RISQL Entry Tool and RISQL Reporter

User's Guide

Informix Red Brick Decision Server

Version 6.0
November 1999
Part No. 000-6372

Published by Informix® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

Documentation Team: Twila Booth, Kathy Eckardt, Laura Kremers, Jerry Tattershall

Table of Contents

Introduction

In This Introduction . . . 3
About This Guide . . . 3
  Types of Users . . . 3
  Software Dependencies . . . 4
New Features . . . 5
Documentation Conventions . . . 5
  Syntax Notation . . . 6
  Syntax Diagrams . . . 7
  Keywords and Punctuation . . . 9
  Identifiers and Names . . . 9
  Icon Conventions . . . 10
Customer Support . . . 11
  New Cases . . . 12
  Existing Cases . . . 12
  Troubleshooting Tips . . . 13
Related Documentation . . . 14
Additional Documentation . . . 16
  Online Manuals . . . 16
  Printed Manuals . . . 17
Informix Welcomes Your Comments . . . 17

Chapter 1 Introduction to the RISQL Entry Tool and RISQL Reporter

In This Chapter . . . 1-3
The RISQL Entry Tool . . . 1-4
The RISQL Reporter . . . 1-4
The Informix Red Brick Environment . . . 1-4

Chapter 2 Using the RISQL Entry Tool and RISQL Reporter

In This Chapter 2-3
Identifying System Shell Prompts 2-4
Accessing Databases 2-4
Defining a Locale 2-5
Starting the Tool 2-5
Usage Notes 2-8
Valid Input 2-8
Output Columns 2-8
Command Syntax 2-9
New-Line Character. 2-9
.rbretrc file 2-10
Error Messages 2-10
Connecting to a Database 2-10
Connecting with the RB_PATH Environment Variable 2-11
Connecting on the Command Line. 2-12
Connecting with the SQL CONNECT Command. 2-13
Ending Your Tool Session 2-15
Submitting Queries to the Database 2-15
Interrupting Command Execution 2-17
Displaying Statistics 2-17
Using File Redirection 2-19
Using System Redirection. 2-19
Using Redirection Within the Tool 2-20
Using an Editor 2-26
Changing the Default Editor 2-26
Starting an Editor 2-27
Adding Comments to Statements 2-32
Accessing the System Shell 2-33
Changing the Tool Prompt 2-35
Changing Database Passwords 2-36
Administering the Database 2-36
Defining Database Objects 2-37
Controlling Database and Table Access 2-37
Changing Information in a Database 2-38
Creating Data Extraction Applications 2-38
Redirecting Extraction Results from the Shell 2-39
Redirecting Extraction Results from Within the Tool. . . . 2-39
Exit Status Codes. 2-40

Chapter 3 **Formatting Reports**

In This Chapter 3-3
Laying Out Reports 3-4
Using the SET COLUMN Commands 3-4
  Column Specification 3-5
  Column Variable Precedence 3-5
  Column Display Options 3-6
Saving Command Settings 3-8
Creating a Report Title 3-9
Defining the Page Length 3-10
Skipping a Line or Starting a New Page When Data Changes . . 3-11
Changing a Column Title 3-12
Underlining Column Titles 3-13
Justifying Data in Columns 3-15
Changing the Width of a Column Field 3-16
Widening Space Between Columns 3-17
Hiding a Column 3-18
Left-Filling Data 3-19
Changing the Format of Numeric Data 3-20
Displaying Signs for Positive Numbers 3-21
Changing the Position of Signs 3-22
Suppressing Duplicate Data in Columns 3-22
Breaking Row Data Across Display Lines 3-23
Defining a Data Label 3-25
Displaying Nulls 3-27
Displaying Current Settings 3-27
Resetting Display Options to Default Values 3-28
Sample Report 1 3-29
  Result Set 3-30
  Format Settings 3-31
  Report 1 3-32
Sample Report 2 3-33
  Result Set 3-34
  Format Settings 3-34
  Report 2 3-36

Chapter 4 Command Reference

In This Chapter 4-3
Commands 4-3
! (shell escape) 4-6
CONNECT 4-7
EDIT 4-9
QUERY 4-11
QUERY COLUMN 4-12
QUIT or EXIT 4-13
RESET 4-14
RESET COLUMN 4-15
RUN 4-18
SAVE 4-19
SET COLUMN 4-20
SET COLUMN column FILL 4-24
SET COLUMN column FOLD LINE 4-26
SET COLUMN column FORMAT 4-28
SET COLUMN column IF NULL 4-30
SET COLUMN column JUSTIFY 4-31
SET COLUMN column NO PRINT 4-33
SET COLUMN column NO REPEAT 4-35
SET COLUMN column POSITIVE SIGN. 4-37
SET COLUMN column SIGN 4-39
SET COLUMN column SKIP 4-41
SET COLUMN column SPACE 4-43
SET COLUMN column TITLE 4-46
SET COLUMN column UNDERLINE. 4-48
SET COLUMN column WIDTH. 4-50
SET ECHO 4-52
SET EDITOR 4-54
SET ERROR OUTPUT 4-55
SET OUTPUT 4-56
SET PAGE LENGTH 4-58
SET PROMPT 4-60
SET ROW DATA 4-62
SET TITLE 4-64

Appendix A **risql and risqlrpt Command Reference**

Appendix B **.rbretrc Files**

Appendix C **Locales**

Index

Introduction

In This Introduction 3
About This Guide 3
Types of Users 3
Software Dependencies 4
New Features. 5
Documentation Conventions 5
Syntax Notation 6
Syntax Diagrams 7
Keywords and Punctuation 9
Identifiers and Names 9
Icon Conventions 10
Comment Icons 10
Platform Icons 10
Customer Support 11
New Cases 12
Existing Cases 12
Troubleshooting Tips 13
Related Documentation 14
Additional Documentation 16
Online Manuals 16
Printed Manuals 17
Informix Welcomes Your Comments 17

In This Introduction

This Introduction provides an overview of the information in this document and describes the conventions it uses.

About This Guide

This user's guide describes how to use the RISQL Entry Tool and the its companion tool, RISQL Reporter.

- The RISQL Entry Tool provides users with access to Informix Red Brick Decision Server running on UNIX or Windows platforms.
- The RISQL Reporter performs the same functions as the RISQL Entry Tool but also allows users to format their reports.

Most features described in this guide apply to both tools. Those specific to RISQL Reporter, however, are clearly noted.

Types of Users

This guide is written for the following users:

- Database users
- Database administrators
- Database-application programmers
- Database developers
- Backup operators
- Performance engineers

This guide assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with computer programming
- Some experience with database server administration, operating-system administration, or network administration

Software Dependencies

This guide assumes that you are using Informix Red Brick Decision Server, Version 6.0, as your database server.

Red Brick Decision Server includes the Aroma database, which contains sales data about a fictitious coffee and tea company. The database tracks daily retail sales in stores owned by the Aroma Coffee and Tea Company. The dimensional model for this database consists of a fact table and its dimensions.

For information about how to create and populate the demonstration database, see the *Administrator's Guide*. For a description of the database and its contents, see the *SQL Self-Study Guide*.

The scripts that you use to install the demonstration database reside in the *redbrick_dir/sample_input* directory, where *redbrick_dir* is the Red Brick Decision Server directory on your system.

New Features

The following section describes new database server features relevant to this document. For a comprehensive list of new features, see the release notes.

- Support for the VARCHAR (variable-length character) data type
- Ability to export the results of an arbitrary query to a data file
- Enhancements to BREAK BY and RESET BY functionality

Documentation Conventions

Informix Red Brick documentation uses the following notation and syntax conventions:

- Computer input and output, including commands, code, and examples, appear in `Courier`.
- Information that you enter or that is being emphasized in an example appears in **`Courier bold`** to help you distinguish it from other text.
- Filenames, system-level commands, and variables appear in *italic* or *`Courier italic`*, depending on the context.
- Document titles always appear in *Palatino italic*.
- Names of database tables and columns are capitalized (Sales table, Dollars column). Names of system tables and columns are in all uppercase (RBW_INDEXES table, TNAME column).

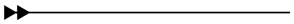
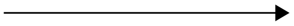
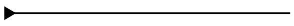
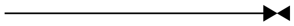
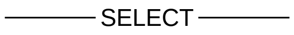
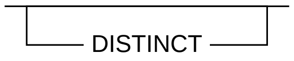
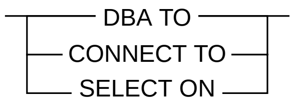
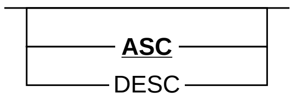
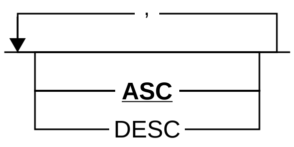
Syntax Notation

This guide uses the following conventions to describe the syntax of operating-system commands.

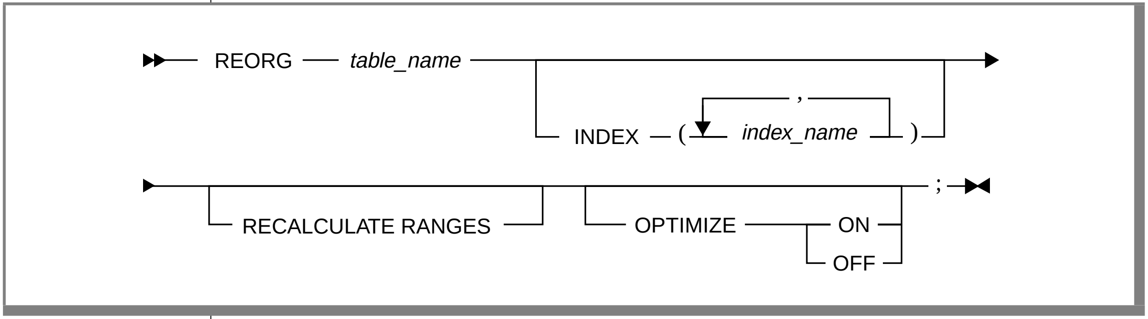
Command Element	**Example**	**Convention**
Values and parameters	*table_name*	Items that you replace with an appropriate name, value, or expression are in *italic* type style.
Optional items	[ ]	Optional items are enclosed by square brackets. Do not type the brackets.
Choices	ONE | TWO	Choices are separated by vertical lines; choose one if desired.
Required choices	{ONE | TWO}	Required choices are enclosed in braces; choose one. Do not type the braces.
Default values	<u>ONE</u> | TWO	Default values are underlined, except in graphics where they are in **bold** type style.
Repeating items	name, …	Items that can be repeated are followed by a comma and an ellipsis. Separate the items with commas.
Language elements	( ) , ; .	Parentheses, commas, semicolons, and periods are language elements. Use them exactly as shown.

Syntax Diagrams

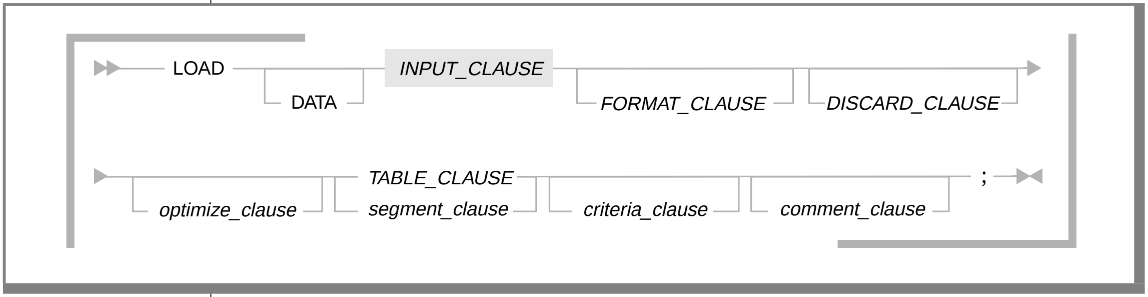
This guide uses diagrams built with the following components to describe the syntax for statements and all commands other than system-level commands.

Component	Meaning
	Statement begins.
	Statement syntax continues on next line. Syntax elements other than complete statements end with this symbol.
	Statement continues from previous line. Syntax elements other than complete statements begin with this symbol.
	Statement ends.
	Required item in statement.
	Optional item.
	Required item with choice. One and only one item must be present.
	Optional item with choice. If a default value exists, it is printed in **bold**.
	Optional items. Several items are allowed; a comma must precede each repetition.

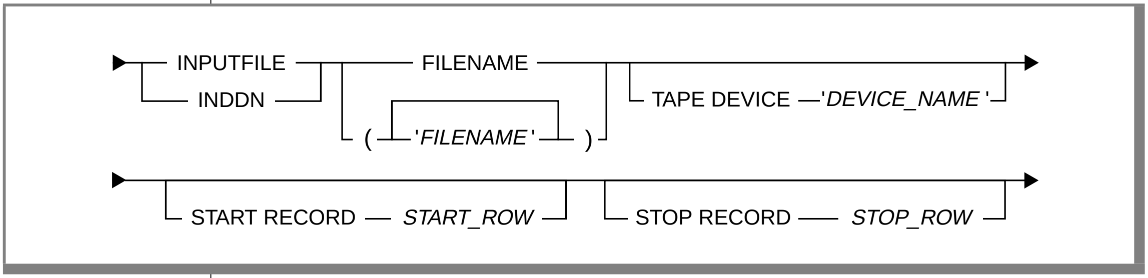
The preceding syntax elements are combined to form a diagram as follows.



Complex syntax diagrams such as the one for the following statement are repeated as point-of-reference aids for the detailed diagrams of their components. Point-of-reference diagrams are indicated by their shadowed corners, gray lines, and reduced size.



The point-of-reference diagram is then followed by an expanded diagram of the shaded portion—in this case, the *INPUT_CLAUSE*.



