DB2 Server for VSE & VM

Interactive SQL Guide and Reference

Version 7 Release 1

SC09-2990-00

DB2 Server for VSE & VM

Interactive SQL Guide and Reference

Version 7 Release 1

SC09-2990-00

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 189.

First Edition (September 2000)

This edition applies to Version 7, Release 1, Modification 0 of the IBM® DATABASE 2™ Server for VSE & VM Program, (product number 5697-F42) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC09-2674-00.

Contents

About This Manual

This manual is a tutorial and reference for IBM DB2 Server for VSE & VM interactive SQL (ISQL) users in a Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA) or Virtual Machine/Enterprise Systems Architecture (VM/ESA) environment. The manual presents reference information for particular topics, followed by tutorial exercises that illustrate the reference information. Screen images (hereafter referred to as *displays*) that are similar to the displays that you view when performing ISQL exercises are included in the text.

The manual begins with an introduction to:

- The DB2 Server for VSE & VM relational database management system (RDBMS).
- The Structured Query Language (SQL) which manipulates and controls data managed by the RDBMS.
- The Interactive SQL (ISQL) facility, through which SQL statements are issued from a display terminal.

The manual continues with query and management techniques for DB2 Server for VSE & VM tables that are used during an ISQL session. Specific guidelines are provided for the use of ISQL commands, SQL statements, routines, and VM functions.

The examples and exercises that are provided throughout this manual to familiarize you with ISQL are supplemented by answers and example table layouts, both of which are in appendixes.

Note:

- The terms *DB2 Server for VM*, *DB2 Server for VSE*, and *DB2 Server for VSE & VM* are used in this guide to refer to DB2 Server for VM Version 7 Release 1 unless otherwise indicated.
- When the term *CICS* is used in this manual, *CICS/VSE* is implied. The CICS/VSE product is required for DBCS support.
- The term *VSE* refers to VSE/Enterprise Systems Architecture Version 2 Release 3 or later.
- The term *VM* is used in this guide to refer to VM/ESA Version 2 Release 2 or later.

For a quick summary of reference information on ISQL functions, refer to the *DB2 Server for VSE & VM Quick Reference* manual.

How to Use This Manual

The following information provides a brief description of each chapter and appendix in the manual.

The Summary of Changes summarizes the technical and library changes made for Version 7 Release 1 of the database manager product.

"Chapter 1. Getting Started" on page 1, introduces the database manager, SQL, and ISQL facilities, and takes the reader through a typical ISQL session using the display terminal.

"Chapter 2. Querying Tables" on page 21, discusses the presentation of queried information both on the display and in a printed report.

"Chapter 3. Managing Table Data" on page 33, provides DB2 Server for VSE & VM table management guidelines for table changes.

"Chapter 4. Using ISQL Commands to Save Time When Executing Statements" on page 35, explains the use of ISQL commands with SQL statements during an ISQL session.

"Chapter 5. Formatting Query Results" on page 43, offers additional methods of displaying queried information. Formatting techniques for both the display and for printed reports are discussed in detail.

"Chapter 6. Storing SQL Statements" on page 67, describes the use of stored SQL statements, including management and processing methods.

"Chapter 7. Creating and Using Routines" on page 73, discusses the use of database manager routines to process a series of ISQL commands or SQL statements.

"Chapter 8. Creating and Managing Tables" on page 83, describes procedures to create and maintain a personal set of database manager tables with an emphasis on the use of table keys. In addition, information is provided that explains the sharing of tables with other database manager users. Guidelines that improve query performance are also provided.

DB2 Server for VM

"Chapter 9. Using VM Functions" on page 99, discusses the use of VM commands and functions to enhance the ISQL session, such as the specification of the number of copies of printed reports.

"Chapter 10. ISQL Commands" on page 105, is a reference chapter for ISQL commands that provides a format diagram, a description, and an example, where appropriate, for each command.

"Appendix A. Answers to the Exercises" on page 165, contains the answers to the exercises in the manual.

"Appendix B. Sample Tables" on page 169, describes the structure and contents of the sample database manager tables used in the examples and exercises in the manual.

"Appendix C. Summary of ISQL PF Keys" on page 183, describes the ISQL commands associated with the PF keys on the keyboard.

"Appendix D. Summary of SQL Statements for Interactive Use" on page 185, lists the SQL statements that you can refer to the *DB2 Server for VSE & VM SQL Reference* manual, if you want more information.

DB2 Server for VSE

"Appendix E. Suppressing the ISQL Sign-On Display for DB2 Server for VSE" on page 187, gives you information on suppressing the ISQL sign-on display and related terminal messages.

The Bibliography lists the full titles and order numbers of related publications, and is followed by the Index.

Components of the Relational Database Management System

Figure 1 depicts a typical configuration with one database and two users.

Figure 2 on page x depicts a typical configuration with one database, one batch partition user, and a CICS® partition with several interactive users.

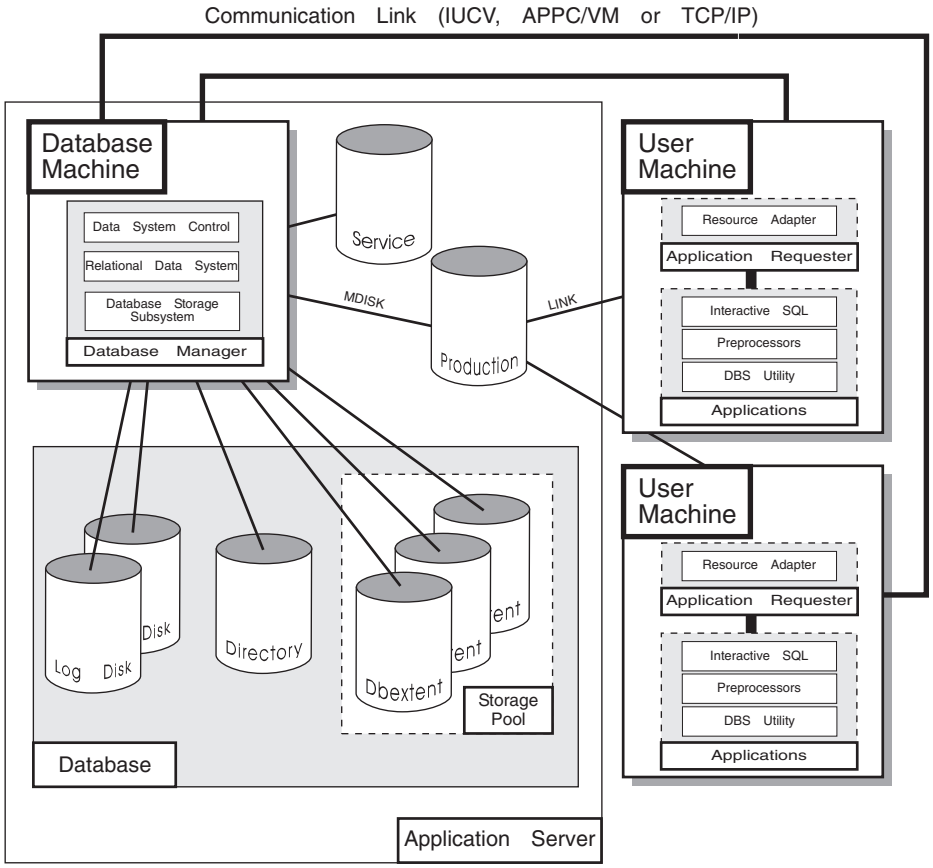


Figure 1. Basic Components of the RDBMS in VM/ESA

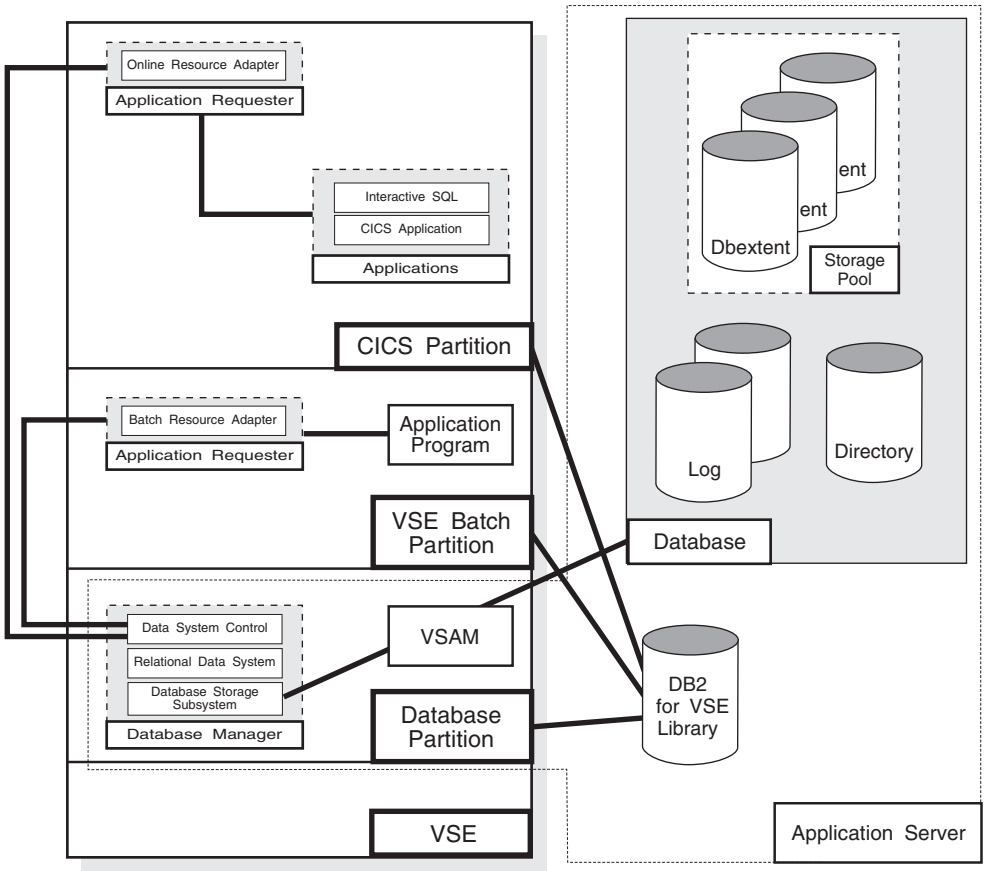


Figure 2. Basic Components of the RDBMS in VSE/ESA

The **database** is composed of :

- A collection of data contained in one or more *storage pools*, each of which in turn is composed of one or more *database extents (dbextents).* A dbextent is a VM minidisk or a VSE VSAM cluster.
- A *directory* that identifies data locations in the storage pools. There is only one directory per database.
- A *log* that contains a record of operations performed on the database. A database can have either one or two logs.

The **database manager** is the program that provides access to the data in the database. In VM it is loaded into the database virtual machine from the production disk. In VSE it is loaded into the database partition from the DB2 Server for VSE library.

The **application server** is the facility that responds to requests for information from and updates to the database. It is composed of the database and the database manager.

The **application requester** is the facility that transforms a request from an application into a form suitable for communication with an application server.

Prerequisite Publications

Although not required, you should have an understanding of material covered in the *DB2 Server for VSE & VM Overview* manual.

Corequisite Publications

The following manuals should be used with this manual:

DB2 Server for VSE & VM SQL Reference
DB2 Server for VSE & VM Database Administration
DB2 Server for VSE & VM Overivew.

How to Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other DB2 Server for VSE & VM documentation:

- Visit our home page at:

  `http://www-4.ibm.com/software/data/db2/vse-vm/`

- A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

  IBM CANADA LTD.
  DB2 Server for VSE & VM
  2S/240/1150/TOR
  1150 Eglinton Ave. East
  North York, Ontario
  Canada M3C 1H7

- Send your comments by electronic mail to one of the following addresses:

Format	Address
Internet	torrcf@ca.ibm.com
Facsimile	(416) 448-6161 (Attention RCF Coordinator)

  Be sure to include the name of the book, the form number (including the suffix), and the page, section title, or topic you are commenting on.

  If you choose to respond through the Internet, please include either your entire Internet network address, or a postal address.

- Fill out the form at the back of this book and return it by mail, by fax, or by giving it to an IBM representative.

Syntax Notation Conventions

Throughout this manual, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right and from top to bottom, following the path of the line.

  The ►►── symbol indicates the beginning of a statement or command.

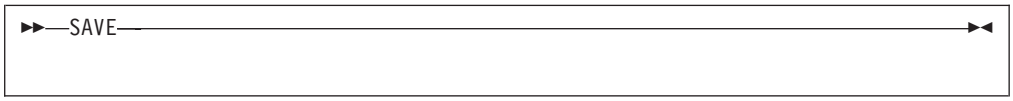
  The ──► symbol indicates that the statement syntax is continued on the next line.

  The ►── symbol indicates that a statement is continued from the previous line.

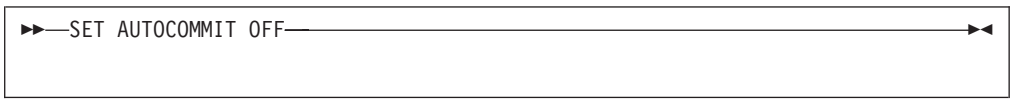
  The ──►◄ symbol indicates the end of a statement.

  Diagrams of syntactical units that are not complete statements start with the ►── symbol and end with the ──► symbol.

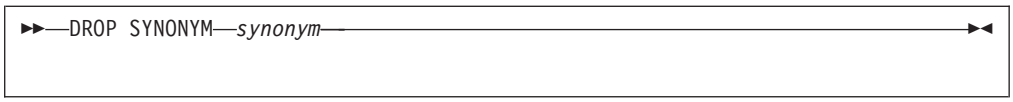
- Some SQL statements, Interactive SQL (ISQL) commands, or database services utility (DBS Utility) commands can stand alone. For example:



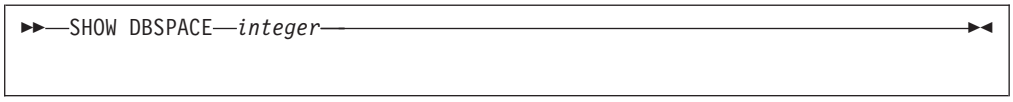
  Others must be followed by one or more keywords or variables. For example:



- Keywords may have parameters associated with them which represent user-supplied names or values. These names or values can be specified as either constants or as user-defined variables called *host_variables* (*host_variables* can only be used in programs).



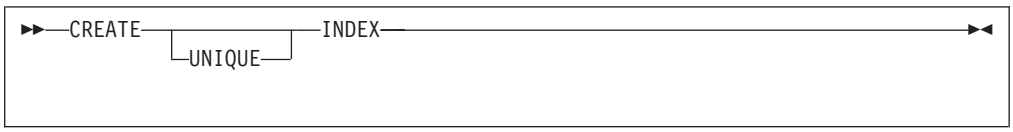
- Keywords appear in either uppercase (for example, SAVE) or mixed case (for example, CHARacter). All uppercase characters in keywords must be present; you can omit those in lowercase.
- Parameters appear in lowercase and in italics (for example, *synonym*).
- If such symbols as punctuation marks, parentheses, or arithmetic operators are shown, you must use them as indicated by the syntax diagram.
- All items (parameters and keywords) must be separated by one or more blanks.
- Required items appear on the same horizontal line (the main path). For example, the parameter *integer* is a required item in the following command:



  This command might appear as:

```
SHOW DBSPACE 1
```

- Optional items appear below the main path. For example:



  This statement could appear as either:

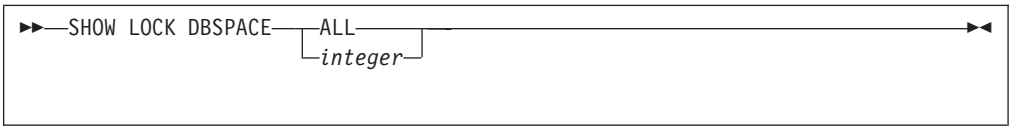
```
CREATE INDEX
```

  or

```
CREATE UNIQUE INDEX
```

- If you can choose from two or more items, they appear vertically in a stack.

If you must choose one of the items, one item appears on the main path. For example:



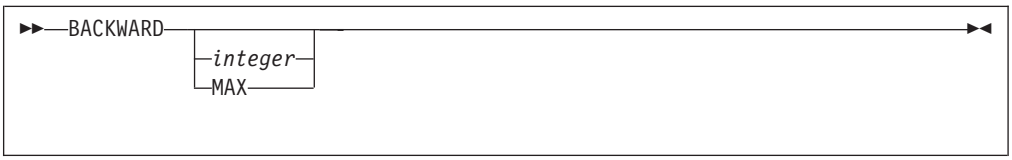
Here, the command could be either:

```
SHOW LOCK DBSPACE ALL
```

or

```
SHOW LOCK DBSPACE 1
```

If choosing one of the items is optional, the entire stack appears below the main path. For example:



Here, the command could be:

```
BACKWARD
```

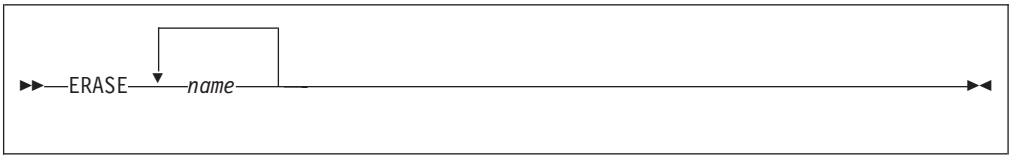
or

```
BACKWARD 2
```

or

```
BACKWARD MAX
```

- The repeat symbol indicates that an item can be repeated. For example:



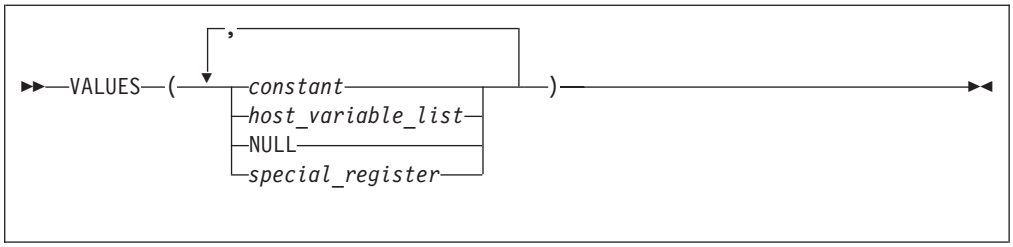
This statement could appear as:

```
ERASE NAME1
```

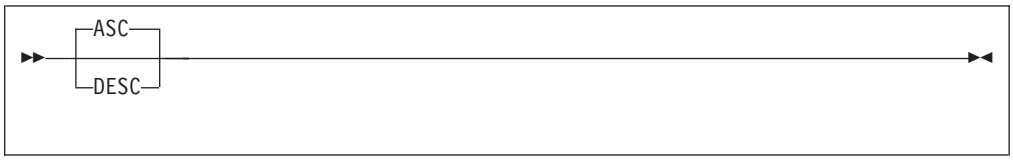
or

```
ERASE NAME1 NAME2
```

A repeat symbol above a stack indicates that you can make more than one choice from the stacked items, or repeat a choice. For example:

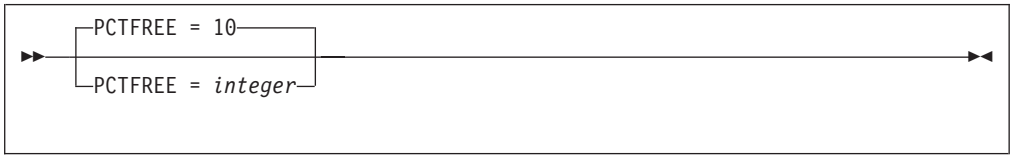


- If an item is above the main line, it represents a default, which means that it will be used if no other item is specified. In the following example, the ASC keyword appears above the line in a stack with DESC. If neither of these values is specified, the command would be processed with option ASC.

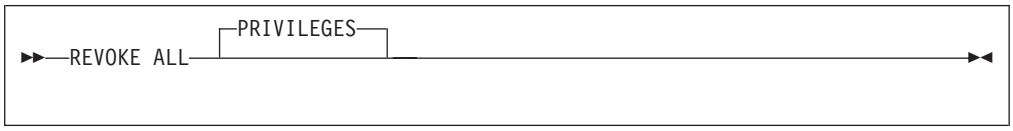


- When an optional keyword is followed on the same path by an optional default parameter, the default parameter is assumed if the keyword is not entered. However, if this keyword is entered, one of its associated optional parameters must also be specified.

  In the following example, if you enter the optional keyword PCTFREE =, you also have to specify one of its associated optional parameters. If you do not enter PCTFREE =, the database manager will set it to the default value of 10.

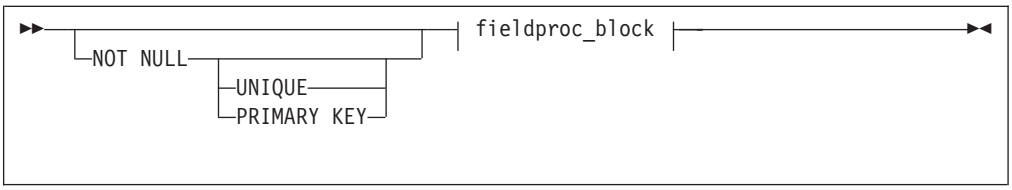


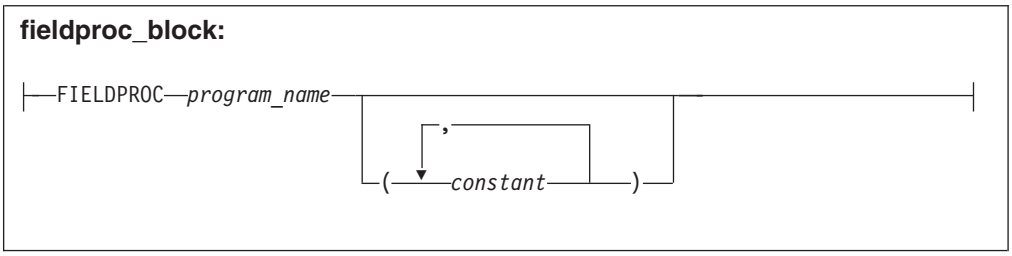
- Words that are only used for readability and have no effect on the execution of the statement are shown as a single uppercase default. For example:



  Here, specifying either REVOKE ALL or REVOKE ALL PRIVILEGES means the same thing.

- Sometimes a single parameter represents a fragment of syntax that is expanded below. In the following example, **fieldproc_block** is such a fragment and it is expanded following the syntax diagram containing it.





SQL Reserved Words

The following words are reserved in the SQL language. They cannot be used in SQL statements except for their defined meaning in the SQL syntax or as host variables, preceded by a colon.

In particular, they cannot be used as names for tables, indexes, columns, views, or dbspaces unless they are enclosed in double quotation marks (").

ACQUIRE
ADD
ALL
ALTER
AND
ANY
AS
ASC
AVG

BETWEEN
BY

CALL
CHAR
CHARACTER
COLUMN
COMMENT
COMMIT
CONCAT
CONNECT
COUNT
CREATE
CURRENT

DBA
DBSPACE
DELETE
DESC
DISTINCT
DOUBLE
DROP

EXCLUSIVE
EXECUTE
EXISTS
EXPLAIN

FIELDPROC
FOR
FROM

GRANT
GRAPHIC
GROUP

HAVING

IDENTIFIED
IN
INDEX
INSERT
INTO
IS

LIKE
LOCK
LONG

MAX
MIN
MODE

NAMED
NHEADER
NOT
NULL

OF
ON
OPTION
OR
ORDER

PACKAGE
PAGE
PAGES
PCTFREE
PCTINDEX
PRIVATE
PRIVILEGES
PROGRAM
PUBLIC

RESOURCE
REVOKE
ROLLBACK
ROW
RUN

SCHEDULE
SELECT
SET
SHARE
SOME
STATISTICS
STORPOOL
SUM
SYNONYM

TABLE
TO

UNION
UNIQUE
UPDATE
USER

VALUES
VIEW

WHERE
WITH
WORK

Authorization Names and Passwords

Authorization names and passwords are limited to 8 characters and cannot have embedded blanks.

Authorization Names without Quotation Marks

The name *must* begin with a letter, $, #, or @ and contain letters, numbers, $, #, @, or underscore. Avoid using SQL reserved words and Database Services Utility reserved words. See the *DB2 Server for VSE & VM Quick Reference* manual for a list of these reserved words.

DB2 Server for VM

Note: # is the usual CP TERMINAL LINEND character.

Authorization Names in Double Quotation Marks

Names can begin with any character and contain any combination of characters when enclosed in double quotation marks. However, the double quotation mark character itself is not allowed within the names, and leading blanks cause errors.

Use of Highlighting in This Guide

Database manager commands and statements are illustrated throughout this manual using

```
indented and highlighted type.
```

You can type these commands and statements. Commands or statements that are

```
indented but not highlighted
```

illustrate additional examples and options. If you type them, you may produce results different from those shown in this manual.

Titles of publications, command variables, parameter values, character strings, and the first use of a term are printed in *italics*.

Any information appearing on the display that is referred to in the text is highlighted in this manual. For example, if the term User ID appears on the display, an instruction to the reader to make an entry beside the term is written in the manual as: Type your user ID in the `User ID` input area. Note the special highlighting for User ID.

Uppercase characters are used for:

- Acronyms (for example, DB2 Server for VSE & VM)
- ISQL commands, statements, and instructions (for example, the CHANGE command)
- Names on top of keys (for example, PF3)
- Names of programs, macros, and EXECs (for example, the PROFILE EXEC)
- Option names, keywords, and special registers (for example, the CASE keyword of the SET command)
- Datasets and files, including tables (for example, the ACTIVITY table).

Italics emphasize the importance of the *italicized phrase*.

Summary of Changes

This is a summary of the technical changes to the DB2 Server for VSE & VM database management system for this edition of the book. All manuals are affected by some or all of the changes discussed here. For your convenience, the changes made in this edition are identified in the text by a vertical bar (|) in the left margin. This edition may also include minor corrections and editorial changes that are not identified.

This summary does not list incompatibilities between releases of the DB2 Server for VSE & VM product; see either the *DB2 Server for VSE & VM SQL Reference*, *DB2 Server for VM System Administration*, or the *DB2 Server for VSE System Administration* manuals for a discussion of incompatibilities.

Summary of Changes for DB2 Version 7 Release 1

Version 7 Release 1 of the DB2 Server for VSE & VM database management system is intended to run on the Virtual Machine/Enterprise Systems Architecture (VM/ESA®) Version 2 Release 3 or later environment and on the Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA™) Version 2 Release 3 Modification 1 or later environment.

Enhancements, New Functions, and New Capabilities

TCP/IP Support for DB2 Server for VSE

TCP/IP support allows:

- VSE online and batch application programs to access remote application servers which support IBM's implementation of the DRDA architecture over TCP/IP.
- Remote application requesters which support IBM's implementation of the DRDA architecture to access the DB2 for VSE application server over TCP/IP.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE & VM Database Administration*
- *DB2 Server for VSE System Administration*
- *DB2 Server for VSE Program Directory*.

DRDA RUOW Application Requester for VSE (Batch)

DRDA Remote Unit of Work Application Requester provides read and update capability in one location in a single unit of work.

This support provides VSE batch application programs with the ability to execute SQL statements to access and manipulate data managed by any remote application server that supports IBM's implementation of the DRDA architecture.

VSE batch application programs can access only one remote application server per unit of work, and must use TCP/IP communications.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE System Administration*
- *DB2 Server for VSE & VM Database Administration*
- *DB2 Server for VSE & VM Application Programming*

- *DB2 Server for VSE Program Directory*.

Stored Procedures Application Requester

A stored procedure is a user-written application program compiled and stored at the server. Stored procedures allow logic to be encapsulated in a procedure that is local to the database manager. The ability to use stored procedures provides distributed solutions that let more people access data faster. SQL statements and replies flowing across the network are reduced and performance is improved.

This support provides VM and VSE (online and batch) application programs with the ability to invoke stored procedures from any remote application servers that support IBM's implementation of the DRDA architecture. It also allows processing of result sets if supported by the remote DRDA application server.

For more information, see the following DB2 Server for VSE & VM documentation:

- *DB2 Server for VSE & VM Application Programming*
- *DB2 Server for VSE & VM SQL Reference*.

Simplified DB2 Server for VSE Installation/Migration

A REXX procedure Job Manager is supplied to assist in the DB2 Server for VSE Installation/Migration process. It controls the overall job flow based on the contents of the job list control tables and the parameter table (supplied as Z-type members). The job manager selects the job control member from the job list file (a Z-type member), extracts the member from the Installation Library, modifies the JCL, submits the job, evaluates the execution, posts the results, and then repeats the process as required. The users are required to modify the parameter table, according to their environment.

This support simplifies the process of installation and migration by reducing user intervention - the Job Manager submits the prepared jobs.

See the *DB2 Server for VSE Program Directory* for further details.

New Code Page and Euro Symbol Code Page Support

The following CCSIDs are now supported:

- 1137: Hindi
- 1142: E-Danish/Norweigan
- 1143: E-Finnish/Swedish
- 1145: E-Spanish.

Additional support has been added for conversions from Unicode (UTF-8) to host CCSIDs.

For a complete list of CCSIDs supported, refer to the *DB2 Server for VM System Administration* and *DB2 Server for VSE System Administration* manuals.

Control Center for VM Enhancements

The following is a list of enhancements that have been made to the Control Center for VM:

- QMF™ Tools: allow the user to list QMF objects, view and unload QMF queries and PROCS, schedule QMF PROCS to execute, and run explain on QMF queries.
- Table Create Tool: allows the user to create new tables.
- Search List improvements.
- Referential Integrity Report tool: A referential integrity map report can now be generated directly from the CMS command interface.

- PL/I prerequisite removal.
- New and improved tape hopper support.
- High density tape drive support: support for high density (non-CMS density) tape drives.

Control Center for VSE Enhancements

The following enhancements have been provided for Control Center for VSE:

- Additional Operator Command Support
- Installation of IBM-provided Stored Procedures.

QMF for VSE & VM Optional Feature

The following enhancements have been provided for QMF for VSE & VM:

- Application Requester support for VSE QMF users
- Command enhancements to default to object type
- Fast path to the QMF home screen
- Cross-platform install capability
- DB2 for AS/400 database access.

QMF for Windows® Optional Feature

The following enhancements have been provided for QMF for Windows :

- Java-based Query
- Aggregating, grouping and formatting directly within query results and automatic Form creation
- Personal portal user interface that launches centrally shared queries and reports, and sends results to spreadsheets, desktop databases, and browsers
- Procedures with REXX.

Reliability, Availability, and Serviceability Improvements

DBNAME Directory Restructuring

ARISDIRD has been restructured to improve readability and flexibility. Each DBNAME entry is now defined explicitly by its type (Local, Remote or Host VM (Guest Sharing)). CICS AXE Transaction TPNs (Transaction Program Names) are still included in the directory as a type of 'LOCALAXE'. The DBNAME Directory Builder program, ARICBDID has been rewritten as a REXX/VSE procedure with extensive error and dependency checking. Support for TCP/IP information is added and 'alias' DBNAMEs are supported. **ALL** DBNAMEs **must** be specified in the new DBNAME Directory, including the Product Default DBNAME "SQLDS". A migration REXX/VSE procedure, ARICCDID, is provided to assist in migrating to the new format. See the *DB2 Server for VSE System Administration* and *DB2 Server for VSE Program Directory* for additional information.

Migration Considerations

Migration is supported from SQL/DS™ Version 3 and DB2 Server for VSE & VM Versions 5 and 6. Migration from SQL/DS Version 2 Release 2 or earlier releases is not supported. Refer to the *DB2 Server for VM System Administration* or *DB2 Server for VSE System Administration* manual for migration considerations.

Library Enhancements

Some general library enhancements include:

- The following books have been removed from the library:
  - *DB2 Server for VM Application Programming*
  - *DB2 Server for VSE Application Programming*

- *DB2 Server for VM Database Administration*
- *DB2 Server for VSE Database Administration*
- *DB2 Server for VSE Installation*
- *DB2 REXX SQL Interface Installation*
- *DB2 REXX SQL Reference*
- *DB2 Server for VM Diagnosis Guide and Reference*
- *DB2 Server for VSE Diagnosis Guide and Reference*
- *DB2 VM Data Spaces Support*

  Note: Information from this book can now be found in the *DB2 Server for VSE & VM Performance Tuning Handbook*
- *DB2 Server for VM Master Index and Glossary*
- *DB2 Server for VSE Master Index and Glossary*.

• The following books have been added to the library:
- *DB2 Server for VSE & VM Database Administration*
- *DB2 Server for VSE & VM Application Programming*
- *DB2 REXX SQL for VM/ESA Installation and Reference*
- *DB2 Server for VSE & VM Diagnosis Guide and Reference*
- *DB2 Server for VSE & VM Master Index and Glossary*.

Refer to the new *DB2 Server for VSE & VM Overivew* for a better understanding of the benefits DB2 Server for VSE & VM can provide.

Chapter 1. Getting Started

This chapter introduces the database manager and the Interactive SQL (ISQL) facility, and shows how to access both.

With the ISQL facility, you can manipulate data contained in a relational database from a display terminal. In the ISQL environment, you will learn such procedures as controlling the display, interpreting database manager messages, entering ISQL commands, and stopping ISQL.

Introducing the DB2 Server for VSE & VM Database Manager

The DB2 Server for VSE & VM relational database management system uses the Structured Query Language (SQL) to manage stored data.

Using SQL, you can query, add, delete, and update data. The language consists of a collection of statements, each of which performs a particular function.

This manual describes how to use the database manager interactively from a CICS display terminal. The terminals supported are IBM 3277, 3278, 3279, or 3290 (or equivalent) with a line length of at least 80 characters and at least 24 lines per display. The database manager also supports the larger display sizes offered by some models of the 3278 and 3279 terminals.

Designed for the interactive user, the manual gives examples of those functions and statements that can be used interactively. For a more comprehensive description of database manager functions, as well as SQL statements used for querying and displaying data, see the *DB2 Server for VSE & VM SQL Reference* manual.

All data stored in the database is in the form of *tables*. The person who creates the table also names it. The table shown in Figure 3 is named `CARS`.

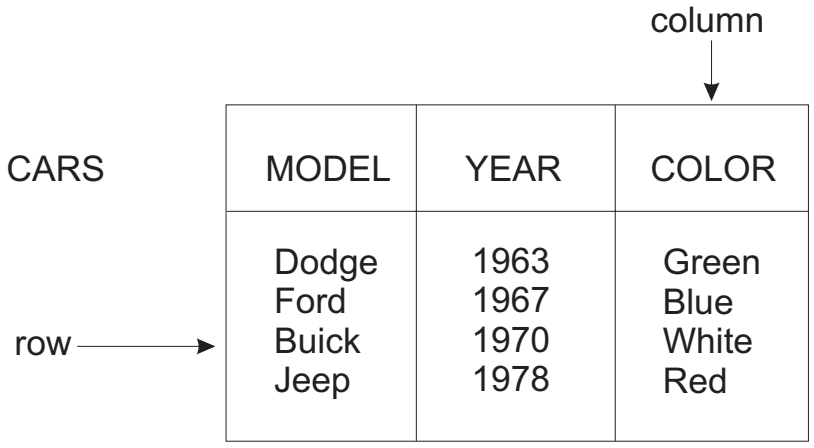


Figure 3. A DB2 Server for VSE & VM Table

A table consists of (vertical) *columns* and (horizontal) *rows*. Each column has a name; the columns in the `CARS` table are `MODEL`, `YEAR`, and `COLOR`.

A *value* is found at the intersection of a column and a row; for example, in the third row of the CARS table, the information in the COLOR column is the value White.

You usually require several tables to adequately store information for an organization. To illustrate how information is stored and used, a set of sample tables is provided for your use. These tables reside in a sample *relational database*. For DB2 Server for VM users, who request access to this relational database are generally granted their own individual online copies. For DB2 Server for VSE users, if your administrator used the IBM-supplied routine ARINEWUS to set you up as a new ISQL user, you receive a copy of the sample tables with all privileges on these tables. Your copy ensures that the table data will remain uncorrupted by other users, which sometimes occurs when multiple users have access to the same data.

If you do not have a copy of the sample tables, you can still do the exercises, but you must use the prefix SQLDBA. with the table names.

This book uses simple examples and samples, but this database manager can readily be used for complex applications in many environments, including scientific, technological, and academic.

Introducing ISQL

SQL statements retrieve, add, delete, and update data in tables, and can be either embedded or interactive. The *Embedded SQL* statements are coded within an application program, and do not begin until the program is being run. SQL statements that are issued interactively, by comparison, create an immediate program or system response for each statement the user issues at a display terminal. This is *interactive processing*, and it is the focus of this manual.

You can issue statements or commands from a display terminal through the interactive SQL (ISQL) facility. Using the following ISQL commands, you can work with the database manager from a display terminal to:

- Control the Display of Data

  You can control the display of data that results from a query in several ways. For example, you can scroll through the results of a query that has more rows than can fit on one display, or look at results that are too wide for the display.
- Print Reports

  You can create reports that are based on data in tables. You can also modify these reports to fit your needs with titles, page numbers, dates and totals.
- Enter Data

  You can enter one or more rows of data into an existing table with the ISQL INPUT command.
- Obtain Online HELP Information from a Display Terminal

  If online HELP information was loaded during installation, you can obtain reference information on your display for ISQL topics. The topics available include reference information about SQL statements, ISQL commands, and messages.

  Online HELP information may also have been installed on your system in other national languages. If you want HELP information in one of these languages, you can specify the language for online HELP by using the SET LANGUAGE command.
- Store SQL Statements for Repetitive Use

You can store SQL statements that are used frequently. A name is assigned to each stored statement to identify it for future use.

- List Operating Characteristics

  You can inquire about operating characteristics that are set using the ISQL SET command. For example, you can see the character that is displayed in null fields.

- Use Routines

  You can store routines, which consist of a series of ISQL commands, SQL statements, or both, and run them at a later time. A routine is especially useful for a frequently used sequence of commands and statements. Routines are discussed in detail in "Chapter 7. Creating and Using Routines" on page 73.

- Switch between Application Servers

  With the CONNECT statement, you can access other application servers. You can access any application servers that have implemented the DRDA protocol. For more information about the CONNECT statement, see the *DB2 Server for VSE & VM SQL Reference* manual.

Introducing the ISQL Display Terminal

ISQL can be run on a variety of display terminals, including the larger display sizes offered by some models of the IBM 3278 and 3279 (or equivalent) display devices. ISQL also supports 5550 terminals with double-byte character sets.

The amount of data displayed varies according to the dimensions of the display terminal being used. Examples in this book are usually a 24-line by 80-character display.

DB2 Server for VSE

Note: The 62 x 160 display requires a CICS/VSE terminal. In an SNA environment, it requires CICS 1.6 ACF/VTAM® Release 1 or later or TCAM Version 2 Release 3 or later.

Using the Program Function Keys

Most keyboards have a group of special keys called program function (or just PF) keys. You use them for quick entry of common ISQL commands. Use of the keys is explained as you proceed through this manual. A summary of the PF keys is in "Appendix C. Summary of ISQL PF Keys" on page 183.

If your PF keys do not match those described in the summary, you can change their functions and tailor them to your needs. Consult the appropriate person in your organization to determine the customized key setting.

Defining the ISQL Session

An ISQL session is signing on, starting ISQL, performing a task (or tasks), and then stopping ISQL.

Before using ISQL from a display terminal, consult the appropriate person in your organization to obtain the following:

DB2 Server for VSE

- A *user ID*. This is a unique user identification that identifies you to the database manager. The user ID and password are optional. If you do not want to type a user ID or password, press ENTER, and the default user ID and password are used. This lets you perform certain activities as defined by your site.
- A *password*. This identification is yours exclusively, and should be kept secret. The user ID and password are optional. If you do not want to type a user ID or password, press ENTER, and the default user ID and password are used. This lets you perform certain activities as defined by your site.
- Access to the sample tables. A description of how your database administrator (DBA) can provide these tables for you is in the *DB2 Server for VSE & VM Database Administration* manual. The tables must be created as described in the *DB2 Server for VSE & VM Database Administration* manual to ensure that the examples and exercises produce the results described in this manual.

DB2 Server for VM

- Access to ISQL. You must have a user ID (to identify you to the VM system) and a password. You must also have authorization to connect with the database manager. This authorization is generally granted by someone with *database administrator* (*DBA*) authority.
- Access to the sample tables. A description of how your DBA can provide these tables for you is in the *DB2 Server for VSE & VM Database Administration* manual. The tables must be created as described in the *DB2 Server for VSE & VM Database Administration* manual to ensure that the examples and exercises produce the results described in this manual.

Note: Your site may have a different signon procedure than that shown on the following pages. Consult the appropriate person in your organization for the correct procedure.

To access ISQL in the VM system environment, you must do the following:

1. Log on the VM system.
2. Start IPL to load the Conversational Monitor System (CMS).
3. Start ISQL.

Each of the activities is described later in this chapter.

Using DBCS for DB2 Server for VM

Some languages, such as Japanese and Korean, require double-byte character sets. If you want to input or see double-byte character sets (DBCS) during your ISQL session, you must enter the following CMS command before starting ISQL:

```
SET FULLSCREEN ON
```

The SET FULLSCREEN ON command lets you input DBCS characters in CMS command mode, and allows input and display of DBCS characters in ISQL command and display mode.

If you are not going to work with DBCS, you do not need this command.

Before Starting ISQL

DB2 Server for VSE

The online resource adapter (ORA) must be enabled before you can start ISQL. ISQL accesses the application server to which the ORA is connected. The ORA connects multiple application servers at a time. You specify the application servers by the DBNAME parameter of the CIRB transaction.

After the online resource adapter is started, you can use the CIRA transaction to add connections to other DB2 Server for VSE & VM application servers. CIRA can be entered multiple times with different *server_name*s to establish connections to the specified application server. With one CIRA command, you can also establish a list of *server_name*s. The system operator or database administrator (DBA) usually performs these tasks.

While in ISQL you can enter a null CONNECT statement to display the connected user ID and application server names.

DB2 Server for VM

Before you can start ISQL, the following steps must be completed for you:

- The DB2 Server for VM disks must be linked.
- The application server must be started.
- The SQLINIT EXEC must be run.

The first two tasks are usually performed by the system operator or the database administrator (DBA), and the SQLINIT EXEC is usually automatically run when you log on your user ID.

The SQLINIT EXEC establishes the required links and defines the name of the application server. If the SQLINIT EXEC is not automatically run for you, run the EXEC before you start ISQL. You must know the name of the application server. If you do not know the name of the application server, speak to your DBA. In the following example, to run the SQLINIT EXEC establishing a link to the SAMPLEDB server, type the following and press ENTER:

```
sqlinit dbname(sampledb)
```

If you want to use the SQLINIT EXEC, refer to the *DB2 Server for VSE & VM Database Administration* manual.

Starting ISQL for DB2 Server for VSE

ISQL runs as a CICS/VSE transaction. A CICS user invokes this transaction just like any other CICS transaction.

After CICS has been activated and the DB2 Server for VSE & VM online support has been started, the CICS user starts ISQL by entering the following four-character CICS transaction identifier from a CICS terminal, and pressing ENTER:

```
isql
```

ISQL responds with a display like the one shown in Figure 4 on page 6.

The screen displays the default application server to which the ORA is connected.

The target application server can be changed by entering the target database information as in the following example:

```
Enter User ID, Password and Target Database, then press Enter

    User ID ==========> ________
    Password =========> ________
    Target Database ==> SQLDB1_TOR_INV
```

This is also the application server to which ISQL will be connected to subsequently. At this point, because ISQL is not connected to the application server to which the ORA is connected, you can end the ORA and restart to another database. If you log on to ISQL again, the signon screen is redisplayed showing the target application server to which the ORA is now connected.

When not connected to a target application server, the ISQL system displays the line `Online Support is not ready. Please exit ISQL`. The online resource adapter (ORA) must be enabled before you can start ISQL. For additional information on enabling the ORA, and other requirements before you can start ISQL, refer to "Before Starting ISQL" on page 5 and the *DB2 Server for VSE System Administration* and *DB2 Server for VM System Administration* manuals.

```
  Welcome to the interactive SQL facility of DB2 for VSE


          IIIIIIII  SSSSSSSS  QQQQQQQQ   LL
             II     SS        QQ     QQ   LL
             II     SSSSSSSS  QQ     QQ   LL
             II           SS  QQ  QQ QQ   LL
          IIIIIIII  SSSSSSSS  QQQQQQQQ   LLLLLLLL
                                    QQ

        Default Target Database is SQLDS

Enter User ID, Password and Target Database, then press Enter

    User ID ==========>
    Password =========>
    Target Database ==>


To exit now, enter EXIT in user ID field with no password, press Enter.
  To exit later, use the EXIT command. Use the HELP command for help.
```

Figure 4. Initial ISQL Screen

Alternative Methods for Starting ISQL

If you decide to press ENTER instead of specifying your user ID and password when the signon display is displayed, then you must use the explicit database manager CONNECT statement as follows:

```
CONNECT authorization_name IDENTIFIED BY password TO server_name
```

An exception to using the CONNECT statement in the above situation is if your installation has defined a default *authorization_name* for you, in which case, you do not have to specify your *authorization_name*, password, nor *server_name*.

If the TO parameter is not specified, then the connection to the previously connected server will be maintained.

An alternate method for invoking ISQL without having to use the ISQL signon display is described in "Appendix E. Suppressing the ISQL Sign-On Display for DB2 Server for VSE" on page 187.

You may run an ISQL routine as part of the ISQL signon procedure. Refer to "Using the ISQL Transaction Identifier (DB2 Server for VSE)" on page 74 and "Appendix E. Suppressing the ISQL Sign-On Display for DB2 Server for VSE" on page 187.

Signing On by Using the Signon Display

Signon is accomplished by:

1. Entering your user ID at the location identified by the cursor (two positions to the right of `User ID ==>`). There must be one, and only one, blank between the > and your user ID.
2. Positioning the cursor two positions to the right of `Password ==>` and typing your DB2 Server for VSE & VM password. There must be one, and only one, blank between the > and your password. (You can use the tab key to position the cursor to the correct position. Tab is the key with the arrow pointing to a vertical line on the right side of the key.)