Keywords and Punctuation

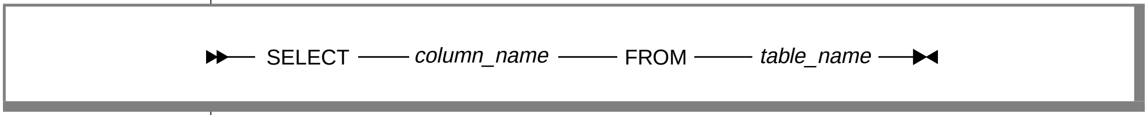
Keywords are words reserved for statements and all commands except system-level commands. When a keyword appears in a syntax diagram, it is shown in uppercase characters. You can write a keyword in uppercase or lowercase characters, but you must spell the keyword exactly as it appears in the syntax diagram.

Any punctuation that occurs in a syntax diagram must also be included in your statements and commands exactly as shown in the diagram.

Identifiers and Names

Variables serve as placeholders for identifiers and names in the syntax diagrams and examples. You can replace a variable with an arbitrary name, identifier, or literal, depending on the context. Variables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a variable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses variables to illustrate the general form of a simple SELECT statement.



When you write a SELECT statement of this form, you replace the variables *column_name* and *table_name* with the name of a specific column and table.

Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Comment Icons

Comment icons identify three types of information, as the following table describes. This information always appears in italics.

Icon	Label	Description
⚠	***Warning:***	Identifies paragraphs that contain vital instructions, cautions, or critical information
⇨	***Important:***	Identifies paragraphs that contain significant information about the feature or operation that is being described
💡	***Tip:***	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described

Platform Icons

Feature, product, and platform icons identify paragraphs that contain platform-specific information.

Icon	Description
UNIX	Identifies information that is specific to UNIX platforms
Windows	Identifies information that is specific to Windows NT, Windows 95, and Windows 98 environments
WIN NT	Identifies information that is specific to the Windows NT environment

These icons can apply to an entire section or to one or more paragraphs within a section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of feature-, product-, or platform-specific information that appears within one or more paragraphs within a section.

Customer Support

Please review the following information before contacting Informix Customer Support.

If you have technical questions about Informix Red Brick Decision Server but cannot find the answer in the appropriate document, contact Informix Customer Support as follows:

Telephone	1-800-274-8184 or 1-913-492-2086 (7 A.M. to 7 P.M. CST, Monday through Friday)
Internet access	http://www.informix.com/techinfo

For nontechnical questions about Red Brick Decision Server, contact Informix Customer Support as follows:

Telephone	1-800-274-8184 (7 A.M. to 7 P.M. CST, Monday through Friday)
Internet access	http://www.informix.com/services

New Cases

To log a new case, you must provide the following information:

- Red Brick Decision Server version
- Platform and operating-system version
- Error messages returned by Red Brick Decision Server or the operating system
- Concise description of the problem, including any commands or operations performed before you received the error message
- List of Red Brick Decision Server or operating-system configuration changes made before you received the error message

For problems concerning client-server connectivity, you must provide the following additional information:

- Name and version of the client tool that you are using
- Version of Informix Red Brick ODBC Driver or Informix Red Brick JDBC Driver that you are using, if applicable
- Name and version of client network or TCP/IP stack in use
- Error messages returned by the client application
- Server and client locale specifications

Existing Cases

The support engineer who logs your case or first contacts you will always give you a case number. This number is used to keep track of all the activities performed during the resolution of each problem. To inquire about the status of an existing case, you must provide your case number.

Troubleshooting Tips

You can often reduce the time it takes to close your case by providing the smallest possible reproducible example of your problem. The more you can isolate the cause of the problem, the more quickly the support engineer can help you resolve it:

- For SQL query problems, try to remove columns or functions or to restate WHERE, ORDER BY, or GROUP BY clauses until you can isolate the part of the statement causing the problem.
- For Table Management Utility load problems, verify the data type mapping between the source file and the target table to ensure compatibility. Try to load a small test set of data to determine whether the problem concerns volume or data format.
- For connectivity problems, issue the *ping* command from the client to the host to verify that the network is up and running. If possible, try another client tool to see if the same problem arises.

Related Documentation

The standard documentation set for Red Brick Decision Server includes the following documents.

Document	Description
Administrator's Guide	Describes warehouse architecture, supported schemas, and other concepts relevant to databases. Procedural information for designing and implementing a database, maintaining a database, and tuning a database for performance. Includes a description of the system tables and the configuration file.
Installation and Configuration Guide	Provides installation and configuration information, as well as platform-specific material, about Red Brick Decision Server and related products. Customized for either UNIX or Windows NT.
Messages and Codes Reference Guide	Contains a complete listing of all informational, warning, and error messages generated by Informix Red Brick Decision Server products, including probable causes and recommended responses. Also includes event log messages that are written to the log files.
The release notes	Contains information pertinent to the current release that was unavailable when the documents were printed.
This manual	Is a complete guide to the RISQL Entry Tool, a command-line tool used to enter SQL statements, and the RISQL Reporter, an enhanced version of the RISQL Entry Tool with report-formatting capabilities.

(1 of 2)

Document	Description
SQL Reference Guide	Is a complete language reference for the Informix Red Brick SQL implementation and RISQL extensions for warehouse databases.
SQL Self-Study Guide	Provides an example-based review of SQL and introduction to the RISQL extensions, the macro facility, and Aroma, the sample database.
Table Management Utility Reference Guide	Describes the Table Management Utility, including all activities related to loading and maintaining data. Also includes information about data replication and the *rb_cm* copy management utility.

(2 of 2)

In addition to the standard documentation set, the following documents are included for specific sites.

Document	Description
Client Connector Pack Installation Guide	Includes procedures for installing and configuring the Informix Red Brick ODBC Driver, the RISQL Entry Tool, and the RISQL Reporter on client systems. Included for sites that purchase the Client Connector Pack.
SQL-BackTrack User's Guide	Is a complete guide to SQL-BackTrack, a command-line interface for backing up and recovering warehouse databases. Includes procedures for defining backup configuration files, performing online and checkpoint backups, and recovering the database to a consistent state.

(1 of 2)

Document	Description
Informix Vista User's Guide	Describes the Informix Vista aggregate navigation and advisory system. Illustrates how Vista improves the performance of queries by automatically rewriting queries using aggregates, describes how the Advisor recommends the best set of aggregates based on data collected daily, and shows how the system operates in a versioned environment.
JDBC Connectivity Guide	Includes information about Informix Red Brick JDBC Driver and the JDBC API, which allow Java programs to access database management systems.
ODBC Connectivity Guide	Includes information about ODBC conformance levels and instructions for using the Informix Red Brick ODBClib SDK to compile and link an ODBC application.

(2 of 2)

Additional references you might find helpful include:

- An introductory-level book on SQL
- An introductory-level book on relational databases
- Documentation for your hardware platform and operating system

Additional Documentation

For additional information, you might want to refer to the following documents, which are available as online and printed manuals.

Online Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print online manuals, see the installation insert that accompanies Answers OnLine.

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and phone number

Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
SCT Technical Publications Department
4100 Bohannon Drive
Menlo Park, CA 94025

If you prefer to send electronic mail, our address is:

doc@informix.com

The **doc** alias is reserved exclusively for reporting errors and omissions in our documentation.

We appreciate your suggestions.

Chapter

Introduction to the RISQL Entry Tool and RISQL Reporter

1

In This Chapter 1-3
The RISQL Entry Tool 1-4
The RISQL Reporter 1-4
The Informix Red Brick Environment 1-4

In This Chapter

Red Brick Decision Server is a relational database management system (RDBMS) designed for data warehouse, data mart, and online analytical processing (OLAP) applications. Compared to online transaction processing (OLTP) or universal database products, Informix Red Brick Decision Server delivers higher query-processing and data-loading performance, greater ease of administration, and a richer set of specialized features for applications that range from a few gigabytes to well over a terabyte, and from a few users to thousands of users.

Informix Red Brick Decision Server can scale from the workgroup to the enterprise, is built for an open client/server environment using industry-standard open database connectivity (ODBC), and is accessed using industry-standard SQL. The server's RISQL extensions simplify analyses that require ranks, ratios, and other commonly used business calculations, while the Vista, STARjoin, STARindex, TARGETjoin, and TARGETindex technologies provide unparalleled ad hoc query and analysis performance against very large databases with various schema designs. Managers and analysts can pose numerous and creative queries to quickly receive the information they need, and make good business decisions with similar speed and confidence.

The RISQL Entry Tool

The RISQL Entry Tool provides interactive access to Informix Red Brick Decision Server databases. You can use the RISQL Entry Tool for writing and executing queries and for database administration tasks.

The RISQL Entry Tool accepts three forms of input.

- Standard SQL statements including those that contain RISQL extensions
- RISQL Entry Tool commands
- Operating-system commands through shell-escape commands

The RISQL Reporter

The RISQL Reporter provides all of the functionality of the RISQL Entry Tool plus the ability to format query output into reports. You can use RISQL Reporter features to define titles for your reports that include a date, time, and page numbers, add spaces or start new pages based on changes in data values, and set format characteristics such as data justification and column width on individual columns.

Most of the features described in this guide apply to both the RISQL Entry Tool and the RISQL Reporter. Those features that are specific to the RISQL Reporter are clearly noted.

The Informix Red Brick Environment

The RISQL Entry Tool and RISQL Reporter can be used from a UNIX or Windows client. The RISQL Entry Tool, RISQL Reporter, and other client query tools communicate with Informix Red Brick Decision Server databases through ODBC software.

For more information about the Informix Red Brick Decision Server environment and query logging, refer to the *Administrator's Guide*.

Chapter

Using the RISQL Entry Tool and RISQL Reporter

2

In This Chapter 2-3
Identifying System Shell Prompts 2-4

Accessing Databases 2-4

Defining a Locale 2-5

Starting the Tool 2-5
Usage Notes 2-8
Valid Input 2-8
Output Columns 2-8
Command Syntax 2-9
New-Line Character 2-9
.rbretrc file 2-10
Error Messages 2-10

Connecting to a Database 2-10
Connecting with the RB_PATH Environment Variable 2-11
Connecting on the Command Line 2-12
Connecting with the SQL CONNECT Command 2-13

Ending Your Tool Session 2-15

Submitting Queries to the Database 2-15

Interrupting Command Execution 2-17

Displaying Statistics 2-17

Using File Redirection 2-19
Using System Redirection 2-19
Using Redirection Within the Tool 2-20
Reading Input 2-20
Including Input in Output 2-21
Saving or Discarding Output 2-23
Specifying an Output Program 2-24
Redirecting or Discarding Error Messages 2-25

Using an Editor 2-26
Changing the Default Editor 2-26
Starting an Editor 2-27
Incomplete-String Prompt 2-30

Adding Comments to Statements 2-32

Accessing the System Shell 2-33

Changing the Tool Prompt 2-35

Changing Database Passwords 2-36

Administering the Database 2-36
Defining Database Objects 2-37
Controlling Database and Table Access. 2-37
Changing Information in a Database 2-38

Creating Data Extraction Applications 2-38
Redirecting Extraction Results from the Shell 2-39
Redirecting Extraction Results from Within the Tool 2-39
Exit Status Codes 2-40

In This Chapter

The RISQL Entry Tool and RISQL Reporter are interactive programs that allow you to access databases. You can use these tools to perform database and server administration tasks and to write queries. The RISQL Reporter also allows you to format query results into attractive reports. All features described in this chapter apply to both the RISQL Entry Tool and the RISQL Reporter, unless RISQL Reporter-only support is specifically noted.

This chapter contains the following sections:

- Accessing Databases
- Defining a Locale
- Starting the Tool
- Connecting to a Database
- Ending Your Tool Session
- Submitting Queries to the Database
- Interrupting Command Execution
- Displaying Statistics
- Using File Redirection
- Using an Editor
- Adding Comments to Statements
- Accessing the System Shell
- Changing the Tool Prompt
- Changing Database Passwords
- Administering the Database
- Creating Data Extraction Applications

Identifying System Shell Prompts

The system default prompt is shell-specific.

C shell: %
Korn and Bourne shells: $
Windows: `C:\>`

The Korn-shell prompt ($) is used in examples throughout this document.

Accessing Databases

Only the system administrator can provide database access to users. Database access authorizations are granted with the SQL GRANT CONNECT command, which is described in detail in both the *Administrator's Guide* and the *SQL Reference Guide*.

The RB_CONFIG environment variable must be defined before you attempt to run the RISQL Entry Tool or RISQL Reporter. This environment variable can be set in your shell-startup file or on the operating-system command line. The following error messages are displayed when the RB_CONFIG environment variable is either not defined or defined incorrectly when you start the RISQL Entry Tool:

- Undefined RB_CONFIG environment variable:

  ```
  ** FATAL ** (803) RB_CONFIG must be defined.
  ```

- Incorrectly defined RB_CONFIG environment variable:

  ```
  Error in opening config file; unable to continue.
  ** FATAL ** (10502) Message base lookup failed for
  message 10502, error code 200040.
  ```

For more information about Informix Red Brick Decision Server error messages, refer to the *Messages and Codes Reference Guide*. For more information about the RB_CONFIG environment variable, refer to page A-3 and the *Administrator's Guide*.

Defining a Locale

The database locale, defined when the Informix Red Brick Decision Server software is installed, specifies the language and location for all the databases in that installation. RISQL Entry Tool and RISQL Reporter users can specify which language they want to use by using the RB_NLS_LOCALE environment variable to override the database locale. For information about setting this environment variable, refer to Appendix C, "Locales."

For detailed information about locale specifications and incompatibilities that might arise when the client locale differs from the database locale, refer to the *Administrator's Guide*.

Starting the Tool

Before you can start the RISQL Entry Tool or RISQL Reporter, you must establish your UNIX or Windows environment. Database administrators typically provide this information so that you have easy access to one or more databases. If the information has not been provided, consult your database administrator.

The database administrator can also describe how to call the tool from a UNIX or Windows system. Generally, you call the tool as follows.

UNIX

- On UNIX, enter the name of the tool, then follow the text entry with Return. ♦

WINDOWS

- On Windows, click **Start** and select the name of the tool from the appropriate directory. ♦

In either case, both the location of the tool and its name depend on how the system administrator configured your machine and how the database administrator configured the Informix Red Brick Decision Server.

Most examples in this guide show how to use these two tools when the server runs on a UNIX computer.

Syntax

The tool startup syntax is as follows:

```
risql [-q] [-s dsn] [-h host] [-d database] [db_username
[db_password]]
risqlrpt [-q] [-s dsn] [-h host] [-d database] [db_username
[db_password]]
```

where:

-q	Quiet mode in which the copyright notice is not displayed when the tool starts.
-s *dsn*	Data source name (DSN) as defined in the *.odbc.ini* file on UNIX systems and the *odbc.ini* file on Windows 95 and Windows 98, and the Registry on Windows NT. A DSN can also be specified with the dRB_DSN environment variable. This option takes precedence over the **-**d *database* option and any pertinent environment variables defined for the session.
-h *host*	RB_HOST value defined in the *rbw.config* file. This RB_HOST value can also be specified with the RB_HOST environment variable.
-d *database*	Logical database name defined in the server's *rbw.config* file. A logical database name can also be specified with the RB_PATH environment variable or with an SQL CONNECT command, as described on page 2-10.
db_username	Valid username for the specified database. If *db_username* is not specified, the tool prompts for the database username.
db_password	Valid password for the specified database. If *db_password* is not specified, the tool prompts for the database password.

To start a RISQL Reporter session, substitute *risqlrpt* for *risql* above.

For more detailed information about risql and risqlrpt commands, command-line options, and environment variables, refer to Appendix A, "risql and risqlrpt Command Reference."

To start a RISQL Entry Tool session and connect to a database

1. Enter the following command at the system prompt:

   ```
   $ risql
   ```

Important: *If the RISQL Entry Tool program (**risql**) or the RISQL Reporter program (**risqlrpt**) is not in your search path, the operating system displays a message, similar to the following, when you try to start the tool:*

```
risql: command not found
risqlrpt: command not found
```

*If this message appears, either type the full pathname for the **risql** or **risqlrpt** program on the command line when you start the tool or add the pathname to your search path. For further assistance, consult your operating-system administrator.*

   The RISQL Entry Tool displays a copyright notice and prompts you for your database username.

   ```
   (C) Copyright 1991-1999 Informix Software, Inc.
   All rights reserved.
   RISQL Entry Tool Version 6.0.1(0)
   Please type username:
   ```

   If you are starting the RISQL Reporter with the *risqlrpt* command, the following copyright notice is displayed:

   ```
   (C) Copyright 1991-1999 Informix Software, Inc.
   All rights reserved.
   RISQL Reporter Version 6.0.1(0)
   Please type username:
   ```

2. Type your database username and press Return.

   ```
   Please type username: curly
   ```

   The RISQL Entry Tool prompts you for your database user password.

3. Type your database password and press Return.

   ```
   Please type password: xxxxxxxx
   ```

   When your password is accepted, you are connected to the database and the tool displays the following prompt:

   ```
   RISQL>
   ```

Usage Notes

When you omit **-**d *database* on the command line, the tool searches for a logical database name defined by the environment variable RB_PATH and connects to that database. If RB_PATH has not been defined, the tool starts but is not connected to a database. In this case, you can use the CONNECT command to access a database. Both the RB_PATH environment variable and the CONNECT command are described in "Connecting to a Database" on page 2-10.

The optional **-***q* parameter causes the RISQL Entry Tool or RISQL Reporter to run in quiet mode. In quiet mode, the copyright notice and the column headings are neither displayed nor printed in a file when output is redirected. This option is useful when you are using the RISQL Entry Tool or RISQL Reporter in data extraction scripts intended for TMU load statements. For more information about operating in quiet mode, refer to "Creating Data Extraction Applications" on page 2-38.

Tip: *You might need to change your password when you log into a database for the first time. For more information, refer to "Changing Database Passwords" on page 2-36.*

Valid Input

When you see the RISQL prompt, you can enter:

- RISQL Entry Tool commands
- Operating-system commands through shell-escape commands

After you make a database connection, you can also enter SQL statements, including those that contain RISQL extensions, to query a database.

Output Columns

The RISQL Entry Tool and RISQL Reporter can return a result set that contains at most 1,000 columns. If the result set exceeds 1,000 columns, the server returns an error message instead of a result set.

Command Syntax

All SQL statements and RISQL Entry Tool or RISQL Reporter commands except the shell-escape command (!) must be followed by a semicolon (;). The semicolon terminates the statement or command.

Warning: *If multiple commands or statements, each one punctuated by a semicolon, occur on the same line (or span multiple display lines without a line break), only the first one will be executed. The server discards the remaining commands or statements without returning a message.*

The examples in this guide include a space character between the last element of a statement or command declaration and the semicolon. This space is allowed by SQL, but not required.

Semicolons do not terminate statements when found inside:

- Quoted strings
- C-program-style comments (**/*…*/**)
- SQL-style comments (**--…**)

If a semicolon as the last character of a line fails to terminate the statement or command, then the statement contains either a quoted string that is missing the closing quote or a C-style comment that is missing the end-comment characters (***/**).

For more information about using comments in your statements, refer to "Adding Comments to Statements" on page 2-32.

New-Line Character

Sometimes, quoted literals span multiple lines. To avoid displaying the INCOMPLETE STRING prompt in these cases, you can insert a line break in a quoted string of characters with a C-program-style new-line character (\n). For more information about the INCOMPLETE STRING prompt, refer to page 2-30.

.rbretrc file

RISQL Entry Tool and RISQL Reporter commands can be entered interactively at the tool prompt or in a tool initialization file named *.rbretrc*. Setting command preferences interactively changes the default settings for the immediate session only. Changing command preferences in the *.rbretrc* file changes the default settings each time a RISQL Entry Tool or RISQL Reporter session is started or until you modify the *.rbretrc* file. Settings made in the *.rbretrc* file during a tool session (for example, during a shell-escape sequence) are not applied until the tool is restarted. For more information about the initialization file, refer to Appendix B, ".rbretc Files."

Error Messages

Error messages generated by the RISQL Entry Tool, RISQL Reporter, and other components of Informix Red Brick Decision Server are documented in the *Messages and Codes Reference Guide*.

Connecting to a Database

To make a database available to the RISQL Entry Tool or RISQL Reporter, the database administrator must assign a logical database name that describes the full pathname to the database in the configuration file (*rbw.config)*. Users can then specify this logical name when requesting a database connection.

After logical database names are defined in the server's *rbw.config* file, there are four ways to connect to a database:

- Users or system administrators can define an environment variable named RB_PATH, which is set to a logical database name. With this method, the RISQL Entry Tool or RISQL Reporter defaults to the database defined in RB_PATH when users start the tool without specifying a database name on the command line.
- Users can enter the -d *database* option to specify a logical database name on the command line when they start the RISQL Entry Tool or RISQL Reporter. This option takes precedence over the RB_PATH environment variable.

- Users can enter the -s *dsn* option to specify a data source name (DSN) or define an RB_DSN environment variable. This option takes precedence over the -d *database* option or pertinent environment variables defined for the session. To access databases on remote servers using this option, the Informix Red Brick Client Connector Pack must be installed on your workstation.
- Users can issue a CONNECT command during a RISQL Entry Tool or RISQL Reporter session and specify a logical database name or a data source name (DSN).

Connecting with the RB_PATH Environment Variable

If you do not specify a database on the command line when you start the RISQL Entry Tool or RISQL Reporter, the tool attempts to connect to a database specified by the environment variable RB_PATH. Defining a database with RB_PATH lets you set a default database, which is useful when you access the same database most of the time.

To set the RB_PATH environment variable

1. Ask your database administrator for a list of logical database names that you can access.
2. For Korn shell and Bourne shell, enter the following command at the system prompt:

   ```
   $ RB_PATH=db_name; export RB_PATH
   ```

   For C shell, enter:

   ```
   % setenv RB_PATH db_name
   ```

   For Windows shell, enter:

   ```
   C:\> set RB_PATH=db_name
   ```

   where *db_name* is a logical name for the database you want to access. For more information about defining logical database names, refer to the *Administrator's Guide*.

Environment variables are specific to your login session. To avoid having to reset RB_PATH each time you log in, you might prefer to set it in your shell-startup file.

Connecting on the Command Line

The -d *database* option, when used at startup, connects you to the database specified by the argument *database*, which can be any logical database name. You can specify your database username and password after *database* on the command line, or you can use the -d option alone and then respond to the tool prompts for your username and password.

The -s *dsn* option connects you to the database specified by the argument *dsn* (data source name). This option takes precedence over the -d *database* option and pertinent environment variables defined for the user's session.

Tip: *The RISQL Entry Tool and RISQL Reporter prompt you for username and password information only if standard input is coming from the terminal and if standard output and error output have not been redirected. For more information about redirecting input and output, refer to "Using File Redirection" on page 2-19.*

For more information about starting the *risql* program, refer to Appendix A, "risql and risqlrpt Command Reference."

Examples

To connect to but not log in to the Aroma database (a logical database name) when you start the RISQL Entry Tool, enter:

```
$ risql -d aroma
```

where *aroma* is a logical database name. The system responds:

```
(C) Copyright 1991-1999 Informix Software, Inc.
All rights reserved.
RISQL Entry Tool Version 6.0.1(0)
RISQL>
Please type username: curly
Please type password: xxxxxxxx
RISQL>
```

Tip: *Passwords are limited to 8 characters when prompted for by the RISQL Entry Tool or RISQL Reporter. Passwords entered on the command line, as shown in the following example, are limited to 128 characters.*

To connect to and log into the Aroma database when you start the RISQL Entry Tool, enter:

```
$ risql -d aroma curly mysecret
(C) Copyright 1991-1999 Informix Software, Inc.
All rights reserved.
RISQL Entry Tool Version 6.0.1(0)
RISQL>
```

Warning: *With this method, on UNIX platforms, your password is displayed on your screen and can be displayed on the screens of other users if they list processes with the operating system ps command.*

To connect and log into the Aroma database by using a DSN when you start the RISQL Entry Tool, enter:

```
$ risql -s aroma
```

where *aroma* is a logical DSN defined in your environment.

Connecting with the SQL CONNECT Command

When you start the RISQL Entry Tool or RISQL Reporter without connecting to a database, you can use the SQL CONNECT command to specify a logical database name. When you omit your database username on the command line, the tool prompts you for both username and password.

If you issue the CONNECT command while a session with another database is still active, the current session is closed before the new session is opened. If your second connection attempt fails because you entered an incorrect password, the tool displays the RISQL prompt but you are no longer connected to the first database. In this situation, you must issue a valid CONNECT statement before you can access a database.

If you have installed the Client Connector Pack, you can use the CONNECT command to connect to a DSN after invoking the RISQL Entry Tool or the RISQL Reporter.

For complete syntax information for the CONNECT command, refer to "CONNECT" on page 4-3.

Examples

To connect to a database during a RISQL Entry Tool or RISQL Reporter session, enter the CONNECT command followed by the database name and a valid database username. For example:

```
RISQL> connect to aroma as moe ;
Please type password: xxxxxxxx
RISQL>
```

In this example, the logical database name aroma has been defined by the database administrator.

Suppose your company has a group database account and password for a particular database. If you start the RISQL Entry Tool and connect with your personal account, you can connect again to the same database under another username. For example:

```
RISQL> connect as keeper ;
Please type password: xxxxxxxx
RISQL>
```

If you have installed the Red Brick Client Connector Pack on a Windows computer and want to use the DSN, enter:

```
RISQL> connect to aroma;
```

where *aroma* is a DSN defined in your environment.

If you want to use the RB_DSN environment variable on a UNIX platform, enter at the prompt:

```
% setenv RB_DSN dsn_name
% risql
RISQL>
```

where *dsn_name* is a DSN defined in your environment.

If you want to use the RB_DSN environment variable on a Windows platform, enter at the prompt:

```
C: \> set RB_DSN=dsn_name
C: \> risql
(C) Copyright 1991-1999 Informix Software, Inc.
All rights reserved.
RISQL Entry Tool Version 6.0.1(0)
RISQL>
```

where *dsn_name* is a DSN defined in your environment.

Ending Your Tool Session

The EXIT and QUIT commands terminate a RISQL Entry Tool or RISQL Reporter session and return you to the system prompt. If you have a current database connection when you issue one of these commands, the connection is closed before the tool terminates.

Example

To terminate the session, enter:

```
RISQL> quit ;
$
```

or:

```
RISQL> exit ;
$
```

Submitting Queries to the Database

To retrieve data from a database with the RISQL Entry Tool or RISQL Reporter:

1. Start the tool and connect to a database.
2. Type an SQL query.
3. Type a semicolon(;) at the end of the statement.
4. Press Return to submit the query to the database.

For a complete description of database-supported SQL and RISQL extensions, refer to the *SQL Reference Guide*. For more query examples, refer to the *SQL Self-Study Guide*.

Important: *SQL keywords including those specific to the RISQL Entry Tool and RISQL Reporter (such as SET, RUN, and EDIT) are always in English, regardless of the current locale.*

Examples

A simple SELECT statement lists all the cities and districts for which data is stored in the Aroma sample database. This sample database is usually created during the installation process.

```
RISQL> select * from market;
MKTKEY HQ_CITY         HQ_STATE  DISTRICT       REGION
     1 Atlanta         GA        Atlanta        South
     2 Miami           FL        Atlanta        South
     3 New Orleans     LA        New Orleans    South
     4 Houston         TX        New Orleans    South
     5 New York        NY        New York       North
     6 Philadelphia    PA        New York       North
     7 Boston          MA        Boston         North
     8 Hartford        CT        Boston         North
     9 Chicago         IL        Chicago        Central
    10 Detroit         MI        Chicago        Central
    11 Minneapolis     MN        Minneapolis    Central
    12 Milwaukee       WI        Minneapolis    Central
    14 San Jose        CA        San Francisco  West
    15 San Francisco   CA        San Francisco  West
    16 Oakland         CA        San Francisco  West
    17 Los Angeles     CA        Los Angeles    West
    19 Phoenix         AZ        Los Angeles    West
```

SELECT * returns all information from a table. Be careful when using it to view large tables, because doing so might inadvertently reduce server performance for other users and make your terminal inaccessible until the query result is returned. The SQL SET ROWCOUNT command is useful for limiting the number of rows returned by a query. For more information about this command, refer to the *SQL Reference Guide*.