   The area to the right of `Password ==>` is a dark field; characters typed in this area remain invisible.
3. Entering the target application server identified by the cursor (two positions to the right of `Target Database ==>`). There must be one, and only one, blank between the > and the target application server.
4. Pressing ENTER.

Note: On some occasions, your display may lock up and you are unable to type data. If this happens, simply press RESET, ensure that the cursor is in the correct position, and retype the information.

When ISQL recognizes your signon name and password, it responds with the display shown in Figure 5 on page 8.

```
ARI7399I The ISQL default profile values are in effect.
ARI7079I ISQL initialization complete.
ARI7080A Please enter an ISQL or SQL command.









_
                                                          Enter a command
```

Figure 5. Initial ISQL Display

Leaving ISQL from the Signon Display

You can exit ISQL from the signon display by typing `EXIT` in the user ID field, leaving the password field blank, and pressing ENTER. If you enter `EXIT` in the user ID field, but also enter a password, ISQL will treat it as a user ID and continue processing.

If you enter `EXIT` to end ISQL, message `ARI7601I` is displayed as follows:

```
ARI7601I ISQL ended normally by your request.
```

Starting ISQL for DB2 Server for VM

Now type ISQL (or the name of your EXEC for ISQL) as follows:

```
isql
```

Press ENTER.

ISQL responds with a display similar to the one shown in Figure 6.

```
Ready; T=0.01/0.03 13:41:49
isql
ARI0659I Line-edit symbols reset:
         LINEND=# LINEDEL=OFF CHARDEL=OFF
         ESCAPE=OFF TABCHAR=OFF
ARI0662I EMSG function value reset to ON.
ARI0320I The default server name is SAMPLEDB.
ARI7716I User SQLUSER1 connected to server SAMPLEDB.
ARI7399I The ISQL default profile values are in effect.
ARI7079I ISQL initialization complete.
ARI7080A Please enter an ISQL command or an SQL statement
```

Figure 6. Initial ISQL Display

Controlling the Display

If your system default is not set for the full-screen environment, you can set it by typing SET FULLSCREEN ON on a CMS command line. Full-screen CMS uses several preset PF keys and displays the current PF key settings in the bottom portion of the display. Before you start ISQL, the PF settings reflect those for the CMS environment. For example, keying PF12 lets you type a command on the command line. After you start ISQL, the PF key settings displayed change to the ISQL settings, and the display is similar to the one shown in Figure 7.

```
                                Fullscreen CMS          Columns 1 - 79 of 81

Ready; T=0.01/0.03 13:41:49
isql
ARI0659I Line-edit symbols reset:
         LINEND=# LINEDEL=OFF CHARDEL=OFF
         ESCAPE=OFF TABCHAR=OFF
ARI0662I EMSG function value reset to ON.
ARI0320I The default server name is SAMPLEDB.
ARI7716I User SQLUSER1 connected to server SAMPLEDB.
ARI7399I The ISQL default profile values are in effect.
ARI7079I ISQL initialization complete.
ARI7080A Please enter an ISQL command or an SQL statement









PF1=HELP      2=START     3=           4=           5=RECALL     6=
PF7=          8=          9=HOLD      10=          11=          12=RETRIEVE
```

Figure 7. Initial ISQL Display with Full-Screen CMS

All commands you type appear in the input area near the bottom of the display. All data returned by ISQL appears on a different display; when you exit from such a display, a display similar to the one shown in Figure 7 is returned.

If you want to temporarily suspend the full-screen option, see the *VM/ESA: CMS Command Reference* manual, for information on the SET FULLSCREEN command.

Interpreting DB2 Server for VSE & VM Messages

The system displays messages about certain operating conditions for your terminal session. (See Figure 6 on page 8 for examples of the system messages.) You can receive messages in the language you want, depending on your site. The text of the messages that you receive may be slightly different from those shown here.

Messages have two parts. The first part is the message number, which remains the same regardless of the language setting. An example of a message number is `ARI0503E`. It starts with the letters `ARI`, which identify it as a DB2 Server for VSE & VM message. Then it contains a four-digit number to identify the message. Finally, it ends with one of the following letters that indicates the message type:

I An informational message is displayed.
W A system wait message is displayed.
E An error has occurred and may require some action on your part.
A An action on your part is required.

D Your decision and reply is required.

The second part is the text of the message; for example, `An SQL error has occurred`. ISQL uses the same language for messages as your CMS language setting. For more information, see "SET" on page 149. In most situations, the text is self-explanatory. If it is not, you can use the message number with the HELP command. (For information about using the HELP command to display the message description, refer to "Chapter 10. ISQL Commands" on page 105.) You can also use the message number to look up the message description in the *DB2 Server for VSE Messages and Codes* and *DB2 Server for VM Messages and Codes* manuals.

Messages `ARI0503E`, `ARI0505I`, and `ARI0504I` are usually encountered when an error is detected while the system is processing an SQL statement. Message `ARI0504I` is an informational message that provides data useful to those who are responsible for locating problems within the system. You can usually ignore this message, but there may be occasions when you are prompted to record its contents.

Message `ARI0504I` always follows message `ARI0505I`. Message `ARI0503E` indicates that the SQL statement being processed was unsuccessful. Message `ARI0505I` follows `ARI0503E` and provides a 3-digit number called an *SQLCODE* in its message text. For example, assume you receive the following messages:

```
ARI0503E An SQL error has occurred.
         SQL command begins properly but is incomplete.
ARI0505I SQLCODE = -106  SQLSTATE = 37501  ROWCOUNT = 0
ARI0504I SQLERRP: ARIXPA1 SQLERRD1: -150 SQLERRD2: 0
```

The SQLCODE provided in message `ARI0505I` is `-106`. Text for this SQLCODE begins on the second line of message `ARI0503E` and describes the cause of the error. If you want further explanation of the error, use the SQLCODE (in this example, `-106`) to view the online help information or to look up the explanation in the *DB2 Server for VSE Messages and Codes* and *DB2 Server for VM Messages and Codes* manuals.

The SQLSTATE information provides a code for error conditions that are common across relational database products. For more information about SQLSTATE, refer to the *DB2 Server for VSE Messages and Codes* and *DB2 Server for VM Messages and Codes* manuals.

The ROWCOUNT information is useful only for certain commands and is explained in the command descriptions. You can ignore the information in message `ARI0504I`, unless you are prompted to record it.

Entering Commands

DB2 Server for VSE

Figure 8 on page 11 shows a diagram of how ISQL divides your display.

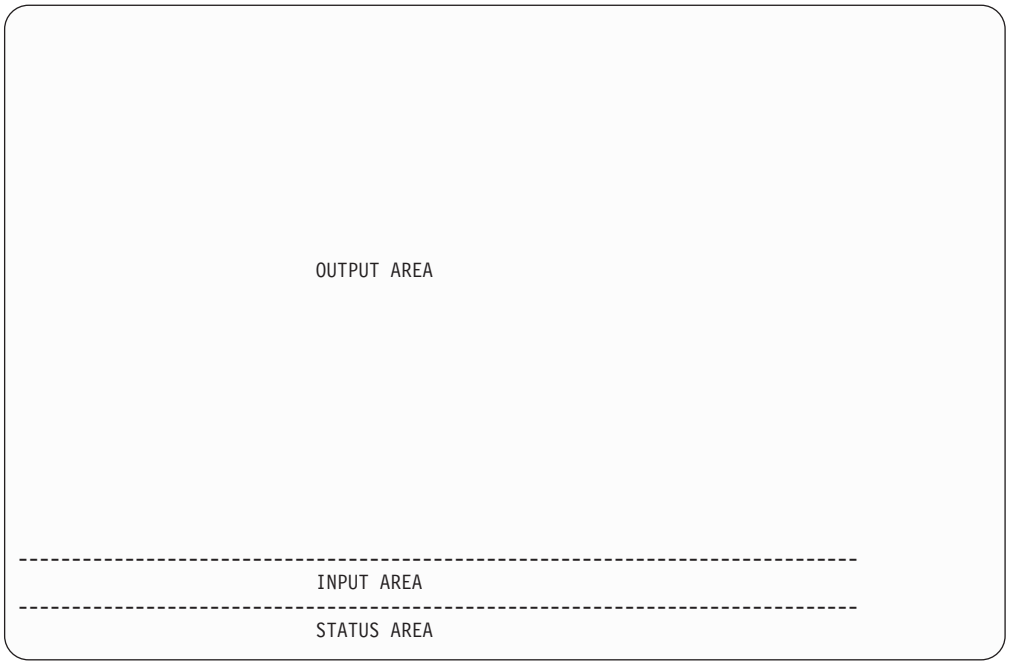


Figure 8. DB2 Server for VSE & VM Display Format

The bottom line is the *status area*. Here, ISQL provides brief messages which indicate its current status during your terminal session. For example, whenever `Enter a command` appears, you know ISQL is ready to receive another command.

Note: For the remainder of this chapter, the general term *command* indicates ISQL commands, SQL statements and data.

You type commands in the input area, which is just above the status area. The input area consists of a single line and begins at the second-character position of the line. You do not have to move the cursor to this location; it is placed there by the database manager.

Although the `-->` cursor-movement key can be used to leave a blank space in the input line, it does not provide a blank character as does the space bar. Use the space bar to insert a blank character; use the cursor key to move the cursor.

The input area is also used by ISQL to provide the following message:

```
ARI7044I Command in progress. Terminal is now free.
```

This message, displayed when you have typed a command that is taking longer to execute than a preset amount of time, is meant for users involved with more than one CICS transaction. This message is only displayed if the ISQL user is connected to a local application server. ISQL is one of several CICS transactions available at your terminal. If you *are not* involved with multiple CICS transactions, ignore the message and wait for the command to complete. If you *are* involved with multiple CICS transactions and want to issue another CICS transaction while waiting for the command to be completed, this message indicates the terminal is free to do so. To enter another CICS transaction in response to this message, press CLEAR and type the desired CICS transaction identifier code. The transaction must not be pseudoconversational. For more information, see the *DB2 Server for VSE System Administration* and *DB2 Server for VM System Administration* manuals.

The output area displays information typed in the input area. It is also used to display any database manager responses to your input. Specific uses of the output area are discussed where appropriate in the manual.

DB2 Server for VM

A diagram of the way ISQL divides your display is shown in Figure 9.
The last 21 characters of the bottom line are the status area. Here, the current VM

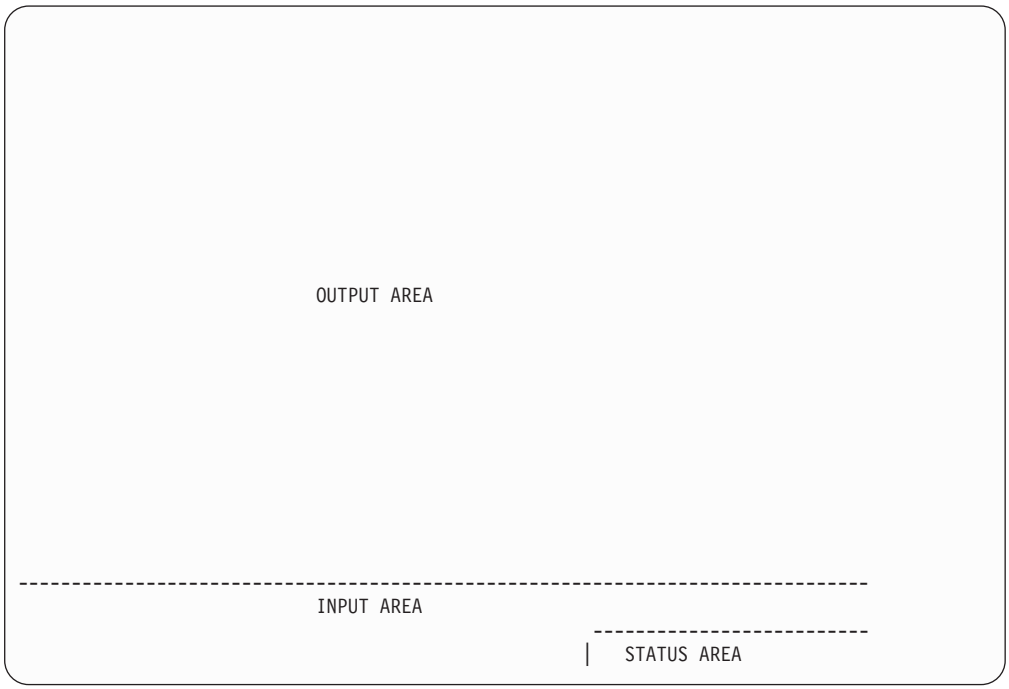


Figure 9. DB2 Server for VSE & VM Display Format

system status is displayed during your terminal session. For example, when the `VM READ` message appears, you know that the system is ready to receive another command from you.

You type commands and data in the input area, which includes all of the line above the status area and the part of the next line that is to the left of the status area. The input area begins at the second character position of the display. You do not have to move the cursor to this location; it is placed there by the ISQL system.

The key marked `-->` only moves the cursor. This key does not insert blank characters into an input line. To insert a blank character, you must use the space bar.

The output area displays your commands and data after they are typed in the input area. It also displays ISQL status messages and any database manager responses to your input. Specific uses of the output area are discussed where appropriate in the manual.

ISQL status messages are displayed as informational or `ARI`-type messages. They appear on the last line of the output area.

The command line is usable at all times. You can type commands or statements even while others are being processed. For example, you can type a SELECT statement and then several FORMAT commands before the query results are displayed.

VM also permits you to stack commands. Commands are stored in the console stack; VM will then execute them one at a time.

Entering Commands While Viewing the Results of a Query

You can type any SQL statement or ISQL command while viewing the results of a query as long as that result does not originate from a query processed in an ISQL routine. Exceptions to this restriction are display commands, which you can use while viewing the results of a query being processed in a routine. (Routines are discussed in "Chapter 7. Creating and Using Routines" on page 73.)

If you do type a command, other than a display command, while viewing the results of a query being processed in a routine, you receive the following message:

```
ARI7956E Command failed.  This command is not valid
         while you view a query result from a routine.
```

The query result, because it is the result of a routine, is not affected.

If you are viewing a query result that is not from a routine, and you type an ISQL command or SQL statement that changes the current display, you receive the following message:

```
ARI7955I The system ended your query result to process your command.
```

In this situation, ISQL ends the current query display and processes the new command or statement.

If you are viewing a query result that is not from a routine, and you type a display command, the display command is processed.

Understanding ISQL Modes

The two modes in ISQL are *wait* and *display*. They provide different displays and, for particular commands and statements, they react differently. Wait mode is indicated by the `VM READ` displayed in the status area (lower right corner) of the display. In wait mode, you can enter any SQL statement and any ISQL command other than display commands. Each command or statement that you type is displayed line by line in the output area. From wait mode, you can query your application server. A query (a SELECT statement) puts the display into display mode.

In display mode, no message is displayed in the status area, and the entire output area displays the results of a query. From display mode, you can type any SQL statement or any ISQL command including ISQL display commands to move through the displayed data. You end display mode when you type END, an SQL statement other than SELECT, an ISQL command other than a display command, or an incorrect SELECT statement. In all instances wait mode is returned.

Using the Continuation Character

Sometimes your input (commands, statements, or data) is too long to fit on the single input area line. When this happens, you can continue typing by using the continuation character, which is usually a hyphen. When the database manager is installed, the hyphen is the continuation character. The continuation character can be changed. For more information on the continuation character, see the SET command description in "Chapter 10. ISQL Commands" on page 105. This continuation character causes what you type to be redisplayed in the output area and frees the input area for more typing.

If a line ends in a complete word, leave a space after it, type the hyphen, and press ENTER. If you have to break a word at the end of a line, just type the hyphen without a space before it, and press ENTER.

DB2 Server for VSE

The continuation character lets ISQL know that you have not finished with the command, and it responds with `continue command` in the status area.

DB2 Server for VM

The continuation character indicates that you are not finished with the command. The system displays `VM READ` in the status area. It also displays the following message in the output area:

```
ARI7068I Your input is being continued. Type more input or press Enter.
```

Type the additional input for the command. When you are finished typing, press ENTER. The entire command cannot exceed 2048 characters and the last line of the command must not end with a continuation character.

If the output area becomes full, you are prompted to clear it. Press CLEAR to clear both the input and output areas to allow the command in progress to continue. Incomplete portions of the command that were in the output area are stored in the SQL command buffer, but are not displayed. You do not have to repeat them. Everything in the input area is removed from the display when you press CLEAR.

You can also press PA2 (Field Mark) to clear the output area. This clears only the output area and leaves anything that you have typed in the input area intact.

Correcting Typing Errors

You can correct a typing error in the input area by backspacing and typing the correct characters before you press ENTER.

If you press ENTER before you notice the typing error, the command or statement containing the typing error is displayed in the output area. It is no longer in the input area. You cannot backspace and retype information already in the output area.

To correct a mistake in a multiple-line command, type the following ISQL command at the beginning of the input area and press ENTER.

```
ignore
```

ISQL responds by telling you that it has ignored all lines previously entered in a multi-line input.

DB2 Server for VSE

The status area contains `Enter a new command`. You can then retype the command correctly.

You can also use the RETRIEVE function to review and correct any typed information. ISQL saves the typed lines.

DB2 Server for VM

The status area contains `VM READ`. You can then retype the information correctly.

You can also use the CMS RETRIEVE function to review and correct typed information. CMS saves the lines that you have typed.

To use the RETRIEVE function, press PF12 (or PF24).

This function retrieves the last line that you typed and redisplays it in the input area for review and correction. The cursor is positioned at the end of the displayed line. You can continue to press PF12 (or PF24) until the command you want to change is redisplayed in the input area. Then, you can make any necessary corrections and press ENTER.

Each time you press F12, ISQL retrieves another line and redisplays it in the input area. After the earlier line is retrieved, command retrieval begins again with the most recent command typed. ISQL retains a varying number of your commands depending on their size. The shorter the commands, the more ISQL retains. For more information on the RETRIEVE function, see "Retrieving and Correcting SQL Lines" on page 35.

Canceling Running Commands

It may take extended time for your commands, statements, or routines to be performed. The processing of your query statement, for example, can be delayed if another user is accessing the same table.

DB2 Server for VM

You can query the status of the command, statement or routine by issuing the CMS Immediate command, SQLQRY. The SQLQRY command returns information that can help you to determine the cause of the delay. For more information on the SQLQRY command, refer to the *DB2 Server for VSE & VM Database Administration* manual.

You have the option of canceling in-progress commands (such as the ISQL INPUT command), long-running SQL commands (in which case message ARI7044I is issued), or LUWs (where AUTOCOMMIT is off), by issuing a CANCEL command. You can type this command anytime during the typing of commands, statements, or data.

To cancel an operation, type:

```
cancel
```

DB2 Server for VSE

This command also performs a ROLLBACK operation.

To cancel a long-running command, wait until you receive the message:

```
ARI7044I  Command in progress. Terminal is now free.
```

Press CLEAR, and type:

```
isql cancel
```

DB2 Server for VM

This command also causes a ROLLBACK RELEASE operation to be performed. The database manager releases your connection to the application server that you are using. If you previously issued an explicit CONNECT command to connect to a particular application server, you must reissue another CONNECT command. If you do not, the next statement or command you type causes you to be implicitly connected to the default application server. If you have been implicitly connected to a default application server, your next statement or command implicitly reconnects you.

When you cancel a routine that is in progress, the routine stops execution immediately, and you are left in ISQL.

For information on the effects of AUTOCOMMIT on the CANCEL command, see "Using the AUTOCOMMIT ON Setting" on page 33, or the description of the CANCEL command in "Chapter 10. ISQL Commands" on page 105.

Obtaining Online HELP Information at a Terminal

The database manager offers online HELP information that provides you with, for example, information about ISQL commands; this saves you time in looking up the commands in "Chapter 10. ISQL Commands" on page 105.

Online HELP information is available for:

- Reference information
- SQL statements
- ISQL commands
- Messages, codes and SQLSTATEs.

Online HELP information is not supported on non-DB2 Server for VSE & VM application servers.

Selecting Online HELP Information

Online HELP information may also have been installed on your system in languages other than English. If you want to read HELP information in one of these other languages, you can specify the language for online HELP using the SET LANGUAGE command, which is discussed in "Language of Messages and HELP Text" on page 62 and in "Chapter 10. ISQL Commands" on page 105.

The database manager online HELP information is stored in a table and displayed in the same manner as queries. After you have retrieved some information, you can use display commands to move through it. The online information is broken up into many topics. Each topic is identified by a name. Most topic names are either a statement name (such as SELECT or INSERT), a message number (such as ARI7399I or ARI7307A), a message code (such as -205 or 100), or an SQLSTATE (such as SQLSTATE 01525). For example, to retrieve the online help information for the UPDATE statement, type:

```
help update
```

To move forward through the UPDATE information, type FORWARD commands until you have reached the end. See "Results That Have Too Many Rows for One Display" on page 22 for more information about FORWARD. Type the END command to end the display.

Some topic names have more than one word. When retrieving these topics, enclose the topic name in single quotation marks. For example, to retrieve the online help information for search conditions, type:

```
help 'search conditions'
```

To see a list of the topics available, type:

```
help contents
```

To obtain a list of topics, and a description of how to use the HELP command, press PF1 or type:

```
help
```

Typing While Viewing Online HELP Information

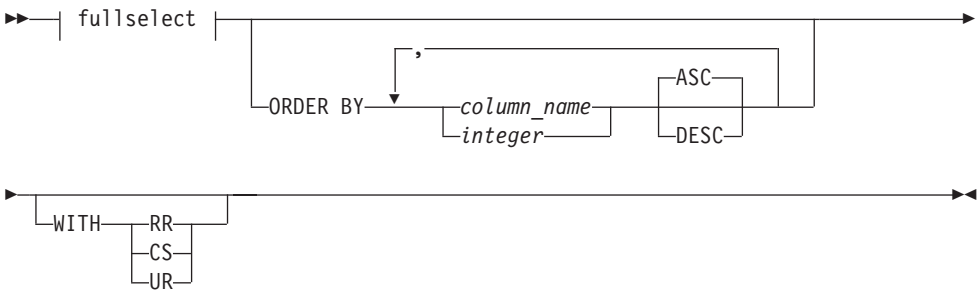
You can type commands and statements while online HELP information is being displayed. This is particularly useful when you want to type a statement, and you know all the necessary column and table name information, but you cannot remember the correct format for the statement.

If you want to execute a SELECT statement but cannot remember its format, type:

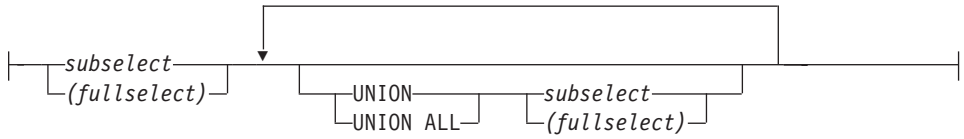
```
help select
```

This displays online HELP information for the SELECT statement similar to the following:

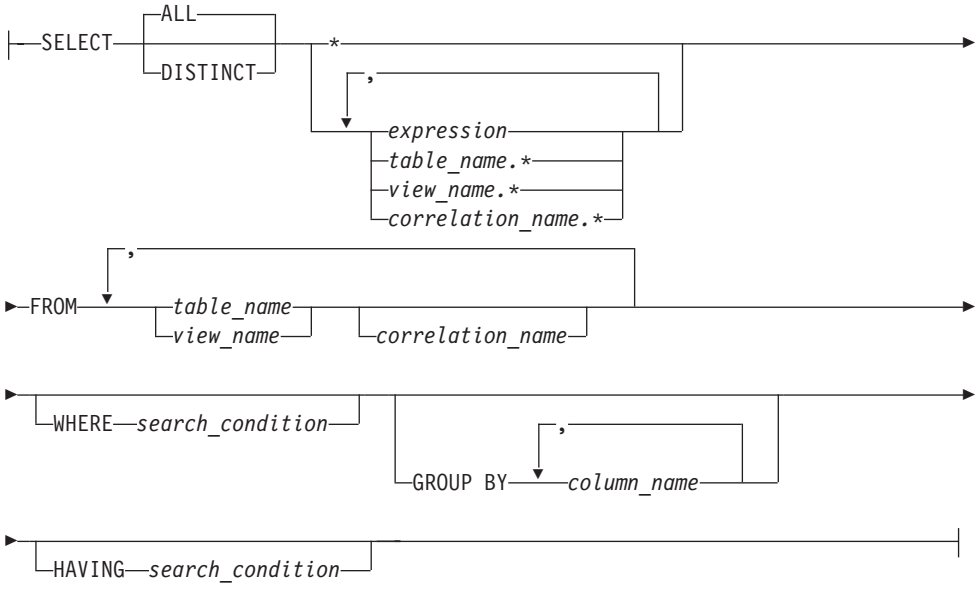
interactive-select-statement



fullselect:



subselect:



```
PROJNO  PRSTAFF
------  -------
IF2000     1.00
OP2012     1.00
OP2013     1.00
OP2011     1.00
PL2100     1.00
AD3112     1.00
MA2111     2.00
IF1000     2.00
AD3113     2.00
AD3111     2.00
* End of Result *** 10 Rows Displayed ***Cost Estimate is 1********************
```

Figure 10. A Query Result after Using Online HELP Information

Using ISQL on a Non-DB2 Server for VM Application Server (VM Only)

With the implementation of the Distributed Relational Database Architecture™ (DRDA) protocol, you can use ISQL to access data on non-DB2 Server for VM application servers that support the DRDA protocol and on which ISQL is loaded. If you plan to use ISQL to access data on a non-DB2 Server for VM application server, you should check with your System Administrator to see whether your application requester and the non-DB2 Server for VM application server are set up for access using the DRDA protocol.

Using ISQL on a Remote Application Server (VSE Only)

With the implementation of the Distributed Relational Database Architecture (DRDA) protocol, you can use ISQL to access data on remote DRDA application servers that support the DRDA protocol and on which ISQL is loaded. If you plan to use ISQL to access data on a remote DRDA application server, you should check with your System Administrator to see whether your application requester and the remote DRDA application server are set up for access using the DRDA protocol.

Stopping ISQL

To stop communication with the application server through ISQL, type the following command in the input area and press ENTER.

```
exit
```

This completes your first ISQL session.

Using CHARNAME and DBCS Options in VSE

ISQL queries the following system catalog tables at initialization:

1. SYSTEM.SYSOPTIONS catalog - to retrieve information such as the default setting for DBCS and CHARNAME. Starting from DB2 Server for VSE Version 6 Release 1, this information is retrieved from the SQLGLOB file. The user DBCS and CHARNAME SQLGLOB values become the default setting for DBCS and CHARNAME, if they exist. Otherwise, the global DBCS and CHARNAME SQLGLOB values become the default setting for DBCS and CHARNAME.

   ISQL uses the user CHARNAME to get the folding table to fold input from the terminal from lowercase to uppercase. However, for CCSID data conversion, ISQL uses the global CHARNAME.

2. SYSTEM.SYSCHARSETS catalog - to retrieve the CHARTRANS information corresponding to the default CHARNAME. Starting from DB2 Server for VSE Version 6 Release 1, this information is retrieved from the ARISSCRD phase file, if it exists. If not, the system defaults are supplied.

Chapter 2. Querying Tables

Querying table data is the most common activity performed by the database manager.

The topics that follow describe query techniques. Before continuing, check with the persons responsible for the system in your organization to ensure the sample tables are stored in the system for your use. If they are, you can use them to practice commands and statements as you learn them. Having access to the sample tables is not absolutely necessary, but being able to work with them from your display terminal is of considerable help as you learn to use the system.

Selecting Columns

Use the interactive SELECT statement to query data in a table. With this statement, you *select columns from a table*. See the *DB2 Server for VSE & VM SQL Reference* manual for details on how to use SELECT statements.

Note: ISQL recognizes (by default) uppercase and lowercase letters as identical in commands. Also by default, all character data in a table is stored as uppercase. Your query on the row containing the word *James*, for example, selects the row containing *JAMES* in uppercase, even if you typed *james* in lowercase in your query. These case defaults can be changed using the SET CASE command, which is discussed in detail in "Chapter 10. ISQL Commands" on page 105.

The displayed length attribute of VARCHAR columns is determined by the default VARCHAR length set in your profile. The system default length for VARCHAR columns is 20 characters. Only the first 20 characters are displayed. You can change the default length by using the SET command. Because the default length for VARCHAR columns can be different, you may see more or fewer characters on your display of VARCHAR columns than are shown in this manual.

You may have seen the following message on your display just before the query result:

```
ARI7960I The query cost estimate for this SELECT statement is 1.
```

The *cost estimate* value for this SELECT statement is also displayed in the last row of output at the end of the query result.

This information is useful for estimating the time needed to obtain query results. Larger cost-estimate values suggest that processing takes longer. The information is provided *before* the query result is displayed so that you can cancel the request if you think it will take too much time. The information is also provided *after* the query result is displayed, because the query result may have been displayed before you read message ARI7960I, or the message may have been suppressed. For additional information on the SET COSTEST command refer to "Chapter 10. ISQL Commands" on page 105.

The cost estimate value is not a unit of time such as seconds. The value becomes more useful as you enter more SELECT statements and become acquainted with the processing times for the cost estimate values.

Using Query Results

When you have the query result on your display, you can look at it, and perhaps print it. You can manipulate query results shown on your display by using ISQL display commands. The following sections show how to look at the entire query results when it does not all fit on your display, and how to direct the query result to a printer.

Displaying Query Results

When your query results are larger than a single display, you must know the display commands with which you can view different portions of the query result. The following examples show how to use the other display commands to query longer and wider tables.

Type the following query and press ENTER:

```
select * -
from proj_act -
where projno <> 'AD3111' -
order by actno,acendate
```

This query selects all the rows that contain project numbers other than *AD3111* in the column named PROJNO from the PROJ_ACT table. It produces a result similar to the display in Figure 11.

```
PROJNO   ACTNO  ACSTAFF  ACSTDATE    ACENDATE
------  ------  -------  ----------  ----------
AD3100      10     0.50  1982-01-01  1982-07-01
MA2100      10     0.50  1982-01-01  1982-11-01
IF2000      10     0.50  1982-01-01  1983-01-01
IF1000      10     0.50  1982-06-01  1983-01-01
IF1000      10     0.50  1982-01-01  1983-01-01
AD3110      10     1.00  1982-01-01  1983-01-01
OP1000      10     0.25  1982-01-01  1983-02-01
OP1010      10     1.00  1982-01-01  1983-02-01
OP2010      10     1.00  1982-01-01  1983-02-01
MA2110      10     1.00  1982-01-01  1983-02-01
MA2100      20     1.00  1982-01-01  1982-03-01
PL2100      30     1.00  1982-02-01  1982-09-01
PL2100      30     1.00  1982-01-01  1982-09-15
MA2111      40     1.00  1982-01-01  1983-02-01
MA2111      50     1.00  1982-01-01  1982-06-01
OP2000      50     0.75  1982-01-01  1983-02-01
AD3112      60     0.75  1982-01-01  1982-03-15
AD3112      60     0.50  1982-02-01  1982-03-15
MA2112      60     2.00  1982-01-01  1982-07-01
AD3113      60     1.00  1982-04-01  1982-09-01
MA2113      60     1.00  1982-02-15  1982-09-01
AD3113      60     0.75  1982-03-01  1982-10-15
```

Figure 11. Example of a Long Query Result

This query result is used in the following topics. Do not end it yet.

Results That Have Too Many Rows for One Display

The query results that you are retrieving are longer than can be shown on one display. To see the remaining portions of the query result, you must know how to scroll forward through it. Usually, the command to scroll forward 20 rows is FORWARD 20.

Instead of entering FORWARD 20, take advantage of the special function of the ENTER key on query results. While viewing a query result, pressing ENTER without having a command in the input area repeats the previous display command. If there was no previous display command entered, the length of the display is scrolled forward. If the previous command is BACKWARD MAX and the display starts from the beginning of the table, press ENTER to scroll forward the length of the display.

Press ENTER. On a 24 X 80 terminal you see a result similar to the display in Figure 12.

```
PROJNO   ACTNO  ACSTAFF  ACSTDATE    ACENDATE
------  ------  -------  ----------  ----------
AD3112      70     0.50  1982-02-01  1982-03-15
AD3113      70     0.50  1982-06-15  1982-07-01
AD3112      70     1.00  1982-03-15  1982-08-15
MA2112      70     1.00  1982-02-01  1982-10-01
AD3112      70     0.75  1982-01-01  1982-10-15
AD3112      70     0.25  1982-08-15  1982-10-15
AD3113      70     0.75  1982-09-01  1982-10-15
AD3113      70     1.25  1982-06-01  1982-12-15
MA2112      70     1.50  1982-02-15  1983-02-01
AD3113      70     1.00  1982-07-01  1983-02-01
AD3113      70     1.00  1982-10-15  1983-02-01
MA2112      70     1.00  1982-06-01  1983-02-01
MA2113      70     2.00  1982-04-01  1983-12-15
AD3113      80     1.75  1982-01-01  1982-04-15
AD3113      80     0.50  1982-03-01  1982-04-15
AD3112      80     0.50  1982-10-15  1982-12-01
AD3112      80     0.35  1982-08-15  1982-12-01
MA2113      80     0.50  1982-10-01  1983-02-01
MA2113      80     1.50  1982-09-01  1983-02-01
MA2113      80     1.00  1982-01-01  1983-02-01
MA2112      80     1.00  1982-10-01  1983-10-01
IF1000      90     0.50  1982-10-01  1983-01-01
```

Figure 12. Display 20 Rows Forward by Pressing ENTER

To scroll through the query result a half display at a time, you can either enter FORWARD or press PF8.

Press PF8.

The rows on the display scroll half the display. The middle row of the previous display is now the first row of the new display.

Now, to move forward to the remaining rows, enter:

```
forward max
```

This results in a display similar to Figure 13 on page 24.

```
PROJNO   ACTNO  ACSTAFF  ACSTDATE    ACENDATE
------  ------  -------  ----------  ----------
IF1000      90     0.50  1982-10-01  1983-01-01
IF1000      90     1.00  1982-01-01  1983-01-01
IF2000     100     0.50  1982-03-01  1982-07-01
IF2000     100     0.75  1982-01-01  1982-07-01
IF1000     100     0.50  1982-10-01  1983-01-01
IF2000     110     0.50  1982-03-01  1982-07-01
IF2000     110     0.50  1982-10-01  1983-01-01
OP1010     130     4.00  1982-01-01  1983-02-01
OP2012     140     0.25  1982-01-01  1983-02-01
OP2011     140     0.75  1982-01-01  1983-02-01
OP2013     140     0.50  1982-01-01  1983-02-01
OP2011     150     0.25  1982-01-01  1983-02-01
OP2012     160     0.75  1982-01-01  1983-02-01
OP2013     170     0.50  1982-01-01  1983-02-01
AD3113     180     1.00  1982-04-15  1982-06-01
AD3113     180     0.50  1982-06-01  1982-07-01
AD3113     180     0.75  1982-03-01  1982-07-01
AD3112     180     0.50  1982-08-15  1983-01-01
MA2113     180     0.50  1982-10-01  1983-01-01
MA2112     180     1.00  1982-07-01  1983-02-01
MA2112     180     1.00  1982-07-15  1983-02-01
* End of Result *** 70 Rows Displayed ***Cost Estimate is 1******************
```

Figure 13. Display after Moving to the End of the Query Result

To move back through the query result, use a BACKWARD command. Also, you can use PF7 to move your view of the result backward one-half display. Moving back through the query result is limited; you can move back only to a limit of one full display from the last FORWARD command. If you want to go farther back, you must return directly to the beginning of the query result.

For example, to view the previous 15 rows, enter:

```
backward 15
```

This command presents the display in Figure 14 on page 25.

```
PROJNO   ACTNO  ACSTAFF  ACSTDATE    ACENDATE
------  ------  -------  ----------  ----------
AD3112      70     0.50  1982-02-01  1982-03-15
AD3113      70     0.50  1982-06-15  1982-07-01
AD3112      70     1.00  1982-03-15  1982-08-15
MA2112      70     1.00  1982-02-01  1982-10-01
AD3112      70     0.75  1982-01-01  1982-10-15
AD3112      70     0.25  1982-08-15  1982-10-15
AD3113      70     0.75  1982-09-01  1982-10-15
AD3113      70     1.25  1982-06-01  1982-12-15
MA2112      70     1.50  1982-02-15  1983-02-01
AD3113      70     1.00  1982-07-01  1983-02-01
AD3113      70     1.00  1982-10-15  1983-02-01
MA2112      70     1.00  1982-06-01  1983-02-01
MA2113      70     2.00  1982-04-01  1983-12-15
AD3113      80     1.75  1982-01-01  1982-04-15
AD3113      80     0.50  1982-03-01  1982-04-15
AD3112      80     0.50  1982-10-15  1982-12-01
AD3112      80     0.35  1982-08-15  1982-12-01
MA2113      80     0.50  1982-10-01  1983-02-01
MA2113      80     1.50  1982-09-01  1983-02-01
MA2113      80     1.00  1982-01-01  1983-02-01
MA2112      80     1.00  1982-10-01  1983-10-01
IF1000      90     0.50  1982-10-01  1983-01-01
```

Figure 14. Results When You Try to Exceed the Limit of a Full Display

The view is moved back to the limit of one full display. Enter the following command to return to the first rows of the query result:

```
backward max
```

Now that you are finished with this query result, end it.

Results That Are Too Wide for One Display

To learn about viewing results that are too wide for a single display, it is necessary to use a sample table of greater width than the PROJ_ACT table. For the next several examples, the EMPLOYEE table is used.

Enter the following query:

```
select * -
from employee
```

The resulting query on an 80-character display is similar to Figure 15 on page 26.

```
EMPNO   FIRSTNME     MIDINIT  LASTNAME        WORKDEPT  PHONENO  HIREDATE
-----   ------------ -------  --------------- --------  -------  ----------
000010  CHRISTINE    I        HAAS            A00       3978     1991-05-27
000110  VINCENZO     G        LUCCHESI        A00       3490     1958-05-16
000120  SEAN                  O'CONNELL       A00       2167     1963-12-05
000020  MICHAEL      L        THOMPSON        B01       3476     1973-10-10
000030  SALLY        A        KWAN            C01       4738     1975-04-05
000130  DOLORES      M        QUINTANA        C01       4578     1971-07-28
000140  HEATHER      A        NICHOLLS        C01       1793     1976-12-15
000060  IRVING       F        STERN           D11       6423     1973-09-14
000150  BRUCE                 ADAMSON         D11       4510     1972-02-12
000160  ELIZABETH    R        PIANKA          D11       3782     1977-10-11
000170  MASATOSHI    J        YOSHIMURA       D11       2890     1978-09-15
000180  MARILYN      S        SCOUTTEN        D11       1682     1973-07-07
000190  JAMES        H        WALKER          D11       2986     1974-07-26
000200  DAVID                 BROWN           D11       4501     1966-03-03
000210  WILLIAM      T        JONES           D11       0942     1979-04-11
000220  JENNIFER     K        LUTZ            D11       0672     1968-08-29
000070  EVA          D        PULASKI         D21       7831     1980-09-30
000230  JAMES        J        JEFFERSON       D21       2094     1966-11-21
000240  SALVATORE    M        MARINO          D21       3780     1979-12-05
000250  DANIEL       S        SMITH           D21       0961     1969-10-30
000260  SYBIL        P        JOHNSON         D21       8953     1975-09-11
000270  MARIA        L        PEREZ           D21       9001     1980-09-30
```

Figure 15. Results of Query That is Too Wide for Display

There is no visual indication that this query result is too wide for the display; however, because the columns extend to the far right side of the display, there is a possibility that additional columns of data may be beyond the last column. To search for any additional columns that may exist, enter the following display command:

```
right 7
```

This moves your view of the query seven columns to the right, resulting in Figure 16 on page 27.

```
JOB      EDLEVEL  SEX  BIRTHDATE        SALARY        BONUS         COMM
-------- -------  ---  ----------  -----------  -----------  -----------
PRES          18  F    1933-08-01     52750.00      1000.00      4220.00
SALESREP      19  M    1929-11-05     46500.00       800.00      3720.00
CLERK         14  M    1942-10-18     29250.00       600.00      2340.00
MANAGER       18  M    1948-02-02     41250.00       800.00      3300.00
MANAGER       20  F    1941-05-11     38250.00       800.00      3060.00
ANALYST       16  F    1925-09-15     23800.00       800.00      1904.00
ANALYST       18  F    1946-01-19     28420.00       800.00      2274.00
MANAGER       16  M    1945-07-07     32250.00       600.00      2580.00
DESIGNER      16  M    1947-05-17     25280.00       500.00      2022.00
DESIGNER      17  F    1955-04-12     22250.00       400.00      1780.00
DESIGNER      16  M    1951-01-05     24680.00       500.00      1974.00
DESIGNER      17  F    1949-02-21     21340.00       500.00      1707.00
DESIGNER      16  M    1952-06-25     20450.00       400.00      1636.00
DESIGNER      16  M    1941-05-29     27740.00       600.00      2217.00
DESIGNER      17  M    1953-02-23     18270.00       400.00      1462.00
DESIGNER      18  F    1948-03-19     29840.00       600.00      2387.00
MANAGER       16  F    1953-05-26     36170.00       700.00      2893.00
CLERK         14  M    1935-05-30     22180.00       400.00      1774.00
CLERK         17  M    1954-03-31     28760.00       600.00      2301.00
CLERK         15  M    1939-11-12     19180.00       400.00      1534.00
CLERK         16  F    1936-10-05      1725.00       300.00      1380.00
CLERK         15  F    1953-05-26     27380.00       500.00      2190.00
```

Figure 16. Display of Query Moved Seven Columns to the Right

Now you can see that there are an additional seven columns, but it is still not clear that you have reached the last column of the table. Enter the following command to move your view two more columns to the right:

```
right 2
```

You should now see the display in Figure 17.

```
SEX  BIRTHDATE        SALARY        BONUS         COMM
---  ----------  -----------  -----------  -----------
F    1933-08-01     52750.00      1000.00      4220.00
M    1929-11-05     46500.00       800.00      3720.00
M    1942-10-18     29250.00       600.00      2340.00
M    1948-02-02     41250.00       800.00      3300.00
F    1941-05-11     38250.00       800.00      3060.00
F    1925-09-15     23800.00       800.00      1904.00
F    1946-01-19     28420.00       800.00      2274.00
M    1945-07-07     32250.00       600.00      2580.00
M    1947-05-17     25280.00       500.00      2022.00
F    1955-04-12     22250.00       400.00      1780.00
M    1951-01-05     24680.00       500.00      1974.00
F    1949-02-21     21340.00       500.00      1707.00
M    1952-06-25     20450.00       400.00      1636.00
M    1941-05-29     27740.00       600.00      2217.00
M    1953-02-23     18270.00       400.00      1462.00
F    1948-03-19     29840.00       600.00      2387.00
F    1953-05-26     36170.00       700.00      2893.00
M    1935-05-30     22180.00       400.00      1774.00
M    1954-03-31     28760.00       600.00      2301.00
M    1939-11-12     19180.00       400.00      1534.00
F    1936-10-05      1725.00       300.00      1380.00
F    1953-05-26     27380.00       500.00      2190.00
```

Figure 17. Display of Query Moved Two More Columns to the Right

Now apparently COMM is the final column in the table, because no other columns appear to the right of it in the query result. If you only want to move your view of

the table one column to the right, you can omit the number and just type `right`. You can also move your view one column to the right by pressing PF11.

To move your view to the left, type:

```
left
```

This command works in the same manner as RIGHT. Again, you can move your view one column to the left by pressing PF10.

Now move your view all the way back to the first column by using the following command:

```
column 1
```

The COLUMN command aligns the column you specify with the left edge of the display. The number refers to the column's position in the query result. If you do not specify a number with the COLUMN command, column 1 is placed at the left edge of the display.

A column of characters that is wider than the display width requires another display command, TAB, to let you see the entire length attribute of a column. The TAB command is described in "Chapter 10. ISQL Commands" on page 105.

Ending a Query Display

Always end a query result when you are finished with it (so that ISQL performance for other users is not affected). The END command removes your query result from the display. Type the following and press ENTER:

```
end
```

DB2 Server for VM

ISQL returns to wait mode.

Obtaining a Printed Report

So far you have learned how to view query results at your display terminal. Now produce a printed copy of a query result on the system printer. The information requested to be printed is a copy of the DEPARTMENT table. Begin by entering the following query:

```
select * -
from department -
order by deptno
```

As you can see, this query selects all the columns and all the rows from the DEPARTMENT table. Request a printed copy of this query result by entering the following display command:

```
print
```

Another way to request a printed copy of the query result is to press PF4.Figure 18 on page 29 illustrates the printed information for the preceding query. Consult the appropriate person at your location for printing procedures.

Each printed page is numbered, dated, and titled.The title consists of the first 100 characters of the SELECT statement that you typed. For information on the creation of report titles, refer to “Chapter 5. Formatting Query Results” on page 43.

```
.      07/13/89     SELECT * FROM DEPARTMENT ORDER BY DEPTNO     PAGE   1 .
.                                                                         .
.                                                                         .
.      DEPTNO  DEPTNAME                       MGRNO   ADMRDEPT           .
.      ______  _____________________________  ______  ________           .
.      A00     SPIFFY COMPUTER SERVICE DIV.   000010  A00                .
.      B01     PLANNING                       000020  A00                .
.      C01     INFORMATION CENTER             000030  A00                .
.      D01     DEVELOPMENT CENTER             ?       A00                .
.      D11     MANUFACTURING SYSTEMS          000060  D01                .
.      D21     ADMINISTRATION SYSTEMS         000070  D01                .
.      E01     SUPPORT SERVICES               000050  A00                .
.      E11     OPERATIONS                     000090  E01                .
.      E21     SOFTWARE SUPPORT               000100  E01                .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
.                                                                         .
```

Figure 18. Example of a Printed Report

Although this query fits on one display, a long query that requires several displays is also printed entirely, regardless of the rows being displayed when you typed the PRINT command. Also, when the PRINT command completes printing the report, the end rows of the query result are displayed regardless of the rows being displayed when you entered the PRINT command.