To display the annual sales stored in the Aroma database (including the number of items sold, the average sales, and total sales) for *Lotta Latte* brand coffee in Los Angeles during 1999, enter the following query:

```
RISQL> select count(*) as num_items,
> string (avg(dollars), 7,2) as avg_sale,
> sum (dollars) as total_sales
> from store st
> join sales sa on st. storekey = sa. storekey
> join product pr on pr. prodkey = sa. prodkey
> and pr.classkey = sa.classkey
> join period pe on pe.perkey = sa. perkey
> where prod_name = 'Lotta Latte'
> and year = 1999
> and city = 'Los Angeles';
NUM_ITEMS   AVG_SAL       TOTAL_SALES
80           171.33        13706.50
```

Interrupting Command Execution

To interrupt an SQL or RISQL statement before it returns data, enter Control-C. (That is, hold down the key marked "control" and press the key marked "C" at the same time.) This keystroke sequence interrupts the current command execution and returns you to the RISQL prompt.

Displaying Statistics

The SET STATS command allows you to display statistics after each statement is executed. You can choose to display:

- No statistics, with SET STATS OFF (default).
- Basic reporting, with SET STATS ON, which typically includes the following types of statistical information:
  - Number of data rows returned
  - Elapsed time during statement processing
  - CPU time spent on statement execution
  - Logical I/O counts
- More detailed reporting, with SET STATS INFO, which includes additional information about how queries are processed.

Statistics reporting varies from statement to statement and from platform to platform; for example, when you display statistics for a CREATE TABLE statement, no data rows are returned so there are no values to report.

The RISQL Entry Tool and RISQL Reporter send statistical output to the same destination as error output, which is the terminal by default. You can redirect statistics output with the SET ERROR OUTPUT command. For more information about redirecting statistical output, refer to "Redirecting or Discarding Error Messages" on page 2-25.

For information about the SET STATS command, refer to the *SQL Reference Guide*.

Examples

To turn statistics reporting on, enter:

```
RISQL> set stats on;
```

Statistics are then displayed in standard output.

```
RISQL> select prod_name, sum(dollars) as sales_99
> from sales natural join product
>   natural join period
>   natural join store
> where year = 1999
>   and city = 'San Jose'
>   and prod_name in
>       (select prod_name
>       from product
>       where pkg_type like 'Gift%')
> group by prod_name;
PROD_NAME                      SALES_99
Spice Sampler                          1860.00
Tea Sampler                            4207.00
Coffee Sampler                         3420.00
Christmas Sampler                       780.00
** STATISTICS ** (500) Time = 00:00:04.69 cp time, 00:00:05.84
time, Logical IO count=156
** INFORMATION ** (256) 4 rows returned.
```

To get more detailed reporting, type the following:

```
RISQL> set stats info;
```

To turn statistics reporting off, type the following:

```
RISQL> set stats off;
```

Using File Redirection

File redirection is useful when you want to execute long SQL statements or generate reports with the RISQL Reporter. There are two methods for using file redirection:

- You can use standard system file-redirection symbols (>, <, >>, <<, |) to redirect input that comes from the terminal and output that goes to the screen.
- You can use tool commands (RUN, SET OUTPUT, SET ERROR OUTPUT) to read from and write to files.

You can also write the result set of a query directly to a disk file using the RISQL EXPORT command. See the *SQL Reference Guide* for additional information on how to use this command.

Using System Redirection

The RISQL Entry Tool and RISQL Reporter use standard syntax for redirecting standard system input and output. To direct file input to the RISQL Entry Tool from the system command line, enter:

```
$ risql -d database db_username db_password < input_file
```

or:

```
$ cat input_file | risql -d database db_username db_password
```

where *input_file* is a file containing SQL or tool commands. (The *cat* command might not be available on your Windows system.)

To read from and write to a file, enter:

```
$ risql -d database db_username db_password < input_file >
output_file
```

where *output_file* contains the output.

Tip: *To avoid typing the database name, username, and password on each command line, you can include a CONNECT statement on the first line of the input file. If you use this method, you must enter your database username and password on one line. For more information about the CONNECT command, refer to "Connecting to a Database" on page 2-10.*

When redirecting output with system-redirection commands, you might want to use the -q option to run in quiet mode. In quiet mode, the copyright notice and the column headings are suppressed and only data output is redirected to the output file. For more information about quiet mode, refer to "Creating Data Extraction Applications" on page 2-38.

Using Redirection Within the Tool

The following tool commands control input and output:

- RUN
- SET OUTPUT
- SET ERROR OUTPUT

Controlling input and output during a tool session is described in the following sections. The SET ECHO command is also described as it relates to input and output.

Reading Input

You can read input from the tool's command buffer or from a file. The RUN command, when issued at the RISQL prompt, executes the contents of the command buffer or a specified file. The output is directed to the screen or to another file depending on the output setting. Input stored in the command buffer is read in the following order:

- If the most recent command was EDIT, issuing the RUN command without a filename executes the previously edited file.
- If the most recent command was not EDIT, issuing the RUN command executes the command buffer, which contains the last command that was executed from standard input.

For complete syntax information for the RUN command, refer to "RUN" on page 4-18.

Examples

To execute the contents of the command buffer, which contains the most recently executed command or SQL statement, type the following:

```
RISQL> run ;
```

Input stored in files can be saved and executed many times.

To execute the contents of a file named *store_rank* located in the directory from which you started the RISQL Entry Tool or RISQL Reporter, enter:

```
RISQL> run store_rank ;
RISQL> select store_name, sum(dollars) as total_sales,
>       rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
>       join sales s on s.storekey = r.storekey
>       join period p on s.perkey = p.perkey
> where year = 1999
>       and month = 'MAR'
>       and region = 'West'
> group by store_name;
STORE_NAME                       TOTAL_SALES     SALES_RANK
Cupertino Coffee Supply                 18801.50           1
San Jose Roasting Company               18346.90           2
Beaches Brew                            18282.05           3
Java Judy's                             17826.25           4
Instant Coffee                          15650.50           5
Roasters, Los Gatos                     12694.50           6
```

If you want to execute a file from another directory, you can specify the full or relative pathname of the file on the RUN command line.

Including Input in Output

You can display input before output. The SET ECHO command controls command echoing when you execute the contents of the command buffer or a file. Command echoing is turned on by default.

For complete syntax information for the SET ECHO command, refer to "SET ECHO" on page 4-52.

Examples

To display SQL statements before the display of output, type the following:

```
RISQL> set echo on;
```

To display the current echo setting, type the following:

```
RISQL> query echo;
```

To restore default echoing, type the following:

```
RISQL> reset echo;
```

If you use the RUN command to execute the contents of the command buffer or a file with SET ECHO ON, input is displayed before the output. To submit a query stored in a file named *lotta_latte_LA_99*, enter:

```
RISQL> run lotta_latte_LA_99;
RISQL> select count(*) as num_items,
> string (avg(dollars), 7,2) as avg_sale,
> sum (dollars) as total_sales
> from market ma join store st
> on ma. mktkey = st.mktkey
> join sales sa on st. storekey = sa. storekey
> join product pr on pr. prodkey = sa. prodkey
> and pr.classkey = sa.classkey
> join period pe on pe.perkey = sa. perkey
> where prod_name = 'Lotta Latte'
> and year = 1999
> and city = 'Los Angeles';

NUM_ITEMS   AVG_SALE    TOTAL_SALES
    80         171.33     13706.50
```

To display output only, when input comes from the command buffer or a file, type the following:

```
RISQL> set echo off;
```

When you start the RISQL Entry Tool or RISQL Reporter in quiet mode, the SET ECHO command is set to OFF by default.

Saving or Discarding Output

The SET OUTPUT command directs input to a specified file or discards it altogether. The SET OUTPUT command has the following options:

- Sends output to a specified file and overwrites the contents of a file, if the file exists when the command is issued or when the tool starts.
- Appends output to the contents of a specified file.
- Sends output to standard output, which, depending on whether output redirection is in effect at the operating-system level, is either the terminal screen or a file.
- Sends output to a specified program, which modifies output when it is displayed on a terminal.
- Discards all output.

This command sends all output to the specified file until the output setting is changed or until the RISQL Entry Tool or RISQL Reporter session is ended. Sending output to a file logs all statements executed during the current session. If the same filename is specified in a subsequent tool session, the contents of the file are overwritten with output from the current session.

Examples

To send output to a file named *test.risql* in the directory */home/curly,* type the following:

```
RISQL> set output /home/curly/test.risql;
```

To send output to a file named *test.risql* and append the output to the contents of an existing file:

```
RISQL> set output /home/curly/test.risql append;
```

To send output to standard output (the default setting), type the following:

```
RISQL> set output stdout;
```

Specifying an Output Program

Specifying an output program is useful when you want to control the display of output on a terminal. When you specify the PROGRAM option of the SET OUTPUT command, the tool sends output to a system program, which processes and displays it. Typical system output programs include *more* and *cat*.

Important: *The cat command may not be available on your Microsoft Windows system.*

The system program *more* displays output one screen at a time, scrolling additional rows after the initial screen.

For syntax information for the SET OUTPUT command, refer to "SET OUTPUT" on page 4-56.

Examples

To specify the *more* program, enter:

```
RISQL> set output program 'more';
```

The name of the program must be enclosed in single quotation marks.

The -c option of the *more* command clears the display area and displays continued output a screen at a time, rather than scrolling it. To specify *more* with the -c option, enter:

```
RISQL> set output program 'more';
```

To discard all output, enter:

```
RISQL> set output off;
```

To restore the output setting to the default (standard output), enter:

```
RISQL> reset output;
```

Redirecting or Discarding Error Messages

The SET ERROR OUTPUT command directs error, warning, and informational messages to a specified file or discards error messages. This command has the following options:

- Sends error messages to a file and overwrites the contents of the file if the file exists when the command is issued or when the tool starts.
- Appends error messages to the contents of a file.
- Sends error messages to standard output, which, depending on whether output redirection is in effect at the operating-system level, is either the terminal or a file.

Important: *Fatal messages, which indicate that your server connection has been lost, are always displayed in standard output and cannot be redirected with the SET ERROR OUTPUT command.*

This command sends all informational messages to the specified file until the setting is changed or the tool session is ended. Sending output to a file logs all error messages for the current session. If the same filename is specified in a subsequent session, the contents of the file are overwritten with error output from the new session.

For complete syntax information for the SET ERROR OUTPUT command, refer to "SET ERROR OUTPUT" on page 4-55.

Examples

To send error messages to a file named *errors* in the directory */home/curly*, enter:

```
RISQL> set error output /home/curly/errors ;
```

To send error messages to a file named *errors* and append the new messages to the list of messages in the file from a previous tool session, enter:

```
RISQL> set error output /home/curly/errors append ;
```

Sending error messages to a file named *stderr* is useful when you have output directed to a file. To send error messages to *stderr* (the default setting), enter:

```
    RISQL> set error output stderr ;
```

or:

```
    RISQL> reset error output ;
```

To send error messages to standard output (the terminal screen or a file, depending on whether output redirection is in effect at the operating-system level), enter:

```
    RISQL> set error output stdout ;
```

Using an Editor

You can use an editor with the RISQL Entry Tool or RISQL Reporter to modify and correct SQL statements and tool commands located in existing files or in the command buffer. Both tools support any editor installed on your system. The default editor is *vi* for UNIX platforms and *Notepad* for Windows platforms.

Changing the Default Editor

You can change the default editor to another system editor with the SET EDITOR command.

For complete syntax information for the SET EDITOR command, refer to "SET EDITOR" on page 4-54.

Examples

To change the editor to *ed*, enter:

```
RISQL> set editor 'ed' ;
RISQL>
```

The editor command declaration must be enclosed in single quotation marks.

To display the current editor, enter:

```
RISQL> query editor ;
```

To restore the default editor, enter:

```
RISQL> reset editor ;
```

Starting an Editor

The EDIT command starts the default editor or the editor you have specified with the SET EDITOR command. You can use the editor to open:

- An existing file that contains SQL statements or tool commands
- A temporary buffer, which can be used for query testing and saved to a permanent file

If you do not specify a filename, the contents of the command buffer are placed in a temporary file for editing. The command buffer contains the most recently executed SQL statement or tool command, with the exception of the EDIT and RUN commands. These cannot be edited in the command buffer.

After editing files, you can use the RUN command to execute them. For more information about the RUN command, refer to "Reading Input" on page 2-20.

To end an editing session, enter the editor's termination command. The editor quits and the tool prompt is displayed.

Examples

Suppose your query named *store_rank* retrieves a ranking of stores in the Western region based on sales during the month of March, and you want to see the same figures for the Southern region. You can open your query text file with an editor, change the search condition on the region column, and use the RUN command to submit the query to the database.

1. First run the existing query and examine the result.

```
RISQL> run store_rank ;
RISQL> select store_name, sum(dollars) as total_sales,
>      rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
>      join sales s on s.storekey = r.storekey
>      join period p on s.perkey = p.perkey
> where year = 1999
>      and month = 'MAR'
>      and region = 'West'
> group by store_name;
STORE_NAME                 TOTAL_SALES     SALES_RANK
Cupertino Coffee Supply          18801.50           1
San Jose Roasting Company        18346.90           2
Beaches Brew                     18282.05           3
Java Judy's                      17826.25           4
Instant Coffee                   15650.50           5
Roasters, Los Gatos              12694.50           6
```

   SET ECHO has been turned on in this example to display the contents of *store_rank* as it executes. For more information about echoing input, refer to "Including Input in Output" on page 2-21.

2. Now use your editor to open and edit the file. (This example shows the ***vi*** editor.)

```
RISQL> edit store_rank ;
```

   The file opens and you see the query.

```
select store_name, sum(dollars) as total_sales,
   rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
   join sales s on s.storekey = r.storekey
   join period p on s.perkey = p.perkey
where year = 1999
   and month = 'MAR'
   and region = 'West'
group by store_name;
~
~
~
~
"store_rank" 10 lines, 292 characters
```

3. Change the search condition from West to South.

```
select store_name, sum(dollars) as total_sales,
   rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
   join sales s on s.storekey = r.storekey
   join period p on s.perkey = p.perkey
where year = 1999
   and month = 'MAR'
   and region = 'South'
group by store_name;
~
~
~
~
"store_rank" 10 lines, 293 characters
```

4. Terminate the editing session and issue the RUN command to submit the new query to the database.

```
RISQL> run ;
```

Tip: *If the last command issued was EDIT, you need not specify a filename as an argument to the RUN command.*

```
RISQL> select store_name, sum(dollars) as total_sales,
>   rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
>   join sales s on s.storekey = r.storekey
>   join period p on s.perkey = p.perkey
> where year = 1999
>   and month = 'MAR'
>   and region = 'South'
> group by store_name;
STORE_NAME                 TOTAL_SALES    SALES_RANK
Miami Espresso                18119.20             1
Moulin Rouge Roasting         17526.25             2
Olympic Coffee Company        16900.25             3
Texas Teahouse                14634.25             4
```

Editing the temporary buffer is useful when you make typing errors while entering statements at the tool prompt. When you use the EDIT command without specifying a filename, the command buffer contains the last RISQL statement that was executed. This buffer is placed in a temporary file, which can be submitted again with the RUN command.

Incomplete-String Prompt

When your statement contains unmatched quotes, the tool displays an incomplete string prompt (INCOMPLETE STRING). This prompt indicates that your statement has not been executed and that the tool is waiting for more input.

For example, a long literal string used in a WHERE clause might be broken across multiple lines. If the tool comes to the end of a line and does not find a closing quote, it displays the incomplete string prompt. This behavior does not indicate that an error has occurred; the prompt is simply sending you a status message about the current line of code. When the tool reads the closing quote on a subsequent line, the continuation or default prompt is displayed.

Suppose you forget to type a closing quote on a literal string in a query statement. You can use the EDIT and RUN commands to correct the statement and execute it again.

Example

1. Type the following query, omitting the closing quote on West as shown, and press Return:

```
RISQL> select store_name, sum(dollars) as total_sales,
>  rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
>  join sales s on s.storekey = r.storekey
>  join period p on s.perkey = p.perkey
> where year = 1999
>  and month = 'MAR'
>  and region = 'West ;
```

2. The tool responds with the incomplete string prompt:

```
INCOMPLETE STRING>
```

   The tool has not read the statement-terminating semicolon because it is placed inside a quoted literal. Instead, the tool is waiting for more input.

3. To execute the statement, enter a closing quote and another semicolon:

```
INCOMPLETE STRING>' ;
```

   The server returns an incorrect result because it does not find a match for the quoted literal, which included the semicolon.

```
STORE_NAME                       TOTAL_SALES    SALES_RANK
RISQL>
```

4. To correct the query statement, edit a temporary copy in the command buffer:

```
RISQL> edit ;
select store_name, sum(dollars) as total_sales,
   rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
   join sales s on s.storekey = r.storekey
   join period p on s.perkey = p.perkey
where year = 1999
   and month = 'MAR'
   and region = 'West ;
~
~
~
~
"/usr/tmp/gaaa06618" 8 lines, 269 characters
```

5. Type the missing quote at the end of line 8:

```
and region = 'West' ;
```

6. Quit the editor. You see the RISQL prompt.
7. Enter the RUN command to execute the contents of the temporary buffer:

```
STORE_NAME                       TOTAL_SALES    SALES_RANK
Cupertino Coffee Supply             18801.50             1
San Jose Roasting Company           18346.90             2
Beaches Brew                        18282.05             3
Java Judy's                         17826.25             4
Instant Coffee                      15650.50             5
Roasters, Los Gatos                 12694.50             6
```

You must issue the RUN command immediately after editing the file, closing it, and submitting it to the database, because the command buffer is overwritten with each data retrieval or command execution.

If you want to save the contents of the buffer, you can do so before terminating the editor session by saving it with a permanent filename in your own directory.

For complete syntax information for the EDIT command, refer to "EDIT" on page 4-3. For more information about the RUN command, refer to "Reading Input" on page 2-20.

Adding Comments to Statements

When you write a complex SQL statement that you plan to use more than once, you might want to describe portions of the code by including comments. Annotating statements is useful when preparing them for less technical users. It is also useful when you use statements only occasionally and want to include reminders about the specific elements of the code.

The RISQL Entry Tool and RISQL Reporter accept two comment formats:

- C-program-style comments (/*…*/), which require a beginning marker and an ending marker and can be embedded in a line or span multiple lines.
- SQL-style comments (--…), which require only a beginning marker, must be placed at the end of a line, and are limited to one line.

Examples

Use C-program-style comments as shown below:

```
select city,
    sum(quantity) as yr_1999,
    tertile(sum(quantity)) as qty_rank
/* calculates H, M, L ranking of quantity totals */
from market ma join store st
    on ma.mktkey = st.mktkey
    join sales sa on st.storekey = sa.storekey
    join product pr on pr.prodkey = sa.prodkey
    join period pe on pe.perkey = sa.perkey
where year = 1999
    and prod_name = 'Expresso XO'
    and region in ('West', 'South')
group by city;
```

Important: *If you use a C-program-style comment (/*…*/) after the semicolon, the ending marker must occur on the same line as the beginning marker.*

Use SQL-style comments as shown below:

```
select city,
> sa.perkey, --Period key column from Sales table
> month,
> dollars,
> ratiotoreport (dollars)*100 as ratio
> --Ratio of $ to total (as percentage)
> from market ma join store st
>       on ma.mktkey = st.mktkey
>       join sales sa on st.storekey = sa.storekey
>       join product pr on pr.prodkey = sa.prodkey
>       and pr.classkey = sa.classkey
>       join period pe on pe.perkey = sa.perkey
> where prod_name like 'Xalapa%'
> and qtr in ('Q3_99','Q4_99')
> and city in ('San Jose', 'Los Angeles')
> order by city, sa.perkey
> --Orders data alphabetically by city, then by period key
> reset by city;
```

Accessing the System Shell

The shell-escape command (*!*) suspends the current tool session, starts a new shell process, and lets you execute system commands. A semicolon is not required to terminate a shell-escape command.

You can issue a single system command by using the command as an argument to the *!* command, or you can start an interactive shell session that suspends your tool session until you indicate that you want to resume it.

The EXIT command ends the shell session and reactivates the tool session. Always remember to terminate the interactive shell session and return to your RISQL Entry Tool or RISQL Reporter session. Otherwise, you might inadvertently leave yourself connected to a database, which might interfere with the work of other users.

For complete syntax information for the *!* command, refer to "!" on page 4-3.

Examples

UNIX

To list all files in the current UNIX directory that end with the suffix *.risql*, type the following:

```
RISQL>!ls -l *.risql

-rw-rw-rw-1 curlypubs13 Oct 23 13:23 cume.risql
-rw-rw-rw-1 curlypubs13 Oct 23 13:24 mvgavg.risql
-rw-rw-rw-1 curlypubs13 Oct 23 13:23 ratio.risql
-rw-rw-rw-1 curlypubs13 Oct 23 13:24 share.risql
RISQL>
```

The operating system executes the command and returns you to the tool prompt.

To start an interactive shell session, type the *!* command and a system shell identifier:

```
RISQL>!csh
%
```

♦

WINDOWS

To list in the current Windows directory all files that end with the suffix *.risql*, type the following:

```
RISQL> !dir * .risql
CUME~1   RIS             80  10-31-98 12:54p cume.risql
AVG~1    RIS             80  10-31-98 12:54p avg.risql
SUM~1    RIS             80  10-31-98 12:54p sum.risql
MOVAG~1  RIS             80  10-31-98 12:55p movag.risql
4 file(s)              320 bytes
```

♦

Changing the Tool Prompt

The RISQL Entry Tool and RISQL Reporter display three unique prompts: the first-line prompt (RISQL), the continuation prompt (>), and the incomplete string prompt (INCOMPLETE STRING). You can change these prompts with the SET PROMPT command. The prompt you specify must be enclosed in single quotation marks.

You can change the first-line prompt without changing the continuation or incomplete string prompts. However, you must change the first-line and continuation prompts or restate their default values if you want to change subsequent prompts.

For complete syntax information for changing these prompts, refer to "SET PROMPT" on page 4-60.

Examples

To change the first-line prompt to **>>>**, the continuation prompt to **>>**, and the incomplete string prompt to UNMATCHED QUOTE, enter:

```
RISQL> set prompt '>>> ' '>> ' 'UNMATCHED QUOTE> ' ;
```

The prompts are displayed as follows until you change them or end the tool session: Windows

```
>>> select * from sales
>> where market = 'Los Angeles ;
UNMATCHED QUOTE>
```

To display the current prompts, enter:

```
RISQL> query prompt ;
```

To restore the default prompts, enter:

```
RISQL> reset prompt ;
```

Changing Database Passwords

The administrator uses the SQL GRANT command to authorize user access to a database and create usernames and passwords. For database security, users must often change assigned passwords to private ones.

For a complete discussion of the GRANT command and character limits on the lengths of usernames and passwords, refer to the *SQL Reference Guide*.

Examples

User *curly* can change his current password *mysecret* to *mystery* by entering:

```
RISQL> grant connect to curly with mystery ;
```

Administering the Database

Provided you have the correct database authorization and table privileges, you can use the RISQL Entry Tool or RISQL Reporter for database administration tasks. These tasks fall into three categories:

- Data definition
- Data control
- Data manipulation

The following sections contain examples of commonly used data administration commands. For complete syntax and option information for all database-supported SQL commands and RISQL extensions, refer to the *SQL Reference Guide*. For a discussion of database administration tasks, refer to the *Administrator's Guide*.

Defining Database Objects

You can use the RISQL Entry Tool or RISQL Reporter to create, alter, and drop database objects such as tables, views, indexes, synonyms, and macros.

Examples

To create a table named Product, enter:

```
RISQL> create table product (
>   classkey integer not null,
>   prodkey integer not null,
>   prod_name char(30),
>   pkg_type char(20),
>   constraint prod_pkc primary key (classkey, prodkey),
>   constraint prod_fkc foreign key (classkey) references
        class
>   (classkey))
>   maxrows per segment 2500;
```

To create a view containing only data for the *Lotta Latte* product, enter:

```
RISQL> create view l_latte
> as select *
> from product
> where prod_name = 'Lotta Latte' ;
```

To drop a view, enter:

```
RISQL> drop view l_latte ;
```

Controlling Database and Table Access

As database administrator, you give database access to users and control access to database tables. As the creator of a table, you grant access to other users. These tasks are referred to as granting database authorization and granting table privileges.

Examples

To create the database user *curly* with the password *mysecret*, enter:

```
RISQL> grant connect to curly with mysecret ;
```

To give *curly* query Select privileges on the Product table, enter:

```
RISQL> grant select on product to curly ;
```

Changing Information in a Database

If you are the database administrator, or if the database administrator has given you data modification privileges, you can use the RISQL Entry Tool or RISQL Reporter to insert new rows, update data in rows, and delete rows.

Examples

To define a row in the Product table for a new brand of coffee named *Aloha Kona*, type the following:

```
RISQL> insert into product
> (classkey,
> prodkey,
> prod_name,
> pkg_type)
> values
> (1,
> 5,
> 'Aloha Kona',
> 'No pkg');
```

To update the Pkg_type data value for *Aloha Kona*, type the following:

```
RISQL> update product
> set pkg_type = 'One-pound bag'
> where prod_name = 'Aloha Kona' ;
```

To delete the *Aloha Kona* row from the Product table, type the following:

```
RISQL> delete from product
> where prod_name = 'Aloha Kona' ;
```

Creating Data Extraction Applications

A useful application of the RISQL Entry Tool or RISQL Reporter is to extract selected data from a database and place that data into a file for transfer to another program. You can also use a pipe (|) to send row data output directly from the RISQL Entry Tool or RISQL Reporter to another program, such as the Table Management Utility (TMU).

When using the RISQL Entry Tool or RISQL Reporter in data extraction applications, you typically want to restrict data output to only query result rows, omitting the product copyright notice, column headings, blank lines, and so on. Quiet mode restricts output to query results only.

Redirecting Extraction Results from the Shell

You can perform data extractions at the operating-system level directly from the UNIX or Windows shell (for example, in a shell script).

When you perform data extraction from the shell by specifying quiet mode (-q tool-startup option), all non-error output from the program is redirected to a file or pipe with standard UNIX or Windows file-redirection commands. Using the -q option also changes the default setting of the following commands:

- SET ECHO is set to OFF.
- SET ROW DATA is set to ONLY.

Examples

Use quiet mode (-q command-line option) to extract only query result rows to a file as shown below:

```
$ risql -q user_name password < weekly_extract > excel_import
```

To pipe extracted data to another program, such as the Table Management Utility, enter:

```
$ risql -q username password < weekly_extract | rb_tmu load.tmu
username password
```

Redirecting Extraction Results from Within the Tool

When used together, the following commands capture query results in a file for transfer to another program during a RISQL Entry Tool or RISQL Reporter session:

- SET ROW DATA ONLY causes the query result columns to be returned without column headings or other extraneous information.
- SET ECHO OFF suppresses the RISQL prompt and input commands, as described on page 2-22.
- SET OUTPUT *filename* sends row data output to a file instead of the terminal (standard output), as described on page 2-23.

Example

To send only the row data for the specified period to a file named *weekly_extract*, enter:

```
RISQL> set row data only;
RISQL> set echo off;
RISQL> set output weekly_extract;
RISQL> select *
> from period join sales
>   on period.perkey = sales.perkey
    and qtr = 'Q1_99' ;
```

Exit Status Codes

The RISQL Entry Tool and RISQL Reporter use the following exit status codes:

Status Code	Meaning
0	Information or statistics messages might have been issued during execution, but no warning, error, or fatal messages were issued.
1	Warning messages cause an exit status of 1 to be returned.
2	Error messages cause an exit status of 2 to be returned.
3	Fatal messages cause an exit status of 3 to be returned.