The printed report does, however, start with the column that is currently at the left edge of the display, and continues for as many print positions as can fit on the specified page size. You can print reports that are too wide for the page size by entering a PRINT command at column 1, moving the display to start with the column that would not fit on the page, entering another PRINT command, and so on. Page-size specification is discussed in “Page Size of Printed Reports” on page 62.

You may have special printing requirements for your query results (such as the type of paper or its size) that you can specify using the CLASS keyword of the PRINT command. See the PRINT command in “Chapter 10. ISQL Commands” on page 105 for a description of this procedure. For DB2 Server for VM, the CLASS value can also be set by a CP SPOOL command.

Obtaining Multiple Copies of a Printed Report

For three copies, type:

```
print copies 3
```

DB2 Server for VM

You can also use the CP SPOOL command to specify the number of copies. The value for copies that is specified on the CP SPOOL command remains in effect until another value is specified by another CP SPOOL command.

Note: If you specify the number of copies on the PRINT command after you specified it using a CP SPOOL command, the PRINT command quantity is used for that print operation. All following PRINT commands use the quantity specified by the CP SPOOL command, unless you specify it using the **COPIES** keyword.

Using More Than One Keyword with the Print Command

You can specify both your print requirements and number of copies with one PRINT command. For example, you can type:

```
print copies 3 class a
```

EXERCISE 1 (Answers are in Appendix A. Answers to the Exercises, page 165.)

Enter the following command:

```
select * -
from proj_act -
where projno <> 'AD3111' -
order by actno,acendate
```

Perform the following:

1. Move the display so that it begins with the 40th row of the query result.
2. Display the end of the query result.
3. Move the display so that it begins with the second column of the query result.
4. Move the display so that it begins with the first column of the query result.
5. Move the display so that it begins with the first row of the query result.
6. Request a printed copy of this result.
7. End the query result.

Chapter 3. Managing Table Data

This chapter shows how to maintain the data in your tables. When the tables are interrelated, you must maintain the accuracy of all the tables whenever you make changes. The following sections describe how to control your changes to the tables.

Controlling Changes to Table Data

Sometimes you want to make updates only with other updates (or deletions). For example, a new employee is hired and is assigned to a project. This involves updates to the EMPLOYEE and EMP_ACT tables. In addition, the employee is made responsible for the project, requiring an update of the PROJECT table (RESPEMP column). The three tables are updated simultaneously, because if only some of the updates are made, and an interval of time passes before the remaining updates are made, anyone accessing the tables during the interval for information about the new employee receives inaccurate information.

For simultaneous updating of tables, you can group SQL statements into a single unit. The statements are run as a group, and only upon your specification. You can also cancel the statements as a group. Such a grouping of one or more statements is called a *logical unit of work* (*LUW*). Two system settings are used in LUW processing: AUTOCOMMIT ON and AUTOCOMMIT OFF.

Using the AUTOCOMMIT ON Setting

With the AUTOCOMMIT ON setting, each statement is treated as an LUW. The work performed by each statement is committed as part of statement processing, resulting in a permanent change. There are, however, a few exceptions. When an INSERT, UPDATE, or DELETE statement that affects more than one row is typed, the LUW is not completed until the next statement is typed. If the next statement is CANCEL or ROLLBACK, the work performed by the INSERT, UPDATE, or DELETE operation before the CANCEL or ROLLBACK is undone instead of being committed. If the next statement is not CANCEL or ROLLBACK, the work performed by the INSERT, UPDATE, or DELETE is committed automatically before the new statement is processed. The CANCEL command is described under “Chapter 10. ISQL Commands” on page 105.

The default setting in ISQL is AUTOCOMMIT ON.

Using the AUTOCOMMIT OFF Setting

Use the AUTOCOMMIT OFF setting to control the committing of information. After you type SET AUTOCOMMIT OFF, any subsequent statements you type are grouped into a single LUW. The statements are processed but not committed until you type the SQL statement COMMIT. As you type the statements, the table data shown on the display looks as if the changes are being committed: they are not. In addition, no other system user can view or modify the changes that you are making until you type the COMMIT statement. Typing COMMIT completes your LUW and commits all processing performed since the beginning of the LUW.

After you set AUTOCOMMIT OFF, and if for any reason, such as an update error, you decide not to commit the LUW processing, you can cancel all processing

performed since the beginning of the LUW by typing ROLLBACK. After you set AUTOCOMMIT OFF, you must explicitly end the LUW by typing COMMIT or ROLLBACK.

Attention Use AUTOCOMMIT OFF cautiously because it prevents other users from accessing the rows of tables used in your processing. In particular, avoid using it for a query.

To return to the condition in which ISQL automatically commits your work, type `SET AUTOCOMMIT ON`. The system displays a message requesting you to commit or rollback any work done during the logical unit of work. You must respond to this message before any further statements can be typed.

Interpreting Messages While Making Changes

Messages are displayed after data has been inserted, deleted, or updated to indicate the number of rows affected by each operation. These messages help you verify the changes.

Interpreting Errors While Making Changes

A single statement can change many rows in a table. If a statement error occurs after only some rows have been changed, all changes are rolled back or withdrawn, and the entire operation fails. If the failed statement is part of an LUW, previously completed statements in the LUW are *not* affected. You still have the option of entering COMMIT or ROLLBACK for the other statements in the LUW.

Chapter 4. Using ISQL Commands to Save Time When Executing Statements

To avoid retyping similar SQL statements or retyping an entire statement just to correct one error, you can use the ISQL commands in this chapter. It explains how to retrieve, modify, and delete SQL statements, how to withhold a statement from processing until you have checked it for errors, and how to reuse the same SQL statement for different tables, columns, and rows.

Reusing the Current SQL Statement

Before the system processes an SQL statement, it inserts it into a special storage area called the SQL *command buffer*. Once in the buffer, the statement becomes the current statement until it is pushed down in the queue by the next statement. ISQL commands are also inserted into the command buffer when you type them in, but not when you invoke them by using a PF key.

Using the command buffer facility, you can rerun a typed statement by using the ISQL START command (PF12).

Process the current SQL statement by performing the following:

1. Type the statement:

   ```
   select * -
   from department
   ```

2. When the result is displayed, type the END command.
3. Reenter the SQL statement by typing:

   ```
   start
   ```

The START command becomes even more useful when you have a typing error in an SQL statement. The error can be corrected (described in the next topic) and the START command used to reenter the corrected statement.

Retrieving and Correcting SQL Lines

You can use the RETRIEVE function to correct a statement already in the command line buffer. RETRIEVE moves the previously-typed line into the input area. The following example shows how to retrieve and correct a line in the output area. Type the following and press ENTER:

```
select actnum,actkwd,actdesc -
```

The line is moved to the output area.

Now type:

```
from activity
```

Press ENTER.

An error message is issued indicating that the column ACTNUM was not found. (ACTNO is the correct name.)

To retrieve the last input line entered, press the PF12 key to invoke the RETRIEVE function. The following line is now displayed in the input area with the cursor positioned at the end of the line:

```
from activity
```

Press the PF12 key again to retrieve the previous input line. The input area now contains:

```
select actnum,actkwd,actdesc -
```

with the cursor positioned at the end of the line. You can now correct the error as you would any other error in the input area by backspacing and typing the correct characters.

After you have corrected the line in the input area, press ENTER.

DB2 Server for VSE

The corrected line is now displayed in the output area, and `CONTINUE COMMAND` has appeared in the status area to confirm that you are to continue entering the statement.

DB2 Server for VM

The corrected line is now displayed in the output area, `VM READ` has appeared in the status area, and message `ARI7068I` has appeared in the output area to confirm that you are to continue typing the statement.

Continue the statement by using either of the following two methods:

- Type the FROM clause in the input area as follows:

  ```
  from activity
  ```

  Press ENTER.
- Press PF12 twice. The FROM clause is displayed in the input area with the cursor at the end of the line:

  ```
  from activity_
  ```

  Press ENTER.

The result is the same using either method: the SELECT statement is reissued.

Note: The command buffer holds a variable number of statements, depending on their length. If your statements are short, the buffer can hold more of them. As you press PF12 repeatedly, the system retrieves statements from the buffer starting with the most recent, and continues until it retrieves the oldest statement. Then it starts over and returns the most recent statement.

DB2 Server for VM

The RETRIEVE facility functions differently in wait mode compared to display mode. In display mode, the system can retrieve any information typed during the ISQL session that is still contained in the command buffer. In wait mode (`VM READ` shows in the status area of the display), the system can retrieve information typed in wait mode or from a time prior to starting ISQL.

Incorrect information typed from wait mode can be retrieved immediately, and corrected as illustrated in the previous example. Incorrect information typed from display mode forces display mode to end, and returns you to wait mode. Pressing PF12 does not retrieve the incorrect information in this case, because you typed it from display mode. The following examples illustrate the RETRIEVE-function differences between wait and display mode.

Ensure that you are in wait mode. If you do not see `VM READ` in the lower-right corner of your display, press PF3. Now, type the following (with *activity* spelled incorrectly):

```
select * from abtivity
```

An error message is issued indicating that ABTIVITY cannot be found. Press PF12. The incorrect `select * from abtivity` statement appears in your input area, where you can correct it.

Erroneous lines typed from display mode cannot be immediately retrieved with RETRIEVE key PF12. To illustrate, type the following correct statement from wait mode.

```
select * from activity
```

The ACTIVITY table is displayed, and the message disappears from the status area to indicate that you are in display mode.

Do not press PF3 to end the display. Instead, type the following incorrect statement from display mode:

```
select * from supply
```

The SUPPLY table is not found; `VM READ` reappears in the lower-right corner of your display to indicate that you are back in wait mode.

Now, press PF12. `select * from activity` appears in the input area. It is not the last statement you typed, but is the last one you typed from wait mode. The line in error that was entered from display mode cannot be retrieved from wait mode. It has been saved, however, and can be retrieved when you return to display mode. To illustrate this, enter the following correct query, which returns you to display mode:

```
select actno from activity
```

Now, from display mode, press PF12. The last line typed, `select actno from activity`, appears in the input area.

Press PF12 again. This time the line in error, `select * from supply`, does appear in the input area.

DB2 Server for VM

All lines typed during an ISQL session can be retrieved from within display mode (up to the maximum number of lines the buffer can hold), including lines entered from display mode and wait mode. Only those lines entered from wait mode can be retrieved from wait mode. Since ISQL commands other than display commands, SQL statements other than SELECT, and incorrect SELECT statements entered from display mode return you to wait mode, you cannot immediately retrieve this type of statement or command. Pressing PF3 after you have finished each query ensures that you enter each statement or command from wait mode and can immediately retrieve and change the information.

Altering and Reusing SQL Lines

The RETRIEVE facility is especially useful when you enter similar commands. To save time, you can retrieve an earlier command, alter it, and then enter it.

For example, enter the following statement:

```
select * -
from employee
```

Your next statement to be typed is `select * from department`. To save time, reuse the statement you just typed as follows.

Press PF12 twice to place `select * -` in the input area. Then, press ENTER.

Press PF12 twice. `from employee` is now contained in the input area. Backspace and type DEPARTMENT over EMPLOYEE. Then, press ENTER.

In this way, `select * from department` is typed in just a few keystrokes.

Remember, a *variable* number of your statements are retained, and you may or may not be able to retrieve a particular statement.

Changing the Current SQL Statement

These statements can be corrected, altered and reused, and portions of them can be deleted.

Correcting Typing Errors in the Statement

Having the facility to change the current SQL statement lets you correct one containing typing errors. For example, enter the following statement (with *activity* misspelled):

```
select actno -
from adtivity -
where actno > 100
```

This returns an error message stating that there is no table owned by you named ADTIVITY. Correct the error by entering the following ISQL command:

```
change /adtivity/activity/
```

The slashes in CHANGE commands separate the data to be changed (between the first two slashes) from the new data to replace it (between the last two slashes). Always include a blank before the first slash and always enter the final slash. ISQL changes the statement at the *first* occurrence of the data you place between the first two slashes, so ensure that the data you want changed is the first occurrence of that data in the statement to be changed.

Note: You do not have to use a slash to separate the data; any character except a blank can be used in its place. To change data that contains a slash, choose a character that does not occur in the data to be changed or in the new data.

Now enter the START command to process the changed SQL statement. If you have made no other mistakes, the query result for this SELECT statement is displayed on your display.

Altering and Reusing the Statement

You can use the CHANGE command to alter and then reuse the statement in the command buffer.

In "Altering and Reusing SQL Lines" on page 38, you used the RETRIEVE function to change the table name from EMPLOYEE to DEPARTMENT. Alternatively, you can query the EMPLOYEE table, end the query, and then type the following statement:

```
change /employee/department/
```

This command changes the statement in the buffer. To perform the new statement, which queries the DEPARTMENT table, type:

```
start
```

Notice that the RETRIEVE command uses *lines* of information from the command line buffer; the CHANGE command uses the *statement* from the command buffer.

Deleting Portions of the Statement

The CHANGE command can also be used to delete a portion of a current statement. For example, suppose you had selected the ACTNO, ACTKWD, and ACTDESC columns from the ACTIVITY table. After you end the query, you can delete the selection of the ACTKWD column table by typing:

```
change /,actkwd//
```

By providing no replacement string (nothing between the second and third slashes), the characters matching the string between the first two slashes are effectively deleted from the statement. You can see the result of your changed query by using the START command.

Ignoring an SQL Line

Another way to correct a typing mistake in a multiple-line statement is to use the IGNORE command.

To illustrate, type the following (*deptname* is misspelled):

```
select depname,mgrno -
```

Press ENTER. Now, you realize the valid column name is DEPTNAME. Type:

```
ignore
```

DB2 Server for VSE

You receive the message:

```
ARI7061I Previous input ignored.
```

in the output area, and the status area contains *ENTER A NEW COMMAND*.

DB2 Server for VM

You receive the message:

```
ARI7061I Previous input ignored.
```

in the output area, and the status area contains `VM READ`.

You can now retype the statement, or use PF12 to retrieve the line for correction.

Preventing the Immediate Processing of an SQL Statement

You can prevent an SQL statement from being processed immediately after being typed. This lets you check the statement for typing errors before it is processed using a START command. It also allows an SQL statement containing *placeholders* to be placed in the SQL command buffer, and values to be substituted for the placeholders when the statement is started using the START command. (The START command and placeholders are discussed in the next section.)

To illustrate, the following example shows how to prevent the statement *SELECT * FROM PROJECT* from being processed immediately. Type the following ISQL command:

```
hold select * from project
```

This statement remains in the buffer as the current statement until you enter another SQL statement (or another HOLD command).

The HOLD command can also be invoked by pressing PF9. If you press PF9 instead of ENTER after typing an SQL statement, it is placed in the command buffer and is not processed.

Your held statement can then be processed using the START command.

Note: HOLD cannot be used with ISQL commands.

Using Placeholders in SQL Statements

You can form SQL statements that contain placeholders. They reserve areas in the statements, which are filled in when the statement is performed. One reason for doing this, for example, is to avoid typing an UPDATE statement for each table update. Suppose you want to update the EMPLOYEE table to reflect a $100 bonus increase for particular employees.

1. To prevent your changes from being automatically committed, type:

   ```
   set autocommit off
   ```

This starts a logical unit of work. Now type:

```
hold update employee -
set bonus = &1 -
where empno = '&2'
```

The use of HOLD at the beginning of this example places the UPDATE statement in the command buffer without processing it.

In the example, &1 and &2 are the placeholders. The number following the ampersand refers to the sequence in which the placeholders are replaced. The database manager performs the replacement as follows: the first item of information replaces &1, the second replaces &2, and so on.

2. Start the UPDATE statement and supply the replacement information:

   ```
   start (bonus+100.00 '000010')
   ```

   This command adds a $100 bonus to employee 000010, and produces a message indicating that one row (ROWCOUNT=1) was updated. The actual UPDATE statement processed is:

   ```
   update employee
   set bonus = bonus + 100.00
   where empno = '000010'
   ```

   The two items of information which replace the placeholders are called *parameters*.

3. Because START is an ISQL command, it does not replace the UPDATE statement in the command buffer. This lets you continue to use the UPDATE statement to update another row as follows:

   ```
   start (bonus+100.00 '000050')
   ```

4. This process can continue for as many updates as needed. Remember though, that the START command uses the statement currently contained in the command buffer. If you typed another SQL statement, it would replace the UPDATE statement in the buffer, and you would have to recall the UPDATE statement to continue. (Statement recalling is discussed in "Chapter 6. Storing SQL Statements" on page 67.)

5. Type the following command to return to normal command processing:

   ```
   set autocommit on
   ```

   Respond to the resulting message with the following to prevent the changes from being committed:

   ```
   rollback
   ```

The following are rules for the use of placeholders and parameters:

- A placeholder for a character data item must be enclosed in single quotation marks.
- A parameter for a character data item must also be enclosed in single quotation marks.
- When using parameters in a START command, enclose them in parentheses and separate each parameter with a blank.
- A parameter may be one word, several words, or an expression.
  - If one parameter (replacing just one placeholder) consists of a list of words separated by commas, such as a list of column names in a *select_list*, the list of words must use commas (and not blanks) as separators.

  - If a parameter contains a list of words separated by blanks, that list of words must be enclosed in single quotation marks to distinguish between the blanks within the one parameter and the blanks that *separate* parameters.
- In a stored SQL command, you can only use the ampersand (&) to create placeholders.

EXERCISE 2 (Answers are in Appendix A. Answers to the Exercises, page 165.)

Perform the following:

1. Enter and hold an SQL statement that retrieves the entire DEPARTMENT table.
2. Change the command created in step 1 to select two columns, and use placeholders to define them.
3. Start the command, replacing the two placeholders with values that retrieve the DEPTNO and DEPTNAME columns. Check the results, and then end the display.
4. Change the current SQL statement to add a WHERE clause. Use a placeholder for the search condition.
5. Start the command, replacing the placeholders with values that select the department name and the manager number for departments that report to E01.

Chapter 5. Formatting Query Results

This chapter builds on the information in previous chapters. When the information appears on your display, you can change the way it looks. You can format the columns so that they appear with different headings, widths, or separators. You can format the whole query result so that it appears as a report, divided into groups, totaled, and titled. You can also use ISQL commands to set the query format characteristics for all queries in a session.

Formatting Columns

In "Chapter 2. Querying Tables" on page 21, you learned how to produce, display, and print query information. In this chapter, you will learn the techniques to arrange the information in a report format. You can:

- Change the number of blanks that separate columns or even change the blanks to some other characters.
- Cease display of a column (and later include it, if desired).
- Change the name of a displayed column heading.
- Specify the number of decimal places to display for numeric columns.
- Control the display of leading zeros on numeric columns.
- Change the display width of a column.

Creating a Report from Query Results

To illustrate the formatting changes that you can make to columns, you first need a query result. Type the following statement:

```
select projno,actno,acstaff,acstaff + .25,acstdate -
from proj_act -
where projno = 'AD3100' -
or projno = 'AD3111' -
or projno = 'AD3112' -
order by projno,actno
```

The query result for this statement (shown in Figure 19 on page 44) is used over the next several topics. Do not end it until asked to do so.

```
PROJNO   ACTNO  ACSTAFF    EXPRESSION 1  ACSTDATE
------  ------  -------  --------------  ----------
AD3100      10     0.50            0.75  1982-01-01
AD3111      60     0.80            1.05  1982-01-01
AD3111      60     0.50            0.75  1982-03-15
AD3111      70     1.50            1.75  1982-02-15
AD3111      70     0.50            0.75  1982-03-15
AD3111      80     1.25            1.50  1982-04-15
AD3111      80     1.00            1.25  1982-09-15
AD3111     180     1.00            1.25  1982-10-15
AD3112      60     0.75            1.00  1982-01-01
AD3112      60     0.50            0.75  1982-02-01
AD3112      60     0.75            1.00  1982-12-01
AD3112      60     1.00            1.25  1983-01-01
AD3112      70     0.75            1.00  1982-01-01
AD3112      70     0.50            0.75  1982-02-01
AD3112      70     1.00            1.25  1982-03-15
AD3112      70     0.25            0.50  1982-08-15
AD3112      80     0.35            0.60  1982-08-15
AD3112      80     0.50            0.75  1982-10-15
AD3112     180     0.50            0.75  1982-08-15
* End of Result *** 19 Rows Displayed ***Cost Estimate is 1********************
```

Figure 19. A Query Result to Be Used to Illustrate Formatting Techniques

Field procedures can affect the order of the rows when you use the ORDER BY clause. For more information about field procedures, see the *DB2 Server for VSE & VM SQL Reference* manual.

Modifying the Separation between Columns

Using the query result described above, change the number of blanks used to separate the columns by typing the following display command:

```
format separator 4 blanks
```

FORMAT is the name of the command; SEPARATOR describes the kind of formatting to be done. After this command has been processed, four blanks are displayed between columns. Now separate the columns with a vertical bar and a couple of blanks by typing the following command:

```
format separator ' | '
```

This defines a separator that consists of a blank, followed by a vertical bar, followed by a blank. The quotation marks were used to include the blanks as part of the separator. Quotation marks are needed whenever the separator you are defining contains a blank. For example, the 3-character separator:

```
|||
```

does not require quotation marks, whereas this separator does:

```
'| |'
```

The result of typing the above FORMAT command is shown in Figure 20 on page 45.

```
PROJNO │  ACTNO │ ACSTAFF │   EXPRESSION 1 │ ACSTDATE   │
------ │ ------ │ ------- │ -------------- │ ---------- │
AD3100 │     10 │    0.50 │           0.75 │ 1982-01-01 │
AD3111 │     60 │    0.80 │           1.05 │ 1982-01-01 │
AD3111 │     60 │    0.50 │           0.75 │ 1982-03-15 │
AD3111 │     70 │    1.50 │           1.75 │ 1982-02-15 │
AD3111 │     70 │    0.50 │           0.75 │ 1982-03-15 │
AD3111 │     80 │    1.25 │           1.50 │ 1982-04-15 │
AD3111 │     80 │    1.00 │           1.25 │ 1982-09-15 │
AD3111 │    180 │    1.00 │           1.25 │ 1982-10-15 │
AD3112 │     60 │    0.75 │           1.00 │ 1982-01-01 │
AD3112 │     60 │    0.50 │           0.75 │ 1982-02-01 │
AD3112 │     60 │    0.75 │           1.00 │ 1982-12-01 │
AD3112 │     60 │    1.00 │           1.25 │ 1983-01-01 │
AD3112 │     70 │    0.75 │           1.00 │ 1982-01-01 │
AD3112 │     70 │    0.50 │           0.75 │ 1982-02-01 │
AD3112 │     70 │    1.00 │           1.25 │ 1982-03-15 │
AD3112 │     70 │    0.25 │           0.50 │ 1982-08-15 │
AD3112 │     80 │    0.35 │           0.60 │ 1982-08-15 │
AD3112 │     80 │    0.50 │           0.75 │ 1982-10-15 │
AD3112 │    180 │    0.50 │           0.75 │ 1982-08-15 │
* End of Result ************ 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 20. A Query Result with a Formatted Separator between Columns

Excluding Columns from the Display

There can be occasions when you obtain a query result and decide that it contains columns that you do not want in your report. You can end the result and retype the statement with the correct columns listed in the SELECT clause, but there is an alternative to retyping.

Suppose you want to exclude the ACSTAFF and ACSTDATE columns in the current display. Type:

```
format exclude (acstaff acstdate)
```

Issuing a FORMAT command with EXCLUDE instructs ISQL to exclude the columns specified from the current display. When specifying more than one column name, enclose them in parentheses and separate the names with a blank. The above FORMAT command displays Figure 21 on page 46.

```
PROJNO │  ACTNO │   EXPRESSION 1 │
------ │ ------ │ -------------- │
AD3100 │     10 │           0.75 │
AD3111 │     60 │           1.05 │
AD3111 │     60 │           0.75 │
AD3111 │     70 │           1.75 │
AD3111 │     70 │           0.75 │
AD3111 │     80 │           1.50 │
AD3111 │     80 │           1.25 │
AD3111 │    180 │           1.25 │
AD3112 │     60 │           1.00 │
AD3112 │     60 │           0.75 │
AD3112 │     60 │           1.00 │
AD3112 │     60 │           1.25 │
AD3112 │     70 │           1.00 │
AD3112 │     70 │           0.75 │
AD3112 │     70 │           1.25 │
AD3112 │     70 │           0.50 │
AD3112 │     80 │           0.60 │
AD3112 │     80 │           0.75 │
AD3112 │    180 │           0.75 │
* End of Result ************ 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 21. A Query Result Formatted to Exclude Two Columns

You can use a number instead of a column name to identify the excluded column. The number refers to the column's position in the SELECT clause. For example, the ACSTAFF and ACSTDATE columns can be excluded by typing:

```
format exclude (3 5)
```

Sometimes it is easier to define columns you want *included*. For example, if you want to include only the third column of a query result that contained many columns, you can type:

```
format exclude all but (3)
```

The parentheses can be omitted when only one column is specified.

Including Columns in the Display

The effects of excluding a column from the display can be reversed. You can use the INCLUDE option to include the ACSTAFF and ACSTDATE columns in the current display. Do this by typing:

```
format include (acstaff acstdate)
```

Again, numbers can be used instead of column names. For example, to include only the first and third columns of a query result, you can type:

```
format include only (1 3)
```

The above FORMAT command displays Figure 22 on page 47.

```
PROJNO | ACSTAFF |
------ | ------- |
AD3100 |    0.50 |
AD3111 |    0.80 |
AD3111 |    0.50 |
AD3111 |    1.50 |
AD3111 |    0.50 |
AD3111 |    1.25 |
AD3111 |    1.00 |
AD3111 |    1.00 |
AD3112 |    0.75 |
AD3112 |    0.50 |
AD3112 |    0.75 |
AD3112 |    1.00 |
AD3112 |    0.75 |
AD3112 |    0.50 |
AD3112 |    1.00 |
AD3112 |    0.25 |
AD3112 |    0.35 |
AD3112 |    0.50 |
AD3112 |    0.50 |
* End of Result *** 19 Rows Displayed ***Cost Estimate is 1********************
```

Figure 22. A Query Result Formatted for Include-Only Columns

To display all the columns again, type:

```
format include
```

Changing a Displayed Column Heading

The query result you are using has EXPRESSION 1 as a column heading. To make the heading more meaningful, use the NAME keyword of the FORMAT command. Type the following command:

```
format column 'expression 1' name 'staff + .25'
```

The above FORMAT command displays Figure 23.

```
PROJNO |  ACTNO | ACSTAFF |    STAFF + .25 | ACSTDATE   |
------ | ------ | ------- | -------------- | ---------- |
AD3100 |     10 |    0.50 |           0.75 | 1982-01-01 |
AD3111 |     60 |    0.80 |           1.05 | 1982-01-01 |
AD3111 |     60 |    0.50 |           0.75 | 1982-03-15 |
AD3111 |     70 |    1.50 |           1.75 | 1982-02-15 |
AD3111 |     70 |    0.50 |           0.75 | 1982-03-15 |
AD3111 |     80 |    1.25 |           1.50 | 1982-04-15 |
AD3111 |     80 |    1.00 |           1.25 | 1982-09-15 |
AD3111 |    180 |    1.00 |           1.25 | 1982-10-15 |
AD3112 |     60 |    0.75 |           1.00 | 1982-01-01 |
AD3112 |     60 |    0.50 |           0.75 | 1982-02-01 |
AD3112 |     60 |    0.75 |           1.00 | 1982-12-01 |
AD3112 |     60 |    1.00 |           1.25 | 1983-01-01 |
AD3112 |     70 |    0.75 |           1.00 | 1982-01-01 |
AD3112 |     70 |    0.50 |           0.75 | 1982-02-01 |
AD3112 |     70 |    1.00 |           1.25 | 1982-03-15 |
AD3112 |     70 |    0.25 |           0.50 | 1982-08-15 |
AD3112 |     80 |    0.35 |           0.60 | 1982-08-15 |
AD3112 |     80 |    0.50 |           0.75 | 1982-10-15 |
AD3112 |    180 |    0.50 |           0.75 | 1982-08-15 |
* End of Result ************* 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 23. A Query Result Formatted to Change a Displayed Column Heading

Numbers can also be used to identify the column heading to change; for example:

```
format column 4 name 'staff + .25'
```

Remember, the 4 refers to the column's position in the SELECT clause, not the position of the column displayed.

Changing the Number of Decimal Places Displayed

The query result on your display shows two decimal places for both the ACSTAFF and ACSTAFF + .25 columns. You control the number of decimal places displayed using the DPLACES option of the FORMAT command. For example, to display only one decimal place for the STAFF + .25 column, type:

```
format column 'staff + .25' dplaces 1
```

This displays a result similar to Figure 24.

```
PROJNO │  ACTNO │ ACSTAFF │    STAFF + .25 │ ACSTDATE   │
------ │ ------ │ ------- │ -------------- │ ---------- │
AD3100 │     10 │    0.50 │            0.7 │ 1982-01-01 │
AD3111 │     60 │    0.80 │            1.0 │ 1982-01-01 │
AD3111 │     60 │    0.50 │            0.7 │ 1982-03-15 │
AD3111 │     70 │    1.50 │            1.7 │ 1982-02-15 │
AD3111 │     70 │    0.50 │            0.7 │ 1982-03-15 │
AD3111 │     80 │    1.25 │            1.5 │ 1982-04-15 │
AD3111 │     80 │    1.00 │            1.2 │ 1982-09-15 │
AD3111 │    180 │    1.00 │            1.2 │ 1982-10-15 │
AD3112 │     60 │    0.75 │            1.0 │ 1982-01-01 │
AD3112 │     60 │    0.50 │            0.7 │ 1982-02-01 │
AD3112 │     60 │    0.75 │            1.0 │ 1982-12-01 │
AD3112 │     60 │    1.00 │            1.2 │ 1983-01-01 │
AD3112 │     70 │    0.75 │            1.0 │ 1982-01-01 │
AD3112 │     70 │    0.50 │            0.7 │ 1982-02-01 │
AD3112 │     70 │    1.00 │            1.2 │ 1982-03-15 │
AD3112 │     70 │    0.25 │            0.5 │ 1982-08-15 │
AD3112 │     80 │    0.35 │            0.6 │ 1982-08-15 │
AD3112 │     80 │    0.50 │            0.7 │ 1982-10-15 │
AD3112 │    180 │    0.50 │            0.7 │ 1982-08-15 │
* End of Result ************* 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 24. A Query Result Formatted to Display One Decimal Place

Controlling the Display of Leading Zeros

You can show leading zeros for a numeric column. Do this for the ACTNO column by typing:

```
format column actno zeros on
```

The above FORMAT command displays Figure 25 on page 49.

```
PROJNO │  ACTNO │ ACSTAFF │    STAFF + .25 │ ACSTDATE   │
------ │ ------ │ ------- │ -------------- │ ---------- │
AD3100 │  00010 │    0.50 │            0.7 │ 1982-01-01 │
AD3111 │  00060 │    0.80 │            1.0 │ 1982-01-01 │
AD3111 │  00060 │    0.50 │            0.7 │ 1982-03-15 │
AD3111 │  00070 │    1.50 │            1.7 │ 1982-02-15 │
AD3111 │  00070 │    0.50 │            0.7 │ 1982-03-15 │
AD3111 │  00080 │    1.25 │            1.5 │ 1982-04-15 │
AD3111 │  00080 │    1.00 │            1.2 │ 1982-09-15 │
AD3111 │  00180 │    1.00 │            1.2 │ 1982-10-15 │
AD3112 │  00060 │    0.75 │            1.0 │ 1982-01-01 │
AD3112 │  00060 │    0.50 │            0.7 │ 1982-02-01 │
AD3112 │  00060 │    0.75 │            1.0 │ 1982-12-01 │
AD3112 │  00060 │    1.00 │            1.2 │ 1983-01-01 │
AD3112 │  00070 │    0.75 │            1.0 │ 1982-01-01 │
AD3112 │  00070 │    0.50 │            0.7 │ 1982-02-01 │
AD3112 │  00070 │    1.00 │            1.2 │ 1982-03-15 │
AD3112 │  00070 │    0.25 │            0.5 │ 1982-08-15 │
AD3112 │  00080 │    0.35 │            0.6 │ 1982-08-15 │
AD3112 │  00080 │    0.50 │            0.7 │ 1982-10-15 │
AD3112 │  00180 │    0.50 │            0.7 │ 1982-08-15 │
* End of Result ************ 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 25. A Query Result Formatted to Display Leading Zeros

Stop the display of leading zeros in the ACTNO column by typing:

```
format column actno zeros off
```

Changing the Displayed Length Attribute of a Column

You may want to modify the displayed length attribute of a column to fit your report on the paper being used for printing. For example, to change the length attribute of the PROJNO column of the current query result, type:

```
format column projno width 8
```

The above FORMAT command displays Figure 26.
Columns defined as variable-character have an additional command to control the

```
PROJNO   |  ACTNO | ACSTAFF |    STAFF + .25 | ACSTDATE   |
-------- | ------ | ------- | -------------- | ---------- |
AD3100   |     10 |    0.50 |           0.75 | 1982-01-01 |
AD3111   |     60 |    0.80 |           1.05 | 1982-01-01 |
AD3111   |     60 |    0.50 |           0.75 | 1982-03-15 |
AD3111   |     70 |    1.50 |           1.75 | 1982-02-15 |
AD3111   |     70 |    0.50 |           0.75 | 1982-03-15 |
AD3111   |     80 |    1.25 |           1.50 | 1982-04-15 |
AD3111   |     80 |    1.00 |           1.25 | 1982-09-15 |
AD3111   |    180 |    1.00 |           1.25 | 1982-10-15 |
AD3112   |     60 |    0.75 |           1.00 | 1982-01-01 |
AD3112   |     60 |    0.50 |           0.75 | 1982-02-01 |
AD3112   |     60 |    0.75 |           1.00 | 1982-12-01 |
AD3112   |     60 |    1.00 |           1.25 | 1983-01-01 |
AD3112   |     70 |    0.75 |           1.00 | 1982-01-01 |
AD3112   |     70 |    0.50 |           0.75 | 1982-02-01 |
AD3112   |     70 |    1.00 |           1.25 | 1982-03-15 |
AD3112   |     70 |    0.25 |           0.50 | 1982-08-15 |
AD3112   |     80 |    0.35 |           0.60 | 1982-08-15 |
AD3112   |     80 |    0.50 |           0.75 | 1982-10-15 |
AD3112   |    180 |    0.50 |           0.75 | 1982-08-15 |
* End of Result ************* 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 26. A Query Result Formatted to Display a Different Column Width

displayed length attribute. See the explanation of the VARCHAR keyword in the FORMAT command description in "Chapter 10. ISQL Commands" on page 105.

You have now finished formatting the report. To produce a copy, type a PRINT command before typing the END command.

You can specify more than one keyword in a single FORMAT command. For example, the following command combines several keywords:

```
format separator ' | ' column 'expression 1' name -
'staff + .25' dplaces 1 column actno zeros off
```

Because ISQL treats this information as a single command, considerable processing time is saved. Multiple-keyword entry is described in more detail under "Using More Than One Keyword in a FORMAT Command" on page 56.

EXERCISE 3 (Answers are in Appendix A. Answers to the Exercises, page 165.)

Perform the following:

1. Retrieve the employee number, project number, and proportion of employee time from the EMP_ACT table for project numbers IF1000 and IF2000. Order the results primarily by project number and secondarily by employee number.
2. Separate all columns with two blanks, an asterisk, and two more blanks.
3. Display all columns except the EMPNO column.
4. Change the column heading for the EMPTIME column to PROPTN and its displayed length attribute to 8 characters.

Formatting Reports

Earlier in this chapter, you learned formatting techniques to create a report. In this section, you learn to prepare totals, create an outline format, and specify report titles.

To illustrate these functions, type the following query statement and format commands:

```
select projno,actno,acstaff,acstaff + .25,acstdate -
from proj_act -
where projno = 'AD3100' -
or projno = 'AD3111' -
or projno = 'AD3112' -
order by projno,actno
format separator ' | ' column 'expression 1' name -
'staff + .25' -
column projno width 8
```

This produces the following result in Figure 27.

```
PROJNO   |  ACTNO | ACSTAFF |    STAFF + .25 | ACSTDATE   |
-------- | ------ | ------- | -------------- | ---------- |
AD3100   |     10 |    0.50 |           0.75 | 1982-01-01 |
AD3111   |     60 |    0.80 |           1.05 | 1982-01-01 |
AD3111   |     60 |    0.50 |           0.75 | 1982-03-15 |
AD3111   |     70 |    1.50 |           1.75 | 1982-02-15 |
AD3111   |     70 |    0.50 |           0.75 | 1982-03-15 |
AD3111   |     80 |    1.25 |           1.50 | 1982-04-15 |
AD3111   |     80 |    1.00 |           1.25 | 1982-09-15 |
AD3111   |    180 |    1.00 |           1.25 | 1982-10-15 |
AD3112   |     60 |    0.75 |           1.00 | 1982-01-01 |
AD3112   |     60 |    0.50 |           0.75 | 1982-02-01 |
AD3112   |     60 |    0.75 |           1.00 | 1982-12-01 |
AD3112   |     60 |    1.00 |           1.25 | 1983-01-01 |
AD3112   |     70 |    0.75 |           1.00 | 1982-01-01 |
AD3112   |     70 |    0.50 |           0.75 | 1982-02-01 |
AD3112   |     70 |    1.00 |           1.25 | 1982-03-15 |
AD3112   |     70 |    0.25 |           0.50 | 1982-08-15 |
AD3112   |     80 |    0.35 |           0.60 | 1982-08-15 |
AD3112   |     80 |    0.50 |           0.75 | 1982-10-15 |
AD3112   |    180 |    0.50 |           0.75 | 1982-08-15 |
* End of Result ************* 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 27. A Formatted Query Result to Be Used to Illustrate Report Format

Obtaining an Outline Report Format

An outline report format suppresses the display of duplicate values in a particular column. To provide this feature for the PROJNO column, type the following command:

```
format group (projno)
```

This produces Figure 28 on page 52.

```
PROJNO   |  ACTNO | ACSTAFF |    STAFF + .25 | ACSTDATE   |
-------- | ------ | ------- | -------------- | ---------- |
AD3100   |     10 |    0.50 |           0.75 | 1982-01-01 |

AD3111   |     60 |    0.80 |           1.05 | 1982-01-01 |
         |     60 |    0.50 |           0.75 | 1982-03-15 |
         |     70 |    1.50 |           1.75 | 1982-02-15 |
         |     70 |    0.50 |           0.75 | 1982-03-15 |
         |     80 |    1.25 |           1.50 | 1982-04-15 |
         |     80 |    1.00 |           1.25 | 1982-09-15 |
         |    180 |    1.00 |           1.25 | 1982-10-15 |

AD3112   |     60 |    0.75 |           1.00 | 1982-01-01 |
         |     60 |    0.50 |           0.75 | 1982-02-01 |
         |     60 |    0.75 |           1.00 | 1982-12-01 |
         |     60 |    1.00 |           1.25 | 1983-01-01 |
         |     70 |    0.75 |           1.00 | 1982-01-01 |
         |     70 |    0.50 |           0.75 | 1982-02-01 |
         |     70 |    1.00 |           1.25 | 1982-03-15 |
         |     70 |    0.25 |           0.50 | 1982-08-15 |
         |     80 |    0.35 |           0.60 | 1982-08-15 |
         |     80 |    0.50 |           0.75 | 1982-10-15 |
         |    180 |    0.50 |           0.75 | 1982-08-15 |
* End of Result ************ 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 28. A Query Result Displayed in Outline Report Format

Outlining is appropriate only on a column that has been ordered into groups of similar values (through an ORDER BY clause in the SELECT statement). Although outlining can be performed on an unordered column, the frequent changes in the values that are likely to occur in that column cause such outlining to be of little value.

Outlining is normally performed when you specify GROUP on a FORMAT command. For a description of how this process is controlled, see the FORMAT command section in "Chapter 10. ISQL Commands" on page 105.

Note: If you have VARCHAR or VARGRAPHIC columns with values that differ only by trailing blanks, the FORMAT GROUP command treats them as duplicates. Therefore, if you have 'AD3100' and 'AD3100 ' in a VARCHAR column, a FORMAT GROUP command eliminates one of them.

Obtaining Totals for Reports

To produce a total for the `STAFF + .25` column, type:

```
format total ('staff + .25')
```

Note: Although included here, the parentheses are necessary only when specifying multiple columns to be totaled.

This FORMAT command displays a result similar to Figure 29 on page 53.

```
PROJNO   |  ACTNO | ACSTAFF |    STAFF + .25 | ACSTDATE   |
-------- | ------ | ------- | -------------- | ---------- |
AD3100   |     10 |    0.50 |           0.75 | 1982-01-01 |

AD3111   |     60 |    0.80 |           1.05 | 1982-01-01 |
         |     60 |    0.50 |           0.75 | 1982-03-15 |
         |     70 |    1.50 |           1.75 | 1982-02-15 |
         |     70 |    0.50 |           0.75 | 1982-03-15 |
         |     80 |    1.25 |           1.50 | 1982-04-15 |
         |     80 |    1.00 |           1.25 | 1982-09-15 |
         |    180 |    1.00 |           1.25 | 1982-10-15 |

AD3112   |     60 |    0.75 |           1.00 | 1982-01-01 |
         |     60 |    0.50 |           0.75 | 1982-02-01 |
         |     60 |    0.75 |           1.00 | 1982-12-01 |
         |     60 |    1.00 |           1.25 | 1983-01-01 |
         |     70 |    0.75 |           1.00 | 1982-01-01 |
         |     70 |    0.50 |           0.75 | 1982-02-01 |
         |     70 |    1.00 |           1.25 | 1982-03-15 |
         |     70 |    0.25 |           0.50 | 1982-08-15 |
         |     80 |    0.35 |           0.60 | 1982-08-15 |
         |     80 |    0.50 |           0.75 | 1982-10-15 |
         |    180 |    0.50 |           0.75 | 1982-08-15 |
         |        |         | ============== |            |
         |        |         |          18.65 |            |
* End of Result ************* 19 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 29. A Query Result Formatted to Produce Totals

You can also create subtotals for this column by typing the following command:

```
format subtotal ('staff + .25')
```

This displays Figure 30.

```
PROJNO   |  ACTNO | ACSTAFF |    STAFF + .25 | ACSTDATE   |
-------- | ------ | ------- | -------------- | ---------- |
AD3100   |     10 |    0.50 |           0.75 | 1982-01-01 |
         |        |         | -------------- |            |
******** |        |         |           0.75 |            |

AD3111   |     60 |    0.80 |           1.05 | 1982-01-01 |
         |     60 |    0.50 |           0.75 | 1982-03-15 |
         |     70 |    1.50 |           1.75 | 1982-02-15 |
         |     70 |    0.50 |           0.75 | 1982-03-15 |
         |     80 |    1.25 |           1.50 | 1982-04-15 |
         |     80 |    1.00 |           1.25 | 1982-09-15 |
         |    180 |    1.00 |           1.25 | 1982-10-15 |
         |        |         | -------------- |            |
******** |        |         |           8.30 |            |

AD3112   |     60 |    0.75 |           1.00 | 1982-01-01 |
         |     60 |    0.50 |           0.75 | 1982-02-01 |
         |     60 |    0.75 |           1.00 | 1982-12-01 |
         |     60 |    1.00 |           1.25 | 1983-01-01 |
         |     70 |    0.75 |           1.00 | 1982-01-01 |
         |     70 |    0.50 |           0.75 | 1982-02-01 |
         |     70 |    1.00 |           1.25 | 1982-03-15 |
         |     70 |    0.25 |           0.50 | 1982-08-15 |
```

Figure 30. A Query Result Formatted to Produce Subtotals

Notice that subtotals are created in the `STAFF + .25` column for each change in the value in the `PROJNO` column.

The conditions on which subtotals are created are defined, like outlining, by the GROUP option of a FORMAT command. Subtotals are created whenever the value changes in the column (or columns) specified with FORMAT GROUP. Columns identified with FORMAT GROUP should have been specified in the ORDER BY clause of the SELECT statement that produced the query results, and should appear in the same sequence as they appeared in the ORDER BY clause. Otherwise, the resulting subtotals can be meaningless.

You can erase these totals and subtotals. For example, to erase the totals in the above report, you would type:

```
format total erase
```

This FORMAT command would display a result similar to Figure 31. Subtotals can be erased and included in the same manner by substituting

```
PROJNO   | ACTNO | ACSTAFF |   STAFF + .25 | ACSTDATE   |
---------|-------|---------|---------------|------------|
         |    60 |    0.50 |          0.75 | 1982-03-15 |
         |    70 |    1.50 |          1.75 | 1982-02-15 |
         |    70 |    0.50 |          0.75 | 1982-03-15 |
         |    80 |    1.25 |          1.50 | 1982-04-15 |
         |    80 |    1.00 |          1.25 | 1982-09-15 |
         |   180 |    1.00 |          1.25 | 1982-10-15 |
         |       |         |---------------|            |
******** |       |         |          8.30 |            |

AD3112   |    60 |    0.75 |          1.00 | 1982-01-01 |
         |    60 |    0.50 |          0.75 | 1982-02-01 |
         |    60 |    0.75 |          1.00 | 1982-12-01 |
         |    60 |    1.00 |          1.25 | 1983-01-01 |
         |    70 |    0.75 |          1.00 | 1982-01-01 |
         |    70 |    0.50 |          0.75 | 1982-02-01 |
         |    70 |    1.00 |          1.25 | 1982-03-15 |
         |    70 |    0.25 |          0.50 | 1982-08-15 |
         |    80 |    0.35 |          0.60 | 1982-08-15 |
         |    80 |    0.50 |          0.75 | 1982-10-15 |
         |   180 |    0.50 |          0.75 | 1982-08-15 |
         |       |         |---------------|            |
******** |       |         |          9.60 |            |
```

Figure 31. A Query Result Formatted with the Totals Erased

SUBTOTAL for TOTAL in the above example. Erasing subtotals also erases totals for the specified columns.

There are some variations in the use of GROUP, SUBTOTAL, and TOTAL with FORMAT commands. See the FORMAT command section in "Chapter 10. ISQL Commands" on page 105 for details.

Creating Titles for Printed Reports

Specify a top title for the current report by typing:

```
format ttitle 'summary of employee time'
```

The quotation marks are needed because the title contains blanks. The command displays the current top title, and prompts you to return to the query result by issuing the following message in the status area:

DB2 Server for VSE

```
Press clear key to continue
```

DB2 Server for VM

```
MORE ...
```

To return to the query result, press CLEAR.

The top title can be erased and replaced with the first 100 characters of the associated SELECT statement by typing:

```
format ttitle erase
```

A bottom title can also be specified. Use the following command to create a bottom title for this report:

```
format btitle 'company confidential'
```

The bottom title can also be erased by typing:

```
format btitle erase
```

Top and bottom titles are centered in the top and bottom margins of the printed report. Although the titles cannot be seen until your report is printed, you can view them by typing FORMAT TTITLE (or FORMAT BTITLE) without specifying a title. For example, type:

```
format ttitle
```

Pressing CLEAR in response to this message returns you to the query result.

Now print a copy of this report by typing:

```
print
```

Your printed report is similar to Figure 32 on page 56.

08/10/89 SUMMARY OF EMPLOYEE TIME

PROJNO	ACTNO	ACSTAFF	STAFF + .25	ACSTDATE
AD3100	10	0.50	0.75	1982-01-01
********			0.75	
AD3111	60	0.80	1.05	1982-01-01
	60	0.50	0.75	1982-03-15
	70	1.50	1.75	1982-02-15
	70	0.50	0.75	1982-03-15
	80	1.25	1.50	1982-04-15
	80	1.00	1.25	1982-09-15
	180	1.00	1.25	1982-10-15
********			8.30	
AD3112	60	0.75	1.00	1982-01-01
	60	0.50	0.75	1982-02-01
	60	0.75	1.00	1982-12-01
	60	1.00	1.25	1983-01-01
	70	0.75	1.00	1982-01-01
	70	0.50	0.75	1982-02-01
	70	1.00	1.25	1982-03-15
	70	0.25	0.50	1982-08-15
	80	0.35	0.60	1982-08-15
	80	0.50	0.75	1982-10-15
	180	0.50	0.75	1982-08-15
********			9.60	
			18.65	

COMPANY CONFIDENTIAL

Figure 32. Example of a Printed Report

Using More Than One Keyword in a FORMAT Command

The following example provides additional practice in using multiple-keyword FORMAT commands.

Assume you want to create a report from a query result that is to include the following modifications:

- Change the blanks that separate columns to another character (SEPARATOR keyword).
- Change the name of a column heading (NAME keyword).
- Leave out a column (EXCLUDE keyword).
- Specify the number of decimal places to be displayed for decimal columns (DPLACES keyword).
- Create subtotals for specific columns (SUBTOTAL keyword and GROUP keyword).
- Create totals for specific columns (TOTAL keyword and GROUP keyword).

First, type the following statement:

```
select actno,acstaff,acstdate -
from proj_act -
where actno between 0 and 20 -
order by actno
```

This produces the display in Figure 33.
Using the above query result, type the following FORMAT command:

```
 ACTNO  ACSTAFF  ACSTDATE
------  -------  ----------
    10     0.50  1982-01-01
    10     1.00  1982-01-01
    10     0.25  1982-01-01
    10     1.00  1982-01-01
    10     0.50  1982-01-01
    10     0.50  1982-01-01
    10     0.50  1982-06-01
    10     0.50  1982-01-01
    10     1.00  1982-01-01
    10     1.00  1982-01-01
    20     1.00  1982-01-01
* End of Result *** 11 Rows Displayed ***Cost Estimate is 1********************
```

Figure 33. A Query Result to Be Used for a Multiple-Keyword Format Command

```
format column 2 name 'mean empl' dplaces 1 -
separator ' | ' exclude (acstdate) -
group actno -
subtotal ('mean empl')
```

The command produces Figure 34.