The RISQL Entry Tool and RISQL Reporter return the highest status code encountered during processing. For example, if the RISQL Entry Tool generates only informational messages, it will return an exit status code of 0; if, however, it generates both information messages and a fatal message, it will return an exit status code of 3.

Chapter

3

Formatting Reports

In This Chapter 3-3

Laying Out Reports. 3-4

Using the SET COLUMN Commands 3-4
  Column Specification. 3-5
  Column Variable Precedence 3-5
  Column Display Options 3-6

Saving Command Settings 3-8

Creating a Report Title 3-9

Defining the Page Length 3-10

Skipping a Line or Starting a New Page When Data Changes 3-11

Changing a Column Title. 3-12

Underlining Column Titles 3-13

Justifying Data in Columns 3-15

Changing the Width of a Column Field 3-16

Widening Space Between Columns 3-17

Hiding a Column 3-18

Left-Filling Data 3-19

Changing the Format of Numeric Data 3-20

Displaying Signs for Positive Numbers 3-21

Changing the Position of Signs. 3-22

Suppressing Duplicate Data in Columns 3-22

Breaking Row Data Across Display Lines 3-23

Defining a Data Label 3-25

Displaying Nulls. 3-27

Displaying Current Settings 3-27

Resetting Display Options to Default Values 3-28

Sample Report 1 3-29
- Result Set 3-30
- Format Settings 3-31
- Report 1 3-32

Sample Report 2 3-33
- Result Set 3-34
- Format Settings 3-34
- Report 2 3-36

In This Chapter

This chapter describes report-formatting commands included with the RISQL Reporter. These features are not available with the RISQL Entry Tool.

This chapter is divided into the following sections:

- Laying Out Reports
- Using the SET COLUMN Commands
- Saving Command Settings
- Creating a Report Title
- Defining the Page Length
- Skipping a Line or Starting a New Page When Data Changes
- Changing a Column Title
- Underlining Column Titles
- Justifying Data in Columns
- Changing the Width of a Column Field
- Widening Space Between Columns
- Hiding a Column
- Left-Filling Data
- Changing the Format of Numeric Data
- Displaying Signs for Positive Numbers
- Changing the Position of Signs
- Suppressing Duplicate Data in Columns
- Breaking Row Data Across Display Lines
- Defining a Data Label
- Displaying Nulls
- Displaying Current Settings

- Resetting Display Options to Default Values
- Sample Report 1
- Sample Report 2

Laying Out Reports

The following procedure describes how to format a report with the RISQL Reporter:

1. Write and execute your SQL query and verify the results.
2. Specify RISQL Reporter SET COLUMN commands to define the layout.
3. Execute the query to view the formatted results.

If you are developing a brand new report format on a large amount of data, you might find it useful to experiment with formatting commands on a subset of the output (for example, one product). When you have designed a satisfactory layout, submit the entire query to display the complete report.

After you have developed a layout, you can use the SAVE command to write the SET COLUMN commands for the format to a file. This file can then be executed with the RUN command in subsequent RISQL Reporter sessions and used as a template for similar reports in the future. For more information about the RUN command, refer to "Reading Input" on page 2-20.

Using the SET COLUMN Commands

Most of the RISQL Reporter formatting features are options of the SET COLUMN command. The syntax for the SET COLUMN command is straightforward:

1. Specify the command.
2. Specify one or more columns on which to work.
3. Specify a formatting option.

SET COLUMN commands can be set interactively at the RISQL prompt, in input files, or in RISQL Reporter initialization files (*.rbretrc* files).

Column Specification

The following table describes valid column variables:

Column Variable	Description
*	Global column list
n	Relative column number
*table.**	Table column list (sales.*)
column_name	Unqualified column name (dollars)
table.column_name	Table-qualified column name (sales.dollars)

Column Variable Precedence

The following list describes each valid column specification and gives its level of precedence when more than one option is specified:

- Options specified for the global column list apply to all columns retrieved unless overwritten by a more specific column option specified later in the command syntax.
- Options specified for a relative column number are applied to the column in that position in the query. Relative column number options modify the characteristics of columns that are literals, subqueries, expressions, or display functions. Relative column numbers are retained even when columns are suppressed with the NO PRINT option of the SET COLUMN command. Relative column number options override global options.

- Options specified for a table column list are applied to all columns in that table. Table column list options override global and relative column number options.
- Options specified for unqualified column names are applied to all columns with that name, regardless of the table to which the column belongs. Column aliases defined in a SELECT statement can also be used as unqualified column names. Unqualified column options override global column lists, table column lists, and relative column number options.

Important: *Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the "SQL Reference Guide" or the "SQL Self-Study Guide."*

- Options specified for table-qualified column names are applied only to the specific column. Column aliases defined in a SELECT statement cannot be used in table-qualified names. Table-qualified column name options take precedence over any other option specified for the column.

Column Display Options

The following table lists and describes column display options.

Format Option	Description
FILL	Sets the left-fill character for data in a column.
FOLD LINE	Breaks a data row into two or more lines.
FORMAT	Sets the format of numeric data in a column (DECIMAL, INTEGER, EXPONENTIAL).
IF NULL	Defines the string that is displayed when null is returned.
JUSTIFY	Sets the justification of data in a column.
NO PRINT	Prevents a column from being displayed in the result.
NO REPEAT	Suppresses the display of duplicate values in a column.
POSITIVE SIGN	Specifies whether to display or suppress the plus sign (+) on positive numeric data.

(1 of 2)

Format Option	**Description**
SKIP LINE	Inserts a blank row between data rows when a value in a column changes.
SKIP PAGE	Starts a new page when a value in a column changes.
SIGN	Defines position, either leading or trailing, for signs (+ or –) on numeric data.
SPACE	Sets lead spacing for data in a column.
TITLE	Defines a column title.
UNDERLINE	Displays a string of repeating characters between column heading and data.
WIDTH	Sets the width of a column.

(2 of 2)

Some of the RISQL Reporter SET COLUMN commands affect other SET commands, so the sequence in which you issue them and the combination you use might be significant. The examples in this chapter are not sequential; to achieve the results shown here, you must reset any options that were previously set during your session or in your *.rbretrc* files so you begin with default settings.

For specific information about command syntax and a complete list of arguments to all options, refer to "SET COLUMN" on page 4-20.

Saving Command Settings

You can save RISQL Reporter command settings made during a session with the SAVE command. Saving formatting settings lets you define and reuse a report style for different queries during different RISQL Reporter sessions. Files created with the SAVE command are standard ASCII files, and can be modified with a system editor.

SAVE writes declarations for the following commands to a specified file:

- SET PAGE LENGTH
- SET TITLE
- SET COLUMN (all options)

SAVE *does not* save settings for the following commands:

- SET ECHO
- SET EDITOR
- SET ERROR OUTPUT
- SET PROMPT
- SET ROW DATA
- SET STATS

If you execute the SAVE command and specify an existing file, the contents of the file are overwritten with the current settings.

The contents of the saved file can be executed with the RUN command (refer to "Reading Input" on page 2-20) to change default format settings for the current session, or copied to the RISQL Reporter initialization files (*.rbretrc*) to change defaults for all subsequent sessions. For complete syntax information for the SAVE command, refer to "SAVE" on page 4-19. For more information about the initialization files, refer to Appendix B, ".rbretc Files."

Creating a Report Title

The SET TITLE command defines a single-line title for a report. Titles are left-justified and can contain one or more of the following predefined title variables, which display current values in the report:

- $DATE displays the date.
- $TIME displays the time.
- $PAGE displays the page number.

Title variables are embedded in text strings and are case sensitive.

For complete syntax information for the SET TITLE command, refer to "SET TITLE" on page 4-64.

Examples

The following command defines a title for the report shown below:

```
RISQL> set title 'WEIGHT RANKING IN WEST & SOUTH Report run: $DATE
    at $TIME' ;
RISQL> run wgt.rank ;
WEIGHT RANKING IN WEST & SOUTH  Report run: 1/15/99 at
    05:12:52

CITY                     YR_1999        W_RK
Miami                          5694.00 M
New Orleans                    3578.00 M
Houston                         367.00 L
San Francisco                 11056.00 H
Los Angeles                    9570.00 H
```

Notice that a blank line is displayed between the report title and the column titles.

To display the current title, enter:

```
RISQL> query title ;
```

To delete the title, enter:

```
RISQL> reset title ;
```

Defining the Page Length

The SET PAGE LENGTH command lets you specify the maximum number of lines printed on each page of a report. Changing the page length is useful when you want to customize your display for a specific terminal screen size, terminal emulator window size, or default printer page size.

The following lines can be displayed at the top of each report page:

- Report title followed by a blank line
- Column title
- Column underline

The default page length is unlimited; the report title, column titles, and column title underlines are printed only once at the beginning of the report. To break the report into several pages, you can provide a line-number argument to the SET PAGE LENGTH command. The column title information is then repeated at the top of each new page.

The RISQL Reporter displays lines of data until either a page break is indicated by a SKIP PAGE option or the maximum number of report lines, as defined by the SET PAGE LENGTH command, is reached. For more information about the SKIP PAGE option, refer to page 3-11.

The number of report lines displayed on a page is determined by subtracting the heading option lines (report title lines, the column title line, and the column underscore line) and the top and bottom margin allowance lines (4) from *the specified page length*. The minimum allowable page length is the number needed for the report title and column titles.

The SET PAGE LENGTH command is also affected by the FOLD LINE option of the SET COLUMN command. The precedence of these format settings is important and is described in detail on "SET COLUMN" on page 4-20.

Examples

The following setting produces a report that prints 20 lines, then breaks and prints the column titles at the top of the next page before printing another 20 lines.

```
    RISQL> set page size 20 ;
```

For an example of page-defined output, refer to the sample reports at the end of this chapter.

To display the current page length setting, enter:

```
    RISQL> query page length ;
```

To restore default page length, enter:

```
    RISQL> reset page length ;
```

Skipping a Line or Starting a New Page When Data Changes

The SKIP option of the SET COLUMN command skips a line or starts a new page in the result each time data changes in a specified column.

The RISQL Reporter compares columns from left to right. The leftmost column encountered with a SKIP setting causes the line to be skipped or the new page to be started.

If both a SKIP LINE and a SKIP PAGE are used on the same column, SKIP PAGE takes precedence. In general, SKIP PAGE should be used on columns to the left of columns with SKIP LINE settings.

Important: *The RISQL Reporter automatically skips to a new page when the number of lines on a page, as specified with SET PAGE LENGTH (refer to page 3-10), is exceeded.*

For complete syntax information for the SKIP option, refer to "SET COLUMN column SKIP" on page 4-41.

Examples

The following report lists cities in the southern and western regions by district. Each time a value changes in the District column, the RISQL Reporter skips a line.

```
RISQL> set column district skip line ;
RISQL> select hq_city, district
> from market
> where region = 'South'
>  or region = 'West'
> order by district ;
HQ_CITY             DISTRICT
Atlanta             Atlanta
Miami               Atlanta

Phoenix             Los Angeles
Los Angeles         Los Angeles

New Orleans         New Orleans
Houston             New Orleans

San Jose            San Francisco
San Francisco       San Francisco
Oakland             San Francisco
```

To print the data for each district on a separate page, enter:

```
RISQL> set column district skip page ;
```

Changing a Column Title

By default, internal column names stored in a database are displayed as titles in a result. Internal names are often abbreviated strings and cannot contain spaces; therefore, they are not always appropriate for report writing. Calculated columns resulting from SQL functions or RISQL extensions have no title by default.

You can use the TITLE option of the SET COLUMN command to define single-word or multi-word column titles in a result. You must enclose your column title definition in single quotation marks.

Important: *Column titles also can be defined with column aliases in a query statement, but column aliases are displayed in uppercase and cannot contain spaces.*

For complete syntax information for the column TITLE option, refer to "SET COLUMN column TITLE" on page 4-46.

Example

To display the title RUNNING TOTAL above the CUME column, enter:

```
RISQL> set column 4 title 'RUNNING TOTAL' right;
RISQL> select prod_name, city, quantity, cume(quantity)
> from store st
>       join sales sa on st.storekey = sa.storekey
>       join product pr on pr.prodkey = sa.prodkey
>       join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>       or prod_name like 'Cafe%')
>       and city like 'New O%'
>       and week = 32
>       and year = 1999
> order by prod_name, quantity asc;
NAME            CITY         QUANTITY        RUNNING TOTAL
Cafe Au Lait    New Orleans         5                    5
Cafe Au Lait    New Orleans         5                   10
Cafe Au Lait    New Orleans        14                   24
Cafe Au Lait    New Orleans        14                   38
Expresso XO     New Orleans        19                   57
Expresso XO     New Orleans        19                   76
```

Underlining Column Titles

The UNDERLINE option of the SET COLUMN command lets you underline one or more column titles with a horizontal character string. The hyphen character (-) is displayed by default, but you can change it to any single character you choose. For example, an equal sign (=) provides a double underline.

The UNDERLINE option has two formats:

- LONG underscores the title across the entire width of the column.
- SHORT underscores only the displayed width of the title text, no matter what the width of the column. When you customize the underline character, SHORT is the default format.

For complete syntax information for the UNDERLINE option, refer to "SET COLUMN column UNDERLINE" on page 4-48.