```
 ACTNO | MEAN EMPL |
------ | --------- |
    10 |       0.5 |
       |       1.0 |
       |       0.2 |
       |       1.0 |
       |       0.5 |
       |       0.5 |
       |       0.5 |
       |       0.5 |
       |       1.0 |
       |       1.0 |
       | --------- |
****** |       6.7 |

    20 |       1.0 |
       | --------- |
****** |       1.0 |
       | ========= |
       |       7.7 |

* End of Result ************* 11 Rows Displayed ****** Cost Estimate is  1   **
```

Figure 34. A Query Result Formatted by a Multiple-Keyword Format Command

Use the END command to end this query and clear the display.

Displaying Null Values and Arithmetic Errors

Null values (indicated by a question mark) and arithmetic errors (indicated by number signs separated by blanks) are displayed in a formatted table. They are treated as zeros in any total or subtotal calculations of the columns they appear in.

In the sample report in Figure 35, the mean employee staff is unknown for one project activity, and contains an arithmetic error for another.

```
          SUMMARY OF EMPLOYEE DISTRIBUTION FOR ACTIVITIES 10 AND 20


02/09/89                                                    PAGE  1


     ACTNO  | MEAN EMPL |
    ________|___________|
        10  |       0.5 |
            | # # # # # |
            |       0.2 |
            |       1.0 |
            |       0.5 |
            |       0.5 |
            |       0.5 |
            |         ? |
            |       1.0 |
            |___________|
    ******  |       6.7 |

        20  |       1.0 |
            |___________|
    ******  |       1.0 |
            |===========|
            |       7.7 |


                            COMPANY CONFIDENTIAL
```

Figure 35. Sample Report Displaying Null Values and Arithmetic Errors

Controlling Null-Field Displays

When formatting reports, you may want something other than a question mark to be used to represent null values. To illustrate, type the following statement:

```
select *  -
from department -
order by mgrno
```

This displays Figure 36 on page 59.

```
DEPTNO  DEPTNAME              MGRNO   ADMRDEPT
------  --------------------  ------  --------
A00     SPIFFY COMPUTER SERV< 000010  A00
B01     PLANNING              000020  A00
C01     INFORMATION CENTER    000030  A00
E01     SUPPORT SERVICES      000050  A00
D11     MANUFACTURING SYSTEM< 000060  D01
D21     ADMINISTRATION SYSTE< 000070  D01
E11     OPERATIONS            000090  E01
E21     SOFTWARE SUPPORT      000100  E01
D01     DEVELOPMENT CENTER    ?       A00
* End of Result *** 9 Rows Displayed ***Cost Estimate is 1*********************
```

Figure 36. A Query Result Displaying a Null Value

To format a report from this query result that replaces the question mark with **NULL**, type:

```
format null *null*
```

This FORMAT command should display a result similar to Figure 37.

```
DEPTNO  DEPTNAME              MGRNO   ADMRDEPT
------  --------------------  ------  --------
A00     SPIFFY COMPUTER SERV< 000010  A00
B01     PLANNING              000020  A00
C01     INFORMATION CENTER    000030  A00
E01     SUPPORT SERVICES      000050  A00
D11     MANUFACTURING SYSTEM< 000060  D01
D21     ADMINISTRATION SYSTE< 000070  D01
E11     OPERATIONS            000090  E01
E21     SOFTWARE SUPPORT      000100  E01
D01     DEVELOPMENT CENTER    *NULL*  A00
* End of Result *** 9 Rows Displayed ***Cost Estimate is 1*********************
```

Figure 37. A Query Result Displaying a Formatted Null Field

The maximum number of characters that can be used as a null field indicator is 20.

Use the END command to end this query and clear the display.

Controlling Query Format Characteristics

You will probably develop a standard style for formatting query results that will be consistent across queries. Given this standardization, you can specify some formatting at the beginning of a display terminal session. The formatting remains effective for the session, unless you change or override it.

The formatting that you can set in this fashion is listed below:

- Punctuation displayed for numeric columns.
- Separation characters displayed between columns.
- Characters displayed for null fields.
- The displayed length attribute of variable character fields. This topic is explained in the description of the VARCHAR item in the FORMAT command section of "Chapter 10. ISQL Commands" on page 105.

The number of copies and page size can also be set for the duration of a display terminal session.

Note: Formatting information can also be set up automatically every time you begin a DB2 Server for VSE & VM display-terminal session, by defining the information in a routine. For an explanation of the routines, refer to "Profile Routines" on page 73.

Setting the Format Characteristics by Using the SET Command

ISQL gives you some control over what you see on your display. You can specify:
- Punctuation displayed for numeric fields
- Separation characters displayed between columns
- Characters displayed for null fields
- Page size of printed reports
- Language of messages and HELP text.

Furthermore, you can specify all of these features in one command and get a list of the current settings.

Punctuation Displayed for Numeric Fields

Punctuation for numeric fields refers to the use of periods, commas, and blanks for the decimal and thousands separators. Valid combinations of the decimal and thousands separators are:

Thousands Separator	Decimal Separator	Example
(nothing)	.	1234.56
,	.	1,234.56
.	,	1.234,56
(a blank)	,	1 234,56

You can set any of these combinations for the duration of a session. For example, set the thousands separator to a comma and the decimal separator to a period for the duration of the current session by typing:

```
set decimal /,/./
```

The character between the first two slashes represents the thousands separator; the character between the last two, the decimal separator. The slashes distinguish the thousands separator from the decimal separator in the command.

Now, observe how a number is displayed using these separators. Type the following statement:

```
select 1000 * acstaff -
from proj_act -
where projno = 'ma2112' and actno = 60 -
and acstdate = '1982-01-01'
```

This SELECT statement displays a result similar to Figure 38 on page 61.

```
  EXPRESSION 1
--------------
      2,000.00
* End of Result *** 1 Rows Displayed ***Cost Estimate is 1*********************
```

Figure 38. A Query Result with a Formatted Numeric Field

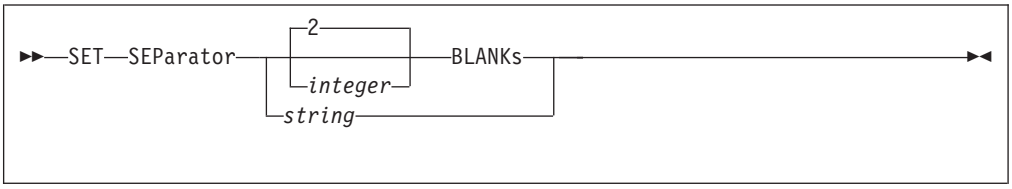
This punctuation remains in effect for all numeric columns until the end of this session. Your next session begins with the normal default (no thousands separator and a period for the decimal separator).

The valid punctuation combinations are set like this:

```
set decimal /,/./
set decimal /./,/
set decimal / /,/
set decimal //./   (nothing for thousands separator)
```

Separation Characters Displayed between Columns

Using the ISQL SET command, you can set the number of blanks or the kinds of characters to be displayed between columns for the duration of a session. The syntax for this command is shown in the following diagram:



This command can set the number of blanks displayed between the columns when an integer is used. For example, the following command causes five blanks to be displayed between columns:

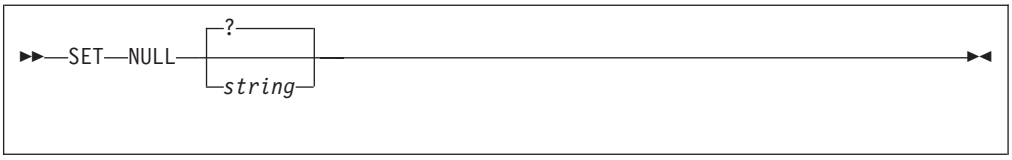
```
set separator 5 blanks
```

You can set a character string to be displayed between columns. To display an asterisk between columns, type the following command:

```
set separator *
```

Characters Displayed for Null Fields

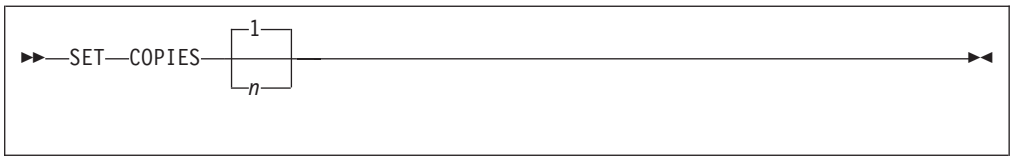
You can set the characters displayed for null fields for the duration of a session by typing a SET NULL command as illustrated in the following diagram:



Replace `string` with the actual characters you want displayed. The maximum string length is 20 characters.

Number of Copies of Printed Reports (DB2 Server for VSE)

In addition to specifying the number of copies desired for printed reports on the PRINT command, you can specify the number of copies for *all* print requests you make during the current session. This command has the following syntax:



Replace `n` with the number of copies required. The maximum number that can be specified is 99.

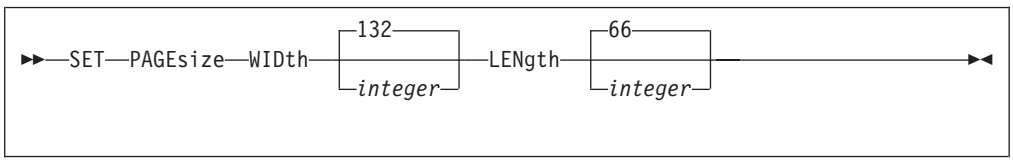
If you specify the number of copies on the PRINT command after also having specified it using a SET command, the PRINT command quantity is used for that print operation. All following PRINT commands use the quantity specified by the SET command unless they too include the `COPIES` keyword.

Page Size of Printed Reports

Defining the page size lets you place printed query results on various paper sizes. Page size is defined in terms of the number of characters that are to be printed on a line and the number of lines that are to be printed on a page.

Before defining the page size, consider the output paper size to be used and the printer characteristics (characters per inch on a line and number of lines per vertical inch) to ensure that your definition fits on the paper. For example, suppose the printer class you are going to use is set up to use 8-1/2 inch wide by 11 inch long paper, and the printer prints 10 characters to the inch horizontally and prints lines at 6 to the inch vertically. This would allow each line to contain 85 characters (8.5 x 10) and 66 lines to be on a page. The maximum page size would therefore be 85 characters wide and 66 lines in length. The maximum number of lines that can actually be printed on a page is 8 less than the length (8 lines are reserved for top and bottom titles and column headings).

Once set, the specified page size remains in effect for the duration of the display terminal session or until it is changed. The SET PAGESIZE command has the following format:



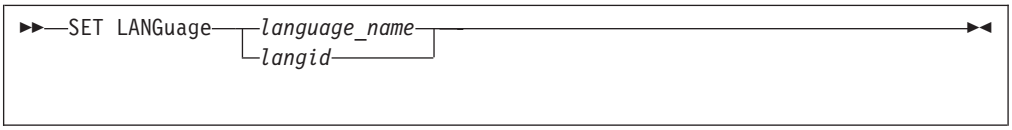
The specified width must be from 19 to 204 and the length must be from 9 to 32767.

When setting the page size, it is not necessary to specify both width and length. In addition, length and width can be specified in either order. The default value for page size is a width of 132 and a length of 66.

Language of Messages and HELP Text

Messages and HELP text can be displayed in any of several national languages. If additional languages were installed on your system, you can use the SET command to display messages and HELP text in another language. Operator

messages are displayed in the national language of the application server. The SET LANGUAGE command takes the following form:



Messages can be displayed in one of the languages listed in the table in Table 1.

Table 1. Alternative Languages for Messages and HELP Text

Language	Language ID
American English (mixed case)	AMENG
American English (uppercase)	UCENG
French	FRANC
German	GER
Simplified Chinese	HANZI
Japanese	KANJI

Either the name of the language or the language ID can be used as the language identifier in the SET LANGUAGE command. For example, you could have messages displayed in French by typing:

```
set language franc
```

If your system uses *french* as the language name you could also type:

```
set language french
```

Languages for which there is no language ID can be set using the name of the language. If your system does not support the language you request, an error message is displayed in the default language.

To find out what languages are available on your system (and what language names or language identifiers you can use to select a language) type:

```
select *  from sqldba.syslanguage
```

A table is displayed listing the names, language IDs, and a brief description of each language available to you.

Multiple Format Characteristics

You can use the SET command with more than one keyword. In this way, you can specify or change multiple characteristics to be effective for the duration of your display session. For example, type the following command to combine several features that you set earlier using separate commands:

```
set autocommit on null *null* separator 2 blanks
```

This command sets AUTOCOMMIT to on, display **NULL** for each null field, and produces a display with columns separated by 2 blanks.

The range of characteristics you can set with the SET command is included in "Chapter 10. ISQL Commands" on page 105.

List of Current Settings

The current settings of all the format characteristics described above can be listed on your display. List them by typing:

```
list set *
```

The asterisk means that you want to list all settings. This displays a series of messages that describe the settings of each operational characteristic.

You can request a specific characteristic by specifying the name of the characteristic instead of the asterisk on the SET command. For example, type the following command to list the current setting for the column separator:

```
list set separator
```

Printing Reports on a Workstation Printer (DB2 Server for VSE)

Your printed output is automatically sent to the system printer designated by your site. You can change or redirect your printed output to a CICS terminal printer or POWER remote workstation.

To route your printed output to a CICS terminal printer, you would specify:

```
print termid  termid
```

or

```
set printroute  termid
```

where *termid* is replaced with the CICS terminal printer identifier.

To route printed output to a POWER remote workstation, specify:

```
print destid  wkstat
```

or

```
set printroute  wkstat
```

where *wkstat* is replaced with the identifier for the required POWER remote workstation.

The system printer is specified as:

```
print system
```

or

```
set printroute system
```

EXERCISE 4 (Answers are in Appendix A. Answers to the Exercises, page 166.)

Perform the following:

1. For DB2 Server for VSE, set the number of copies requested for printed reports to two.
2. List the current settings for format characteristics.
3. Retrieve all of the information from PROJ_ACT where the project number is AD3112; order the result by activity end date.
4. Create an outline format for the activity end date column and exclude the activity start date column.
5. Create a top title called *PERSONNEL PROGRAMMING DEADLINES*.
6. For DB2 Server for VSE, request printed copies of this report.
7. For DB2 Server for VM, request two printed copies of this report.

Chapter 6. Storing SQL Statements

Storing frequently-used SQL statements saves you from retyping them. On longer statements, such as that used in the previous chapter to format a report, typing errors can be avoided by storing the statement and reusing it. A stored SQL statement may also contain placeholders.

Storing the Current Statement

Type the following:

```
select deptno,mgrno -
from department -
order by mgrno
```

Now, type the following format commands:

```
format separator ' | ' -
column mgrno width 8
```

End the query.

Store the current SQL statement using the storage name *dept* by typing the following:

```
store dept
```

If you know that the stored command DEPT already exists, and you wish to replace it, include the keyword REPLACE in your STORE command, as follows:

```
store dept replace
```

When the STORE command is processed, a message informs you that the SQL statement is stored. Use storage names that are 8 characters or less in length. Do not use the name *previous* because ISQL always saves the current SQL statement, and names it *previous* when a new SQL statement is typed.

Protecting a Stored Statement

If the name you use with the STORE command is the name of a statement that is already stored, and you do *not* use the REPLACE keyword, you receive a warning message. The warning gives you the following choices:

- REPLACE the existing stored SQL statement with the new statement.
- END the processing of the STORE command, leaving the existing stored SQL statement intact.
- Enter a new name under which your new statement is to be stored.

The message itself appears as in Figure 39 on page 68.

```
ARI7955I The system ended your query result to process your command.
ARI7577D A stored SQL statement named DEPT already exists.
         Enter a new name to store the SQL statement, or enter
         one of the following keywords:
         REPLACE - to replace the existing stored SQL statement, or
         END - to end the store command processing.
```

Figure 39. Warning Message Displayed If You Try to Store a Name Already Stored

You can use the REPLACE function to replace an old statement with a new one. If you have already stored a statement under the name DEPT, you can still store the current statement under the same name by typing:

```
replace
```

and pressing ENTER.

If you do not want to replace the previously stored statement, simply type:

```
end
```

Press ENTER.

The END command returns you to the ISQL environment.

Starting a Stored Statement

When you start a stored SQL statement, ISQL moves it to the command buffer and then processes it. Start the statement stored as DEPT by typing the following:

```
start dept
```

When the results appear on your display, verify that the FORMAT commands you typed with the SELECT statement stored as DEPT are still in effect.

Note: Only the SELECT portion of the stored statement is recalled. The stored formatting commands are not shown although you can see their effect on the query.

Type END. The stored SELECT statement called DEPT is not erased. It remains in storage until you decide to erase it.

Starting a Stored Statement That Contains Placeholders

First, store a statement that contains placeholders. Type:

```
hold select &1 -
from &2 -
where &3

store myquery
```

Now start the statement, and add parameters to complete it. Type:

```
start myquery (* employee empno='000010')
```

This produces the following statement:

```
select * from employee where empno='000010'
```

and displays Figure 40.

```
EMPNO   FIRSTNME      MIDINIT  LASTNAME         WORKDEPT  PHONENO  HIREDATE
------  ------------  -------  ---------------  --------  -------  ----------
000010  CHRISTINE     I        HAAS             A00       3978     1965-01-01
* End of Result *** 1 Rows Displayed ***Cost Estimate is 1*********************
```

Figure 40. A Query Result from Starting a Stored Statement with Placeholders

Note: A message is displayed, indicating that any formatting performed while viewing this query result is not saved. This condition is explained in detail in "Saving the Format Information".

Because the placeholders are replaced by the parameters when the statement is processed, the stored SQL statement remains unchanged and can be reused.

Recalling a Stored Statement

A stored SQL statement can be recalled to the command buffer to become the current SQL statement without being processed. You can do this to make changes to it and store it again, or to simply verify that it is the statement that you want to process next. For example, recall DEPT by typing:

```
recall dept
```

The recalled statement resembles:

```
SELECT DEPTNO,MGRNO FROM DEPARTMENT ORDER BY MGRNO
```

You can also use the RECALL command to display the current SQL statement, or to place the previous SQL statement in the command buffer at the top of the stack so that it becomes the current SQL statement. To recall the previous SQL statement, type:

```
recall previous
```

The statement that was current in the buffer becomes the new *previous* SQL statement.

To display the current SQL statement in the command buffer, type:

```
recall
```

or press PF5. The absence of a name on the RECALL command instructs ISQL to display the current SQL statement.

Saving the Format Information

The formatting information performed while viewing the query result is always saved when the query does not contain placeholders. When the query contains placeholders, however, the formatting information is saved only if the placeholders are not in the SELECT or FROM clause. To illustrate how format information is saved, remove the placeholders and insert actual values into the SELECT and FROM clauses of MYQUERY, but retain the placeholder in the WHERE clause by typing:

```
hold select * -
from employee -
where &1

store myquery replace
```

Notice the keyword REPLACE; it instructs ISQL that the statement you are now storing replaces the previous version of MYQUERY. If the keyword REPLACE is not used, ISQL issues the warning message described earlier. Of course, you can choose another name and save both the old and new queries.

Now start MYQUERY and include the parameter that completes the WHERE clause:

```
start myquery (empno='000010')
```

This processes the following statement:

```
select * -
from employee -
where empno='000010'
```

Add formatting information to the query result. Type:

```
format separator ' | '
```

End the query, and then save the formatting information by typing:

```
store myquery replace
```

Start the query again, but substitute a different value for the placeholder:

```
start myquery (empno='000150')
```

You see that the query result has retained the separator that you formatted and stored.

Changing a Stored Statement

Changes can be made to a stored SQL statement using the CHANGE command discussed earlier in this chapter under "Changing the Current SQL Statement" on page 38. For example, to change the ORDER BY clause in the statement stored as DEPT, perform the following:

1. Use the RECALL command to display the stored statement. Type:

   ```
   recall dept
   ```

   The recalled statement appears as:

   ```
   SELECT DEPTNO,MGRNO FROM DEPARTMENT ORDER BY MGRNO
   ```

2. For this example, change the ORDER BY feature to the DEPTNO column. Type:

   ```
   change /by mgrno/by deptno/
   ```

   The changed statement is displayed as:

   ```
   SELECT DEPTNO,MGRNO FROM DEPARTMENT ORDER BY DEPTNO
   ```

3. Now, store the changed SELECT statement as *DEPT2*:

   ```
   store dept2
   ```

4. To verify the new statement, use the RECALL command:

   ```
   recall dept2
   ```

If you look at the query results of DEPT and DEPT2, you will notice that the FORMAT information is maintained. As long as the statement to be changed or stored contains an unchanged SELECT or FROM clause, FORMAT information is saved.

Listing the Names of Stored Statements

To view the names and contents of stored statements, type:

```
list sql *
```

This displays information similar to Figure 41.

```
DEPT         SELECT DEPTNO,MGRNO FROM DEPARTMENT ORDER BY MGRNO
DEPT2        SELECT DEPTNO,MGRNO FROM DEPARTMENT ORDER BY DEPTN
MYQUERY      SELECT * FROM EMPLOYEE WHERE &1;
ARI7620I You have          3 stored SQL statements.
```

Figure 41. A Display of the Stored SQL Statements

Notice that only the first 50 characters of each statement are displayed.

You may wish to store a statement but are unsure if the stored name you want to use is already in use. Using the LIST SQL * command becomes tedious if there are many stored statements. Instead, you can list a specific stored statement by using its name in the LIST command. Type:

```
list sql getsup
```

You receive a message indicating that GETSUP is not found, and you can use it as the new store name.

You can also use this form of the LIST command to view statements whose name you know. For example, type the following to view DEPT2:

```
list sql dept2
```

You can use multiple statement names with the LIST command. For example, type:

```
list sql dept myquery
```

Renaming a Stored Statement

You can rename stored SQL statements. For example, type:

```
rename dept olddept
```

The stored statement DEPT is now called OLDDEPT. Do not use *previous* with the RENAME command.

Erasing a Stored Statement

When you no longer need a stored SQL statement, you can erase it. Type:

```
erase olddept
```

The stored statement called OLDDEPT has been erased.

You can erase several stored statements with a single ERASE command:

```
erase first second
```

EXERCISE 5 (Answers are in Appendix A. Answers to the Exercises, page 166.)

1. Recall the stored query MYQUERY.
2. Change the query so that information is retrieved for salaries between $25000 and $30000, and is ordered by the employee's first name.
3. Start the query.
4. Exclude the middle initial from the result.
5. Separate all columns with a blank, a vertical bar, an asterisk, a vertical bar, and a blank.
6. Change the EDLEVEL column heading to SCHOOL YEARS.
7. End the display and store the command along with all its formatting information using the name EXER11.
8. List all the stored SQL statements.
9. Retrieve the help information for the ISQL STORE command.

Chapter 7. Creating and Using Routines

A routine is a series of ISQL commands, SQL statements, or both, which is stored under an identifying name. When a routine is run, each command or statement in the routine is performed as if it had been typed from the keyboard. Routines save (and later run) frequently-used sequences of commands and statements, such as a series of ISQL SET commands that set certain operational characteristics.

Routines offer a number of features to users. In addition to using them to perform a frequently used set of commands or statements, you can:

- Define a profile routine that is run automatically when you start ISQL
- Use a routine to start one or more stored SQL statements
- Share routines with other users
- Use a routine to display and print query results.

Running Routines When ISQL Is Started

Profile Routines

A routine named *PROFILE*, that was previously set up by you, is performed automatically when you start ISQL. This allows unique terminal-session characteristics to be established before any information is typed. You can set these characteristics yourself by using the SET command described in "Chapter 10. ISQL Commands" on page 105.

DB2 Server for VM

Note: The PROFILE routines described in this section are ISQL PROFILE routines and should not be confused with a VM PROFILE EXEC.

Placeholders are not allowed in a profile routine. Because the routine is run automatically, there is no way to pass parameters to substitute for the placeholders.

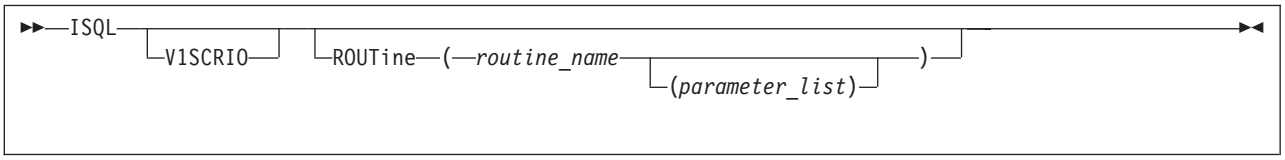
In addition, a unique master profile routine can be created by someone having DBA authority. It is created with an authorization ID of SQLDBA and is run automatically for each user who starts ISQL as if it were the user's own profile routine.

Both the master and your own profile routines run automatically when you start ISQL. Your routine runs second, and its operational characteristics override those in the master routine.

Routines to Which Parameters Can Be Passed (DB2 Server for VM)

After the profile routine is run, you can invoke a subsequent routine to run as part of the ISQL signon process. Parameters can be passed on to this routine. The ISQL EXEC procedure invokes ISQL and runs this subsequent routine.

The ISQL EXEC procedure has the following format:



You can create your own EXEC procedure from the ISQL EXEC procedure. For example, you can add commands to your EXEC to change the PF-key settings or to route printed reports.

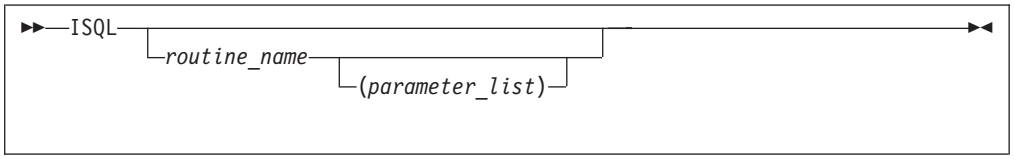
You set your own terminal-session characteristics by using the SET command described in "Chapter 10. ISQL Commands" on page 105.

Using the ISQL Transaction Identifier (DB2 Server for VSE)

A profile routine cannot contain parameters, but an ISQL routine that runs as part of the ISQL signon procedure can contain parameters. You use the 4-character transaction identifier `ISQL` to start ISQL and to run a routine. Two ways of using the transaction identifier are:

1. Start ISQL, type your user ID and password, which the signon screen prompts you to do.
2. Start ISQL from another CICS transaction that suppresses the signon screen and the related terminal messages. The format for doing this along with its explanation are found in "Appendix E. Suppressing the ISQL Sign-On Display for DB2 Server for VSE" on page 187.

For method 1, the ISQL transaction identifier and routine have the following format:



Both of the above-mentioned methods let you specify a routine, and any parameters to be passed to that routine, to be run by ISQL after it has run a profile routine. For method 1, the ISQL transaction identifier and the routine name and all its parameters must not be longer than the screen width or 132, whichever is less. For method 2, the command which suppresses the ISQL signon display as described in "Appendix E. Suppressing the ISQL Sign-On Display for DB2 Server for VSE" on page 187, must not be longer than the screen width or 132, whichever is less.

You can insert any number of blanks after the routine name or the parameters and the surrounding parentheses. For example:

```
isql    routine  (abc (1 2   3))
```

You can create your own routine or profile routine. For example, in your profile, you can add commands to change the number of copies of printed reports or route printed reports.

You can change the terminal-session characteristics. You can do this each time you use ISQL (SET command, page 149), or you can have ISQL do it for you (SET commands in the PROFILE routine).

Establishing Where Routines Are Stored

Routines are stored in a special table called *ROUTINE*. It consists of four columns: NAME, SEQNO, COMMAND, and REMARKS (optional). An example of a routine table is shown in Table 2.

Table 2. Example of a Routine Table

NAME	SEQNO	COMMAND	REMARKS
EMPLREP	10	SELECT PROJNO,ACTNO,ACSTAFF -	GENERATE AND PRINT A
EMPLREP	20	FROM PROJ_ACT -	REPORT FOR PROJECT
EMPLREP	30	WHERE PROJNO IN (&1) -	STAFF
EMPLREP	40	ORDER BY PROJNO,ACTNO	
EMPLREP	50	FORMAT GROUP PROJNO	
EMPLREP	60	FORMAT SUBTOTAL ACSTAFF	
EMPLREP	70	FORMAT TTITLE 'AVERAGE PROJECT STAFF'	
EMPLREP	80	PRINT	
EMPLREP	90	END	
PRINTDEP	10	SELECT * FROM DEPARTMENT	FORMAT DEPARTMENT TABLE
PRINTDEP	20	FORMAT SEPARATOR ' | '	AND PRINT IT
PRINTDEP	30	FORMAT COLUMN DEPTNAME WIDTH 30	
PRINTDEP	40	PRINT	
PRINTDEP	50	END	

The `NAME` column identifies the rows that belong to a particular routine. `SEQNO` specifies the sequence in which the commands and statements are executed. Use sequence numbers that are increments of ten to allow for later additions. The `COMMAND` column contains the SQL statements and ISQL commands.

Before creating routines, you must have a routine table. You can create your own routine table if you have Resource authority or your own private dbspace (DB2 Server for VM user only). If you do not have Resource authority, ask the appropriate person to create a routine table for you. The following SQL statement illustrates the creation of a routine table:

```
create table routine (name char(8) not null, -
                      seqno integer not null, -
                      command varchar(254) not null, -
                      remarks varchar(254))
```

The CREATE TABLE statement is discussed in detail in "Chapter 8. Creating and Managing Tables" on page 83.

When creating this table, use the CREATE TABLE statement exactly as shown above, except for the REMARKS column which is optional. If it is included, specifying a data type of VARCHAR with a length of 40 is usually sufficient. The size selected for the REMARKS column should accommodate the largest entry used. Allow nulls so that remarks are not required. The COMMAND column can be a maximum length of 254 characters.

Once the table is created, create an index for it so that when the table is referenced, the commands or statements are displayed or executed in the correct order. To create an index, type a statement similar to the following (substitute a name of your choice for *RINDEX*):

```
create unique index rindex on routine -
              (name, seqno)
```

For detailed information on the CREATE INDEX statement, refer to "Chapter 8. Creating and Managing Tables" on page 83.

Storing a Routine

When you insert new commands or statements for a routine into the ROUTINE table, always assign the same routine name in the NAME column. Commands and statements are inserted into the routine table in the same manner as data is inserted into any database manager table by using SQL INSERT statements or the ISQL INPUT command.

The use of ampersands (&) in a routine is allowed only for creating placeholders (see the description of the RUN command for more information on placeholders).

In the COMMAND column of the routine, be sure to:

- Put single quotation marks around any placeholder that stands for a character data item. Enclose the COMMAND column (CHARACTER data type) in single quotation marks, and use a pair of single quotation marks for each single quotation mark that is to appear inside an SQL statement. This is shown in the example below.
- Use the continuation character as if you were typing a long SQL statement.

In the following example, the INPUT command inserts the commands and statements for a routine named *QREPORT* into the routine table.

```
input routine
'qreport',10,'select projno,acstaff -','begin:'
'qreport',20,'from proj_act -',null
'qreport',30,'where projno = '&1'' -',null
'qreport',40,'or actno = &2 -',null
'qreport',50,'order by projno',null
'qreport',60,'format group projno',null
'qreport',70,'format subtotal acstaff',null
'qreport',80,'print',null
'qreport',90,'end','done!'
 end
```

The stored information should resemble Table 3.

Table 3. Routine Statements Inserted in the Routine Table

NAME	SEQNO	COMMAND	REMARKS
QREPORT	10	SELECT PROJNO,ACSTAFF -	BEGIN:
QREPORT	20	FROM PROJ_ACT -	
QREPORT	30	WHERE PROJNO = '&1' -	
QREPORT	40	OR ACTNO = &2 -	
QREPORT	50	ORDER BY PROJNO	
QREPORT	60	FORMAT GROUP PROJNO	
QREPORT	70	FORMAT SUBTOTAL ACSTAFF	
QREPORT	80	PRINT	
QREPORT	90	END	DONE!

To display your stored QREPORT routine, type:

```
select * -
from routine -
where name='qreport'
```

The display resembles Figure 42.

```
NAME            SEQNO  COMMAND               REMARKS
--------  ------------  --------------------  --------------------
QREPORT             10  SELECT PROJNO,ACSTAF<  BEGIN:
QREPORT             20  FROM PROJ_ACT          ?
QREPORT             30  WHERE PROJNO = '&1'    ?
QREPORT             40  OR ACTNO = &2 -        ?
QREPORT             50  ORDER BY PROJNO        ?
QREPORT             60  FORMAT GROUP PROJNO    ?
QREPORT             70  FORMAT SUBTOTAL ACST<  ?
QREPORT             80  PRINT                  ?
QREPORT             90  END                    DONE!
* End of Result *** 9 Rows Displayed ***Cost Estimate is 1 ********
```

Figure 42. Display of Routine Statements Inserted in the Routine Table

Managing a Routine

Routines stored in a table are managed in the same manner as data in any table by using SQL UPDATE, INSERT, and DELETE statements.

For example, to modify the PRINTDEP routine (shown in Table 2 on page 75) to use a double bar instead of a single bar to separate columns, you can type:

```
update routine -
set command = 'format separator  '' || ''' -
where seqno = 20 and name = 'printdep'
```

As another example, you can delete the entire QREPORT routine by typing:

```
delete from routine -
where name = 'qreport'
```

Running a Routine

You run routines by using the ISQL RUN command. Use placeholders and parameters in the same manner as you would use them in stored SQL statements.

For example, type the following command and placeholder replacement values to run the QREPORT routine you created:

```
run qreport ('ad3100' 180)
```

Routine QREPORT creates a printed report similar to that shown in Figure 43 on page 78. The query result is printed rather than displayed because of the PRINT command contained in the routine. Producing a query result on the display is discussed later in this chapter under “Using SELECT Statements in a Routine” on page 79.

```
08/27/89   SELECT PROJNO,ACSTAFF FROM PROJ_ACT WHERE PROJNO = 'AD3


PROJNO  ACSTAFF
______  _______
AD3100    0.50
        _______
******    0.50

AD3111    1.00
        _______
******    1.00

AD3112    0.50
        _______
******    0.50

AD3113    0.75
          1.00
          0.50
        _______
******    2.25

MA2112    1.00
          1.00
```

Figure 43. Example of a Report Created from a Routine

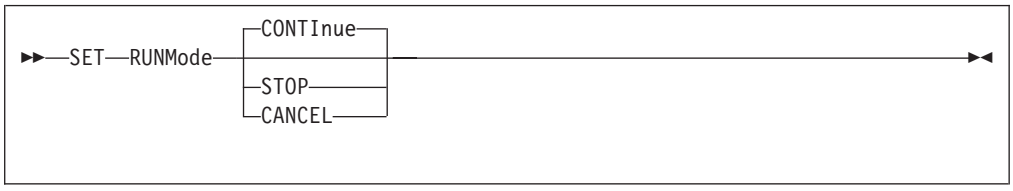
Running Shared Routines

You can run another user's routine if you have obtained the SELECT privilege (using a GRANT statement as described in "Sharing Your Tables with Other Users" on page 91) on that user's routine table.

Error Mode Processing in a Routine

When an error is detected in a routine, continued processing depends on a setting you make to a special RUNMODE indicator. It indicates the mode of operation for running routines.

The format for the command that sets the indicator is:



CONTInue
indicates that your routine continues to the next command or statement even if errors are detected. You are likely to use this option when your routine contains several independent commands or statements. Failure of a particular command or statement does not affect the remaining commands and statements.

If you do not specify a RUNMODE option, CONTINUE becomes the default value.

STOP
indicates that your routine is ended, but a ROLLBACK operation is not performed. Select this option when commands and statements within a routine are not interrelated, and you want to maintain any changes made to the database. The remaining commands and statements are not executed if errors exist in any of their predecessors.

CANCEL
indicates that your routine is ended and a ROLLBACK operation is performed. Select this option when your routine contains a series of interrelated commands and statements that update tables. When an error is detected, all changes generated by the routine are erased and the integrity of the database is maintained.

You can display the current RUNMODE setting by issuing the following command:

```
list set runmode
```

You can set the RUNMODE indicator at any time, either from the display terminal or in a routine.

Using INPUT Commands in a Routine

If the ISQL INPUT command is used in a routine, all data and INPUT subcommands (SAVE and BACKOUT) must also be entered from the routine.

Using SELECT Statements in a Routine

The use of SELECT statements in a routine (either an actual statement contained in the routine or a stored statement started by the routine) is slightly different from usage of the statements at a keyboard. SELECT statements contained in a routine do not cause query results to be displayed automatically at the terminal.

The SELECT statement results can be displayed at the terminal by placing an ISQL DISPLAY command in the routine at the desired location. This command can be placed anywhere between the SELECT statement and its associated END command. The DISPLAY command allows results to be formatted before they are displayed. It also lets you type display commands from the keyboard after the routine results are displayed.

To illustrate this process, type the following command:

```
insert into routine (name,seqno,command) -
values ('qreport',75,'display')
```

The QREPORT routine now appears as Table 4.

Table 4. Modified QREPORT Routine

NAME	SEQNO	COMMAND	REMARKS
QREPORT	10	SELECT PROJNO, ACSTAFF -	BEGIN:
QREPORT	20	FROM PROJ_ACT -	
QREPORT	30	WHERE PROJNO = '&1' -	
QREPORT	40	OR ACTNO = &2 -	
QREPORT	50	ORDER BY PROJNO	
QREPORT	60	FORMAT GROUP PROJNO	
QREPORT	70	FORMAT SUBTOTAL ACSTAFF	
QREPORT	75	DISPLAY	
QREPORT	80	PRINT	
QREPORT	90	END	DONE!

Run the routine by typing:

```
run qreport ('ad3100' 180)
```

The formatted query result is now displayed at your terminal. If required, you can alter it using additional FORMAT commands. The display resembles Figure 44.

```
PROJNO  ACSTAFF
------  -------
AD3100     0.50
        -------
******     0.50

AD3111     1.00
        -------
******     1.00

AD3112     0.50
        -------
******     0.50

AD3113     0.75
           1.00
           0.50
        -------
******     2.25

MA2112     1.00
           1.00
        -------
******     2.00
```

Figure 44. Formatted Query Result from Running Routine QREPORT

Type an END command to end the display of the query result and return to the routine. The routine issues its remaining commands, causing a report containing the modified query results to be printed.

EXERCISE 6 (Answers are in Appendix A. Answers to the Exercises, page 166.)

1. Create a routine named EXER13 to:
   a. Select the ACTNO and ACTDESC columns from the ACTIVITY table.
   b. Provide a separation between columns consisting of three blanks.
   c. Display the results on the screen.
   d. Request three copies of the resulting report.
   e. End the query.

Note: Remember to take the REMARKS column of the ROUTINE table into account if you have such a column.

Chapter 8. Creating and Managing Tables

Some users of the database manager are authorized to create and manage their own set of tables. There are two ways you can obtain the authority to create and manage your own tables. You may have RESOURCE authority, or you may ask someone with DBA authority to acquire a dbspace on your behalf. The latter lets you create tables in a private dbspace only (dbspaces are discussed later in this chapter.)

To determine what authority you have, type:

```
select resourceauth from sqldba.sysuserlist -
where name = user
```

A `Y` under `RESOURCEAUTH` indicates that you have resource authority and can create and manage your own set of tables.

You can use the following sections if you have resource authority. They describe how to manage your own tables, share them with other users, and improve query performance.

Managing Your Own Tables

Managing your tables consists of the following:

- Determining what your information is
- Creating tables
- Creating referential structures
- Determining where your tables are stored
- Dropping (deleting) tables
- Dropping primary or foreign keys
- Activating or deactivating primary or foreign keys
- Copying data from one table into another
- Identifying the minimum contents of a table
- Adding columns to a table.

Querying Information about Your Tables

The database manager automatically maintains a catalog that contains several catalog tables. The catalog holds information about the application server. Table names, view names, table owners, view owners, and column names are just a part of the information that can be found in the catalog tables. The data in the catalog tables is available to SQL users through the normal SQL query facilities. Whenever you use the catalog tables, remember to prefix the table name with the owner name *system*.

To find out which tables and views belong to you, type the following:

```
select tname,remarks -
from system.syscatalog -
where creator = user
```

This query presents results similar to Figure 45 on page 84.

```
TNAME              REMARKS
-----------------  ---------------------------------
ACT10
ACT60
PROJ1
PROJ2
ROUTINE
* End of Result *** 5 Rows Displayed ***Cost Estimate is 1*******************
```

Figure 45. A Query Result Displaying Which Tables and Views Belong to You

Your display may provide additional information if you have created views and tables of your own.

The column TNAME contains the table or view name. The REMARKS column contains information about the table or view. You enter information into this field for your tables or views using the COMMENT command. (For information on using the COMMENT command, see the *DB2 Server for VSE & VM SQL Reference* manual.)

USER in the above query instructs the system to use your authorization ID when selecting information from the SYSCATALOG table. You can use your authorization ID in place of USER in the WHERE clause. If your authorization ID is VELDA, for example, you can type the previous statement as:

```
select tname,remarks -
from system.syscatalog -
where creator = 'velda'
```

To determine the column names of the EMPLOYEE table, type the following:

```
select cname -
from system.syscolumns -
where tname = 'employee'
```

This query provides results similar to Figure 46.

```
CNAME
-----------------
BIRTHDATE
BONUS
COMM
EDLEVEL
EMPNO
FIRSTNME
HIREDATE
JOB
LASTNAME
MIDINIT
PHONENO
SALARY
SEX
WORKDEPT
* End of Result *** 14 Rows Displayed ***Cost Estimate is 1*******************
```

Figure 46. A Query Result Displaying the Column Names of the EMPLOYEE Table

For more information on the catalog, see the *DB2 Server for VSE & VM SQL Reference* manual.

Creating Your Own Tables

You can create your own table by providing the database manager with a name for the table and the columns that you want it to contain. The names chosen can consist of letters, numbers, and some special characters. If you want a blank or other special character in the name, you must enclose the name in double quotation marks.

When you provide the column names, you must also indicate the type of data for each column. For more information on data types that can be defined for a column, refer to the *DB2 Server for VSE & VM SQL Reference* manual.

For example, type the following SQL statement to create a table with employee information for a particular quarter:

```
create table empl -
(empno char(6), -
 lastname varchar(15), -
 edlevel smallint, -
 birthdate date, -
 quarter integer)
```

Note: ISQL does *not* support INSERT, UPDATE, or SELECT for tables or views with VARCHAR>254, VARGRAPHIC>127, LONG VARCHAR or LONG VARGRAPHIC columns.

ISQL supports hexadecimal constants and graphic constants that can be used to insert (INSERT or INPUT commands) or update data into DBCS columns of length <= 127. Hexadecimal constants and graphic constants can also be used in WHERE clauses with DBCS columns. See the *DB2 Server for VSE & VM SQL Reference* manual for more information about this data type.

Defining Column Data

When you create a table, you can indicate that character columns will contain single-byte characters, single or double-byte characters, or bit data by specifying FOR SBCS DATA, FOR MIXED DATA, or FOR BIT DATA respectively.

You can also indicate that character or graphic columns will contain data represented using a specific character set, code page, and encoding scheme by specifying a Coded Character Set Identifier (CCSID). CCSIDs are important for applications that use the DRDA protocol. With the DRDA protocol, data at the application server and application requester could be represented by different CCSIDs, as for example in the ASCII and EBCDIC environments. The CCSID clause consists of the keyword CCSID followed by an integer and assigns the integer identifier as the CCSID attribute of the column. This attribute defines the specific character set, code page, and encoding scheme used to represent the data in the character or graphic column.

For more information on column clauses and CCSIDs, see the *DB2 Server for VSE & VM SQL Reference* manual.

Storing Your Tables

When you create a table, the database manager inserts it into a section of the database that is reserved for you. For DB2 Server for VSE, these sections are called *dbspaces*. For DB2 Server for VM, these sections are called *private dbspaces*.

There are private dbspaces and public dbspaces. If a DBA has given you RESOURCE authority, you can acquire private dbspaces to contain your tables. You

can also create your tables in public dbspaces if you have RESOURCE authority. If you do not have RESOURCE authority, you can create tables only if a DBA has acquired a private dbspace for you.

If you have more than one dbspace, you can let the database manager choose the one in which to place your tables, or you can specify the dbspace yourself. For example, assume that you have had two dbspaces reserved for you named *JOHN1* and *JOHN2*. To place the EMPL table in JOHN2, you type the following when you create the table:

```
create table empl -
(empno char(6), -
 lastname varchar(15), -
 edlevel smallint, -
 birthdate date, -
 quarter integer) -
in john2
```

Copying Data from Other Tables

Data can be loaded into your table by copying data from another table. This is performed using a variation of the INSERT statement. For example, load (copy) data from the EMPLOYEE table into the EMPL table using the following INSERT statement:

```
insert into empl (lastname,edlevel,birthdate,quarter) -
  select lastname,edlevel,birthdate,'014'  -
  from employee
```

This form of the INSERT statement uses a subquery instead of the VALUES clause. The information retrieved by the subquery is placed into the table as if multiple INSERT statements had been entered.

In the example above, the database manager is instructed to copy the employee information from each row of the EMPLOYEE table into the EMPL table. The database manager is also instructed to place the value *014* in the QUARTER column for each row inserted.

Dropping a Table

You can drop (delete) one of your tables using a DROP TABLE statement. For example, drop the table just created by typing:

```
drop table empl
```

The DROP TABLE statement deletes all the rows of the table, as well as the definition of the table. (Recall that the DELETE statement, deletes individual rows.)

Identifying the Minimum Contents of a Table

When you create a table, you can identify which columns must contain entries. You would typically want key fields such as EMPNO in the EMPLOYEE table to always contain valid (not null) values. You prevent the use of nulls by specifying the NOT NULL option in the column definitions of CREATE TABLE statements.

For example, the following statement can be used to prevent nulls from being used for the EMPNO and LASTNAME columns of the EMPL table:

```
create table empl -
(empno char(6) not null, -
 lastname varchar(15) not null, -
 edlevel smallint, -
 birthdate date, -
 quarter integer)
```

Specifying NOT NULL can be very useful in copying data from other tables because it prevents incomplete rows from being inserted into your table.

In the above example, the BIRTHDATE column was defined as NOT NULL, because nulls are allowed for the corresponding column in the EMPLOYEE table.

Adding a Column to a Table

When you create a table, it is not necessary to know or specify all columns. Additional columns can be added later using the SQL statement ALTER TABLE. For example, you can add a column to the EMPL table to contain employee salary information by typing:

```
alter table empl -
add salary decimal(9,2)
```

You cannot specify the NOT NULL option in an ALTER TABLE statement. All existing rows of the table assume a null value for the SALARY column as a result of the ALTER TABLE statement.

After a column has been added, you can insert values using the UPDATE statement.

Specifying Referential Constraints

Referential constraints can be specified when tables are defined, or they can be added later.

When a primary key is added to an existing table, the database manager checks the table to ensure that all keys are unique. When a foreign key is added, the database manager checks all non-null foreign keys to ensure that they exist in the parent table.

Constraints can also be dropped, activated, or deactivated. When referential constraints have been deactivated, the database manager suspends checking, and the tables become unavailable for access by anyone other than the owner of the table or by someone possessing DBA authority.

Tables are deactivated, for example, to load large amounts of data onto the table. Since checking of the data has been suspended, the speed of the loading process increases considerably. When loading is complete, the table is activated.

When the primary key is deactivated, the primary key index on the parent table is automatically dropped and all active dependent foreign keys are implicitly deactivated. When a primary key is deactivated, all associated foreign keys are implicitly deactivated. When a primary key or a dependent foreign key is deactivated, all tables involved in the referential constraint become unavailable until the keys are activated once again. Activating the keys causes the database manager to validate the references in the data, and referential constraints are automatically enforced once again.

Additional information is provided for activation and deactivation in "Activating and Deactivating Primary Keys, Foreign Keys, or Unique Constraints" on page 90.

You can remove a referential constraint by dropping the foreign key. Dropping a table containing foreign keys removes the constraints associated with its keys. When a table containing a primary key is dropped, any foreign keys that reference the primary key are dropped automatically, thereby removing all the constraints that reference the primary key.

Identifying Required Privileges

The following table identifies the privileges required for changes to the referential structure.

Table 5. Privileges Required

ALTER TABLE Clause	Privilege on Parent Table	Privilege on Dependent Table
Add Column	ALTER	
Add Primary Key	ALTER	
Add Foreign Key	REFERENCES	ALTER
Drop Primary Key	ALTER REFERENCES [1]	ALTER
Drop Foreign Key	REFERENCES	ALTER
Deactivate Primary Key	ALTER REFERENCES [1]	ALTER
Deactivate Foreign Key	REFERENCES	ALTER
Activate Primary Key	ALTER REFERENCES [1]	REFERENCES
Activate Foreign Key	REFERENCES	REFERENCES[2]
Create Foreign Key	REFERENCES	not applicable
Create Primary Key	not applicable	not applicable

Notes:

1. The REFERENCES privilege is required only if the parent table has any dependents.
2. The ALTER privilege is required if the primary key has any foreign keys defined on it.

To add, drop, deactivate, activate, or create a unique constraint, you must have the ALTER privilege on the table. See the discussions of the CREATE TABLE and ALTER TABLE statements in the *DB2 Server for VSE & VM SQL Reference* manual.

Creating a Table That Contains a Primary Key

You can create a table with a primary key by using a PRIMARY KEY clause in the CREATE TABLE statement. This clause specifies the column that is the primary

key. For example, create a new table for students that contains a student first name, last name, and student number (the primary key) with the following statement:

```
create table students -
(firstname varchar(12)    not null, -
 lastname  varchar(15)    not null, -
 studentno char(6)        not null, -
 primary key (studentno))
```

The last line in the CREATE TABLE statement defines the STUDENTNO column as the primary key for this table. A column named as a primary key must have been defined with the NOT NULL option.

Adding a Primary Key to an Existing Table

It is not necessary to define the primary key in the CREATE TABLE statement. It can be added later using the ALTER TABLE statement. You can create the table first using:

```
create table students -
(firstname varchar(12)    not null, -
 lastname varchar(15)     not null, -
 studentno char(6)        not null)
```

Then add a primary key:

```
alter table students -
add primary key (studentno)
```

This makes the STUDENTNO column the primary key in the STUDENTS table if there are no duplicate values in that column. If duplicate values exist when you attempt to add a primary key on an existing column, the ALTER TABLE statement fails. If the column named for the primary key allows nulls, the statement also fails.

Creating a Table That Contains a Foreign Key

You can create a table with a foreign key by adding the FOREIGN KEY clause to the CREATE TABLE statement. This clause specifies the column that will be the foreign key and the table containing the primary key to be referenced. The parent table referenced must already exist and must have a primary key defined.

Create a table for a computer science class that contains a row for each student enrolled in the class and references the STUDENTS table as follows:

```
create table cs110 -
(studentno char(6) not null, -
 midterm   integer, -
 final     integer, -
 foreign key r_studt (studentno) references -
           students on delete cascade)
```

This creates a table where every row must represent a student who is listed in the STUDENT table. If a student is deleted from the STUDENT table, that student is also automatically deleted from this class list because of the delete cascade rule specified in the foreign key definition.

The referential constraint defined in the above example is `r_studt`. This name is used when the foreign key is deactivated, activated, or dropped.

Adding a Foreign Key to an Existing Table

Foreign keys can be added after a table has been created by using the ALTER TABLE statement. The dependent table CS110 can be created by creating the table first and without the foreign key:

```
create table cs110 -
(studentno char(6) not null, -
 midterm   integer, -
 final     integer)
```

You then add the foreign key using the following statement:

```
alter table cs110 add -
foreign key r_studt (studentno) references -
            students on delete cascade
```

When a foreign key is added in this way, all foreign key values currently in the table must match existing values in the primary key referenced, or the attempt to add a foreign key fails.

When creating referential constraints involving two tables that reference each other, at least one of the foreign keys must be added after the table has been created. It is impossible to reference a table (and its primary key) if that table has not been created. To create this type of structure, create one table with its primary key. Create the second table with its primary key and the foreign key referencing the first table. Then, add a foreign key to the first table which references the primary key in the second.

Activating and Deactivating Primary Keys, Foreign Keys, or Unique Constraints

The constraints placed on altering tables that contain primary or foreign keys, or unique constraints, can be suspended by deactivating the keys in the table. For example, the primary key in the STUDENTS table is deactivated with the following statement:

```
alter table students deactivate primary key
```

The above statement causes the restrictions on inserting, deleting, and updating to be suspended until the key is reactivated. No other users are allowed access to a table while it has an inactive key. In addition, keys that are related to an inactive key through a referential constraint are also considered inactive by the database manager. If the primary key in STUDENTS is deactivated, the foreign key in CS110 becomes inactive, and that table cannot be accessed.

To activate an inactive key, you must alter the table as follows:

```
alter table students activate primary key
```

If you make changes to the STUDENTS table while its primary key is inactive, the result of those changes cannot violate any of the constraints on the primary key, or of the referential constraint. If the changes produced any duplicate primary key values, dependent foreign key values without matching primary key values, or null primary key values, the primary key activation fails.

Foreign keys can be deactivated and then activated in the same way as primary keys by using an ALTER TABLE statement. When activating or deactivating a foreign key, the ALTER TABLE statement must include the name of the referential constraint. Deactivating the foreign key in the referential constraint *r_studt* for the CS110 table is accomplished by typing:

```
alter table cs110 deactivate foreign key r_studt
```

You can also use the ALTER TABLE statement to deactivate and activate unique constraints. As with a foreign key, you have to use the constraint name when you deactivate or activate it. To deactivate the unique constraint *empno* for the table TEACHERS, type:

```
alter table teachers deactivate unique empno
```

This statement causes the restrictions on inserting and updating to be suspended until the constraint is reactivated. For more information about adding a unique constraint to an existing table, or creating a table that contains a unique constraint, refer to the *DB2 Server for VSE & VM Database Administration* manual.

Determining Effects on Stored Format Information

Performing management tasks on your tables also affects any stored queries you have. The result of these tasks depends on the type of query stored and the type of changes made to the table.

If the table referred to by a stored query is changed by DROP TABLE, CREATE TABLE, or ALTER TABLE statements, the formatting information stored with that query may no longer be valid. For example, suppose a stored query performed grouping on columns 1 and 2 of a table. These two table columns are defined as VARCHAR.

The table is dropped and recreated with column 1 now defined as CHAR and column 2 again defined as VARCHAR. The formatting information saved in the stored query for column 1 is no longer valid. The formatting for the other columns is still valid if the data types defined for these columns is the same in the recreated table as in the original table.

In this last example, you get the same result if the stored query did not contain a GROUP BY clause.

Sharing Your Tables with Other Users

When you create a table, you become its owner. Only you and a person with DBA authority can use the table. However, you may want to share access to your tables.

You can give access to your data using an SQL GRANT statement, and you can take away access using the SQL REVOKE statement. See the *DB2 Server for VSE & VM SQL Reference* manual for more information.

Granting Privileges to Multiple Users

If you want to grant the identical capabilities to a *group* of users, you use a single GRANT statement. For example, assume Mona is part of a department responsible for maintaining the quarter information. Users Jim, Dan, and Dee also need update privileges on the QUARTER column of the EMPL table. You can extend the update capability to all three by typing:

```
grant update(quarter) -
on empl -
to jim,dan,dee
```

You can also grant privileges to the public within a list of grantees as illustrated below:

```
grant update(quarter) -
on empl -
to jim,public,dee
```

Using a View to Restrict Privileges to Certain Rows

Views can be used to restrict a privilege to specific rows of your tables. For example, assume Jim and Dee maintain quarter information for different groups of employees based on level of education. The following view supplies the information Jim needs for employees of education level 16 or less:

```
create view to16 -
as select empno,lastname,edlevel,birthdate,quarter -
   from empl -
   where edlevel <= 16
```

Similarly, the following view supplies all the information needed by Dee:

```
create view past16 -
as select empno,lastname,edlevel,birthdate,quarter -
   from empl -
   where edlevel > 16
```

The next step is to grant Jim and Dee privileges on the views. Type the following GRANT statements:

```
grant select,update(quarter) -
on to16 -
to jim

grant select,update(quarter) -
on past16 -
to dee
```

Using a View to Restrict Privileges to Certain Columns

Views can also be used to restrict the columns on which a user can type SELECT or INSERT statements. For example, you can give Jim and Dee the capability to select salary information from the EMPLOYEE table, but restrict them from viewing commission information simply by specifying the SALARY column and omitting the COMM column when you create the view. Create the view using:

```
create view blindempl -
as select salary -
from employee
```

Now give Jim and Dee table privileges:

```
grant select -
on blindempl -
to jim,dee
```

Creating Tables That You Want to Share

In an earlier section of this chapter, a private dbspace as a section of the database reserved for your tables was described. A private dbspace is suitable for your personal tables and is not really appropriate for tables that are to be shared.

Instead, a public dbspace should be used. When a table is inserted into a public dbspace, multiple users can update the table simultaneously. No two users, however, can use *the same row* at the same time.

To insert a table into a public dbspace, provide the database manager with the name of the dbspace using the IN clause of the CREATE TABLE statement. For example, the EMPL table can be created in a public dbspace named *SAMPLEDB* by typing:

```
create table empl -
(empno char(6), -
 lastname varchar(15), -
 edlevel smallint, -
 birthdate date, -
 quarter integer) -
in sampledb
```

When the dbspace name is not supplied on the CREATE TABLE statement, the table is created in one of your private dbspaces.

Accessing Tables Belonging to Other Users

When you refer to a table (or view) in an SQL statement, the database manager assumes you are referring to a table that you own. If you want to access a table that belongs to another user, you must identify the user and the table name. This is done by inserting the owner's authorization ID before the table name and separating the two with a period.

For example, you can access the system copy of the EMPLOYEE table (created during the installation of the database manager) by using the following query:

```
select * -
from sqldba.employee
```

This query retrieves the copy of the EMPLOYEE table that is owned by the sample user, SQLDBA. Your personal copy of the EMPLOYEE table is ignored. Similarly, other users have to use this method to access your EMPL table.

If an authorization ID does not begin with a letter, number, $, #, or @, you must enclose it in double quotation marks. For example:

```
select * -
from "%A23C".payroll
```

For detailed information about identifier naming conventions, see the *DB2 Server for VSE & VM SQL Reference* manual.

Using Synonyms

To avoid specifying an authorization ID for another user's table or view that you access frequently, you can create *synonyms* for their tables and views. For example, you can assign a synonym to be used in place of SQLDBA.EMPLOYEE by typing:

```
create synonym dbaemp -
for sqldba.employee
```

Now you can refer to SQLDBA.EMPLOYEE by the synonym DBAEMP. For example, if you type:

```
select * -
from dbaemp
```

The database manager displays all information from the EMPLOYEE table.

You can also use the CREATE SYNONYM statement to assign a synonym to one of your own tables. For example, if you are user JOHN, you can assign the synonym *ih* to your EMPL table by typing:

```
create synonym ih -
for john.empl
```

When you finish using the table or view for which you have defined the synonym, you should drop the synonym. To drop the synonym *ih*, for example, type:

```
drop synonym ih
```

The table or view on which the synonym was defined is not affected by this command.

Note: Because the performance of a DROP TABLE or DROP VIEW statement does not drop associated synonyms, you must drop the synonym(s) yourself.

Improving Query Performance

There are several ways you can improve data-access performance. This section describes three methods: indexing a table, updating statistics, and locking data.

Indexing a Table

The database manager uses an *index* to locate particular rows of a table.

Although you create these indexes, you do not use them directly. You simply enter your query and the database manager searches for, and attempts to use, an appropriate index to locate the information. The database manager finds the information whether an index exists or not, but it may find the information faster using an index.

A good table index is one that anticipates the kinds of queries to be used for the table.

For example, if you typically look for departments in the DEPARTMENT table by department name, create an index for the DEPTNAME column by typing the following SQL statement:

```
create index dptnme -
on department -
(deptname)
```

You can delete an index by using the DROP INDEX statement. For example, drop the DPTNME index by typing:

```
drop index dptnme
```

You can create a unique index when you create a table by using the CREATE TABLE statement, or you can add a unique index to an already existing table by using the ALTER TABLE statement. For more information about the UNIQUE attribute of these statements, see the *DB2 Server for VSE & VM SQL Reference* manual.

A unique index is also automatically created for each primary key and dropped automatically when the primary key is dropped. The primary key is dropped either by using an ALTER TABLE statement or by dropping the table or dbspace.

You cannot create a unique index on a VARCHAR or VARGRAPHIC column that has values that differ only by trailing blanks. Trailing blanks are ignored for values with these data types. Therefore, 'Adm ' is the same as 'Adm'.

If a VARCHAR or VARGRAPHIC column is not defined as unique, the trailing blanks on values do not affect the index. The order of 'Adm ' and 'Adm' is unpredictable because they only differ in trailing blanks.

Maintaining Updated Statistics

Another performance consideration concerns data statistics that are kept in the database manager catalogs. These statistics provide information about tables such as the number of rows in a particular table, or the number of different values contained in a particular column. This information is used by the database manager to determine the best method to satisfy query requests.

It may be important to keep these statistics up to date. For example, some tables are rarely updated, and their statistics change very little over the life of the table. Other tables, however, are updated frequently, and their statistics should be updated periodically.

You can update the statistics on a table by using an SQL UPDATE STATISTICS statement. For example, update the statistics on the ACTIVITY table by typing:

```
update statistics -
for table activity
```

Updating statistics involves a scan of both the rows and indexes of a table. This can be time-consuming for a large table; performing the update during off-peak hours is a good idea. As a general rule, try to develop a rule of thumb for determining when to update table statistics; for example, you might decide to update a table when it has changed by 20% or more. The full description of the UPDATE STATISTICS statement is shown in the *DB2 Server for VSE & VM SQL Reference* manual.

Locking Data

When you update or delete data in a table, the table is considered unstable because its data is changing. The database manager protects you and other users from obtaining unreliable query results by limiting access to unstable tables. It also prevents deadlocks when several users are trying to update the same data.

The database manager isolates the data by *locking* it. You can control the amount of data locked from other users, as well as the length of time that the lock is applied. Controlling the amount of locked data affects system performance. For example, a small amount of locked data requires less processing time than a large amount.

If you try to access an SQL object while another user is locking it, your processing is suspended until the other user has finished with the object. If you do not want to wait indefinitely, you can type the following to stop your transaction:

```
cancel
```

The database manager then rolls back any uncommitted work, and issues messages ARI7043I and ARI7040I. You can find more information about the CANCEL command under “CANCEL” on page 108.

Specifying the Isolation Level

You control locks by specifying the *isolation level*. The isolation-level setting affects only those tables stored in public dbspaces.

The isolation level you set is related to the task you are performing. The selection, insertion, updating, and deletion of table data are all affected by the isolation level. Specifically, the isolation level determines *how soon* data you have read can be changed by other users.

The isolation level has three settings: *repeatable read* (RR), *cursor stability* (CS), and *uncommitted read* (UR).

You can use the SET ISOLATION command only when the target AS is a local application server. Otherwise, the isolation level of CS is used and the SET ISOLATION command will have no effect.

Using the Repeatable Read Setting: Use the RR setting when you are modifying data. The RR setting ensures that data is completely isolated for your use. No other user can update the data until your work has completed.

Using the Cursor Stability Setting: Use the CS setting when you are simply querying (selecting) committed data. The term *cursor* in this case refers to the database manager cursor that points to the data in the table that you are using. The data involved in a CS setting is unlocked as soon as possible by the database manager for other users.

Using the Uncommitted Read Setting: Use the UR setting when you are querying (selecting) either committed or uncommitted data. With this setting, data can be read without waiting for other logical units of work that are updating the data and reading data will not prevent other application processes from updating it. However, you should remember that data integrity may be compromised with this setting and that UR should only be used when it is not necessary that the data you are reading be committed.

Using the SET Command

You use the SET command to control the isolation level. The SET command format is given on page 149, and an explanation of isolation level settings is found on page 154. See "Chapter 7. Creating and Using Routines" on page 73 for information on how to set the isolation level from a routine.

Note: Do not forget to change the isolation level back to the default isolation level when you have completed the operation for which you changed it.

Handling Lock Contention

Use an RR setting unless lock contention is a significant problem on the application server you are accessing. If lock contention is significant, use a CS setting whenever you can. The UR setting minimizes lock contention but it should only be used when it is not necessary that the data you are reading be committed.

Determining the Isolation-Level Setting

Table 6 and Table 7 on page 98 describe isolation level settings for various tasks. The first table discusses cursor stability, and the second, repeatable read. There is no table for the uncommitted read setting because this setting is not recommended for regular use.

Table 6. Guidelines for Using Cursor Stability

Activity	Description
Browsing a query result	When you are simply displaying data, the isolation level cursor stability is sufficient. At this time, do not plan to update the table or to print a formal report.

Table 6. Guidelines for Using Cursor Stability (continued)

Activity	Description
Preparing sample reports	While you are preparing a draft of a report, which you are using to check format and general content, use isolation level cursor stability.
Selecting or printing read only data	Read-only data resides in tables that are subject to controlled maintenance (insert, update, or delete). That is, any maintenance is done on a known periodic basis (for example, tables that are only updated overnight).
Browsing HELP information	ISQL HELP text information is read-only data, and isolation level cursor stability is sufficient.
Using SQL data definition operations	Read and update access to a catalog table during the performance of data definition statements (CREATE, ACQUIRE, GRANT) is always done with an isolation level setting of repeatable read. You do not have to set the isolation level to protect your definitions.
Using EXPLAIN	Using the EXPLAIN statement to access a catalog table is always performed with isolation level repeatable read, regardless of your isolation level setting.
Accessing data in private dbspaces	Accessing private dbspaces is effectively isolation level repeatable read, because locking is only performed at the dbspace level. You do not need to adjust your setting for isolation level.
Accessing data in public dbspaces with dbspace level locking	For public dbspaces with dbspace locking, access is always effectively isolation level repeatable read, because locking is only performed at the dbspace level. You need not change your isolation level setting.
Working with stored queries	Stored queries are always stored and recalled with an isolation level repeatable read. You control the isolation level used for starting a stored query. Therefore, you must set the isolation level to the desired value before starting the stored query.
Invoking routines	Retrieving from a routine is performed with an isolation level repeatable read. You control the isolation level used for running a routine. You can change the isolation level setting before running the routine or within the routine (as many times as needed).
Using ISQL commands	For ISQL commands that access the application server, the isolation level used is repeatable read. For the ISQL commands RUN, START, and HELP, you can control the setting of the isolation level.
Using the INSERT statement and the INPUT command	Using INSERT statements with values and INPUT commands are not affected by the isolation level setting because no data is read from the application server. Using INSERT statements with subselect are affected by the isolation level setting because of the SELECT clause. You must follow the guidelines for selecting data given above when you use INSERT through subselect statements.

Table 7. Guidelines for Using Repeatable Read

Activity	Description
Using DELETE, INSERT, or UPDATE from a display	If you type a DELETE, INSERT, or UPDATE statement based on a SELECT statement result, do not set the isolation level to repeatable read before you type the SELECT statement. It is then impossible for the displayed data to change before you have typed the DELETE, INSERT, or UPDATE statement. **Note:** In addition, set AUTOCOMMIT OFF so the SELECT and data change statements are contained in the same LUW. You need not set AUTOCOMMIT OFF and isolation level repeatable read if any of the following is true: • Data selected is read-only. • You are the only person authorized to modify the selected data. • You have other ways of ensuring the data selected does not change. • A command that changes the contents of the table is valid even if the selected data does change.
Using DELETE or UPDATE statements	Use an isolation level repeatable read when you delete or update data unless: • You are the only person authorized to modify the data. • You have other ways of ensuring the data selected does not change.
Preparing formal reports	To prevent data from changing, set the isolation level to repeatable read when you prepare a formal report unless: • Data selected is read-only. • You are the only person authorized to modify the selected data. • You have other ways of ensuring the data selected does not change.

Chapter 9. Using VM Functions

In this chapter, you progress beyond ISQL and learn some VM commands and functions that can enhance your use of the database manager. Although this chapter does not show you how to use VM, it does identify many of the VM features that help you get the most out of ISQL and this RDBMS.

The commands described in this chapter are neither SQL statements nor ISQL commands; they are CMS and CP commands that can be used to control the characteristics of your virtual machine, direct printed output, and provide additional VM facilities. As CMS and CP commands, they can only be used in CMS or CMS subset mode.

CMS-Subset Processing

During your ISQL session, you can enter CMS or CP commands without terminating your ISQL session. When finished you can return to ISQL.

To enter CMS-subset mode, type:

```
cms
```

You can now type CMS or CP commands, EXEC procedures, or user programs.

Note: Do not enter any EXEC procedure or program that accesses the application server while you are in CMS subset mode.

Returning to ISQL from CMS Subset Mode

When you finish entering your CMS and CP commands and are ready to return to ISQL, type:

```
return
```

Entering CP Commands

You can enter CP commands in three ways:

- A CP command can be typed at any time during ISQL processing by prefixing the command with *#CP*. This procedure immediately interrupts the current ISQL processing, enters CMS subset mode and performs the designated CP command, and then returns to ISQL when the command is complete.
- You can press PA1 at any time. `CP READ` appears in the status area and you can type any CP commands.

  If you are running with CP SET RUN ON, control automatically returns to ISQL after each CP command is executed. ISQL resumes processing from the point of interruption. If you are running with CP SET RUN OFF, you must type a B or begin on the input line and press ENTER to resume processing in ISQL.

  The SET FULLSCREEN ON command cancels the interrupt action of PA1. Suspending the full-screen option or turning it off does not reset PA1 to the original interrupt setting. If PA1 does not interrupt ISQL, you can reestablish it as an interrupt key with the following command:

  ```
  #cp terminal brkkey pa1
  ```

- While in CMS subset mode, you can type CP commands (with or without the CP prefix) in the same manner as that described above for CMS commands.

As a DB2 Server for VM user, you probably use CP commands only when you want to change printer spooling or routing characteristics, or change PF key definitions.

Obtaining Printed Reports on a Workstation Printer

Your printed output is automatically sent to the printer designated by your site. You can change or redirect your printed output to another printer and specify other print characteristics by using the CP commands TAG and SPOOL. The TAG command identifies the receiving destination; the SPOOL command directs the printed output to the network machine.

First, start the statement stored as DEPT in "Chapter 6. Storing SQL Statements" on page 67 by typing:

```
start dept
```

Suppose that you want to print this information on a remote printer. You have to know the node ID for that printer and the user ID of the network machine. You can use the CMS IDENTIFY command for information about network identifiers (NETIDs). For this example, assume that you want to send your output to a 3262-13 printer with a destination ID of RMT3262. Enter CMS subset mode by typing the following:

```
cms
```

Then type the following CMS command:

```
identify
```

The results of this command give you the necessary information to use the SPOOL and TAG commands. The format of the output produced by typing this command is:

```
user AT nodeid VIA netid
```

Now you would type the CP SPOOL and TAG commands to route your printed output to the 3262 printer:

```
#cp spool printer to netid nohold
#cp tag dev printer RMT3262 system
```

To return to ISQL and print your report, type:

```
return
print
```

Specifying the Number of Copies of Printed Reports

You can use the CP SPOOL command to specify the number of copies for *all* reports you print during the current terminal session.

For example, you can specify three copies by modifying the previous CP SPOOL command example as follows:

```
#cp spool printer to netid copy 3
```

All following PRINT commands use the quantity specified by the most recently typed CP SPOOL command or ISQL PRINT command with the COPIES keyword.

Note: This section assumes your site is running with the Remote Spooling Communications Subsystem (RSCS) Networking Program Product (Version 3 Release 2 or later).

Using EXEC Files

An EXEC is a file with a file type of EXEC. It contains a series of commands and statements that are executed when the file name of the EXEC file is typed.

You can use EXECs to stack SQL statements and ISQL commands before you begin your ISQL session. As soon as you start ISQL, this information is read from the stack and executed. By using EXEC processing in this way, you can predefine all default settings to be used during your display session, establish PF key values, and designate print specifications. This method can also be used to automatically start procedures tailored to the needs of specific users.

The EXEC examples in this chapter do not include the CONNECT statement as part of the stacked set; it is assumed that you are working with your own sample tables. If you must gain access to another user ID to use ISQL, use a CONNECT statement as the first stacked statement. See the *DB2 Server for VSE & VM SQL Reference* manual for additional information on the CONNECT statement.

For information on creating CMS EXEC files, see the *VM/ESA: CMS User's Guide* manual. For information on the REXX language, see the *VM/ESA REXX/VM User's Guide* and the *VM/ESA REXX/VM Reference* manuals.

Stacking Commands in an EXEC File

In the CMS environment, you can run EXECs that stack SQL statements and ISQL commands to be processed automatically as soon as ISQL begins. If the EXEC also starts ISQL, the user of the EXEC need not know anything about ISQL.

You can write specialized EXECs for your own use or for other users. Figure 47 shows an EXEC that is written in the REXX language:

```
/* this exec develops a report for project mean numbers */
   Queue 'SELECT * FROM PROJ_ACT ORDER BY PROJNO'
   Queue 'FORMAT GROUP PROJNO'
   Queue 'FORMAT SUBTOTAL ACSTAFF'
   Queue "FORMAT TTITLE 'PROJECT MEAN EMPLOYEES'"
   Queue 'PRINT'
   Queue 'END'
   Queue 'EXIT'
'EXEC ISQL'
Exit   /* end of exec */
```

Figure 47. Example of Stacked ISQL Commands in a REXX EXEC

Prompting by Using an EXEC File

To create a prompt environment for a casual SQL user, you can write an EXEC like the one shown in Figure 48 on page 102. The user would run the EXEC from the CMS environment without starting ISQL.

```
/* This exec prompts for a table to be viewed or printed */
Trace Value 'OFF'
Say 'WHICH TABLE WOULD YOU LIKE TO SEE?'
Parse Upper Pull tablename
Say 'WOULD YOU LIKE TO HAVE THIS TABLE PRINTED? (Y OR N)'
Parse Upper Pull printoption
Say 'YOU MUST ENTER END TO LEAVE THE DISPLAY OF THE TABLE'
If printoption = 'Y' then Do
   Queue 'SELECT * FROM' tablename
   Queue 'DISPLAY'
   Queue 'PRINT'
   Queue 'END'
End
Else If printoption = 'N' then Do
   Queue 'SELECT * FROM' tablename
   Queue 'DISPLAY'
   Queue 'END'
End
If printoption = 'Y' | printoption = 'N' then Do
   Queue 'EXIT'
   'EXEC ISQL'
End
Exit /* End of exec */
```

Figure 48. Example of an EXEC That Prompts the User

When you run this EXEC, the following user/system dialog occurs. If the EXEC has a file name of MYTABLES, you type:

```
mytables
```

The system prompts you with the message `WHICH TABLE WOULD YOU LIKE TO SEE?`. You type:

```
department
```

The system prompts you with the message `WOULD YOU LIKE TO HAVE THIS TABLE PRINTED? (Y OR N)`. You type:

```
y
```

The system prompts you with the message `YOU MUST ENTER END TO LEAVE THE DISPLAY OF THE TABLE`.

The ISQL startup messages immediately follow. Then the partial display in Figure 49 should appear.

```
DEPTNO  DEPTNAME              MGRNO   ADMRDEPT
------  --------------------  ------  --------
A00     SPIFFY COMPUTER SERV< 000010  A00
B01     PLANNING              000020  A00
C01     INFORMATION CENTER    000030  A00
D01     DEVELOPMENT CENTER    ?       A00
D11     MANUFACTURING SYSTEM< 000060  D01
D21     ADMINISTRATION SYSTE< 000070  D01
E01     SUPPORT SERVICES      000050  A00
E11     OPERATIONS            000090  E01
E21     SOFTWARE SUPPORT      000100  E01
* End of Result *** 9 Rows Displayed ***Cost Estimate is 1**********************
```

Figure 49. Query Result from an EXEC

You see the ISQL signoff messages when you end the display.

Starting ISQL from a Terminal

Because ISQL is a full-screen interactive program, it must be started from a display terminal. When a user is automatically logged on using the CP AUTOLOG command, or when a user tries to start ISQL while disconnected, ISQL is not started. The user must be logged on to a display terminal to start ISQL.

Disconnecting after Starting ISQL

Although the restriction of a logged-on display terminal applies to the starting of ISQL, it does not apply if a user disconnects *after* starting ISQL. For example, if you have an EXEC that starts ISQL and then executes a number of ISQL SELECT statements, you can disconnect your display terminal after ISQL is started and the remainder of your SELECT statements are executed.

Avoid using commands and statements that result in the issuance of ISQL decision-type messages. For example, stacking a SET AUTOCOMMIT OFF command causes the following message to be issued:

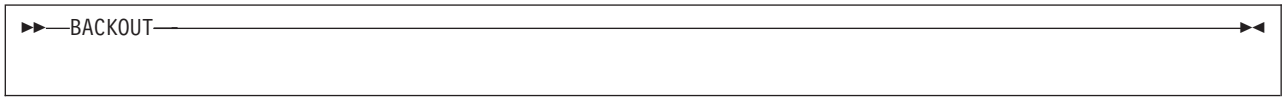
```
ARI7602D  You are in a logical unit of work. Type COMMIT to
          have a COMMIT issued for you or ROLLBACK to
          have a ROLLBACK issued for you.
```

Your response to this message cannot be accepted from the command stack because of the interactive design of ISQL. You must type the response yourself.

Chapter 10. ISQL Commands

This chapter contains syntax diagrams, semantic descriptions, rules, and situations where you would use ISQL or operator commands. The commands are organized alphabetically.

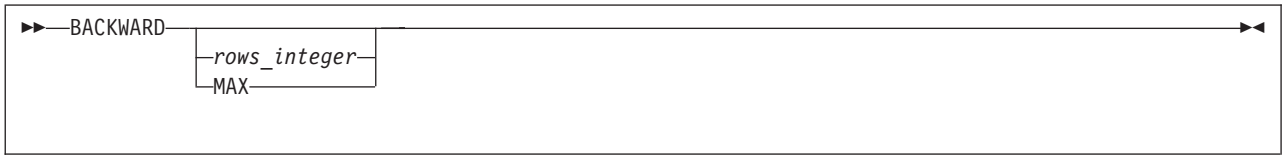
BACKOUT



BACKOUT is an ISQL command that is meaningful only while you are using the INPUT command with AUTOCOMMIT on. It nullifies all changes made since the last SAVE command. If no SAVE command has been issued, all changes are nullified since the start of the INPUT command.

With AUTOCOMMIT off, BACKOUT has no effect. You must enter a ROLLBACK statement to nullify all input data. This also nullifies all changes made during the current logical unit of work. You must end the INPUT command (by using the END command) before issuing the ROLLBACK statement.

BACKWARD



This ISQL display command displays the rows above or before those currently being displayed.

rows_integer
is a number that indicates the number of rows backward that you want to move the display.

MAX
moves the display to the beginning of the query result.

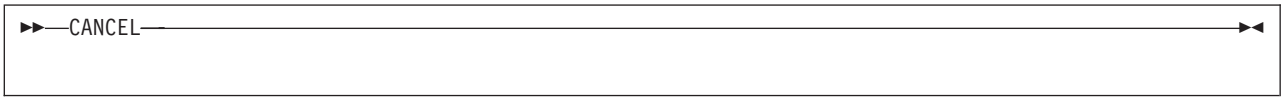
If neither *rows_integer* nor MAX is specified, the display moves backward half a display.

Moving backward through the query result is limited: you can go backward to a limit of one full display from the last FORWARD command, or you can return to the beginning of the query result by issuing BACKWARD MAX.

Scrolling backward a half display at a time can be done by pressing PF7 (or PF19). Because you can reset your PF keys, you can assign the BACKWARD function to any PF key. The default function key is PF7 (or PF19).

Note: You cannot receive a prompt response when you enter a BACKWARD MAX command, because the query must be reexecuted to move the display back to the first display.

CANCEL



CANCEL is an ISQL command that you can use to cancel a command, an SQL statement, or logical unit of work in progress. CANCEL can be typed anytime you are entering data, commands, or statements.

DB2 Server for VSE

Canceling a command or statement also causes a ROLLBACK to be processed.

DB2 Server for VM

Canceling a command or statement also causes a ROLLBACK RELEASE to be processed. If you have previously issued an explicit CONNECT command, you must reconnect to the database manager.

If AUTOCOMMIT is on and the previous statement was an SQL INSERT, UPDATE, or DELETE statement that affected more than one row, all changes made by the previous statement are rolled back.

If AUTOCOMMIT is off, all work since the last COMMIT, or since the beginning of the logical unit of work performed by the following, is rolled back:
- SQL statements
- ISQL commands that cause changes to table data, stored SQL statements, or routines.

DB2 Server for VSE

Except for long-running SQL statements, to start a cancel operation, type:

```
cancel
```

DB2 Server for VM

To start a cancel operation, type:

```
cancel
```

When canceling a command with AUTOCOMMIT off, you are prompted to verify if a CANCEL should be performed. If you type NO, the CANCEL processing is not performed. If you answer YES, all changes made to the data since the last COMMIT, or since the start of the logical unit of work if there was no COMMIT, are rolled back.

When you cancel an ISQL INPUT command with AUTOCOMMIT on, all changes since the last SAVE command, or since the start of the INPUT command if there was no SAVE command, are rolled back.

DB2 Server for VSE

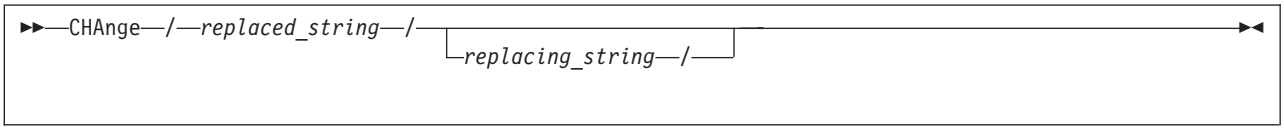
Long-running SQL statements are those for which message ARI7044I is issued to tell you that the terminal is free. These statements are cancelled by pressing CLEAR and typing the CANCEL command prefixed by the ISQL transaction identifier, ISQL. (If your location has redefined the ISQL transaction identifier, prefix the CANCEL command with the transaction identifier that your location has defined.)

The command looks like:

```
isql cancel
```

When canceling long-running SQL statements, a ROLLBACK is always performed.

CHANGE



CHANGE is an ISQL command that modifies the current SQL statement in the SQL command buffer and displays the results. If data in the SELECT or FROM clauses of a SELECT statement is changed, associated formatting information for that statement is erased. However, if the changed information is contained in the WHERE, GROUP BY, ORDER BY, or HAVING clauses, associated formatting information for the statement is saved.

/
: is any non-blank character that identifies the beginning and end of a string. The slash is a good choice unless you are changing data that contains a slash. This character must be separated from the command name by at least one blank and must not occur in either string.

replaced_string
: is the characters to be replaced in the current SQL statement. The string can contain DBCS characters.

replacing_string
: is the characters to replace the first occurrence of the characters in *replaced_string*. If *replacing_string* is omitted, the first occurrence of replaced_string is deleted. The string can contain DBCS characters.

Example

If the current SQL statement is:

```
select * from activity
```

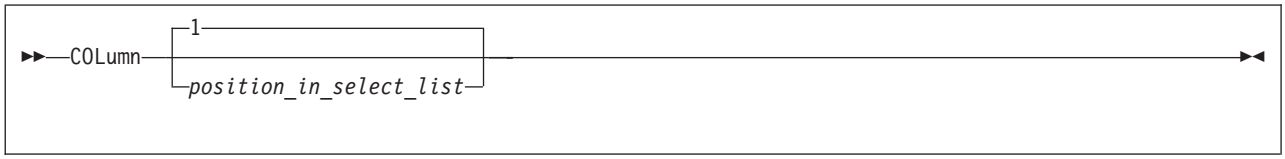
and you type the following ISQL CHANGE command:

```
change /*/actno/
```

the result in the SQL command buffer is:

```
select actno from activity
```

COLUMN



COLUMN is an ISQL display command that displays the query result to be formatted so that it begins with the specified column at the left edge of the display.

blank
causes column 1, or the first displayable column if column 1 is being excluded, to be displayed at the left edge of the display.

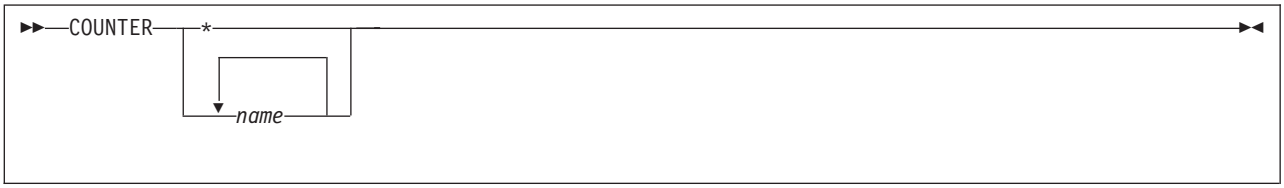
position_in_select_list
is the number of the specified column.

The number refers to the columns provided by the SELECT statement, not those currently displayed. You must choose a number that represents the desired column's position in the SELECT clause of the SELECT statement that provided the query result.

If the value of *position_in_select_list* represents a column being excluded, the display starts at the next column to the right that is not being excluded. If there are no more displayable columns to the right, a blank display is the result.

Specifying a value greater than the number of columns in the query result displays the last column or a blank display if the last column is being excluded.

COUNTER



COUNTER is an operator command that is used primarily to monitor system activity. It is not allowed during a logical unit of work. Before typing COUNTER, you must end any logical unit of work that is in progress.

The SHOW command can only be used when the target application server is a local application server. It cannot be used when the application server is a remote application server.

* specifies that all counters are to be displayed.

name
is the name of the counters to be specified. Valid names are:

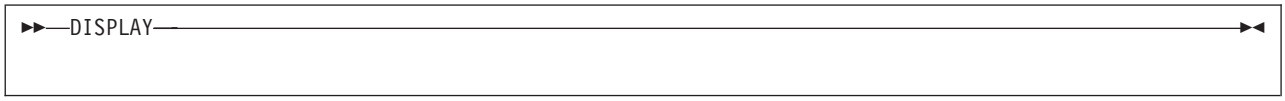
```
BEGINLUW     DBSSCALL     LDIRBUFF     PAGEREAD
CHKPOINT     DEADLCK      LOCKLMT      PAGWRITE
DASDIO       DIRREAD      LOGREAD      RDSCALL
DASDREAD     DIRWRITE     LOGWRITE     ROLLBACK
DASDWRIT     ESCALATE     LPAGBUFF     WAITLOCK
```

For an explanation of these counters, see the *DB2 Server for VSE & VM Operation* manual.

The COUNTER command results in one or more displays. To proceed to the next display, press CLEAR. It is not possible to move backward. You must first type the END command, and then retype the COUNTER command. To exit from the COUNTER command, type END. The COUNTER command is not available on a non-DB2 Server for VSE & VM application server or if you are using DRDA protocol.

Note: If the national language of the application server differs from the national language that the user set for the ISQL session, the messages generated by this operator command are issued in the national language of the application server.

DISPLAY



DISPLAY is an ISQL display command that can *only be used in a routine*. When encountered, DISPLAY causes its associated SELECT statement results to be displayed at your display terminal.

Any ISQL display commands placed in the routine after the associated SELECT statement and before the DISPLAY command affect the display at the terminal. You can, in addition, format the display or print the query result by typing ISQL display commands from the keyboard after the query result is displayed, or from the routine after the display is ended and before the END statement that is associated with the SELECT statement whose results are being displayed. To end the display of the query result initiated by the DISPLAY command and to return to the routine, type the ISQL END command from the keyboard.

Example

In the following routine, you select all the rows from the DEPARTMENT table, specify the format commands to be performed on the rows, and display the formatted results on the display terminal. You must type `end` to end the display.