Examples

To display a double underline across the entire width of all columns, enter:

```
RISQL> set column * underline '=' long;
RISQL> select prod_name, city, quantity,
>       cume(quantity) as running_total
>   from store st
>       join sales sa on st.storekey = sa.storekey
>       join product pr on pr.prodkey = sa.prodkey
>       join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>       or prod_name like 'Cafe%')
>       and city like 'New O%'
>       and week = 32
>       and year = 1999
> order by prod_name, quantity asc;
PROD_NAME          CITY          QUANTITY   RUNNING_TOTAL
==================  =============  ========  ===============
Cafe Au Lait        New Orleans           5               5
Cafe Au Lait        New Orleans           5              10
Cafe Au Lait        New Orleans          14              24
Cafe Au Lait        New Orleans          14              38
Expresso XO         New Orleans          19              57
Expresso XO         New Orleans          19              76
```

To display asterisks that underline column titles only, type the following:

```
RISQL> set column * underline '*' short;
RISQL> select prod_name, quantity, cume(quantity) as
running_total
>  from store st
>               join sales sa on st.storekey = sa.storekey
>               join product pr on pr.prodkey = sa.prodkey
>               join period pe on pe.perkey = sa.perkey
>  where (prod_name like 'Expresso%'
>               or prod_name like 'Cafe%')
>               and city like 'New O%'
>               and week = 32
>             and year = 1999
>  order by prod_name, quantity asc;
PROD_NAME                     QUANTITY    RUNNING_TOTAL
*********                     ********    *************
Cafe Au Lait                         5                5
Cafe Au Lait                         5               10
Cafe Au Lait                        14               24
Cafe Au Lait                        14               38
Expresso XO                         19               57
Expresso XO                         19               76
```

Justifying Data in Columns

The JUSTIFY option of the SET COLUMN command lets you right- or left-justify data in a column. By default, numeric data is right-justified and character data is left-justified.

For complete syntax information for the JUSTIFY option, refer to "SET COLUMN column JUSTIFY" on page 4-31.

Example

To left- and right-justify numeric columns, type the following:

```
RISQL> set column 2 justify right;
RISQL> set column 3 justify left;
RISQL> select prod_name, quantity,
>                  rank(quantity) as qty_rank
>    from store st
>                  join sales sa on st.storekey = sa.storekey
>                  join product pr on pr.prodkey = sa.prodkey
>                  join period pe on pe.perkey = sa.perkey
>    where (prod_name like 'Expresso%'
>                  or prod_name like 'Cafe%')
>                  and city like 'New O%'
>                  and week = 32
>                  and year = 1999
>   order by prod_name, quantity asc;
PROD_NAME                      QUANTITY    QTY_RANK
*********                      ********    ********
Cafe Au Lait                          5 5
Cafe Au Lait                          5 5
Cafe Au Lait                         14 3
Cafe Au Lait                         14 3
Expresso XO                          19 1
Expresso XO                          19 1
```

Changing the Width of a Column Field

The default width of a column field is the datatype width that was defined by a table-definition statement when the table was created in the database. For example, if a column is defined as *char (10)*, the default column width is 10 characters in the result.

You can change the width of a column field in your report with the WIDTH option of the SET COLUMN command. This option accepts a numeric value for the number of display spaces in a column and is valid for any column type.

Column width settings can be influenced by the JUSTIFY option of the SET COLUMN command. If the alignment of data appears irregular in your report, consider the effect of justification and column width settings.

Column titles returned from the database, defined with the TITLE option of the SET COLUMN command, or specified as column aliases in the select list are truncated when they exceed the defined width of a column. For information about column aliases, refer to the *SQL Reference Guide*.

For complete syntax information for the WIDTH option, refer to "SET COLUMN column WIDTH" on page 4-50.

Example

The following command sets a width of 9 display spaces for the Dollars column.

```
    RISQL> set column dollars width 9 ;
```

Important: *If you set the SIGN TRAILING option on a right-justified decimal or integer column, the rightmost character position in the column is always reserved for a sign character. Therefore, the number of characters of a numeric result that this column can contain is reduced by one. For more information about the SIGN option, refer to page 3-21.*

Widening Space Between Columns

The SPACE option of the SET COLUMN command defines lead spacing, which widens the margin between columns and makes output easier to read. The default column spacing is one space for all columns except the leftmost column, which has no leading spaces.

For complete syntax information for the SPACE option, refer to "SET COLUMN column SPACE" on page 4-43.

Example

To indent the result display, set the lead spacing on the leftmost column to 5 as follows:

```
RISQL> set column prod_name space 5;
RISQL> select prod_name, quantity, dollars
>  from store st
>              join sales sa on st.storekey = sa.storekey
>              join product pr on pr.prodkey = sa.prodkey
>              join period pe on pe.perkey = sa.perkey
>  where (prod_name like 'Expresso%'
>     or prod_name like 'Cafe%')
>              and city like 'New O%'
>              and week = 32
>              and year = 1999
>  order by prod_name, quantity asc;
     PROD_NAME                      QUANTITY    DOLLARS
     *********                      ********    *******
     Cafe Au Lait                          5      23.75
     Cafe Au Lait                          5      23.75
     Cafe Au Lait                         14     112.00
     Cafe Au Lait                         14     112.00
     Expresso XO                          19     114.00
     Expresso XO                          19     114.00
```

Tip: *You can set spacing to zero to simulate concatenated columns. For example, you might want to concatenate columns when working with a database of address information.*

Hiding a Column

The NO PRINT option of the SET COLUMN command hides a column in the result. Hiding columns is often useful when you want to order your data by a particular column, but do not want that column to be displayed or printed in the report.

When using relative column numbers to replace column variables, remember that all columns are counted, even those that are hidden. For example, when you hide column 3, column 4 of the result is actually displayed in the column 3 position. Any settings on the displayed column in position 3 must be set on column 4.

For complete syntax information for the NO PRINT option, refer to "SET COLUMN column NO PRINT" on page 4-33.

Example

Suppose you do not want certain columns of data to display in your report. You can use the NO PRINT option to hide those columns:

```
RISQL> set column state no print;
RISQL> set column store_type no print;
RISQL> select store_name, store_type, state,
>                 sum(dollars) as sales
>  from store natural join sales
>  where store_type = 'Large'
>  group by store_name, store_type, state;
STORE_NAME                    SALES
**********                    *****
Miami Espresso                507022.35
San Jose Roasting Company     509819.15
Beaches Brew                  503493.10
Olympic Coffee Company        514830.00
```

Left-Filling Data

The FILL option of the SET COLUMN command left-fills data in a column. Left-filling data is often useful when you have numbers of unequal length in a column (for example, employee numbers). The default fill character is a space, but you can change it to a character of your choice.

When you format integer or decimal data with leading zero characters, the plus and minus signs (+ and –) are placed in the leftmost character position. When you fill with any other character, the minus sign is displayed immediately to the left of the leftmost numeric data character and the plus sign is not displayed at all.

When the FILL option is in use, the JUSTIFY LEFT option has no effect on numeric data in the column because every space in a data field is filled. For more information about justifying data, refer to "Justifying Data in Columns" on page 3-15.

For complete syntax information for the FILL option, refer to "SET COLUMN column FILL" on page 4-24.

To left-fill a column with leading zeros, enter:

```
RISQL> set column emp_no fill '0';
RISQL> select emp_no from employees;
```

Data in the Emp_No column is displayed in the result as follows:

```
EMP_NO
00012
00036
00037
00052
00123
00320
```

Changing the Format of Numeric Data

Data is displayed in its original datatype by default. The FORMAT option of the SET COLUMN command offers three options for changing the format of numeric data:

- DECIMAL format displays all real, float, double-precision, decimal, and numeric types in typical decimal format (2 decimal places, by default). The number of decimal places can be changed by supplying a numeric argument to the DECIMAL option. The maximum number of decimal places is 99. Data values that have more decimal places than specified by this option are rounded off.
- INTEGER format displays all numbers as integers. In this format, decimal values are rounded off and no decimal places are displayed. Negative numbers are also rounded off; –1.2 becomes –1 and –1.5 becomes –2.
- EXPONENTIAL format displays all numbers in exponential format. The number of decimal places displayed is equal to the column width minus 8. If the column width is less than 9, asterisks are displayed.

If the value exceeds the width or format limitation of the column, asterisks (*) are printed in the field. Asterisks occur most often in columns that are formatted as decimal or integer. However, if exponential format is specified and the column width is less than 9, asterisks are displayed.

For complete syntax information for the numeric format options, refer to "SET COLUMN column FORMAT" on page 4-28.

Examples

```
RISQL> set column weight format decimal 4 ;
123.4567
RISQL> set column weight format integer ;
123
RISQL> set column weight format exponential ;
1.2345E+2
```

Displaying Signs for Positive Numbers

By default, negative integer and decimal values in a result set are shown with a minus sign (–), but no sign is shown for positive numbers. The POSITIVE SIGN option of the SET COLUMN command lets you specify that plus signs (+) for positive numbers should also be shown.

For complete syntax information for the POSITIVE SIGN option, refer to "SET COLUMN column POSITIVE SIGN" on page 4-37.

Examples

To display the plus sign, specify the SHOW keyword in your SET COLUMN command, as follows:

```
RISQL> set column dollars show positive sign;
RISQL> select dollars from sales_chg;
    DOLLARS
    -132.13
    +566.00
    -212.98
```

To hide the plus sign (after specifying that it is to be displayed), specify the HIDE keyword in your SET COLUMN command, as follows:

```
RISQL> set column dollars hide positive sign;
RISQL> select dollars from sales_chg;
    DOLLARS
    -132.13
     566.00
    -212.98
```

Changing the Position of Signs

By default, any displayed plus (+) and minus (–) signs for integer and decimal values in a result set are placed to the left of the numbers in a column. The SIGN option of the SET COLUMN command lets you reposition the signs to the right of a displayed number.

When you specify the TRAILING option, the rightmost character position in a column is reserved for the sign. When positive values are present in this column, either a plus sign or a space character occupies the rightmost space, depending on the setting of the SET COLUMN POSITIVE SIGN command. Minus signs are always displayed for negative numbers.

For complete syntax information for the SIGN option, refer to "SET COLUMN column SIGN" on page 4-39.

Example

To move the sign to the right of column data, specify the TRAILING keyword in your SET COLUMN command, as shown below:

```
RISQL> set column dollars sign trailing;
RISQL> select dollars from sales_chg;
    DOLLARS
    132.13-
    566.00
    212.98-
```

Note that the plus signs are not displayed in this example.

Suppressing Duplicate Data in Columns

The NO REPEAT option of the SET COLUMN command suppresses the display of duplicate values in a result column. When you specify NO REPEAT and data in a named column is the same as the data in the preceding row, the RISQL Reporter hides succeeding identical data in the result. Thus, the NO REPEAT option is useful in reports ordered by a column with multiple identical entries.

When multiple NO REPEAT columns are specified, the RISQL Reporter searches for NO REPEAT values from left to right.

If a page break occurs when data in a column is suppressed, the suppressed values are displayed at the top of the new report page.

For complete syntax information for the NO REPEAT option, refer to "SET COLUMN column NO REPEAT" on page 4-35.

Example

To prevent the state and store type from repeating on each row, enter:

```
RISQL> set column state no repeat;
RISQL> set column store_type no repeat;
RISQL> select store_name, state, store_type,
>  sum(dollars) as sales
>  from store natural join sales
>  where store_type = 'Large'
>  group by store_name, store_type, state
>  order by state;
STORE_NAME                    STATE STORE_TYPE SALES
Beaches Brew                  CA    Large           503493.10
San Jose Roasting Company                           509819.15
Miami Espresso                FL    Large           507022.35
Olympic Coffee Company        GA    Large           514830.00
```

Breaking Row Data Across Display Lines

The FOLD LINE option of the SET COLUMN command wraps data rows across multiple lines. This command causes the RISQL Reporter to insert a new-line character (\n) before displaying a specified column, letting you break a long row of data into several rows so the total length of the line does not exceed the width of your output device (such as a terminal screen or printer).

For complete syntax information for the FOLD LINE option, refer to "SET COLUMN column FOLD LINE" on page 4-26.

Example

To display the product name on a line above the city, quantity, and running total information in the following report, set the FOLD LINE option on the City column as follows:

```
RISQL> set column city fold line;
RISQL> select prod_name, city, quantity,
>  cume(quantity) as running_total
>  from store st
>        join sales sa on st.storekey = sa.storekey
>        join product pr on pr.prodkey = sa.prodkey
>        join period pe on pe.perkey = sa.perkey
>  where (prod_name like 'Expresso%'
>     or prod_name like 'Cafe%')
>  and city like 'New O%'
>  and week = 32
>  and year = 1999
>  order by prod_name, quantity asc;
PROD_NAME
 CITY                 QUANTITY    RUNNING_TOTAL
Cafe Au Lait
 New Orleans                 5                5
Cafe Au Lait
 New Orleans                 5               10
Cafe Au Lait
 New Orleans                14               24
Cafe Au Lait
 New Orleans                14               38
Expresso XO
 New Orleans                19               57
Expresso XO
 New Orleans                19               76
```

Defining a Data Label

The LABEL option of the SET COLUMN SPACE command lets you annotate data in the result. For example, suppose your result returns dollar sales figures; when you define a dollar-sign label for the Dollars column, each value in the column is prefixed with a dollar sign (*$*).

Data labels are displayed to the left of each value in a column. The label string you define must be enclosed in single quotation marks.

Tip: *The width of a column is automatically expanded by the length of your label string. If you use the LABEL option of the SET COLUMN command with the WIDTH option, you need not include the length of the label string in the width specification. For information about the WIDTH option, refer to "Changing the Width of a Column Field" on page 3-16.*

For complete syntax information for the SPACE and LABEL options, refer to "SET COLUMN column SPACE" on page 4-43.