```
select * from department
format separator ' | '
format ttitle 'departments'
display
end
```

To get the same result as the above routine, you can type these lines in a routine:

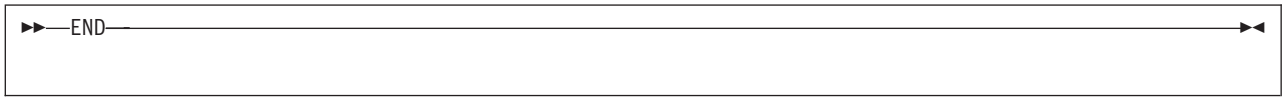
```
select * from department
display
end
```

to see the DEPARTMENT table. Next, type these commands from the keyboard:

```
format separator ' | '
format ttitle 'departments'
```

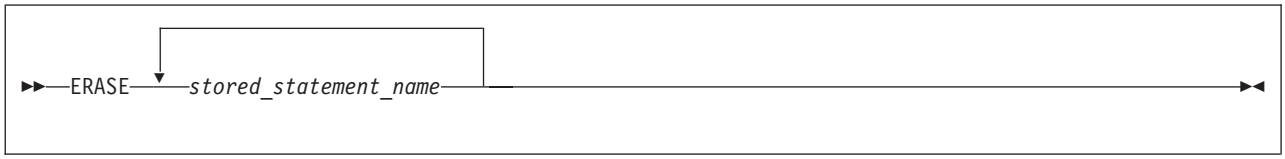
and you see the DEPARTMENT table with column separators and a title just like the first example.

END



END is an ISQL command that ends the display of a query result, an operator command, a SELECT statement in a routine, or an ISQL INPUT command. Query results, operator commands, or the INPUT command can also be ended by pressing the PF3 key (or PF15 key). Because you can reset your PF keys, you can assign the END function to any PF key. The default function key is PF3 (or PF15).

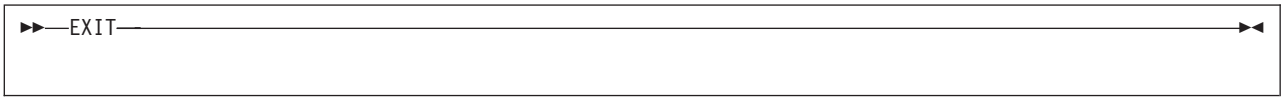
ERASE



ERASE is an ISQL command that erases one or more stored SQL statements.

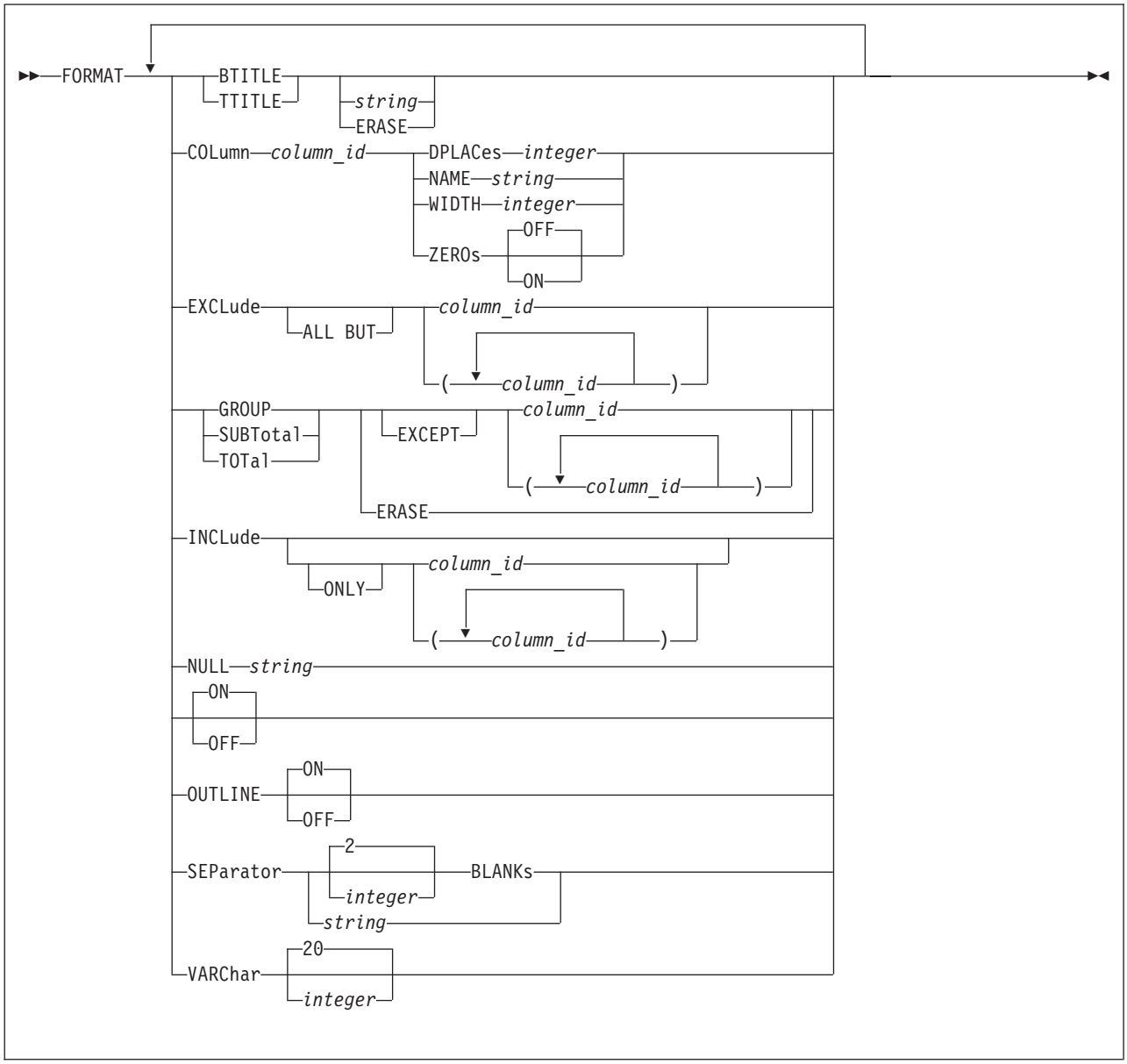
stored_statement_name
is the name of the stored SQL statement.

EXIT



EXIT is an ISQL command that ends the current ISQL session. If AUTOCOMMIT is on, the session ends immediately. If AUTOCOMMIT is off, you are prompted for a response so that the current logical unit of work can be committed or rolled back before you exit the session.

FORMAT



FORMAT is an ISQL display command that controls the format of the query result currently being displayed. Modifications made to the display format by a FORMAT command are reflected on the current display and the printed output obtained from any subsequent PRINT command.

You can perform formatting only on the first 45 columns of a query result.

Any formatting command containing the keywords EXCLUDE, GROUP, INCLUDE, ON/OFF, OUTLINE, SUBTOTAL, or TOTAL, causes the query to be reexecuted.

Note: You can use DBCS characters in strings or column lists.

BTITle
specifies the bottom title to be printed on reports. This bottom title is centered on the bottom line of the report. Unless specified, no bottom title is printed.

string
is the characters to use for the bottom title on a subsequent PRINT command.

The maximum length of the bottom title is that which fits on a line of the page (up to a maximum of 100 characters). If the bottom title exceeds the maximum, only those characters that fit on the line are printed.

Enclose the title in *single* quotation marks if it contains any blanks, but do not include any single quotation marks in the title itself.

ERASE
causes the current bottom title to be deleted, resulting in no bottom title.

If FORMAT BTITLE is issued without either ERASE or a character string, the current bottom title is displayed if it does not contain DBCS data. If the title *does* contain DBCS data, message `ARI7970I` is issued.

TTITle
specifies the top title to be printed on reports. This top title is centered on the top line of the report between the date and page number. If no top title is specified, the first 100 characters of the SELECT statement are used for the top title on printed reports.

string
is the characters to use for the top title on a subsequent PRINT command.

The maximum length of the top title is that which fits on the top line between the date and page number (to a maximum of 100 characters). If the top title exceeds the maximum, only those characters that fit between the date and page number are printed.

Enclose the title in *single* quotation marks if it contains any blanks, but do not include any single quotation marks in the title itself.

ERASE
causes the current top title to be deleted and the default to be used. The default TTITLE is the first 100 characters of the SELECT statement.

If FORMAT TTITLE is issued without ERASE or a character string, the current top title is displayed if it does not contain DBCS data. If the title *does* contain DBCS data, message `ARI7970I` is issued.

COLumn
provides display formatting for a particular column.

column_id
specifies the column to be formatted.

If a number *n* is specified, it identifies the column to be formatted as the nth column of the query result. The number refers to the position of the columns provided by the SELECT statement, not those currently displayed. Therefore, you must choose a number that represents the position of the desired column in the SELECT clause of the SELECT statement that provided the query result.

If a number is not specified, *column_id* refers to the current column heading of the column to be formatted. Enclose the name in single quotation marks if it either contains a blank or refers to a column with a solely numeric heading.

If the *column_id* represents a column being excluded, the formatting specified is performed on the excluded column although you cannot see the formatting until the column is included.

The following keywords describe how the current display is to be formatted for the column specified by *column_id*:

DPLACes
specifies the number of decimal places (*integer*) to be displayed for a numeric field. Rounding is *not* performed. The DPLACes must be less than the column width.

NAME
specifies a column heading to be used for the display.

string
is the actual column heading to be displayed.

A maximum of 30 characters can be used for the column heading. All characters except single quotation marks are valid. Enclose the column heading in single quotation marks if it contains a blank, but do not include any single quotation marks in the column heading itself.

WIDth
specifies the display length attribute (*integer*) of the column.

integer
For character type columns, the leftmost *integer* characters are displayed.

For numeric type columns, the leftmost *integer* significant digits counting the sign and decimal marker, if any, are displayed. The sign character for a positive number is a blank.

For VARCHAR type columns, characters up to the current setting of VARCHAR (using a SET or FORMAT command) are displayed. For GRAPHIC type columns, integer represents the number of DBCS characters. Two bytes are reserved for the SO/SI characters.

ZEROs
specifies whether leading zeros for numeric columns are displayed (ON) or not (OFF).

EXCLude
specifies columns to be excluded (omitted) from the display. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result.

ALL BUT
indicates that all columns except those specified are excluded. For example:

```
format exclude all but (1 job 4)
```

includes the JOB column, and columns 1 and 4, but excludes all other columns.

column_id
specifies the column to be formatted.

If a number is specified, it identifies the column to be formatted as the nth column of the query result. The number refers to the position of the columns provided by the SELECT statement, not those currently being displayed. Therefore, you must choose a number that represents the position of the desired column in the SELECT clause of the SELECT statement that provided the query result.

If a number is not specified, *column_id* refers to the current column heading of the column to be formatted. Enclose the name in single quotation marks if it contains a blank or refers to a column with a solely numeric heading.

When more than one column is specified, separate the column_ids with a blank and enclose them in parentheses.

For example, the command:

```
format exclude (3 5)
```

causes the third and fifth column of the original position in the query result to be excluded from the display. The command:

```
format exclude job
```

prevents the JOB column from being displayed during the current query result. If JOB is selected two or more times, only the first occurrence of the JOB column is excluded.

GROUP
specifies the columns to outline (when outlining is on) and the columns to use for determining when subtotals are taken. Subtotals are taken whenever the values change in the columns specified. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result.

SUBTotal
specifies the columns in which subtotals are to be calculated. Subtotals are taken whenever the values change in the columns being grouped. A (final) total is also provided for the columns unless otherwise specified by a FORMAT TOTAL command. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result.

TOTal
specifies the columns in which (final) totals are to be calculated. If not specified, totals are provided for all columns being subtotaled. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result.

EXCEPT
specifies the columns to be grouped or totalled or subtotaled except those specified by *column_id*.

column_id
is the name or position of each column in the query result to be grouped for outlining, subtotals, or totals.

When used with the SUBTOTAL and TOTAL keywords, column_id specifies the columns on which subtotals or totals are to be calculated.

If a number is specified, it identifies the column to be formatted as the nth column of the query result. The number refers to the position of the columns provided by the SELECT statement, not those currently displayed. Therefore, you must choose a number that represents the position of the desired column in the SELECT clause of the SELECT statement that provided the query result.

If a number is not specified, *column_id* refers to the current column heading of the column to be formatted. Enclose the name in single quotation marks if it contains a blank or refers to a column with a solely numeric heading.

When specifying more than one column, separate the column_ids with a blank and enclose them in parentheses.

If the column represents one that you are excluding, ISQL groups the columns using that excluded column, although you do not see the excluded column.

Subtotals or totals are calculated on the excluded column. You cannot see the subtotals or totals until the column is included. When arithmetic errors occur, the value of the column in error is calculated as zero.

The usual use of GROUP is to first order the rows of the columns that you want grouped. Use an ORDER BY clause in the SELECT statement issued to obtain the query result. The left to right ordering of the columns themselves depends on the order in which they appear in the SELECT clause of the SELECT statement.

Rows containing arithmetic errors from an outer select are displayed together at the end of the list, followed by rows containing NULL values. Rows containing arithmetic errors are displayed as asterisks (*).

ERASE
deletes all the previous specifications of GROUP for the current query result. When used with SUBTOTAL, ERASE suspends subtotals. When used with TOTAL, ERASE suspends totals.

INCLude
reverses the effect of a previous FORMAT EXCLUDE command. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result.

ONLY
indicates that only those columns specified are displayed; all others are excluded.

Note: Do not use ONLY for a column name with an INCLUDE keyword. In this case, use its column number.

column_id
specifies the column to be formatted.

If a number *n* is specified, it identifies the column to be formatted as the nth column of the query result. The number refers to the position of the

columns provided by the SELECT statement, not those currently being displayed. You must choose a number that represents the position of the desired column in the SELECT clause of the SELECT statement that provided the query result.

If a number is not specified, *column_id* refers to the current column heading of the column to be formatted. Enclose the name in single quotation marks if it contains a blank, or if it refers to a column with a solely numeric heading.

When more than one column is specified, separate the column IDs with a blank and enclose them in parentheses.

When arithmetic errors occur, the value of the column in error is calculated as zero.

The columns that are not mentioned in the INCLUDE command, and that are not currently being excluded, continue to participate in the display. If INCLUDE is issued by itself and with no options, all excluded columns are restored.

NULL
specifies the characters to display for null fields.

string
specifies the actual characters to display (up to a maximum of 20). If blanks are included, you must enclose the string in single quotation marks, but do not use single quotation marks in the string itself.

For example, the following command,

```
format null empty
```

causes the word EMPTY to be displayed for all null fields.

A question mark (?) is displayed for null values unless otherwise specified with a FORMAT or SET command.

ON

OFF
controls the status of outlining, subtotals, and totals on query results. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result. Until OFF is specified, outlining, subtotals, and totals are active.

ON
permits outlining, subtotals, and totals. The ON status stays in effect for the current query result until you type FORMAT OFF.

OFF
suspends outlining, subtotals, and totals.

OUTLINE
controls the outline report format for columns specified with FORMAT GROUP. When SQL processes a FORMAT command containing this keyword, it reexecutes the query and repositions the display to the top of the query result.

If OUTLINE is not specified, outlining is performed whenever GROUP is specified unless you type FORMAT OFF.

ON
specifies that successive duplicate values in grouped columns are repeated only when they are the first line at the top of the display or at the beginning of each page on printed reports. If the first line is a subtotal line or a blank line between groups, successive duplicate values are not displayed or printed at the beginning of the next group.

OFF
specifies that successive duplicate values in grouped columns are to be displayed wherever they occur.

SEParator *integer* **BLANKs**
specifies the number of spaces (*integer*) to be displayed between columns. The maximum number of blanks that can be specified is 254. Unless otherwise specified with a FORMAT or SET command, the separation between columns consists of two blanks.

SEParator *string*
specifies the characters to be displayed between columns. If blanks are included, the string must be enclosed in single quotation marks, but do not include any single quotation marks in the separator itself.

For example, if you want a vertical line between the columns, you type:

```
format separator ' | '
```

which places a blank, a vertical bar, and a blank between all columns. The maximum number of characters that can be used for a separator is 254.

VARChar
specifies the display width of variable length columns.

integer
is the length desired up to 254. Unless otherwise specified with a SET command, ISQL displays only the first 20 characters of a variable-length column.

The SET command value for VARCHAR columns can be overridden for a particular query by specifying the desired value with this keyword on the FORMAT command.

When you type a FORMAT VARCHAR, the SELECT statement is reissued and you are returned to the beginning of the query result.

Example

The following FORMAT commands used during a query exclude the first column, change the name of the PRSTAFF column heading to ESTMEAN, display all projected mean staff numbers in this column with three decimal places, and display leading zeros:

```
format exclude 1
format column prstaff name estmean
format column estmean dplaces 3
format column estmean zeros on
```

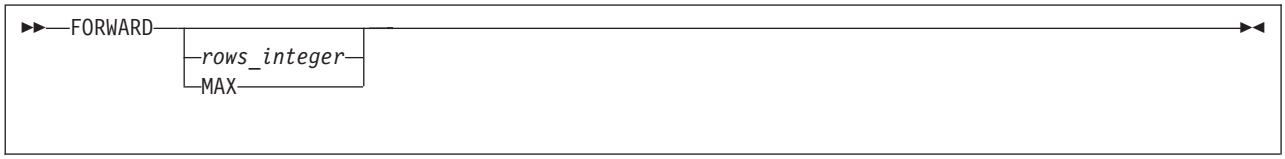
Note: If you rename a column heading, further FORMAT commands that refer to that column by name must use the new name. When referring to the column by a number, you must specify its original position in the SELECT clause of the SELECT statement.

You can specify more than one keyword in a FORMAT command. For example, the FORMAT commands described above can be expressed with a single command:

```
format exclude 1 column prstaff name estmean dplaces 3 zeros on
```

By using more than one keyword in a single FORMAT command, you can reduce the amount of data you must type and improve the performance of ISQL.

FORWARD



FORWARD is an ISQL display command that lets you move your display forward through a query result.

rows_integer
is the number of rows you want to move forward.

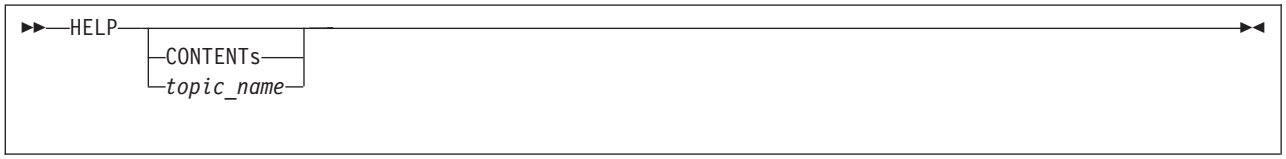
MAX
causes the last display of the query result to be displayed along with an indication of the number of rows in the query result and the Query Cost Estimate (QCE) message. For query results that contain many rows, a FORWARD MAX command can take a long time.

If nothing is specified, the display moves forward one-half of a display.

Scroll forward through the query result one-half of a display at a time by typing FORWARD. (You can also press PF8 or PF20.) Because you can reset your PF keys, you can assign the FORWARD function to another PF key.

You can activate FORWARD by pressing ENTER if it is the first display command to be issued for a query result. The display moves forward one *entire* display each time you press ENTER.

HELP



This ISQL command retrieves online HELP information. This online HELP information is for ISQL users who need quick *reference* information at their display for:

SQL statements
ISQL commands
Messages, codes, and SQLSTATEs.

Note: The online HELP information is not serviced by the IBM Support Center. The information is extracted from this book, and from the *DB2 Server for VSE Messages and Codes* and *DB2 Server for VM Messages and Codes* manuals. Use the readers' comment form in the back of these books to express concerns and comments on this information.

You can view this information in the language of your choice if your site chose to install different language versions of the online HELP information. See the SET command later in this chapter for more information on setting the language of your choice.

Online HELP information is stored as a table. After you retrieve a topic, you can use any of the ISQL display commands, or PF keys that provide display commands, to assist in viewing the text. You can also type SQL statements or ISQL commands at this time. You can format retrieved topics with the ISQL FORMAT command and print them with the ISQL PRINT command. A top title is provided for printed topics.

CONTENTs
displays the available online topics and the correct names to use in retrieving specific topics.

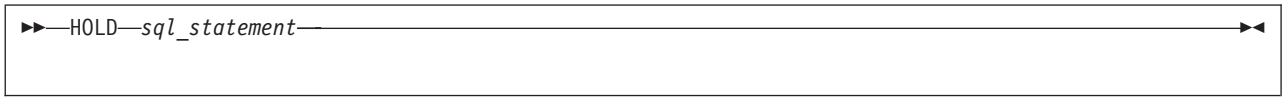
topic_name
is the name of the topic to be retrieved. It can be one or more words and can be placed within quotation marks. Most *topic_names* are either a statement name (such as *SELECT* or *INSERT*), a message number (such as *ARI7399I* or *ARI7307A*), a message code (such as *-205* or *100*), or an SQLSTATE (such as *SQLSTATE 01512*). For example, to retrieve online HELP information for the UPDATE statement, type:

```
help update
```

If the HELP command is issued with no parameters, a description of how to use the HELP command is returned along with a list of the available topics. The HELP command without additional parameters can be invoked by pressing PF1 (or PF13). Because you can reset your PF keys, you can assign the HELP function to any PF key. The default function key is PF1 (or PF13).

You can type SQL statements while the online HELP information is displayed.

HOLD



HOLD is an ISQL command that prevents an SQL statement from being processed when it is typed. The SQL statement is placed in the SQL command buffer and remains there until it is replaced with another SQL statement. You can check the SQL statement for typing errors before it is processed by a START command. You can also type an SQL statement containing placeholders and substitute values for the placeholders when the statement is started using a START command.

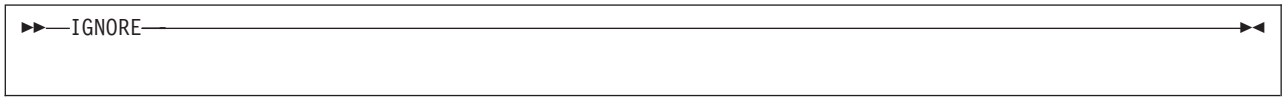
sql_statement
is the SQL statement to be held. (ISQL commands cannot be held).

The following example illustrates the use of a HOLD command to place an SQL statement in the command buffer without executing it:

```
hold select * from employee
```

The HOLD command can also be invoked by pressing PF9 (or PF21) prior to, during, or after typing. By pressing PF9 instead of ENTER, the SQL statement typed is placed in the SQL command buffer without being processed. The HOLD PF key does not store the command in the SQL command buffer; you must press ENTER after pressing HOLD PF. The default key for the HOLD function is PF9 (or PF21).

IGNORE

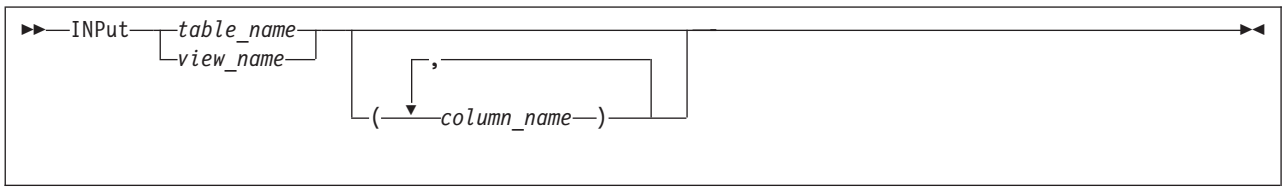


IGNORE is an ISQL command that nullifies a command, statement, or data that is being typed on multiple input lines. Type it at the start of the input area.

The following example shows how the IGNORE command can be used to correct a table-naming error:

```
select projno,actno,acstaff -
from emp_act -
ignore
select projno,actno,acstaff -
from proj_act -
where projno = 'MA2100'
```

INPUT



INPUT is an ISQL command that enables you to insert several rows of data into a table or view.

INPUT is issued to identify the table, view, or column(s) where the data is inserted.

table_name

view_name
is the name of the table or view in which data is to be inserted. A *view_name* can only be used if it applies to a single table. The names can contain DBCS characters.

column_name
is one or more column names, separated by commas, into which data is to be inserted. The order in which the column names are specified determines the order in which the data must be typed. All columns of the new row that are not listed receive a null value, and unlisted columns must have been defined to accept null values or an error occurs. This is because the INPUT command is essentially inserting a new row of data, and null values are inserted into any columns not specified. Omitting the column_name is the same as naming all the columns of the table in their created order. The names can contain DBCS characters.

Successful execution of the INPUT command causes the column names and their data types to be displayed in the order in which the data must be typed. Data can then be typed one row at a time. Press the ENTER key after each row is typed.

When typing data:

- Use commas to separate each data item of a row.
- Enclose the data item in single quotation marks if it is a CHAR, VARCHAR, DATE, TIME, or TIMESTAMP data type.
- Do not enclose the keyword **NULL**, or any of the special registers, such as CURRENT SERVER, in single quotation marks.
- If your data includes a single quotation mark, type two single quotation marks. When you type:

  ```
  'julie's book shop'
  ```

  JULIE'S BOOK SHOP is inserted into the table.
- If you are typing graphic data, you type:

  ```
  G'so...si'
  ```

  where `so` stands for shift-out character, `si` stands for shift-in character, and `...` is graphic data (a DBCS character string).

  Note: You can use N' as a synonym for G'

- If you are typing hexadecimal data, you type X'F140F2'.
- If all the data for a single row does not fit in the input area, type the continuation character. Press ENTER to continue.
- Be sure to include a space before the continuation character, if required, because the continuation character does *not* provide one.
- When null values are allowed, you can type NULL to insert a null value for the data-item value.

After you press ENTER for a row of data, the data is moved to the output area of the display and the input area is cleared so that another row of data can be typed. When all the data has been typed, type the END command to signify the end of input data.

With AUTOCOMMIT on, the data you type is not committed to the table until the INPUT command is ended by the END command or the END PF key (usually PF3). You can use the ISQL SAVE command to commit data in the table prior to typing an END command.

The SAVE command stores all data typed since the previous SAVE command or, if one had not been typed, since the start of the INPUT command. It has the same effect as the END command but lets you continue to type data.

In addition to storing data prior to ending an INPUT command, you can also prevent the storing of data typed since the last SAVE command or since the start of the INPUT command if no SAVE has been issued. You type the ISQL BACKOUT command instead of more data.

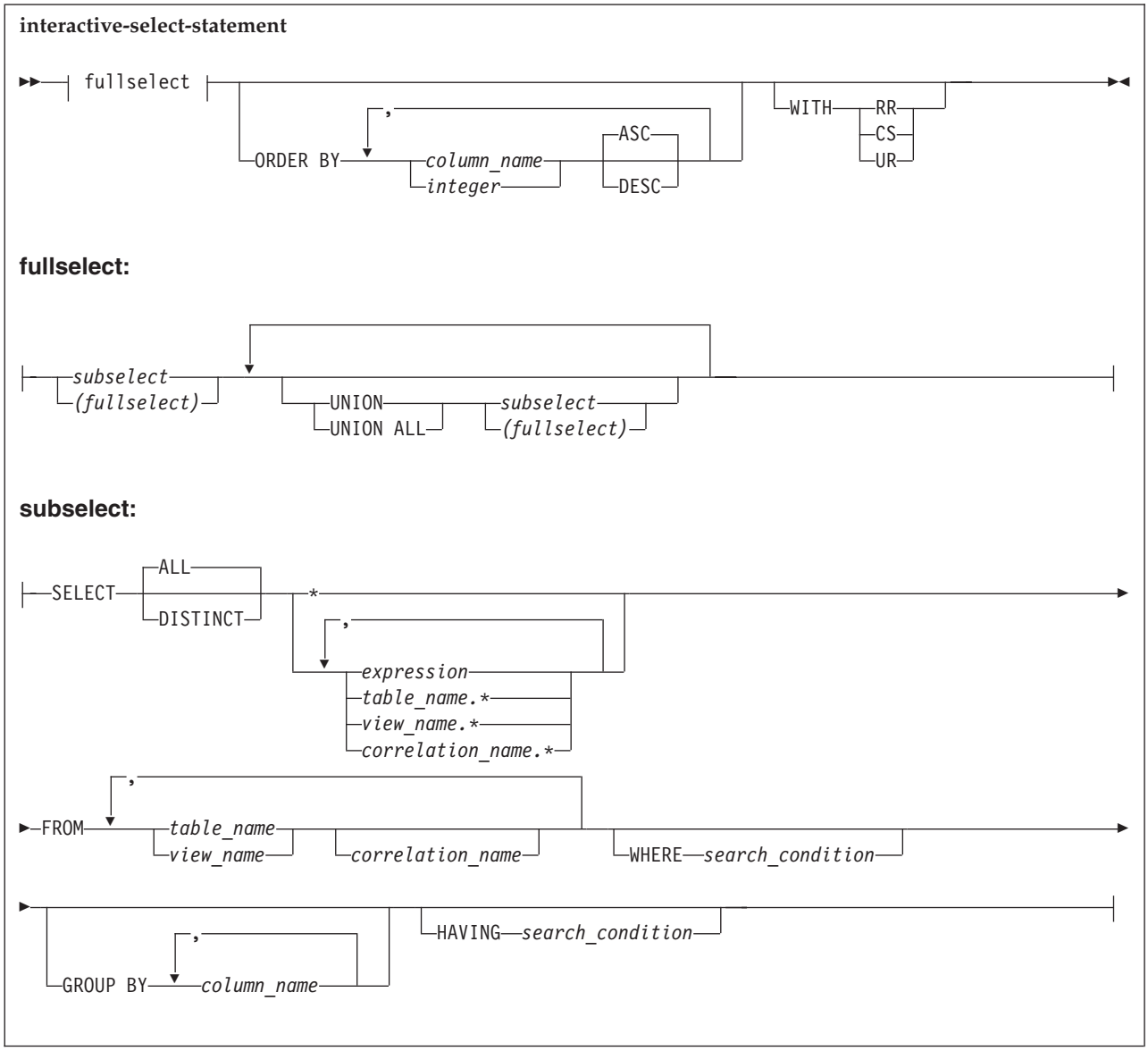
If AUTOCOMMIT is off, the data you type on an INPUT command is not committed to the table until you type a COMMIT command after ending the INPUT command. With AUTOCOMMIT off, the SAVE and BACKOUT commands have no effect.

Example

The following illustrates how you can fill the ACTIVITY table by using the INPUT command:

```
input activity
  .
  .
  .
160,'ADMDB','Adm databases'
  .
  .
  .
100,'TEACH','Teach classes'
end
```

Interactive Select



The interactive-select statement retrieves data from a table or view. The data retrieved can only be viewed; you cannot change it. For more information about the terminology used in this diagram, refer to the *DB2 Server for VSE & VM SQL Reference* manual.

If a SELECT statement contains a placeholder in the WHERE, GROUP BY, ORDER BY, or HAVING clauses, FORMAT information is saved until the next SELECT statement is entered. Formatting information is saved permanently if the SELECT statement is stored. However, if the SELECT or FROM clauses contain placeholders or parameters, FORMAT information is not saved. In addition, FORMAT information is not saved if you change any data in the SELECT or FROM clauses by using the CHANGE command.

When you enter a SELECT statement from a terminal, the row length of the information you can see at one time is limited by the terminal screen size.

The number of bytes per row includes the bytes used as column separators.

UNION
combines two or more queries into a single query by merging the rows returned by each query. Duplicate rows are eliminated.

ALL
combines the results of two or more queries without eliminating the duplicate rows.

ALL
specifies that duplicate values are to be selected. If ALL is specified when you select all department numbers from the PROJECT table, each and every department number listed in the department number column is selected. This is the default.

DISTINCT
specifies that duplicate values are not to be selected. DISTINCT can be used only once for each SELECT statement. You can use it to eliminate duplicates from the SELECT result:

```
select distinct deptno from project
```

You can also use it to eliminate duplicates from a column function:

```
select count (distinct deptno) from project
```

If you specify DISTINCT, the maximum number of columns you can put in the *expression* list is 16. In addition, the length of the encoded key derived from the *expression* list must not exceed 255 characters. That is, the sum of the lengths of the columns in the *expression* list, plus approximately 25% of the lengths of those columns that are of varying-length character type, must not exceed 255 EBCDIC or 127 DBCS characters.

* indicates that the data in all the columns is to be selected.

Note: If the table contains more than 45 columns, only the first 45 columns will be displayed. To see the other columns in the table, you can create a view. A view may contain no more than 140 columns. If more columns are needed, you can create additional views.

expression
is a definition of the data desired. An *expression* may be a column name, a constant, a character expression, a special register, a column function, a scalar function, an arithmetic expression, or a labeled duration.

You can specify a list of expressions, separating each with a comma, and the items are retrieved in the same left-to-right order as they appear in the list. The value of USER is interpreted as a CHAR(8) string whose value is the user ID of the user currently connected. The maximum number of columns that you can specify in the list is 45.

You can use the following operators to connect numeric data types:
+ (plus, add)
– (minus, subtract)
* (times, multiply)
/ (divided by)

If you use these operators for numeric data types, see the *DB2 Server for VSE & VM SQL Reference* manual for information about data conversion.

You can use the concatenation operator, CONCAT, to join two or more compatible operands to form a string. An operand may be a column, a name, a constant, or an expression.

An operand can be the result of an expression. For example, if the USER special register is used, it is treated as CHAR(8). If the CURRENT DATE, CURRENT TIME, CURRENT TIMESTAMP, or CURRENT TIMEZONE special registers are used, the values are treated as the character representation of the value in the format defined by the system. A datetime value can be concatenated with a character string because the datetime data types are compatible with character data types.

When varying-length operands are concatenated, only the actual length of the operand is concatenated.

The result of the expression is the concatenation of the operand expressions. The resulting data type is null if either operand is nullable. The resulting data type is character if both operands are CHARACTER. If both operands are GRAPHIC, the result data type is GRAPHIC. If both operands are fixed length, the resulting data type is fixed length. If either operand is varying length, the resulting data type is a varying length string. The associated defined length is the sum of the defined lengths not exceeding 254 bytes.

In the following example, the employee number and the first name of the employee are concatenated with a hyphen between them:

```
select empno concat '-' concat
firstnme from employee
```

If a column function is used in an *expression*, all expressions in the list must contain a column function unless grouping is being performed. An example is:

```
select min(edlevel),avg(bonus) from employee
```

*table_name.**

*view_name.**

*correlation_name.**
identifies the table or view to which the column belongs. The asterisk (*) can be replaced with a column name. These prefixes are especially useful for differentiating columns that have the same name, but that belong to different tables or views. The *correlation_name* can be used to simplify a query, or to join a table to itself. See "*correlation_name*" below for more information.

FROM *table_name*

FROM *view_name*
identifies the table or view from which data is to be selected.

table_name

view_name
is the name of the table or view.

You can further qualify the table or view by specifying the owner of the table or view. You must separate the owner's name from the table name or view name with a period. The owner's name is unnecessary for tables or views that you own. You must have the SELECT privilege or DBA authority to select information from tables or views owned by other users.

correlation_name
is the name you define as an alternative name for the table or view to be selected. It can be any string up to 18 characters long, and must begin with a letter.

WHERE *search_condition*
is one or more conditions to apply in selecting data.

GROUP BY *column_name*
A query can have the column functions SUM, AVG, MAX, MIN, and COUNT applied to groups of rows that have matching values in a column. Rows can also be grouped by matching values in more than one column. The definition of the groups is specified with a GROUP BY clause.

column_name
is the name of one or more columns, separated by commas, to be used when forming a group.

For example, the maximum, minimum, and average activity staff *for each project* in the PROJ_ACT table could be selected by the following query:

```
select projno,max(acstaff),min(acstaff),avg(acstaff) -
from proj_act -
group by projno
```

When a query uses the grouping feature, it returns only one result row for each group. Therefore, the items selected by such a query must be properties of the groups, not properties of individual rows. The *expression* list may contain columns that are also in the GROUP BY clause together with column functions on any columns. It may not contain any non-grouped column without a column function. If the column function COUNT(*) is used, it evaluates to the number of rows in the group.

If any rows have a null value in a grouped column, ISQL groups the null values in those columns together. The null values returned may be due either to unknown column values or to arithmetic exception errors.

A grouping query may have a standard WHERE clause that serves as a filter, keeping only those rows which satisfy the *search_condition*. The WHERE clause filters out the non-qualifying rows before the groups are formed and the column functions are computed.

The following example query finds the average and minimum activity staff for each project, considering only activities whose starting date is 1 January 1982:

```
select projno,avg(acstaff),min(acstaff) -
from proj_act -
where acstdate = '1982-01-01' -
group by projno
```

HAVING *search_condition*
is one or more conditions that apply to groups. ISQL returns a result only for those groups that satisfy the condition. The HAVING clause may contain one or more group-qualifying conditions connected by ANDs and ORs. Each group-qualifying condition compares some property of the group, such as AVG(ACSTAFF), with another group property or with a constant.

The following example query lists the maximum and minimum activity staff for various projects in the PROJ_ACT table, considering only projects that have more than three activities:

```
select projno,max(acstaff),min(acstaff) -
from proj_act -
group by projno -
having count(*) > 3
```

One of the functions in a HAVING clause may specify DISTINCT (for example, COUNT(DISTINCT PROJNO)). However, DISTINCT may be used only once in a query. It may not be used in both the *expression* list and the HAVING clause.

It is possible, though unusual, for a query to contain a HAVING clause but no GROUP BY clause. In this case, the entire table is treated as one group. If the HAVING condition is true for the table as a whole, the selected result, which must consist entirely of column functions, is returned.

ORDER BY

orders, or sorts, the rows to be retrieved by the column(s) specified. A maximum of 16 columns may be specified in the ORDER BY clause.

column_name

refers to a column name in the *expression* list. For example, to order a query primarily by the values in the ACTNO column and secondarily by the values in the PROJNO column, you would type:

```
select actno,projno,acstdate,acendate -
from proj_act -
order by actno,projno
```

integer

refers to the items in the *expression* list. For example, to order a query primarily by the first item and secondarily by the third item of the list, you would type:

```
select projno,acstdate,acendate,acstaff -
from proj_act -
order by 1,3
```

These items may be columns or more complex expressions such as BONUS+COMM.

ASC

DESC

indicates the order in which results are returned: either ascending (ASC) or descending (DESC). The default is ASC. For example, to indicate ascending order on item 3 and descending order on item 5, you could type:

```
order by 3,5 desc
```

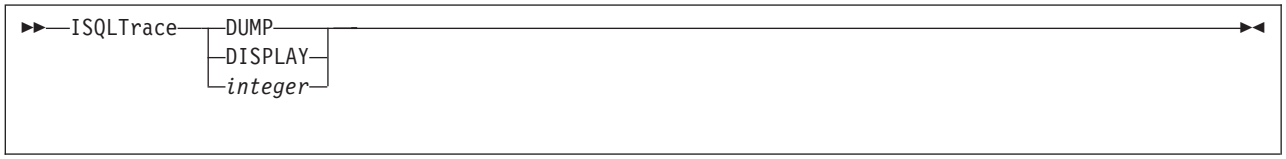
because ASC is the default.

In contrast, to indicate descending order on item 3 and item 5, you would type:

```
order by 3 desc,5 desc
```

Blanks sort first in ascending order, last in descending order, and are ignored if the data type is VARCHAR or VARGRAPHIC.

ISQLTRACE



The ISQLTRACE command traces activity within ISQL. This ISQL command traces calls to and returns from other ISQL modules, SQL return codes, and ISQL messages. Trace information is saved in storage for dump debugging.

DUMP
specifies that the trace information is to be printed. ISQL creates an unformatted storage dump hardcopy of the trace table.

DB2 Server for VM

ISQL uses the CP DUMP command to dump to the lowest priority virtual printer defined.

DB2 Server for VSE

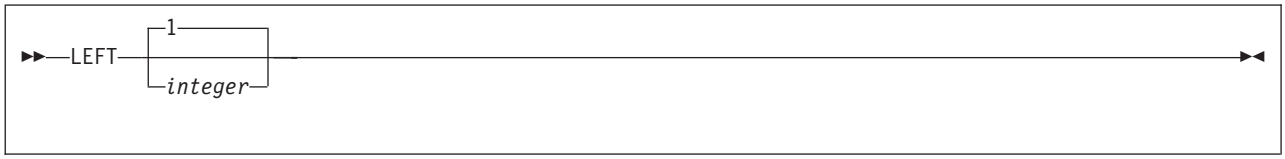
ISQL issues a CICS dump to the CICS dump data set.

You must run a job to print the dump data set.

DISPLAY
specifies that the formatted trace table is to be displayed on the display. If more than 50 entries or more than the specified *integer* number of entries have been made in the trace table, the entries wrap. The more recent entries are written over the earlier entries. As a result, only the last 50 entries or *integer* number of entries in the table are displayed. The entries are displayed in reverse order of the order that they were put into the trace table.

integer
changes the size of the trace table. Replace *integer* with the number of trace entries that are to be contained in the trace table. *integer* must be a number from 50 to 1000.

LEFT

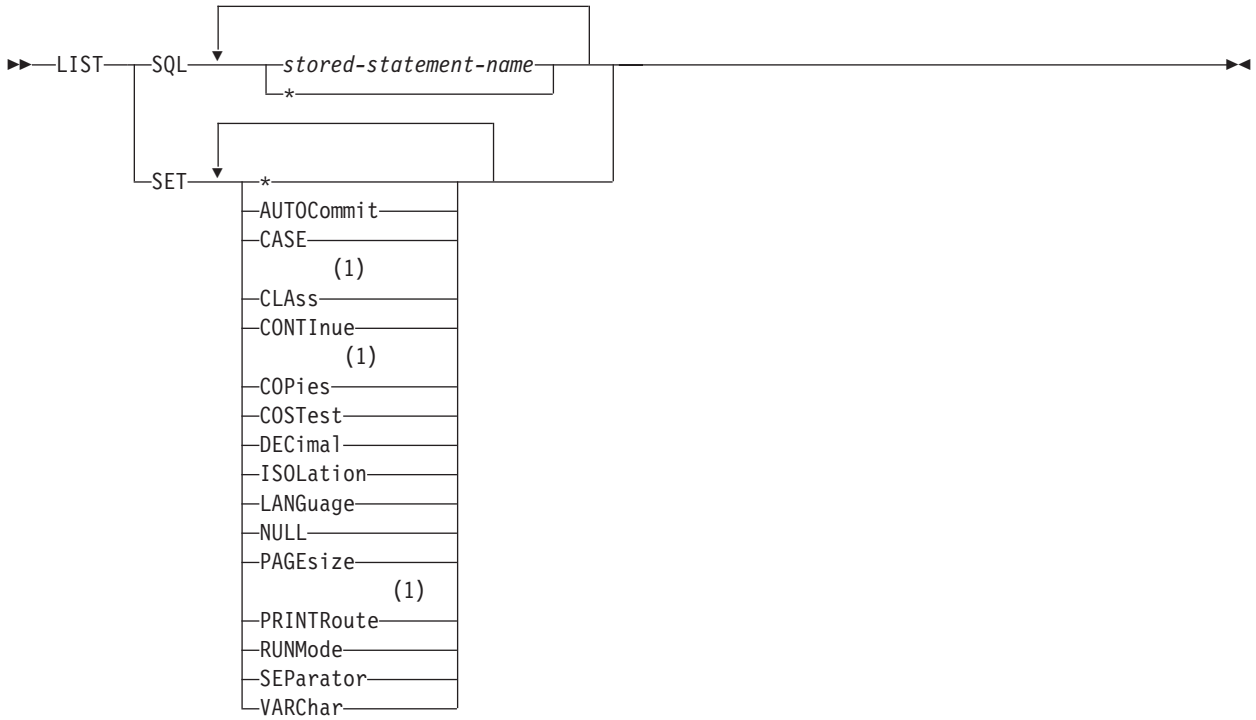


LEFT is an ISQL display command that causes the display to start *integer* columns to the left, counting from the leftmost column on the display and as long as no columns are being excluded. The number refers to the columns provided by the SELECT statement, not those currently displayed. You must use the order of the columns specified in the SELECT clause of the SELECT statement that provided the query result to determine the value to specify for integer. If the value of *integer* represents a column being excluded, the display starts at the next displayable column to the right of the excluded column.

Specifying a value greater than the number of columns remaining to the left, starts the display at column 1 or at the first displayable column if column 1 is excluded.

If no number is specified, the display moves one column to the left. No change in the display occurs if that column is excluded.

LIST



Notes:

1 The parameters CLAss, COPies, and PRINTRoute are not applicable to VM.

LIST is an ISQL command that lists information about stored SQL statements or the settings of operational characteristics set by the SET command.

SQL
lists stored SQL statement(s) on the display. See "STORE" on page 161 for more information.

stored_statement_name
is the name of the stored SQL statement to be listed. The entire statement stored with the specified name is displayed.

***** specifies that all your stored SQL statements are to be listed. The name of each statement and its first 50 characters are displayed.

SET
lists the current operational characteristics in effect for the specified function of the SET command. For example, if you type:

```
list set continue
```

the current continuation character is displayed. Valid functions are any of those functions performed by the SET command.

If * is specified, the current operational characteristics in effect for all SET command functions are listed. For a description of the SET characteristics, see "SET" on page 149. for more information.

Examples

The LIST command can be used with more than one keyword option in a single command. For example, if you want to display more than one operational characteristic, you can type:

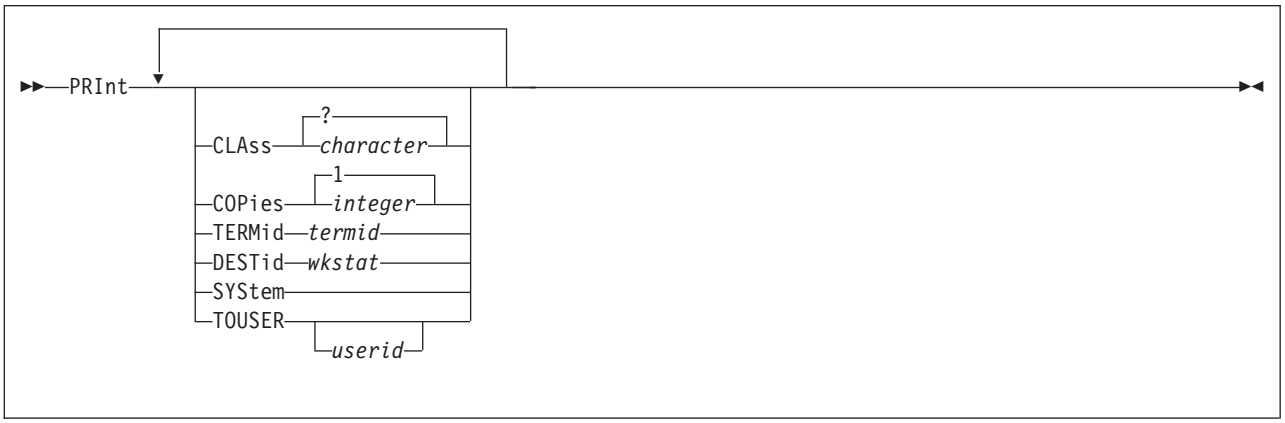
```
list set page case null
```

If you want to display more than one stored SQL statement, you can type:

```
list sql myquery query1
```

Stored SQL statements and operational characteristics may not be listed at the same time.

PRINT



DB2 Server for VSE

PRINT is an ISQL display command that requests printed copies of a query result by sending it to the system printer, POWER remote printer, or CICS/VSE terminal.

DB2 Server for VM

PRINT is an ISQL display command that requests printed copies of a query result by sending it to the system printer.

The data that is printed is based on the query result obtained by the SELECT statement.

All the rows of the query result, as modified by FORMAT commands, are printed regardless of the position of your display of the result. A query result that is longer than what can be viewed on one display, can be printed with one PRINT command. The printed report starts with the column and the character position within the column that is at the left edge of the display when the PRINT command is typed. The number of characters of each row that is printed depends on the setting of the PAGESIZE WIDTH value specified by the SET command. You receive a message if the width of the row of data exceeds the WIDTH setting.

The output class to use can be obtained from the people responsible for data processing at your location.

DB2 Server for VSE

Note: CLASS and COPIES cannot be specified for a CICS/VSE terminal printer.

DB2 Server for VM

Note: You must use the CP TAG and CP SPOOL commands to direct printed output.

CLAss
specifies the output class.

character
is the output class desired. For DB2 Server for VSE, the output class can be a letter (A to Z).

For DB2 Server for VM, the CLASS value can be an integer (0 to 9) or a letter (A to Z). The default is class A.

? (DB2 Server for VSE only)
specifies that the default printer class of the system is to be used.

For DB2 Server for VSE, if CLASS is not included in the PRINT command at all, the current class set by the SET command is used. If no class has been set by a SET command, the default printer class for the system is used.

For DB2 Server for VM, there is no SET command for CLASS. You can set the CLASS value by a CP SPOOL command.

COPies
specifies the number of copies.

integer
is the number of copies desired. For DB2 Server for VSE, you can request up to 99 copies. For DB2 Server for VM, you can specify a COPIES value from 1 to 255.

If this keyword is not specified, one copy is the default unless otherwise determined by a SET command (in DB2 Server for VSE).

DB2 Server for VM

You can also use the CP SPOOL command to specify the number of copies. The value for copies that is specified on the CP SPOOL command remains in effect until another value is specified by another CP SPOOL command.

Note: If you specify the number of copies on the PRINT command after you specified it using a CP SPOOL command, the PRINT command quantity is used for that print operation. All following PRINT commands use the quantity specified by the CP SPOOL command, unless you specify it using the **COPIES** keyword.

Because you can reset your PF keys, you can assign the PRINT function to any PF key. The default function key is PF4 or PF16.

TERMid (DB2 Server for VSE only)
specifies that printed output is to be directed to the designated CICS/VSE terminal.

termid
is the terminal identifier of the CICS/VSE terminal that you want. *termid* must be from one to four alphanumeric characters.

DESTid (DB2 Server for VSE only)
specifies that the printed output is to be directed to the designated POWER remote workstation.

wkstat
is the ID of the desired remote workstation. *wkstat* can be any number from 0 to 250. When *wkstat* is 0, the printed output is to be directed to the system printer.

SYStem (DB2 Server for VSE only)
specifies that the printed output is to be directed to the system printer.

TOUSER (DB2 Server for VSE only)
specifies that the printed output is to be directed to a user identified by *userid*.

userid
is the VSE POWER user identifier of the user to whom the output is being spooled. An identifier cannot be longer than 8 alphanumeric characters.

If you enter the PRINT TOUSER command without specifying a *userid*, ISQL spools the output to the ISQL user ID used at time of log on.

The PRINT TOUSER *nnn* command has the same effect as the PRINT DESTid *nnn* command, where *nnn* is the ID of the remote workstation to which you want to direct your printed output. This ID can be any number from 1 to 250.

DB2 Server for VSE

Any TERM, DEST, or SYS indication on the PRINT command applies for that particular PRINT operation only.

The PRINT command can be started by pressing PF4.

Each printed page is numbered and dated at the top. A title is automatically provided at the top of each printed page. This title consists of the first 100 characters of the SELECT statement issued unless you specified your own title with an ISQL FORMAT command.

You can specify more than one keyword option on a single PRINT command. For example, the following command specifies both the class and number of copies:

```
print copies 3 class a
```

DB2 Server for VM

If there is an error in the command, all valid changes are made until the mistake is determined. For example, in the following command, you type a $ sign instead of the letter a:

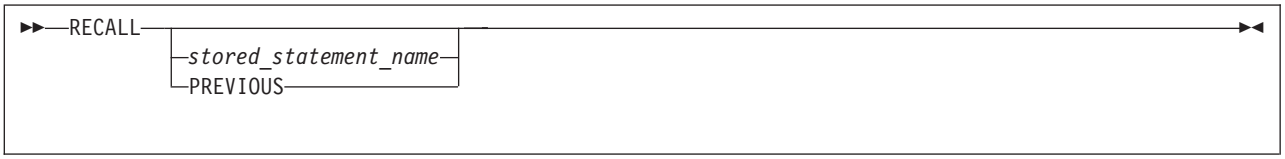
```
print copies 3 class $
```

The command sets the number of print copies to 3. However, the class remains at the default class A, because $ is an incorrect output class. This output class error does not change the number of copies to the original value. Neither does the number of copies or the class change from their current settings if the following command is issued:

```
print class $ copies 3
```

The output class value is again incorrect, so processing stops when this error is detected.

RECALL



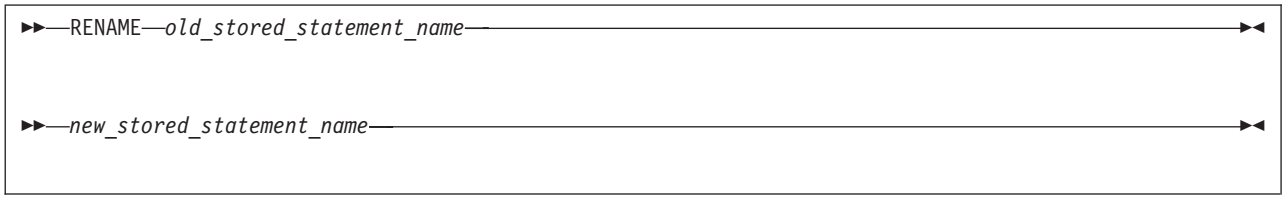
RECALL is an ISQL command that retrieves a stored SQL statement. The stored statement is inserted into the SQL command buffer and is shown in your display. You must enter the START command before the statement is started.

stored_statement_name
is the name of the stored SQL statement to be recalled.

PREVIOUS
specifies that the previous SQL statement (the one typed prior to the current SQL statement) is to be recalled. The current SQL statement is then saved with the name PREVIOUS.

Entering RECALL with no name specified, or pressing the appropriate PF key, displays the statement currently contained in the SQL command buffer. The RECALL command can be invoked by pressing PF5. You can assign the RECALL function to any PF key; the default is PF5 or PF17.

RENAME

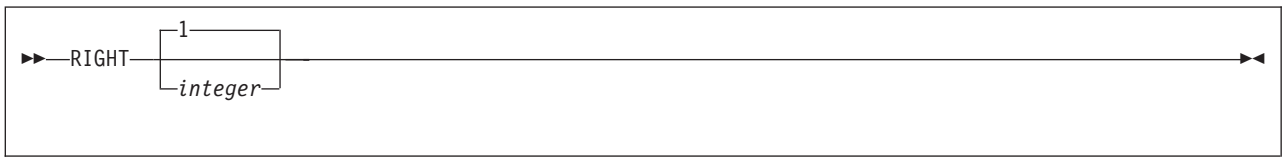


RENAME is an ISQL command that changes the name of a stored SQL statement.

old_stored_statement_name
is the current name of the stored SQL statement. The name PREVIOUS should not be used.

new_stored_statement_name
is the new name for the stored SQL statement. It can be up to 8 characters. Do not use the name PREVIOUS.

RIGHT



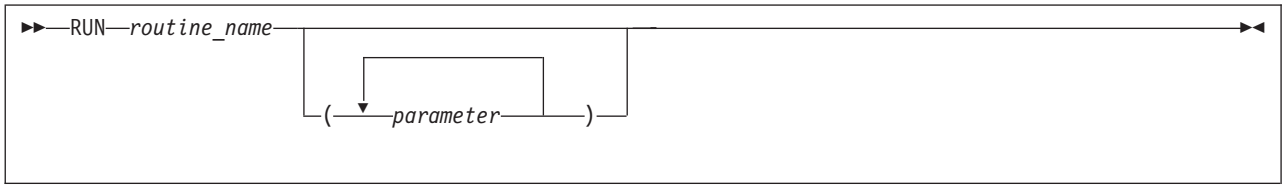
RIGHT is an ISQL display command that causes the display to start *integer* columns to the right, counting from the leftmost column of the display, provided no columns are being excluded. The number refers to the columns provided by the SELECT statement, not those currently displayed. You must use the order of the columns specified in the SELECT clause of the SELECT statement that provided the query result to determine the value for *integer*. If the value of *integer* represents a column being excluded, the display starts at the next displayable column to the right of the excluded column. If no more columns to the right can be displayed, a blank display results.

Specifying a value greater than the number of columns remaining to the right displays the last column or a blank display if the last column is excluded.

If no number is specified, the display moves one column to the right or to the next displayable column to the right if that column is excluded.

The RIGHT 1 command can be invoked by pressing PF11. Because you can reset your PF key values, you can assign the RIGHT 1 function to any PF key. The default function key is PF11 or PF23.

RUN



RUN is an ISQL command that starts the processing of a routine.

You can run another user's routine if you have obtained the SELECT privilege by a GRANT statement on that user's ROUTINE table. Once you have the SELECT privilege on the user's ROUTINE table, you can run those routines by qualifying the routine name with the owner's authorization ID. For example, to run a routine called REPORTS that is owned by a user whose authorization ID is MIKE, you use:

```
run mike.reports
```

You do not have to identify the table name, ROUTINE, because the name of everyone's routine table is the same. Care must be taken in running another user's routine because any stored SQL statements or synonyms used in a routine are not recognized unless you have also defined them.

Parameters can also be passed to the routine by including them on the RUN command.

routine_name
is the name of the routine to be run.

parameter
is the parameter to be substituted for placeholders in the commands contained in the named routine. If more than one parameter is used, separate them by blanks.

Enclose a parameter in single quotation marks if it contains a blank. Any characters between the terminating single quotation mark and the next blank are ignored.

If there are more placeholders in the routine than there are parameters on the RUN command, the extra placeholders are replaced by null characters. Extra parameters on the RUN command are ignored.

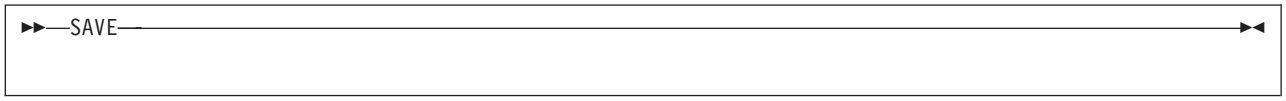
Example

The following command executes your routine named SAMPLE and provides three parameters:

```
run sample (BLUE RED,GREEN 'WHITE BLACK')
```

The placeholders (indicated by &) in the routine are replaced by:
- BLUE to be substituted for all occurrences of &1
- RED,GREEN to be substituted for all occurrences of &2
- WHITE BLACK to be substituted for all occurrences of &3.

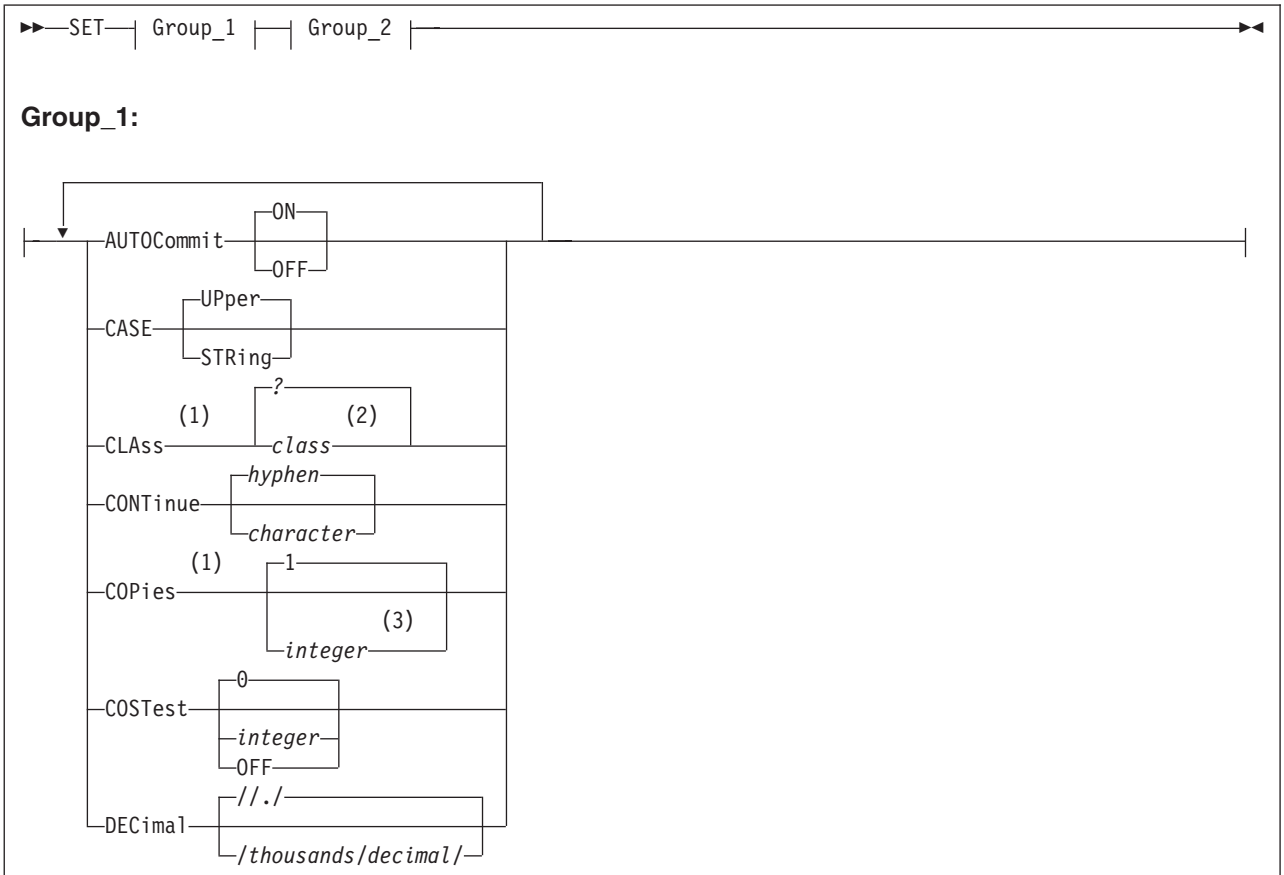
SAVE



SAVE is an ISQL command that is only meaningful while using the INPUT command with AUTOCOMMIT on. If AUTOCOMMIT is on, it saves all changes made since the last SAVE command or since the start of the INPUT command if no SAVE command has been typed.

SAVE has no effect with AUTOCOMMIT off. With AUTOCOMMIT off, a COMMIT is necessary to save all input. This also commits all changes made during the current logical unit of work. The COMMIT must be typed after the INPUT command is ended.

SET



Notes:

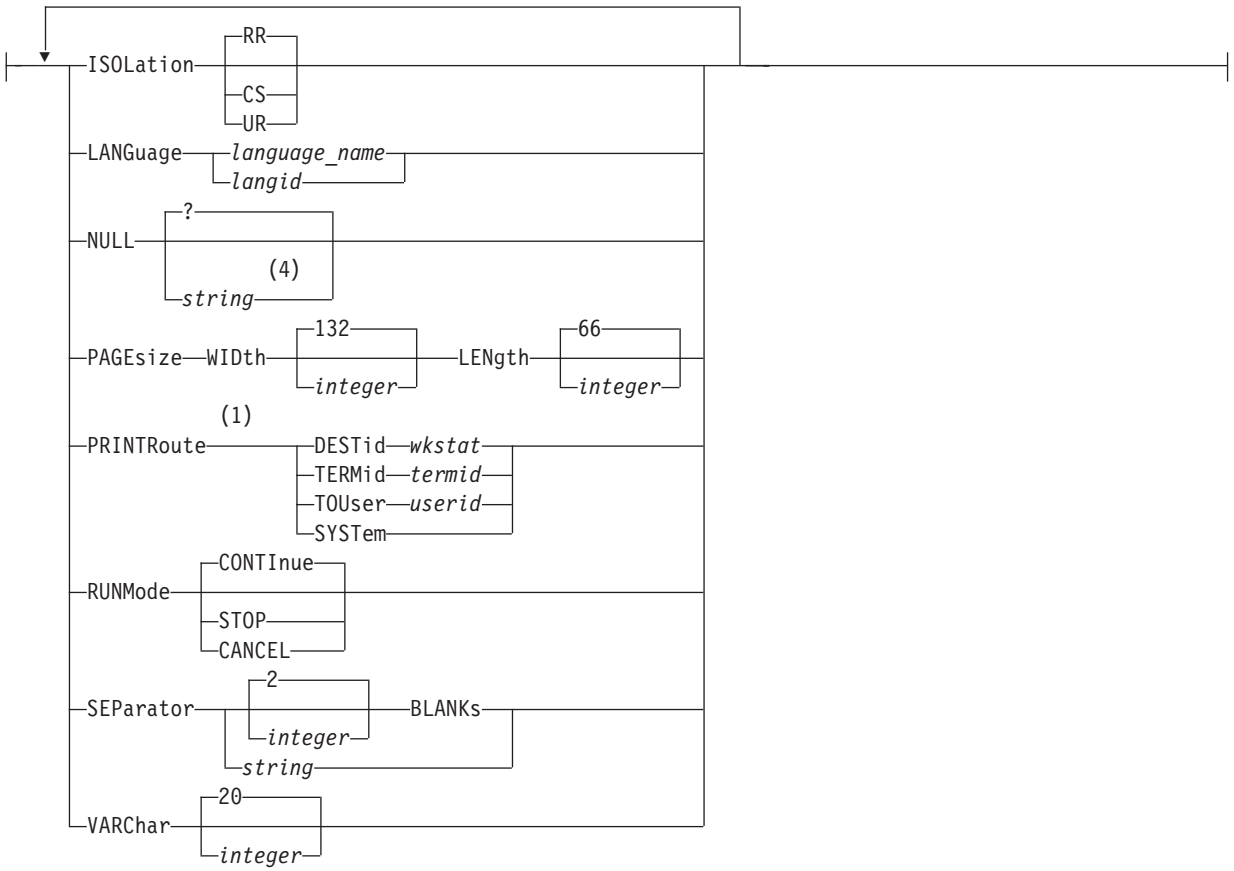
1 The parameters CLAss, COPies, and PRINTRoute are not applicable to VM.

2 Output class wanted (letters from A to Z).

3 Number of copies to be printed.

4 A maximum of 20 characters can be used for null values.

Group_2:



Notes:

1 The parameters CLAss, COPies, and PRINTRoute are not applicable to VM.

2 Output class wanted, (letters from A to Z).

3 Number of copies to be printed.

4 A maximum of 20 characters can be used for null values.

SET is an ISQL command that controls a number of functions.

When ISQL is started, the following operational characteristics are in force:
- AUTOCOMMIT is ON.
- CASE is UPPER.
- CLASS is a question mark (?). (DB2 Server for VSE only)
- CONTINUE is a hyphen (-).
- COPIES is 1. (DB2 Server for VSE only)
- COSTEST is zero.
- DECIMAL is //./.
- ISOLATION is repeatable read.
- LANGUAGE is the default language as defined in the SYSOPTIONS catalog.
- NULL is a question mark (?).
- PAGESIZE is LENGTH=66 WIDTH=132.
- PRINTROUTE is to the system printer. (DB2 Server for VSE only)
- RUNMODE is CONTINUE.

- SEPARATOR is two blanks.
- VARCHAR is 20.

The current setting of these functions can be listed with the LIST command. Some of these settings may already have been changed by a PROFILE routine, or you may want to change the settings for your session in your PROFILE routine, which is run automatically when you start ISQL.

Notes:

1. In the VM environment, the user must use the CP commands SPOOL and TAG to change the routing of the print output.
2. When you are using DRDA protocol in the VSE or VM environment, the isolation level is set to CS.

AUTOCommit
specifies if commands are to be committed automatically. Unless otherwise specified, commands are committed automatically.

ON
specifies that changes to tables resulting from an SQL statement are committed automatically when the statement is processed. This is the default.

There is an exception to the automatic committing of changes. If the SQL statement is INSERT, UPDATE, or DELETE, and it affects more than one data row, the changes are not immediately committed. Instead, the system issues a message that lets you either commit the work, or cancel or rollback the changes. If you type CANCEL or ROLLBACK, the changes are not committed; if you type any other statement, the changes are committed.

OFF
specifies that changes are not to be committed to a table until a COMMIT statement is typed.

CASE
specifies if characters enclosed in single quotation marks are to be converted to uppercase. Unless otherwise specified, all characters typed from the keyboard are converted to uppercase.

If you specify that the characters are to be converted, SQL uses the conversion information from the SYSCHARSETS system catalog to handle the conversion.

UPper
specifies that all characters typed from the keyboard or a routine are converted to uppercase characters. This is the default.

STRing
specifies that all characters enclosed in single quotation marks typed from the keyboard or a routine are not converted to uppercase characters.

CLAss (DB2 Server for VSE only)
specifies the printer output class.

class
is the output class desired (a letter from A to Z).

? specifies the default printer class of the system.

Unless otherwise specified, the default printer class for the system is used.

Note: Class cannot be specified for a CICS/VSE terminal printer.

CONTInue
specifies the continuation character.

character
is the continuation character to use for continuation of input lines.

Choose a character that is not normally the last character of a statement. Also, do not use a single or double quotation mark, a semicolon, or a blank. Unless otherwise set, the continuation character is the hyphen (-).

COPies (DB2 Server for VSE only)
specifies the number of copies for printed reports.

integer
is the number of copies that are to be printed when subsequent PRINT commands are issued.

Unless otherwise specified, the number of copies for printed reports is one. A maximum of 99 copies can be specified.

Note: The number of copies cannot be specified for a CICS/VSE terminal printer.

COSTest
specifies when the ISQL *Query Cost Estimate* (QCE) is displayed. Unless OFF is specified, the QCE message is always displayed.

OFF
specifies that the ISQL QCE message should not be displayed.

integer
specifies that the ISQL QCE message should be displayed. Replace *integer* with any number from 0 to 9999. The QCE is displayed if it is greater than *integer*.

While you cannot specify any number greater than 9999, the QCE message displays '>9999' to indicate the cost is greater than 9999.

For additional information about the ISQL Query Cost Estimate, refer to the *DB2 Server for VSE & VM Database Administration* manual.

DECimal
specifies the type of punctuation to use when displaying a decimal column.

/thousands/
is the thousands separator character. Valid characters are a period, a comma, and a blank.

/decimal/
is the decimal separator character. Valid characters are a period and a comma.

The slash distinguishes the thousands separator character from the decimal separator character.

Unless otherwise specified, no thousands separator is used, and the decimal separator is a period.

Valid combinations of t and d are:

Thousands Separator	Decimal Separator	Example
(nothing)	.	1234.56
,	.	1,234.56
.	,	1.234,56
(a blank)	,	1 234,56

For example, assume the value *123456* is contained in a decimal column that was created with two decimal places. Then, specifying:

```
set decimal /,/./
```

provides the following punctuation when this field is displayed:

```
1,234.56
```

The DECIMAL keyword does not change how input is specified, only how output is displayed. For example, to reference the above number (1,234.56) in a column called SALES in a WHERE clause, you use:

```
where SALES=1234.56
```

ISOLation
specifies the isolation level placed on data when you read information from the application server. You specify the type of lock that the system places on the data. This is specifying the *isolation level*. The isolation level specified sets the degree of independence one terminal user has from another terminal user.

For guidelines on using this setting, see "Specifying the Isolation Level" on page 95.

RR
requests the isolation level repeatable read. This setting holds locks on the data you are using until a COMMIT or ROLLBACK is performed. These locks isolate the data from other users. No one can modify any rows you have read until your work has been committed or rolled back. Use this setting when it is important to keep data completely isolated. This is the default.

Note: For DB2 Server for VSE & VM, if you are using the DRDA protocol, the default isolation level is set to CS. If you specify any other setting, it will be ignored.

CS
requests the isolation level cursor stability. Isolation level cursor stability has meaning only for data in public dbspaces with row and page level locking. The system locks individual rows or pages depending on the lock specified in ACQUIRE DBSPACE, ALTER DBSPACE, and LOCK statements. Use the CS setting to free the data you are reading as soon as possible.

UR
requests the isolation level uncommitted read. Isolation level uncommitted read has meaning only for data in public dbspaces with row and page level locking. This setting applies only to read-only operations (SELECTs). For other operations (UPDATE, DELETE, and INSERT), the rules of CS apply. This setting reduces lock contention on data being read; however, data integrity may be compromised because read-only access to

uncommitted data is allowed. Use the UR setting only when it is not necessary that the data you are reading be committed.

The isolation level specification affects the UPDATE, DELETE, INSERT with subselect form, and SELECT statements. The correct value to specify depends on what activity you are performing. Locking problems can be reduced if the isolation level can be set to cursor stability in your system.

Note: Read and update access to the catalog is performed with a repeatable read setting, regardless of how you set the isolation level. This access is activated by dynamic preprocessing of SQL statements and by SQL data definition statements such as CREATE, ACQUIRE, and GRANT. Your selects (and updates and deletes, if you are a DBA) against the catalog are performed according to the isolation level you set.

SET ISOLATION is acceptable only when the target application server is a local application server. Otherwise the isolation level of CS is assumed and the SET ISOLATION command will have no effect.

LANGuage
specifies the language in which online HELP and error messages are displayed. Operator messages are displayed in the national language of the application server.

language_name
is the language being specified; for example, French. The description of the language can be either the IBM-supplied description or the description chosen by your site. For example, your site may prefer to use *Francais* instead of *French*. *Language-name* can be up to 40 characters long.

langid
is a 5-character language identifier that can be specified instead of the language name. The language identifiers are:

AMENG	Mixed Case American English
UCENG	Uppercase American English
FRANC	French
GER	German
KANJI	Japanese
KOR	Korean
HANZI	Simplified Chinese

If the language you specify is not supported, the current language remains unchanged.

NULL
specifies the characters to be displayed in null fields. Unless otherwise specified, a question mark (?) is used.

string
specifies the characters (up to a maximum of 20) to use for null fields. Enclose the string in single quotation marks if it contains a blank.

For example, the following command:

```
set null empty
```

causes the word EMPTY to be displayed in all null fields.

PAGEsize
specifies the page size for printed output.

WIDth
specifies the width of the paper being used.

integer
is the number of characters that can fit on a line of the output paper. Unless otherwise specified, the page width is 132. You can specify values from 19 to 204.

Note: If you are sending your output to the terminal printer, you should set the page width to the printer width-1 to avoid spacing problems. For example, if the terminal printer width is 132, then set the width to 131 with the command SET PAGE WID 131.

LENgth
specifies the length of the page being used.

integer
is the number of printed lines that can fit on the output paper. Unless otherwise specified, the page length is 66. You can specify values from 9 to 32767. The maximum number of lines that can actually be printed on a page is 8 less than the length. Eight lines are reserved for top and bottom titles and margins.

PRINTRoute (DB2 Server for VSE only)
specifies where print output is to be sent.

DESTid
specifies that printed output is to be directed to the designated POWER remote workstation.

wkstat
is the ID of the desired remote workstation. *wkstat* can be any number from 0 to 250. When *wkstat* is 0, the printed output is to be directed to the system printer.

TERMid
specifies that printed output is to be directed to the designated CICS/VSE terminal.

termid
is the terminal identifier of the desired CICS/VSE terminal. The *termid* must be from one to four alphanumeric characters.

TOUser
spools the print output the same way it will spool the print output when the user enters `PRINT TOUser userid`

userid
is the VSE POWER user identifier of the user to whom the output is being spooled. An identifier cannot be longer than eight alphanumeric characters. If the *userid* is any number from 1 to 250, ISQL will spool the print output to the POWER remote workstation whome ID is the number specified.

SYStem
specifies that printed output is to be directed to the system printer.

RUNMode
specifies whether processing should continue when an error is detected in a routine. Unless otherwise specified, processing continues.

CONTInue
specifies that processing is to continue to the next command even if errors are detected in the routine. This is the default.

STOP
specifies that processing is to stop if an error is detected in the routine. No rollback is performed, and processing is terminated.

CANCEL
specifies that an internal CANCEL is to be issued if an error is detected. A rollback is issued internally.

SEParator
specifies the separation between columns. Unless otherwise specified, two blanks are used.

integer **BLANKs**
specifies the number of spaces to be displayed between columns. Replace *integer* with the number of blanks desired. The maximum number of blanks that can be specified is 254.

string
specifies the characters to be displayed between columns. Enclose the string in single quotation marks if it contains a blank. For example, if you want to draw a vertical line between columns, you type:

```
set separator ' | '
```

which places a blank, a vertical bar, and a blank between all columns. The string can be up to 254 characters long.

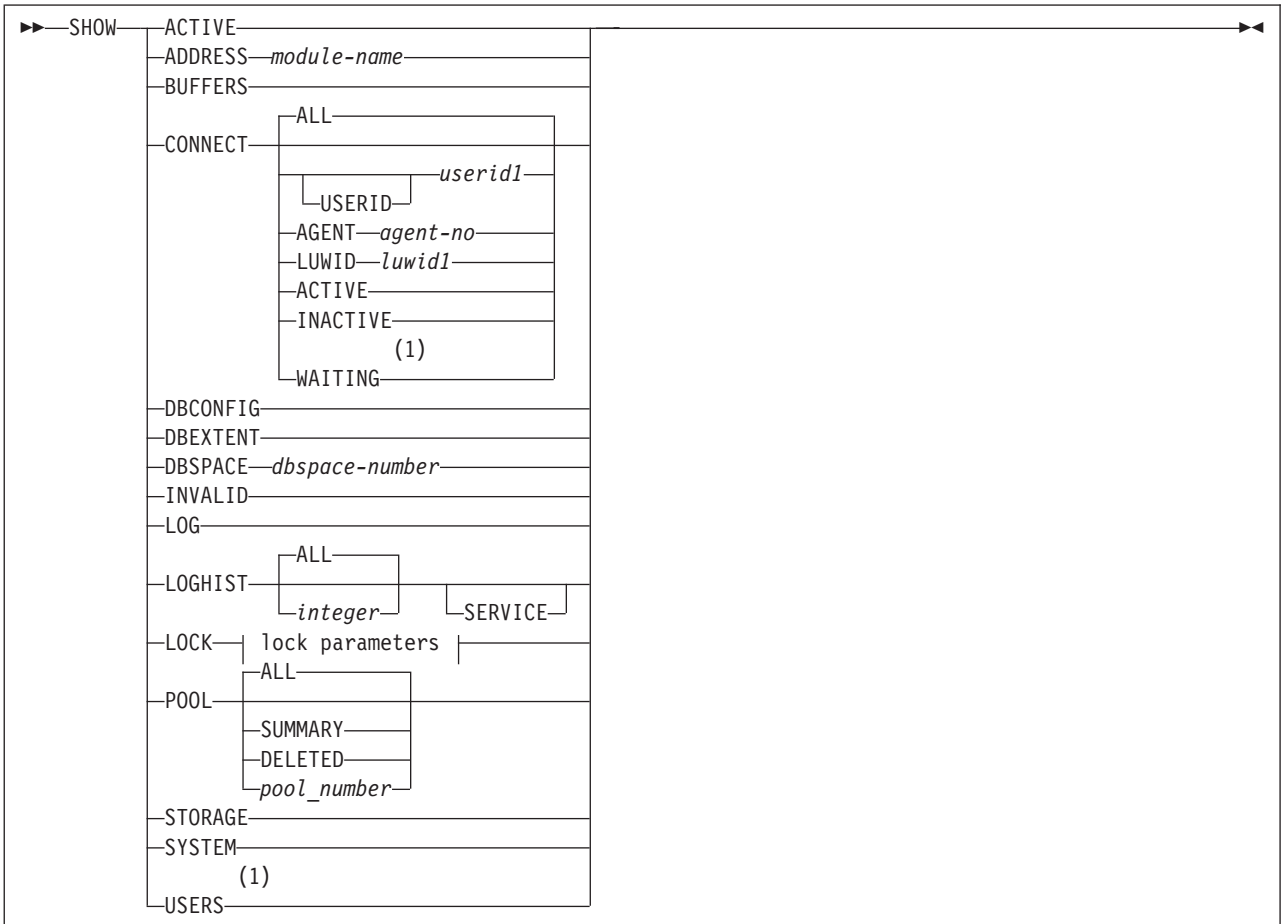
VARChar
specifies the display width of variable length columns.

integer
is the length desired. The maximum width that can be specified is 254. Unless otherwise set, the VARCHAR length is 20. Since only the first 20 characters of a variable length column are displayed, this command or a FORMAT VARCHAR is necessary to view those columns that are wider than 20 characters.

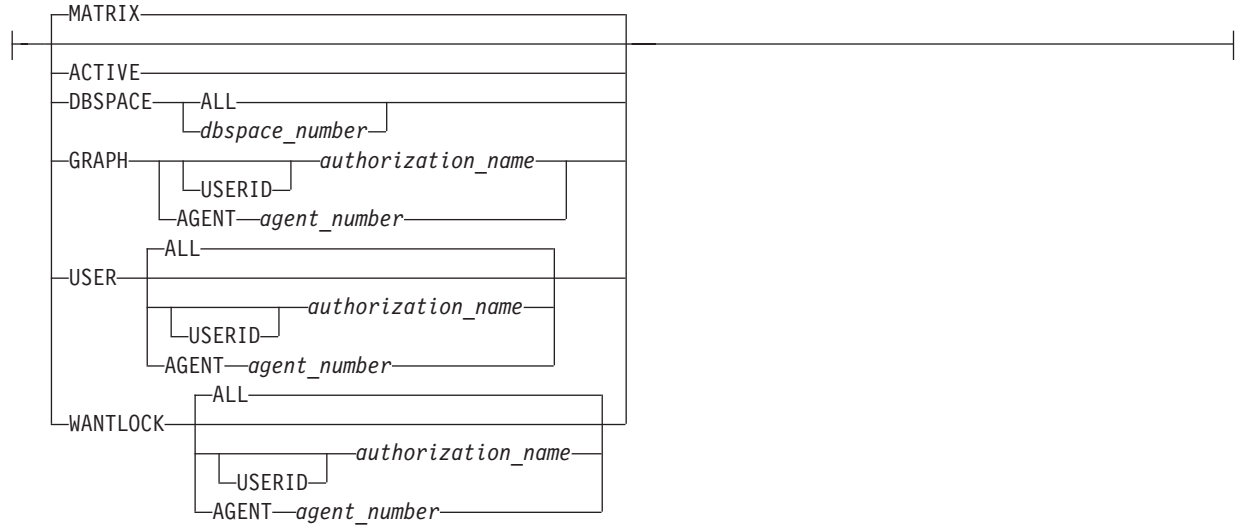
The SET command can be used with more than one keyword option, allowing you to set several operational characteristics with a single SET command. These characteristics are effective for the duration of your terminal session. In the following example, a single SET command sets operational characteristics for AUTOCOMMIT, NULL, and SEPARATOR:

```
set autocommit on null 'no data' separator 2 blanks
```

SHOW



lock parameters



Notes:

1 The parameters WAITING and USERS are not applicable to DB2 Server for VSE

SHOW is an operator command used primarily to monitor system activity. It is not allowed during an LUW, and you must end an LUW that is in progress before issuing SHOW.

The SHOW command can only be used when the target application server is a local application server. It cannot be used when the application server is a remote application server.

See the *DB2 Server for VSE & VM Operation* manual for a description of the keywords used on this command and for a description of the information displayed.

This command results in one or more displays.

DB2 Server for VSE

Instructions for proceeding to the next display of data or ending the display are provided in the status area.

DB2 Server for VM

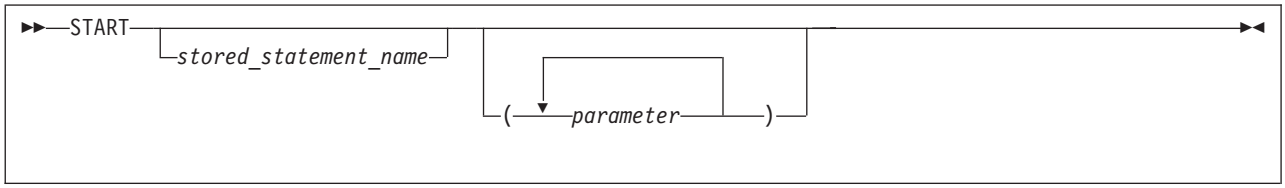
Instructions for proceeding to the next display of data or ending the display are provided in the output area.

When using the SHOW DBSPACE command, replace *dbspace-number* with the number of the particular dbspace for which you want information.

The SHOW command is not available on non-DB2 Server for VSE & VM application servers or if you are using the DRDA protocol.

Note: If the national language of the application server differs from the national language that the user has set for the ISQL session, the messages generated by this operator command are issued in the national language of the application server.

START



START is an ISQL command that causes a stored SQL statement or the current SQL statement to be processed.

The START command can be invoked by pressing PF2. The START command can be invoked by pressing PF2 (or PF14). Because you can reset your PF keys, you can assign the START function to any PF key.

stored_statement_name
: is the name of the stored SQL statement to be started. The name PREVIOUS is not allowed. To start the previous SQL statement, you must first recall it to the command buffer and enter START without supplying a name. If *stored_statement_name* is not specified, or START PF is pressed, the current SQL statement is started.

parameter
: is the parameter to be substituted for the placeholders in the SQL statement. If you use more than one parameter, separate them with blanks. Enclose a parameter in single quotation marks if it contains a blank. Any characters between the terminating single quotation mark and the next blank are ignored.

  All placeholders in the SQL statement that do not have a corresponding parameter on the START command are replaced by null characters or are erased. Extra parameters on the START command are ignored.

  The parameter can contain DBCS characters.

Example

Assume you have the following SQL statement stored as *SELPD*:

```
select &1,&2,&3 -
from emp_act -
where actno = &4 -
  or emstdate = '&5'
```

You could type the START command as:

```
start selpd (empno actno emptime 90 '1982-06-01')
```

The resulting statement started is:

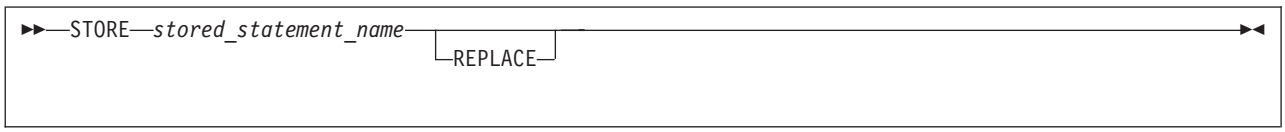
```
select empno,actno,emptime -
from emp_act -
where actno = 90 -
  or emstdate = '1982-06-01'
```

Notice that single quotation marks are needed around the &5 placeholder in the stored SQL statement because that placeholder stands for a character data item. Single quotation marks are not needed around &4 because &4 stands for a numeric data item. Single quotation marks are not needed for &1, &2, and &3 because they do not contain any blanks.

In a stored SQL statement, you can only use the ampersand (&) to create placeholders.

In the above example, the formatting information is not saved because placeholders are used in the SELECT clause.

STORE



STORE is an ISQL command that causes the current SQL (only) statement to be saved for later use. SQL statements remain stored until erased. When storing an SQL statement, you must associate a name with it. Use this name to recall the SQL statement when you want to execute it. The SQL statement, when stored, also remains as the current SQL statement.

Stored statements can be started, listed, renamed, erased, or recalled. When a stored statement is started, renamed, or recalled, it becomes the current SQL statement.

stored_statement_name
is the name you want to use to refer to the stored statement. Names can be up to 8 characters long. The name PREVIOUS is not allowed because ISQL always stores the current statement under that name when a new SQL statement is typed.

REPLACE
specifies that the current statement is to replace any existing stored SQL statement with the same name. If REPLACE is not specified and there is an existing stored SQL statement with the same name, a warning message is issued. This message gives you three options to select from:

- REPLACE, to replace the existing stored SQL statement with the current SQL statement
- END, to end the processing of the STORE command.
- Type a different name to be used as the name for the current SQL statement being stored.

When you store a SELECT statement, related display formatting information is also stored. Formatting is defined by FORMAT commands or current DB2 Server for VSE & VM formatting defaults. This information, which cannot be "seen", remains stored with the SELECT statement and formats the display when the statement is recalled and executed. However, the following exceptions exist:

- Formatting information is not saved when SELECT statements are stored that contain placeholders in the SELECT or FROM clauses. Formatting information *is saved* when the placeholders occur in clauses other than the SELECT and FROM clauses.
- If a stored SELECT command is changed by a CHANGE command, formatting information is saved only when the change occurs in the WHERE, GROUP BY, ORDER BY, or HAVING clauses (that is, when the change does not occur in the SELECT or FROM clauses).
- If a table referenced by a stored query is dropped and later created with different column characteristics (such as DECIMAL instead of INTEGER), existing formatting information is not used.

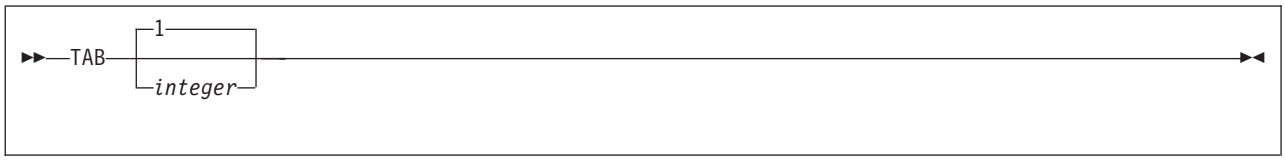
The notation *SELECT * FROM table_name* is not recommended for use in stored SELECT statements. It is possible that someone can add a new column to the table, which causes any stored format information for the SELECT statement to be

erased. By explicitly naming the columns required, or by referring to a view instead of a table (the view can have SELECT *), you can avoid this problem.

The steps necessary to store FORMAT command information along with an SQL statement are:

1. Execute a SELECT statement to retrieve data.
2. Type one or more FORMAT commands to format the display.
3. Type an END command to end the display.
4. Type a STORE command to store the statement (which is still the current SQL statement) and any formatting information.

TAB



TAB is an ISQL display command that enables you to view all the characters of a column that is too wide to fit on the display. It lets you display any adjacent characters of the column; the number of characters displayed is equal to the width of your display. Before issuing TAB, place the desired column at the left edge of the display.

integer
is the number that represents the character's position where the display is to start. If no number is specified, the display starts at the first character position of the column.

The TAB command is valid only for CHAR or VARCHAR columns.

Example

Suppose you are viewing a column whose length attribute is 100 characters. If you are using a 24 x 80 display and you want to view the characters beyond the 80th, you type:

```
tab 81
```

This displays the column, starting with the 81st character at the left edge of the display.

Appendix A. Answers to the Exercises

The following answers are shown with each clause on a separate line so that you can easily check your commands. You can, of course, put the entire command on one line if it fits. Or, you can use several lines and break each line at a different place than shown here. It is your choice. Just follow the rules for using the continuation character.

Exercise 1:

(page 31)

```
1.   forward 39
2.   forward max
3.   right 1     or    column 2
4.   left 1      or    column 1
5.   backward max
6.   print
7.   end
```

Exercise 2:

(page 42)

```
1.   hold select * from department
2.   change /*/&1,&2/;
3.   a.   start (deptno deptname)
     b.   end
4.   change /nt/nt where &3/;
5.   start (deptname mgrno admrdept='e01')
```

Exercise 3:

(page 50)

```
1.   select empno,projno,emptime -
     from emp_act -
     where projno='if1000' or projno='if2000' -
     order by projno,empno
2.   format separator '  *  '
3.   format exclude empno
         OR
     format exclude 1
         OR
     format include only (projno emptime)
         OR
     format include only (2 3)
4.  a.  format column emptime name proptn
            OR
        format column 3 name proptn

    b.  format column proptn width 8
            OR
        format column 3 width 8
```

Note: You can also do items 2 through 4 using a single FORMAT command:

```
format separator ' * ' exclude 1 -
column emptime name proptn width 8
```

Exercise 4:

(page 65)

DB2 Server for VSE

```
1.   set copies 2
2.   list set *
3.   select * -
     from proj_act -
     where projno='ad3112' -
     order by acendate
4.   format group acendate
         AND
     format exclude (acstdate)
         OR
     format group acendate exclude (4)
5.   format ttitle 'personnel programming deadlines'
6.   print
```

DB2 Server for VM

```
1.   list set *
2.   select * -
     from proj_act -
     where projno='ad3112' -
     order by acendate
3.   format group acendate
         AND
     format exclude (acstdate)
         OR
     format group acendate exclude (4)
4.   format ttitle 'personnel programming deadlines'
5.   print copies 2
```

Exercise 5:

(page 72)

```
1.   recall myquery
2.   change /&1/salary between 25000 and 30000 order by 2/
3.   start
4.   format exclude (3)
         OR
     format exclude (midinit)
5.   format separator ' |*| '
6.   format column edlevel name 'school years'
7.   a.  end
     b.  store EXER11
8.   list sql *
9.   help store
```

Exercise 6:

(page 81)

This answer assumes you have a REMARKS column in your ROUTINE table.

```
1.   input routine
     a.  'exer13',10,'select actno,actdesc -',null
         'exer13',20,'from activity',null
     b.  'exer13',30,'format separator 3 blanks',null
     c.  'exer13',40,'display',null
     d.  'exer13',50,'print copies 3',null
     e.  'exer13',60,'end',null
     end
```

Note: Any sequence numbers are valid as long as they are in ascending order.

Appendix B. Sample Tables

The sample tables illustrated in this appendix are used in examples throughout the library. These tables simulate a database created for use in organization or project management applications. As a group, the tables include information that describes employees, departments, projects, and activities. Figure 50 shows the relationships among the tables. These relationships are established by referential constraints, where a foreign key in the dependent table references a primary key in the parent table. In the figure, the referential constraint is symbolized by lines joining the keys; the arrowheads point from the primary key to the foreign key. Only those columns named as foreign or primary keys are listed in the figure. All tables have additional columns. You can easily review the contents of any table by executing an SQL statement, such as `SELECT * FROM SQLDBA.DEPARTMENT`.

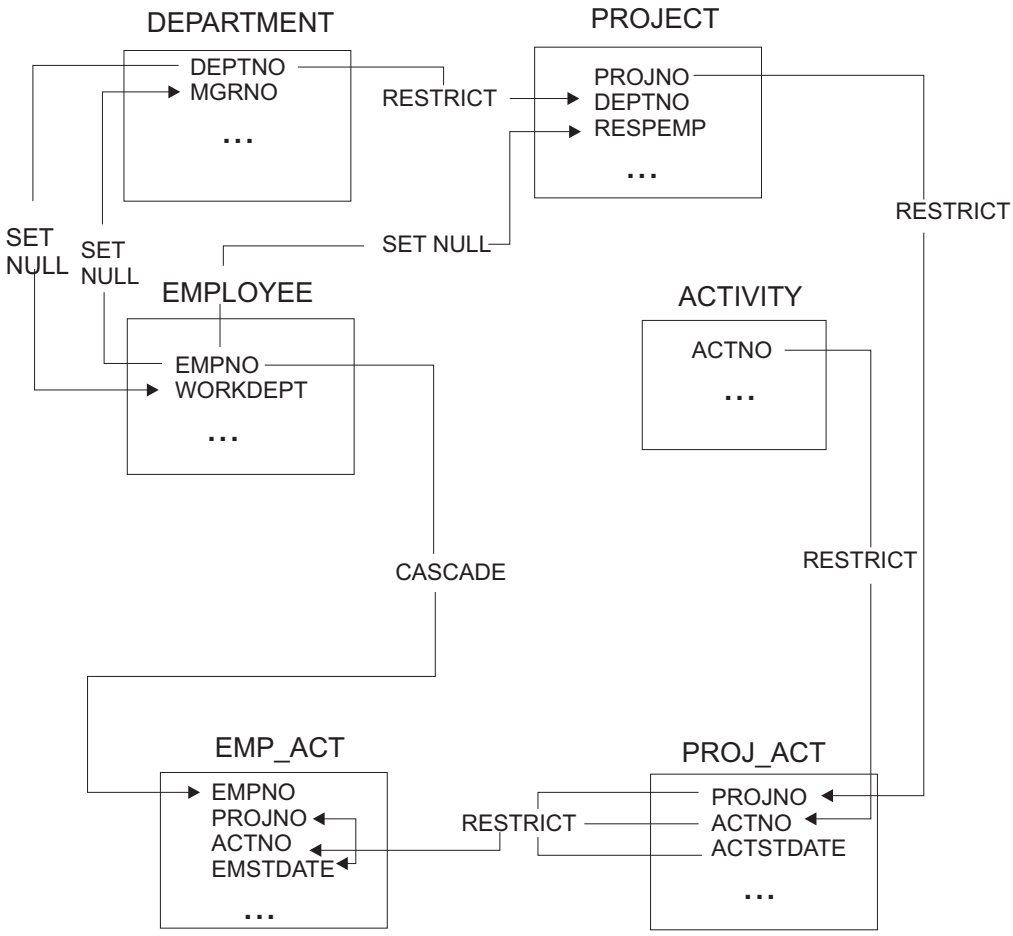


Figure 50. Relationships among Tables in the Sample Application

DEPARTMENT Table

The DEPARTMENT table describes each department in the business and identifies its manager and the department to which it reports. The table contents are shown in Figure 51 on page 170; a description of the columns is shown in Figure 52.

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
A00	SPIFFY COMPUTER SERVICE DIV.	000010	A00
B01	PLANNING	000020	A00
C01	INFORMATION CENTER	000030	A00
D01	DEVELOPMENT CENTER	?	A00
E01	SUPPORT SERVICES	000050	A00
D11	MANUFACTURING SYSTEMS	000060	D01
D21	ADMINISTRATION SYSTEMS	000070	D01
E11	OPERATIONS	000090	E01
E21	SOFTWARE SUPPORT	000100	E01

Figure 51. DEPARTMENT Table Contents

Column Name	Description
DEPTNO	Department number, the primary key
DEPTNAME	A name describing the general activities of the department
MGRNO	Employee number (EMPNO) of the department manager
ADMRDEPT	Number of the department to which this department reports; the department at the highest level reports to itself

Figure 52. Columns of the DEPARTMENT Table

The DEPARTMENT table is created with:

```
CREATE TABLE DEPARTMENT
    (DEPTNO    CHAR(3)          NOT NULL,
     DEPTNAME  VARCHAR(36)      NOT NULL,
     MGRNO     CHAR(6)                 ,
     ADMRDEPT  CHAR(3)          NOT NULL,
     PRIMARY KEY (DEPTNO)              )
```

After the EMPLOYEE table has been created, a foreign key is added to the DEPARTMENT table with this statement:

```
ALTER TABLE DEPARTMENT ADD
     FOREIGN KEY R_EMPLY1 (MGRNO) REFERENCES EMPLOYEE
          ON DELETE SET NULL
```

Relationship to Other Tables

DEPARTMENT is a parent of the EMPLOYEE and PROJECT tables.

The DEPARTMENT table is a dependent of the EMPLOYEE table; the MGRNO column is the foreign key in the DEPARTMENT table and references EMPNO, the primary key in the EMPLOYEE table.

EMPLOYEE Table

The EMPLOYEE table identifies all employees by an employee number and lists basic personnel information. The table in Table 8 on page 172 shows the contents of the EMPLOYEE table; Table 9 on page 174 shows a description of the columns.

Table 8. EMPLOYEE Table Contents

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
char(6) not null	varchar(12) not null	char(1) not null	varchar(15) not null	char(3)	char(4)	date	char(8)	smallint not null	char(1)	date	dec(9,2)	dec(9,2)	dec(9,2)
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750	1000	4220
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250	800	3300
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250	800	3060
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175	800	3214
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250	500	2580
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170	700	2893
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750	600	2380
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150	500	2092
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500	900	3720
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250	600	2340
000130	DOLORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800	500	1904
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420	600	2274
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280	500	2022
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250	400	1780
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680	500	1974
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340	500	1707
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450	400	1636
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740	600	2217
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270	400	1462
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840	600	2387
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180	400	1774
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760	600	2301
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180	400	1534
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250	300	1380
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380	500	2190

Table 8. EMPLOYEE Table Contents (continued)

EMPNO	FIRSTNME	MID INIT	LASTNAME	WORK DEPT	PHONE NO	HIREDATE	JOB	ED LEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250	500	2100
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340	300	1227
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750	400	1420
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900	300	1272
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950	400	1596
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370	500	2030
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840	500	1907

Table 9. Columns of the EMPLOYEE Table

Column Name	Description
EMPNO	Employee number (the primary key)
FIRSTNME	First name of the employee
MIDINIT	Middle initial of the employee
LASTNAME	Last name of the employee
WORKDEPT	Number of department in which the employee works
PHONENO	Employee telephone number
HIREDATE	Date of hire
JOB	Job held by the employee
EDLEVEL	Number of years of formal education
SEX	Sex of the employee (M or F)
BIRTHDATE	Date of birth
SALARY	Yearly salary
BONUS	Yearly bonus
COMM	Yearly commission

The EMPLOYEE table has a foreign key referencing the primary key in the DEPARTMENT table. The DEPARTMENT table must, therefore, be created first. The EMPLOYEE table is then created with:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(6)         NOT NULL,
    FIRSTNME  VARCHAR(12)     NOT NULL,
    MIDINIT   CHAR(1)         NOT NULL,
    LASTNAME  VARCHAR(15)     NOT NULL,
    WORKDEPT  CHAR(3)                 ,
    PHONENO   CHAR(4)                 ,
    HIREDATE  DATE                    ,
    JOB       CHAR(8)                 ,
    EDLEVEL   SMALLINT        NOT NULL,
    SEX       CHAR(1)                 ,
    BIRTHDATE DATE                    ,
    SALARY    DECIMAL(9,2)            ,
    BONUS     DECIMAL(9,2)            ,
    COMM      DECIMAL(9,2)            ,
    PRIMARY KEY (EMPNO)               ,
    FOREIGN KEY R_DEPT1 (WORKDEPT) REFERENCES DEPARTMENT
          ON DELETE SET NULL       )
```

Relationship to Other Tables

The EMPLOYEE table is a parent of the DEPARTMENT table, the PROJECT table, and the EMP_ACT table.

The EMPLOYEE table is a dependent of the DEPARTMENT table; the foreign key on the WORKDEPT column in the EMPLOYEE table references the primary key on the DEPTNO column in the DEPARTMENT table.

PROJECT Table

The PROJECT table describes each project that the business is currently undertaking. Data contained in each row includes the project number, name, person responsible, and schedule dates as shown in Table 10; Table 11 describes the columns.

Table 10. PROJECT Table Contents

PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ
AD3100	ADMIN SERVICES	D01	000010	6.5	1982-01-01	1983-02-01	?
AD3110	GENERAL ADMIN SYSTEMS	D21	000070	6	1982-01-01	1983-02-01	AD3100
AD3111	PAYROLL PROGRAMMING	D21	000230	2	1982-01-01	1983-02-01	AD3110
AD3112	PERSONNEL PROGRAMMING	D21	000250	1	1982-01-01	1983-02-01	AD3110
AD3113	ACCOUNT PROGRAMMING	D21	000270	2	1982-01-01	1983-02-01	AD3110
IF1000	QUERY SERVICES	C01	000030	2	1982-01-01	1983-02-01	?
IF2000	USER EDUCATION	C01	000030	1	1982-01-01	1983-02-01	?
MA2100	WELD LINE AUTOMATION	D01	000010	12	1982-01-01	1983-02-01	?
MA2110	W L PROGRAMMING	D11	000060	9	1982-01-01	1983-02-01	MA2100
MA2111	W L PROGRAM DESIGN	D11	000220	2	1982-01-01	1982-12-01	MA2110
MA2112	W L ROBOT DESIGN	D11	000150	3	1982-01-01	1982-12-01	MA2110
MA2113	W L PROD CONT PROGS	D11	000160	3	1982-02-15	1982-12-01	MA2110
OP1000	OPERATION SUPPORT	E01	000050	6	1982-01-01	1983-02-01	?
OP1010	OPERATION	E11	000090	5	1982-01-01	1983-02-01	OP1000
OP2000	GEN SYSTEMS SERVICES	E01	000050	5	1982-01-01	1983-02-01	?
OP2010	SYSTEMS SUPPORT	E21	000100	4	1982-01-01	1983-02-01	OP2000
OP2011	SCP SYSTEMS SUPPORT	E21	000320	1	1982-01-01	1983-02-01	OP2010
OP2012	APPLICATIONS SUPPORT	E21	000330	1	1982-01-01	1983-02-01	OP2010
OP2013	DB/DC SUPPORT	E21	000340	1	1982-01-01	1983-02-01	OP2010
PL2100	WELD LINE PLANNING	B01	000020	1	1982-01-01	1982-09-15	MA2100

Table 11. Columns of the PROJECT Table

Column Name	Description
PROJNO	Project number (the primary key)
PROJNAME	Project name

Table 11. Columns of the PROJECT Table (continued)

Column Name	Description
DEPTNO	Number of department responsible for the project
RESPEMP	Number of employee responsible for the project
PRSTAFF	Estimated mean project staffing (mean number of persons) needed between PRSTDATE and PRENDATE to achieve the whole project, including any subprojects
PRSTDATE	Estimated project start date
PRENDATE	Estimated project end date
MAJPROJ	Number of any major project of which the subject project may be a part

The PROJECT table has foreign keys referencing DEPARTMENT and EMPLOYEE. The EMPLOYEE and DEPARTMENT tables must be created before the PROJECT table. Once EMPLOYEE and DEPARTMENT are created, the following statement creates the PROJECT table:

```
CREATE TABLE PROJECT
    (PROJNO    CHAR(6)        NOT NULL,
     PROJNAME  VARCHAR(24)    NOT NULL,
     DEPTNO    CHAR(3)        NOT NULL,
     RESPEMP   CHAR(6)                ,
     PRSTAFF   DECIMAL(5,2)           ,
     PRSTDATE  DATE                   ,
     PRENDATE  DATE                   ,
     MAJPROJ   CHAR(6)                ,
     PRIMARY KEY (PROJNO)             ,
     FOREIGN KEY R_DEPT2 (DEPTNO) REFERENCES DEPARTMENT
          ON DELETE RESTRICT          ,
     FOREIGN KEY R_EMPLY2 (RESPEMP) REFERENCES EMPLOYEE
          ON DELETE SET NULL          )
```

Relationship to Other Tables

PROJECT is a parent of the PROJ_ACT table.

PROJECT is a dependent of:

- The DEPARTMENT table; the foreign key on the DEPTNO column in PROJECT references the primary key in the DEPARTMENT table.
- The EMPLOYEE table; the foreign key on the RESPEMP column in PROJECT references the primary key in the EMPLOYEE table.

ACTIVITY Table

The ACTIVITY tables describes the activities that can be performed during a project. The table acts as a master list of possible activities, identifying the activity number, and providing a description of the activity. Figure 53 on page 177 shows table contents; Figure 54 on page 177 shows a description of the columns.

ACTNO	ACTKWD	ACTDESC
160	ADMDB	Adm databases
170	ADMDC	Adm data comm
90	ADMQS	Adm query system
150	ADMSYS	Adm operating sys
70	CODE	Code programs
110	COURSE	Develop courses
30	DEFINE	Define specs
180	DOC	Document
20	ECOST	Estimate cost
40	LEADPR	Lead program/design
60	LOGIC	Describe logic
140	MAINT	Maint software sys
10	MANAGE	Manage/advise
130	OPERAT	Oper computer sys
50	SPECS	Write specs
120	STAFF	Pers and staffing
100	TEACH	Teach classes
80	TEST	Test programs

Figure 53. ACTIVITY Table Contents

Column Name	Description
ACTNO	Activity number (the primary key)
ACTKWD	Activity keyword (up to six characters)
ACTDESC	Activity description

Figure 54. Columns of the ACTIVITY Table

The ACTIVITY table is created with:

```
CREATE TABLE ACTIVITY
     (ACTNO    SMALLINT       NOT NULL,
      ACTKWD   CHAR(6)        NOT NULL,
      ACTDESC  VARCHAR(20)    NOT NULL,
      PRIMARY KEY (ACTNO)             )
```

Relationship to Other Tables

ACTIVITY is a parent of the PROJ_ACT table.

PROJ_ACT Table

The PROJ_ACT table lists the activities performed for each project. The table contains information on the start and completion dates of the project activity as well as staffing requirements as shown in Figure 55 on page 178. Figure 56 on page 179 shows a description of the columns.

PROJNO	ACTNO	ACSTAFF	ACSTDATE	ACENDATE
AD3100	10	0.50	1982-01-01	1982-07-01
AD3110	10	1.00	1982-01-01	1983-01-01
AD3111	60	0.80	1982-01-01	1982-04-15
AD3111	70	1.50	1982-02-15	1982-10-15
AD3111	80	1.25	1982-04-15	1983-01-15
AD3111	180	1.00	1982-10-15	1983-01-15
AD3112	60	0.75	1982-01-01	1982-05-15
AD3112	60	0.75	1982-12-01	1983-01-01
AD3112	70	0.75	1982-01-01	1982-10-15
AD3112	80	0.35	1982-08-15	1982-12-01
AD3112	180	0.50	1982-08-15	1983-01-01
AD3113	60	0.75	1982-03-01	1982-10-15
AD3113	70	1.25	1982-06-01	1982-12-15
AD3113	80	1.75	1982-01-01	1982-04-15
AD3113	180	0.75	1982-03-01	1982-07-01
IF1000	10	0.50	1982-01-01	1983-01-01
IF1000	90	1.00	1982-01-01	1983-01-01
IF1000	100	0.50	1982-01-01	1983-01-01
IF2000	10	0.50	1982-01-01	1983-01-01
IF2000	100	0.75	1982-01-01	1982-07-01
IF2000	110	0.50	1982-03-01	1982-07-01
IF2000	110	0.50	1982-10-01	1983-01-01
MA2100	10	0.50	1982-01-01	1982-11-01
MA2100	20	1.00	1982-01-01	1982-03-01
MA2110	10	1.00	1982-01-01	1983-02-01
MA2111	40	1.00	1982-01-01	1983-02-01
MA2111	50	1.00	1982-01-01	1092-06-01
MA2111	60	1.00	1982-06-01	1983-02-01
MA2112	60	2.00	1982-01-01	1982-07-01
MA2112	70	1.50	1983-02-01	1983-02-01
⋮	⋮	⋮	⋮	⋮

Figure 55. Partial Contents of PROJ_ACT Table

Column Name	Description
PROJNO	Project number
ACTNO	Activity number
ACSTAFF	Estimated mean number of employees needed to staff the activity
ACSTDATE	Estimated activity start date
ACENDATE	Estimated activity completion date

Figure 56. Columns of the PROJ_ACT Table

The table has a composite primary key and was created with:

```
CREATE TABLE PROJ_ACT
     (PROJNO     CHAR(6)      NOT NULL,
      ACTNO      SMALLINT     NOT NULL,
      ACSTAFF    DECIMAL(5,2)         ,
      ACSTDATE   DATE         NOT NULL,
      ACENDATE   DATE                  ,
      PRIMARY KEY (PROJNO, ACTNO, ACTSTDATE),
      FOREIGN KEY R_PROJ2 (PROJNO) REFERENCES PROJECT
              ON DELETE RESTRICT,
      FOREIGN KEY R_ACTIV (ACTNO) REFERENCE ACTIVITY
              ON DELETE RESTRICT)
```

Relationship to Other Tables

PROJ_ACT is a parent of the EMP_ACT table.

It is a dependent of:

- The ACTIVITY table; the foreign key on ACTNO in the PROJ_ACT table references the primary key, ACTNO, in the ACTIVITY table.
- The PROJECT table; the foreign key on PROJNO in the PROJ_ACT table references the primary key, PROJNO, in the PROJECT table.

EMP_ACT Table

The EMP_ACT table identifies the employee performing each activity listed for each project. The table in Figure 57 on page 180 shows some of the rows in this table. Figure 58 on page 180 shows a description of the columns.

EMPNO	PROJNO	ACTNO	EMPTIME	EMSTDATE	EMENDATE
000130	IF1000	90	1.00	1982-01-01	1982-10-01
000130	IF1000	100	.50	1982-10-01	1983-01-01
000140	IF1000	90	.50	1982-10-01	1983-01-01
000030	IF1000	10	.50	1982-06-01	1983-01-01
000030	IF2000	10	.50	1982-01-01	1983-01-01
000140	IF2000	100	1.00	1982-01-01	1982-03-01
000140	IF2000	100	.50	1982-03-01	1982-07-01
000140	IF2000	110	.50	1982-03-01	1982-07-01
000140	IF2000	110	.50	1982-10-01	1983-01-01
000010	MA2100	10	.50	1982-01-01	1982-11-01
000110	MA2100	20	1.00	1982-01-01	1982-03-01
000020	PL2100	30	1.00	1982-01-01	1982-09-15
000010	MA2110	10	1.00	1982-01-01	1983-02-01
000220	MA2111	40	1.00	1982-01-01	1983-02-01
⋮	⋮	⋮	⋮	⋮	⋮

Figure 57. Partial Contents of EMP_ACT Table

Column Name	Description
EMPNO	Employee number
PROJNO	Project number of the project to which the employee is assigned
ACTNO	Activity number within a project to which an employee is assigned
EMPTIME	A proportion of the employee's full time (between 0.00 and 1.00) to be spent on the project from EMSTDATE to EMENDATE
EMSTDATE	Date the activity starts
EMENDATE	Completion date of the activity

Figure 58. Columns of the EMP_ACT Table

Since the table has foreign keys referencing EMPLOYEE and PROJ_ACT, those tables must be created first.

This table was created with:

```
CREATE TABLE EMP_ACT
      (EMPNO     CHAR(6)          NOT NULL,
       PROJNO    CHAR(6)          NOT NULL,
       ACTNO     SMALLINT         NOT NULL,
       EMPTIME   DECIMAL(5,2)             ,
       EMSTDATE  DATE                     ,
       EMENDATE  DATE                     ,
       FOREIGN KEY R_PROACT (PROJNO,ACTNO,EMSTDATE)
```

```
        REFERENCES PROJ_ACT ON DELETE RESTRICT,
FOREIGN KEY R_EMPLY3 (EMPNO) REFERENCES EMPLOYEE
        ON DELETE CASCADE          )
```

Relationship to Other Tables

The EMP_ACT table is a dependent of:

- The EMPLOYEE table; the foreign key on EMPNO in the EMP_ACT table references the primary key, EMPNO, in the EMPLOYEE table.
- The PROJ_ACT table; the foreign key on the set of PROJNO, ACTNO, EMSTDATE in the EMP_ACT table references the primary key, PROJNO, ACTNO, ACSTDATE, in the PROJ_ACT table.

IN_TRAY Table

The IN_TRAY table contains a person's note log. The table contents are shown in Figure 59; a description of the columns is shown in Figure 60.

RECEIVED	SOURCE	SUBJECT	NOTE_TEXT
1965-01-01-07.00.00	SQLDBA	English	Here is a note from your DBA.

Figure 59. IN_TRAY Table Contents

Column Name	Description
RECEIVED	Date and time note was received
SOURCE	User id of person sending note
SUBJECT	Brief description
NOTE_TEXT	The text of the note

Figure 60. Columns of the IN_TRAY Table

This table was created with:

```
CREATE TABLE IN_TRAY
     (RECEIVED     TIMESTAMP   NOT NULL,
      SOURCE       CHAR(8)     NOT NULL,
      SUBJECT      CHAR(64)              ,
      NOTE_TEXT    VARCHAR(4000)       )
```

CL_SCHED Table

The CL_SCHED table describes a classroom schedule. The table contents are shown in Figure 61; a description of the columns is shown in Figure 62 on page 182.

CLASS_CODE	DAY	STARTING	ENDING
101:KAR	2	14.10.00	16.10.00
202:LMM	3	14.40.00	16.40.00
303:RAR	4	09.00.00	09.40.00

Figure 61. CL_SCHED Table Contents

Column Name	Description
CLASS_CODE	Class Code (room:teacher)
DAY	Day number of four day schedule
STARTING	Class start time
ENDING	Class end time

Figure 62. Columns of the CL_SCHED Table

This table was created with:

```
CREATE TABLE CL_SCHED
     (CLASS_CODE      CHAR(7)  NOT NULL,
      DAY             SMALLINT NOT NULL,
      STARTING        TIME     NOT NULL,
      ENDING          TIME     NOT NULL)
```

Note: For more information about data types, refer to the *DB2 Server for VSE & VM Application Programming* manual.

Appendix C. Summary of ISQL PF Keys

Listed below is a summary of the default PF key functions:

PF1, PF13 Issues a HELP command, which retrieves an explanation of how to use HELP information and a list of the topics available.

PF2, PF14 Issues a START command, which starts the statement in the SQL command buffer (the current SQL statement).

PF3, PF15 Issues an END command.

PF4, PF16 Issues a PRINT command, which requests the currently displayed query result to be printed on the designated printer.

PF5, PF17 Issues a RECALL command, which displays the contents of the SQL command buffer.

PF6, PF18 Not assigned.

PF7, PF19 Issues a BACKWARD command, which moves your view of the query result backward one-half display.

PF8, PF20 Issues a FORWARD command, which moves your view of the query result forward one-half display.

PF9, PF21 Issues a HOLD command.

PF10, PF22 Issues a LEFT 1 command, which moves your view of the query result one column to the left.

PF11, PF23 Issues a RIGHT 1 command, which moves your view of the query result one column to the right.

PF12, PF24 Issues a RETRIEVE command, which displays the last input line from the SQL command buffer and places it in the input area. Each successive use of RETRIEVE retrieves an earlier input line. When there are no more lines in the SQL command buffer to be retrieved, the newest line (the last one entered in the buffer) is again retrieved.

Using PF Keys in CMS FULLSCREEN Mode (DB2 Server for VM)

If you are using ISQL in CMS FULLSCREEN mode, you may want to reset the functions of some PF keys. You can reset your keys by executing a routine. The routine can be a PROFILE routine which is executed automatically every time you start ISQL. To reset the PF key functions, you can include the following example commands in the routine:

```
CMS
SET LINEND OFF
SET CMSPF 07 BACKWARD NOECHO #WM SCROLL BACKWARD CMS 1
SET CMSPF 08 FORWARD NOECHO #WM SCROLL FORWARD CMS 1
SET LINEND ON
RETURN
```

This addition to your routine lets you use PF7 and PF8 in ISQL command mode to scroll forward and backward in a manner similar to the CMS FULLSCREEN mode.

For information about creating and running routines, see “Chapter 7. Creating and Using Routines” on page 73.

You can also reset the PF keys every time you use ISQL. For more information on resetting the PF keys, see “Routines to Which Parameters Can Be Passed (DB2 Server for VM)” on page 73.

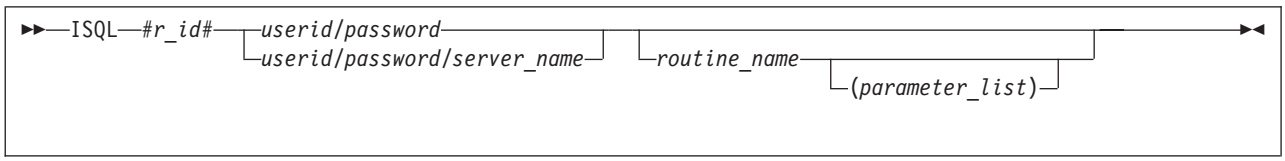
Appendix D. Summary of SQL Statements for Interactive Use

For syntax diagrams, semantic descriptions, rules, and situations where you would use the following SQL statements, refer to the *DB2 Server for VSE & VM SQL Reference* manual.

- ACQUIRE DBSPACE
- ALTER DBSPACE
- ALTER TABLE
- COMMENT ON
- COMMIT
- CONNECT
- CREATE INDEX
- CREATE SYNONYM
- CREATE TABLE
- CREATE VIEW
- DELETE (searched DELETE only)
- DROP DBSPACE
- DROP INDEX
- DROP PROGRAM
- DROP SYNONYM
- DROP TABLE
- DROP VIEW
- EXPLAIN
- GRANT PROGRAM PRIVILEGES
- GRANT SYSTEM AUTHORITIES
- GRANT TABLE OR VIEW PRIVILEGES
- INSERT
- Interactive select (see SELECT in the *DB2 Server for VSE & VM SQL Reference* manual)
- LABEL ON
- LOCK DBSPACE
- LOCK TABLE
- REVOKE
- ROLLBACK
- UPDATE (searched UPDATE only)
- UPDATE STATISTICS

Appendix E. Suppressing the ISQL Sign-On Display for DB2 Server for VSE

In addition to typing **isql** at a CICS terminal, to start ISQL, you can type the following command at a CICS terminal. This command suppresses the ISQL signon display and related terminal messages.



Where:

r_id	Is a 1–4 character CICS transaction identifier or 1 to 4 blanks.
#	Stands for a hexadecimal byte X'FF' positioned immediately before and after r_id to mark the beginning and end of r_id.
userid/password	Is the ISQL signon user ID and password. You must type the slash (/). Since the *server_name* is not specified, the *userid/password* will be used to connect to the default server.
userid/password/server_name	Is the ISQL signon user ID, password, and the server-name. You must type the slash (/). The *userid/password* will be used to connect to the specified server.
routine_name	Is optional. Refer to "Using the ISQL Transaction Identifier (DB2 Server for VSE)" on page 74.
(*parameter_list***)**	Is optional. Refer to "Using the ISQL Transaction Identifier (DB2 Server for VSE)" on page 74.

This method of invoking ISQL is primarily designed for VSE system programs:

- Enter into an ISQL session directly from another interactive session. The ISQL signon display is suppressed so that the user of the interactive session can enter an ISQL session without doing the formal ISQL signon steps. The user ID and password supplied in the command are processed as if they were supplied by the signon display.
- Return to the interactive session when the ISQL session ends, passing any ISQL ending message to the interactive session. Before ISQL is ended, it starts the CICS transaction identified by *r_id* (provided *r_id* is not all blank characters) using the CICS START command. Ending messages from ISQL are passed as data in the START command. ISQL returns to CICS if *r_id* is blank.

The hexadecimal byte X'FF' is usually not available with terminal keyboards. The sample program, as shown in Figure 63 on page 188, illustrates one method of displaying the command on the terminal and prompting the user to start ISQL.

```
         TITLE 'STARTING ISQL WITHOUT THE SIGN-ON SCREEN'
*****************************************************************
* THIS PROGRAM WRITES 2 LINES TO THE USER TERMINAL (24X80):     *
* - LINE 1 IS FOR SETTING UP THE INVOCATION OF ISQL WITHOUT     *
*   DISPLAYING THE SIGN-ON SCREEN.                              *
* - LINE 2 PROMPTS THE USER TO EXECUTE LINE 1 WITH THE ENTER    *
*   KEY, OR QUIT WITH THE CLEAR KEY.                            *
*****************************************************************
         PRINT GEN
DFHEISTG DSECT
INSTRUCT CSECT
         SPACE
         EXEC CICS SEND FROM(ISQLSTR) FLENGTH(SENDLEN) ERASE
         EXEC CICS RETURN
ISQLSTR  EQU   *
LINE1    DC    X'1140401D4D'   LINE 1: INVISIBLE, MDT ON,
*                                      UNPROTECTED.
         DC    C'ISQL'
         DC    X'FF'           DELIMITER BYTE
         DC    C'TRX0'         TRANS-ID TO BE INVOKED AT ISQL
*                                                    EXIT.
         DC    X'FF'           DELIMITER BYTE
         DC    C'SQLDBA/SQLDBAPW'      USER-ID/PASSWORD TO ISQL
*
```

Figure 63. Starting ISQL Without the Sign-on Display (Part 1 of 2)

```
LINE2    DC    X'11C1501D40'    LINE 2: VISIBLE, MDT OFF,
*                                       UNPROTECTED.
         DC    C'PRESS ENTER KEY TO INVOKE ISQL,'
         DC    C' OR CLEAR KEY TO QUIT'
STRLEN   EQU   *-ISQLSTR
SENDLEN  DC    A(STRLEN)
         LTORG
         END
```

Figure 63. Starting ISQL Without the Sign-on Display (Part 2 of 2)

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

ACF/VTAM
IBMLink
C/370
CICS
CICS/VSE
DATABASE 2
DB2
Distributed Relational Database Architecture
DRDA
IBM
MVS
QMF
SQL/DS
VM/ESA
VSE/ESA

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

This bibliography lists publications that are referenced in this manual or that may be helpful.

DB2 Server for VM Publications

- *DB2 Server for VSE & VM Application Programming*, SC09-2889
- *DB2 Server for VSE & VM Database Administration*, SC09-2888
- *DB2 Server for VSE & VM Database Services Utility*, SC09-2983
- *DB2 Server for VSE & VM Diagnosis Guide and Reference*, LC09-2907
- *DB2 Server for VSE & VM Overivew*, GC09-2995
- *DB2 Server for VSE & VM Interactive SQL Guide and Reference*, SC09-2990
- *DB2 Server for VSE & VM Master Index and Glossary*, SC09-2890
- *DB2 Server for VM Messages and Codes*, GC09-2984
- *DB2 Server for VSE & VM Operation*, SC09-2986
- *DB2 Server for VSE & VM Quick Reference*, SC09-2988
- *DB2 Server for VM System Administration*, SC09-2980
- *DB2 Server for VSE & VM Performance Tuning Handbook*, GC09-2987
- *DB2 Server for VSE & VM SQL Reference*, SC09-2989

DB2 Server for VSE Publications

- *DB2 Server for VSE & VM Application Programming*, SC09-2889
- *DB2 Server for VSE & VM Database Administration*, SC09-2888
- *DB2 Server for VSE & VM Database Services Utility*, SC09-2983
- *DB2 Server for VSE & VM Diagnosis Guide and Reference*, LC09-2907
- *DB2 Server for VSE & VM Overivew*, GC09-2995
- *DB2 Server for VSE & VM Interactive SQL Guide and Reference*, SC09-2990
- *DB2 Server for VSE & VM Master Index and Glossary*, SC09-2890
- *DB2 Server for VSE Messages and Codes*, GC09-2985
- *DB2 Server for VSE & VM Operation*, SC09-2986
- *DB2 Server for VSE System Administration*, SC09-2981
- *DB2 Server for VSE & VM Performance Tuning Handbook*, GC09-2987
- *DB2 Server for VSE & VM SQL Reference*, SC09-2989

Related Publications

- *DB2 Server for VSE & VM Data Restore*, SC09-2991
- *DRDA: Every Manager's Guide*, GC26-3195
- *IBM SQL Reference, Version 2, Volume 1*, SC26-8416
- *IBM SQL Reference*, SC26-8415

VM/ESA Publications

- *VM/ESA: General Information*, GC24-5745
- *VM/ESA: VMSES/E Introduction and Reference*, GC24-5837
- *VM/ESA: Installation Guide*, GC24-5836
- *VM/ESA: Service Guide*, GC24-5838
- *VM/ESA: Planning and Administration*, SC24-5750
- *VM/ESA: CMS File Pool Planning, Administration, and Operation*, SC24-5751
- *VM/ESA: REXX/EXEC Migration Tool for VM/ESA*, GC24-5752
- *VM/ESA: Conversion Guide and Notebook*, GC24-5839
- *VM/ESA: Running Guest Operating Systems*, SC24-5755
- *VM/ESA: Connectivity Planning, Administration, and Operation*, SC24-5756
- *VM/ESA: Group Control System*, SC24-5757
- *VM/ESA: System Operation*, SC24-5758
- *VM/ESA: Virtual Machine Operation*, SC24-5759
- *VM/ESA: CP Programming Services*, SC24-5760
- *VM/ESA: CMS Application Development Guide*, SC24-5761
- *VM/ESA: CMS Application Development Reference*, SC24-5762
- *VM/ESA: CMS Application Development Guide for Assembler*, SC24-5763
- *VM/ESA: CMS Application Development Reference for Assembler*, SC24-5764

- *VM/ESA: CMS Application Multitasking*, SC24-5766
- *VM/ESA: CP Command and Utility Reference*, SC24-5773
- *VM/ESA: CMS Primer*, SC24-5458
- *VM/ESA: CMS User's Guide*, SC24-5775
- *VM/ESA: CMS Command Reference*, SC24-5776
- *VM/ESA: CMS Pipelines User's Guide*, SC24-5777
- *VM/ESA: CMS Pipelines Reference*, SC24-5778
- *VM/ESA: XEDIT User's Guide*, SC24-5779
- *VM/ESA: XEDIT Command and Macro Reference*, SC24-5780
- *VM/ESA: Quick Reference*, SX24-5290
- *VM/ESA: Performance*, SC24-5782
- *VM/ESA: Dump Viewing Facility*, GC24-5853
- *VM/ESA: System Messages and Codes*, GC24-5841
- *VM/ESA: Diagnosis Guide*, GC24-5854
- *VM/ESA: CP Diagnosis Reference*, SC24-5855
- *VM/ESA: CP Diagnosis Reference Summary*, SX24-5292
- *VM/ESA: CMS Diagnosis Reference*, SC24-5857
- CP and CMS control block information is not provided in book form. This information is available on the IBM VM/ESA operating system home page (http://www.ibm.com/s390/vm).
- *IBM VM/ESA: CP Exit Customization*, SC24-5672
- *VM/ESA REXX/VM User's Guide*, SC24-5465
- *VM/ESA REXX/VM Reference*, SC24-5770

C for VM/ESA Publications

- *IBM C for VM/ESA Diagnosis Guide*, SC09-2149
- *IBM C for VM/ESA Language Reference*, SC09-2153
- *IBM C for VM/ESA Compiler and Run-Time Migration Guide*, SC09-2147
- *IBM C for VM/ESA Programming Guide*, SC09-2151
- *IBM C for VM/ESA User's Guide*, SC09-2152

Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA) Publications

- *IBM VSE/ESA Administration*, SC33-6505
- *IBM VSE/ESA Diagnosis Tools*, SC33-6514
- *IBM VSE/ESA General Information*, GC33-6501
- *IBM VSE/ESA Guide for Solving Problems*, SC33-6510
- *IBM VSE/ESA Guide to System Functions*, SC33-6511
- *IBM VSE/ESA Installation*, SC33-6504
- *IBM VSE/ESA Messages & Codes*, SC33-6507
- *IBM VSE/ESA Networking Support*, SC33-6508
- *IBM VSE/ESA Operation*, SC33-6506
- *IBM VSE/ESA Planning*, SC33-6503
- *IBM VSE/ESA System Control Statements*, SC33-6513
- *IBM VSE/ESA System Macros User's Guide*, SC33-6515
- *IBM VSE/ESA System Macros Reference*, SC33-6516
- *IBM VSE/ESA System Utilities*, SC33-6517
- *IBM VSE/ESA Unattended Node Support*, SC33-6512
- *IBM VSE/ESA Using IBM Workstations*, SC33-6509

CICS/VSE Publications

- *CICS/VSE Application Programming Reference*, SC33-0713
- *CICS/VSE Application Programming Guide*, SC33-0712
- *CICS Application Programming Primer (VS COBOL II)*, SC33-0674
- *CICS/VSE CICS-Supplied Transactions*, SC33-0710
- *CICS/VSE Customization Guide*, SC33-0707
- *CICS/VSE Facilities and Planning Guide*, SC33-0718
- *CICS/VSE Intercommunication Guide*, SC33-0701
- *CICS/VSE Performance Guide*, SC33-0703
- *CICS/VSE Problem Determination Guide*, SC33-0716
- *CICS/VSE Recovery and Restart Guide*, SC33-0702
- *CICS/VSE Release Guide*, GC33-1645
- *CICS/VSE Report Controller User's Guide*, SC33-0705
- *CICS/VSE Resource Definition (Macro)*, SC33-0709
- *CICS/VSE Resource Definition (Online)*, SC33-0708
- *CICS/VSE System Definition and Operations Guide*, SC33-0706
- *CICS/VSE System Programming Reference*, SC33-0711
- *CICS/VSE User's Handbook*, SX33-6079
- *CICS/VSE XRF Guide*, SC33-0704

CICS/ESA Publications

- *CICS/ESA General Information*, GC33-0803

VSE/Virtual Storage Access Method (VSE/VSAM) Publications

- *VSE/VSAM Commands and Macros*, SC33-6532
- *VSE/VSAM Introduction*, GC33-6531
- *VSE/VSAM Messages and Codes*, SC24-5146
- *VSE/VSAM Programmer's Reference*, SC33-6535

VSE/Interactive Computing and Control Facility (VSE/ICCF) Publications

- *VSE/ICCF Administration and Operation*, SC33-6562
- *VSE/ICCF Primer*, SC33-6561
- *VSE/ICCF User's Guide*, SC33-6563

VSE/POWER Publications

- *VSE/POWER Administration and Operation*, SC33-6571
- *VSE/POWER Application Programming*, SC33-6574
- *VSE/POWER Networking*, SC33-6573
- *VSE/POWER Remote Job Entry*, SC33-6572

Distributed Relational Database Architecture (DRDA) Library

- *Application Programming Guide*, SC26-4773
- *Architecture Reference*, SC26-4651
- *Connectivity Guide*, SC26-4783
- *DRDA: Every Manager's Guide*, GC26-3195
- *Planning for Distributed Relational Database*, SC26-4650
- *Problem Determination Guide*, SC26-4782

C/370 for VSE Publications

- *IBM C/370 General Information*, GC09-1386
- *IBM C/370 Programming Guide for VSE*, SC09-1399
- *IBM C/370 Installation and Customization Guide for VSE*, GC09-1417
- *IBM C/370 Reference Summary for VSE*, SX09-1246
- *IBM C/370 Diagnosis Guide and Reference for VSE*, LY09-1805

VSE/REXX Publication

- *VSE/REXX Reference*, SC33-6642

Other Distributed Data Publications

- *IBM Distributed Data Management (DDM) Architecture, Architecture Reference, Level 4*, SC21-9526
- *IBM Distributed Data Management (DDM) Architecture, Implementation Programmer's Guide*, SC21-9529
- *VM/Directory Maintenance Licensed Program Specification*, GC20-1836
- *IBM Distributed Relational Database Architecture Reference*, SC26-4651
- *IBM Systems Network Architecture, Format and Protocol Reference*, SC30-3112
- *SNA LU 6.2 Reference: Peer Protocols*, SC31-6808
- *Reference Manual: Architecture Logic for LU Type 6.2*, SC30-3269
- *IBM Systems Network Architecture, Logical Unit 6.2 Reference: Peer Protocols*, SC31-6808
- *Distributed Data Management (DDM) General Information*, GC21-9527

CCSID Publications

- *Character Data Representation Architecture, Executive Overview*, GC09-2207
- *Character Data Representation Architecture Reference and Registry*, SC09-2190

DB2 Server RXSQL Publications

- *DB2 REXX SQL for VM/ESA Installation and Reference*, SC09-2891

C/370 Publications

- *IBM C/370 Installation and Customization Guide*, GC09-1387
- *IBM C/370 Programming Guide*, SC09-1384

Communication Server for OS/2 Publications

- *Up and Running!*, GC31-8189
- *Network Administration and Subsystem Management Guide*, SC31-8181
- *Command Reference*, SC31-8183
- *Message Reference*, SC31-8185
- *Problem Determination Guide*, SC31-8186

Distributed Database Connection Services (DDCS) Publications

- *DDCS User's Guide for Common Servers*, S20H-4793
- *DDCS for OS/2 Installation and Configuration Guide*, S20H-4795

VTAM Publications

- *VTAM Messages and Codes*, SC31-6493
- *VTAM Network Implementation Guide*, SC31-6494
- *VTAM Operation*, SC31-6495
- *VTAM Programming*, SC31-6496
- *VTAM Programming for LU 6.2*, SC31-6497
- *VTAM Resource Definition Reference*, SC31-6498
- *VTAM Resource Definition Samples*, SC31-6499

CSP/AD and CSP/AE Publications

- *Developing Applications*, SH20-6435
- *CSP/AD and CSP/AE Installation Planning Guide*, GH20-6764
- *Administering CSP/AD and CSP/AE on VM*, SH20-6766
- *Administering CSP/AD and CSP/AE on VSE*, SH20-6767
- *CSP/AD and CSP/AE Planning*, SH20-6770
- *Cross System Product General Information*, GH23-0500

Query Management Facility (QMF) Publications

- *Introducing QMF*, GC27-0714
- *Installing and Managing QMF for VSE*, GC27-0721
- *QMF Reference*, SC27-0715
- *Installing and Managing QMF for VM*, GC27-0720
- *Developing QMF Applications*, SC27-0718
- *QMF Messages and Codes*, GC27-0717
- *Using QMF*, SC27-0716

Query Management Facility (QMF) for Windows Publications

- *Getting Started with QMF for Windows*, SC27-0723
- *Installing and Managing QMF for Windows*, GC27-0722

DL/I DOS/VS Publications

- *DL/I DOS/VS Application Programming*, SH24-5009

COBOL Publications

- *VS COBOL II Migration Guide for VSE*, GC26-3150
- *VS COBOL II Migration Guide for MVS and CMS*, GC26-3151
- *VS COBOL II General Information*, GC26-4042
- *VS COBOL II Language Reference*, GC26-4047
- *VS COBOL II Application Programming Guide*, SC26-4045
- *VS COBOL II Application Programming Debugging*, SC26-4049
- *VS COBOL II Installation and Customization for CMS*, SC26-4213
- *VS COBOL II Installation and Customization for VSE*, SC26-4696
- *VS COBOL II Application Programming Guide for VSE*, SC26-4697

Data Facility Storage Management Subsystem/VM (DFSMS/VM) Publications

- *DFSMS/VM RMS User's Guide and Reference*, SC35-0141

Systems Network Architecture (SNA) Publications

- *SNA Transaction Programmer's Reference Manual for LU Type 6.2*, GC30-3084
- *SNA Format and Protocol Reference: Architecture Logic for LU Type 6.2*, SC30-3269
- *SNA LU 6.2 Reference: Peer Protocols*, SC31-6808
- *SNA Synch Point Services Architecture Reference*, SC31-8134

Miscellaneous Publications

- *IBM 3990 Storage Control Planning, Installation, and Storage Administration Guide*, GA32-0100
- *Dictionary of Computing*, ZC20-1699
- *APL2 Programming: Using Structured Query Language*, SH21-1056
- *ESA/390 Principles of Operation*, SA22-7201

Related Feature Publications

- *DB2 for VM Control Center Operations Guide*, GC09-2993
- *DB2 for VSE Control Center Operations Guide*, GC09-2992
- *DB2 Replication Guide and Reference*, SC26-9920

Index

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Readers' Comments — We'd Like to Hear from You

DB2 Server for VSE & VM
Interactive SQL Guide and
Reference
Version 7 Release 1

Publication No. SC09-2990-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

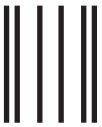
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC09-2990-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CANADA LTD.
DB2 Server for VSE & VM
2S/240/1150/TOR
1150 Eglinton Avenue East
North York, Ontario, Canada M3C 1H7

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

File Number: S370/4300-50
Program Number: 5697-F42

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC09-2990-00

Spine information:

DB2 Server for VSE & VM Interactive SQL Guide and Reference Version 7 Release 1