Examples

To include dollar signs in the sales figures column, first narrow the width of the column so that the *$* symbol appears immediately to the left of the data, then define the label as follows:

```
RISQL> set column total width 9;
RISQL> set column total space 1 '$';
RISQL> select district,
> sum(dollars) as total
> from market ma join store st
>   on ma.mktkey = st.mktkey
>   join sales sa on st.storekey = sa.storekey
>   join product pr on pr.prodkey = sa.prodkey
>   join period pe on pe.perkey = sa.perkey
> where prod_name like 'Cafe%'
> and year = 1999
> group by district
> order by total asc;
DISTRICT              TOTAL
Boston               $ 59623.00
Minneapolis          $ 62439.00
New Orleans          $ 62515.50
New York             $ 64072.00
Chicago              $ 66264.00
Atlanta              $ 69294.00
Los Angeles          $ 71858.00
San Francisco        $141803.00
```

The LABEL option also works well for custom report formatting when used with SET COLUMN * FOLD LINE and SET ROW DATA ONLY.

```
RISQL> set column * fold line;
RISQL> set row data only;
RISQL> set column mktkey space 1 'Mkt: ';
RISQL> set column hq_city space 4 'City: ';
RISQL> set column hq_state space 4 'State: ';
RISQL> set column district space 4 'District: ';
RISQL> set column region space 4 'Region: ';
RISQL> set column city skip line;
RISQL> select * from market
> where region = 'West';
    Mkt:          14
    City: San Jose
    State: CA
    District: San Francisco
     Region: West
 Mkt:          15
    City: San Francisco
    State: CA
    District: San Francisco
    Region: West
     Mkt:          16
    City: Oakland
    State: CA
    District: San Francisco
    Region: West
    Mkt:          17
    City: Los Angeles
    State: CA
    District: Los Angeles
    Region: West
    Mkt:          19
    City: Phoenix
    State: AZ
    District: Los Angeles
    Region: West
```

Displaying Nulls

The IF NULL option of the SET COLUMN command defines a null-replacement string. The null string you supply must be enclosed in single quotation marks when defined. The default string is NULL.

Example

To display nulls as N / A, enter:

```
RISQL> set column mgrno if null 'N/A';
RISQL> select name, department, title, mgrno, empno
> from dept natural join employees
> where department like 'Sales%';
NAME           DEPARTMENT  TITLE    MGRNO  EMPNO
Blake, Joe     Sales       AcctRep  N/A    1874
Peach, Etta    Sales       AcctRep  N/A    1051
Slo, Monica    Sales       Mgr      089    0498
Tepio, Curtis  Sales       AcctRep  N/A    1179
…
```

For complete syntax information for the IF NULL option, refer to "SET COLUMN column IF NULL" on page 4-30.

Displaying Current Settings

The QUERY COLUMN command displays settings for SET COLUMN options defined for the current session.

You can display option settings for:

- A single column, which displays the specified options set on that column.
- A column list, which displays options set on columns with that list specification (for example, market.*).
- All columns with options set with the global column wildcard (*).

Important: *Displaying global options displays only those options that were set with the global column wildcard character (*), not all column settings.*

Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the *SQL Reference Guide*.

For complete syntax information for the QUERY COLUMN command, refer to "QUERY COLUMN" on page 4-12.

Example

To display the format settings on the Dollars column, enter:

```
RISQL> query column dollars;
set column dollars width 8;
set column dollars title 'Dollars';
```

Resetting Display Options to Default Values

The RESET COLUMN command restores one or more of the column display options to its default value.

You can reset options on:

- A single column, which resets the specified options set on that column.
- A column list, which resets options set on columns with that list specification (for example, market.*).
- All columns with options set with the global column wildcard (*).

When you reset an option for a single column, options on other columns in the result are left intact. When you reset an option on a column list, only the specified option that was set with that particular column-list variable is reset. Resetting global options affects only those options that were set with the global column wildcard character (*), not all column settings.

Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the *SQL Reference Guide*.

For complete syntax information for the RESET COLUMN command, refer to "RESET COLUMN" on page 4-15.

Examples

To reset an option, you must use the same type of column variable and option argument that you used when setting the option. For example, the following command sets the width of the sales column:

```
set column sales.* width 8;
```

To reset the default column width for the sales column, enter:

```
reset column sales.* width;
```

However, this command does not reset the column width if it was set with the following command:

```
set column sales.dollars width 8;
```

Sample Report 1

The following report lists coffee sales in San Jose for the first quarter of 1998. The data is listed by product. Percentages are provided to show the most profitable month for each product for the quarter.

```
RISQL> > select pj.prod_name, dj.month, sum(dollars) as
mon_sales_98,
> dec(100 * sum(dollars)/
> (select sum(si.dollars)
> from store ri natural join sales si
> natural join product pi
> natural join period di
> where di.qtr = dj.qtr
> and di.year = dj.year
> and pi.prod_name = pj.prod_name
> and pi.pkg_type = pj.pkg_type
> and ri.city = rj.city), 7, 2) as pct_qtr1
> from store rj natural join sales sj
> natural join product pj
> natural join period dj
> where rj.city = 'San Jose'
> and dj.year = 1998
> and dj.qtr = 'Q1_98'
> and pj.pkg_type = 'One-pound bag'
> group by pj.prod_name, pj.pkg_type, dj.month,
> dj.qtr, dj.year, rj.city
> order by pj.prod_name, pct_qtr1 desc;
```

Result Set

```
PROD_NAME                    MONTH   MON_SALES_98   PCT_QTR1
Aroma Roma                     FEB         688.75      39.91
Aroma Roma                     JAN         594.50      34.45
Aroma Roma                     MAR         442.25      25.63
Cafe Au Lait                   MAR         742.00      40.61
Cafe Au Lait                   JAN         600.50      32.86
Cafe Au Lait                   FEB         484.50      26.51
Colombiano                     JAN         770.75      41.66
Colombiano                     FEB         593.50      32.08
Colombiano                     MAR         485.75      26.25
Demitasse Ms                   MAR         789.25      35.98
Demitasse Ms                   JAN         768.75      35.04
Demitasse Ms                   FEB         635.50      28.97
Expresso XO                    JAN         849.75      50.99
Expresso XO                    MAR         528.00      31.68
Expresso XO                    FEB         288.75      17.32
La Antigua                     MAR        1131.50      39.70
La Antigua                     JAN         888.75      31.18
La Antigua                     FEB         829.25      29.10
Lotta Latte                    JAN         901.00      36.55
Lotta Latte                    MAR         892.50      36.20
Lotta Latte                    FEB         671.50      27.24
NA Lite                        FEB         531.00      43.38
NA Lite                        JAN         441.00      36.02
NA Lite                        MAR         252.00      20.58
Veracruzano                    FEB        1008.00      45.32
Veracruzano                    JAN         632.00      28.41
Veracruzano                    MAR         584.00      26.25
Xalapa Lapa                    MAR         864.00      50.52
Xalapa Lapa                    JAN         639.00      37.36
Xalapa Lapa                    FEB         207.00      12.10
```

Format Settings

To improve the readability of the preceding report, format settings are used to make the following changes:

- Define a default page length and a report title.
- Underline the column titles.
- Suppress repeated values in the Prod_Name column.
- Skip a line after each Product.
- Specify titles for each column that include uppercase and lowercase characters and spaces.
- Adjust the column width and spacing.

The format settings are as follows:

```
set page length 43 using blank lines;
set title 'Q1 Sales in San Jose $DATE $TIME Pg: $PAGE';
set column * underline '-' short;
set column prod_name no repeat;
set column prod_name skip line;
set column 1 title 'Product';
set column 1 width 15;
set column 1 space 2;
set column 2 title 'Month';
set column 2 space 3;
set column 3 title 'Monthly Sales';
set column 3 space 5;
set column 3 width 8;
set column 4 title 'Pct. of Qtr';
set column 4 space 5;
set column 4 width 12;
```

Report 1

The reformatted report is now easier to read and its overall organization more obvious.

```
Q1 Sales in San Jose 10/28/98 05:13:34 Pg: 1
Product          Month     Monthly      Pct. of Qtr
-------          -----     --------     -----------
Aroma Roma             FEB        688.75            39.91
                       JAN        594.50            34.45
                       MAR        442.25            25.63

Cafe Au Lait           MAR        742.00            40.61
                       JAN        600.50            32.86
                       FEB        484.50            26.51

Colombiano             JAN        770.75            41.66
                       FEB        593.50            32.08
                       MAR        485.75            26.25

Demitasse Ms           MAR        789.25            35.98
                       JAN        768.75            35.04
                       FEB        635.50            28.97
Expresso XO            JAN        849.75            50.99
                       MAR        528.00            31.68
                       FEB        288.75            17.32

La Antigua             MAR       1131.50            39.70
                       JAN        888.75            31.18
                       FEB        829.25            29.10

Lotta Latte            JAN        901.00            36.55
                       MAR        892.50            36.20
                       FEB        671.50            27.24

NA Lite                FEB        531.00            43.38
                       JAN        441.00            36.02
                       MAR        252.00            20.58

Veracruzano            FEB       1008.00            45.32
                       JAN        632.00            28.41
                       MAR        584.00            26.25


Q1 Sales in San Jose 10/28/98 05:13:34 Pg: 2

Product          Month     Monthly      Pct. of Qtr
-------          -----     --------     -----------
Xalapa Lapa            MAR        864.00            50.52
                       JAN        639.00            37.36
                       FEB        207.00            12.10
```

Sample Report 2

The following report lists quarterly coffee sales in San Jose for the Aroma Roma, Colombiano, and Lotta Latte brands. This report includes subtotals for each quarter and each year and shows the average cost per unit by quarter.

```
select prod_name, year, qtr, sum(dollars),
sum(quantity), dec(sum(dollars)/sum(quantity),7,2)
> from market ma join store st
> on ma.mktkey = st.mktkey
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
where prod_name in ('Aroma Roma', 'Lotta Latte',
'Colombiano')
> and city = 'San Jose'
> and year in (1998, 1999)
group by prod_name, year, qtr
order by prod_name, year, qtr
break by year summing 4, 5;
```

Result Set

```
PROD_NAME         YEAR      QTR
Aroma Roma 1998 Q1_98              14417.50    2398       6.01
Aroma Roma 1998 Q2_98              12975.00    2220       5.84
Aroma Roma 1998 Q3_98              16423.50    2594       6.33
Aroma Roma 1998 Q4_98              14242.00    2304       6.18
Aroma Roma 1998 NULL               58058.00    9516       NULL
Aroma Roma 1999 Q1_99              17172.50    2766       6.20
Aroma Roma 1999 Q2_99              16658.00    2666       6.24
Aroma Roma 1999 Q3_99              15858.00    2582       6.14
Aroma Roma 1999 Q4_99              15942.50    2614       6.09
Aroma Roma 1999 NULL               65631.00   10628       NULL
Aroma Roma NULL NULL              123689.00   20144       NULL
Colombiano 1998 Q1_98              16048.00    2726       5.88
Colombiano 1998 Q2_98              15498.00    2628       5.89
Colombiano 1998 Q3_98              13278.50    2260       5.87
Colombiano 1998 Q4_98              12390.50    2176       5.69
Colombiano 1998 NULL               57215.00    9790       NULL
Colombiano 1999 Q1_99              16006.50    2752       5.81
Colombiano 1999 Q2_99              17382.00    2882       6.03
Colombiano 1999 Q3_99              15299.50    2624       5.83
Colombiano 1999 Q4_99              17457.00    2702       6.46
Colombiano 1999 NULL               66145.00   10960       NULL
Colombiano NULL NULL              123360.00   20750       NULL
Lotta Latte 1998 Q1_98              24633.00    3258      7.56
Lotta Latte 1998 Q2_98              17433.00    2266      7.69
Lotta Latte 1998 Q3_98              17842.00    2482      7.18
Lotta Latte 1998 Q4_98              18047.00    2528      7.13
Lotta Latte 1998 NULL               77955.00   10534      NULL
Lotta Latte 1999 Q1_99              18753.00    2650      7.07
Lotta Latte 1999 Q2_99              24777.00    3310      7.48
Lotta Latte 1999 Q3_99              17091.00    2306      7.41
Lotta Latte 1999 Q4_99              14935.00    2096      7.12
Lotta Latte 1999 NULL               75556.00   10362      NULL
Lotta Latte NULL NULL              153511.00   20896      NULL
NULL NULL NULL              400560.00   61790      NULL
```

Format Settings

To improve the readability of the preceding report, format settings are used to make the following changes:

- Define a title for the report.
- Specify titles for each column that include uppercase and lowercase characters and spaces.
- Underline column titles.
- Adjust column widths and spacing.

- Skip a line between each year.
- Add labels.
- Suppress nulls (which appear in some BREAK BY fields).
- Suppress repeated values in the Product column.
- Custom-justify data in specified columns.
- Print data for each product on a separate page.

The format settings are as follows:

```
set title '1998-1999 Sales in S. J. by Brand $DATE Pg: $PAGE';
set column * if null ' ';
set column * underline '=' short;
set column prod_name title 'Product';
set column 1 no repeat;
set column 1 skip page;set column year title 'Year';
set column year width 4;
set column year no repeat;
set column year skip line;
set column qtr title 'Qtr';
set column qtr width 5;
set column qtr space 2;
set column qtr justify left;
set column qtr no repeat;
set column 4 title 'Sales';
set column 4 width 10;
set column 4 space 2 '$';
set column 5 title 'Quantity';
set column 5 width 8;
set column 5 space 2;
set column 6 title 'Unit Cost';
set column 6 width 9;
set column 6 space 2;
```

Report 2

The reformatted report is now easier to read and its overall organization more obvious.

```
1998-1999 Sales in S. J. by Brand 10/28/97 Pg: 1
Product           Year     Qtr      Sales      Quantity     Unit Cost
=======           ====     ===      =====      ========     =========
Aroma Roma        1998     Q1_98  $  14417.50    2398         6.01
                           Q2_98  $  12975.00    2220         5.84
                           Q3_98  $  16423.50    2594         6.33
                           Q4_98  $  14242.00    2304         6.18
                                  $  58058.00    9516

                  1999     Q1_99  $  17172.50    2766         6.20
                           Q2_99  $  16658.00    2666         6.24
                           Q3_99  $  15858.00    2582         6.14
                           Q4_99  $  15942.50    2614         6.09
                                  $  65631.00   10628
                                  $ 123689.00   20144
1998-1999 Sales in S. J. by Brand 10/28/97 Pg: 2
Product         Year      Qtr      Sales        Quantity    Unit Cost
=======         ====      ===      =====        ========    =========
Colombiano      1998     Q1_98  $  16048.00       2726         5.88
                         Q2_98  $  15498.00       2628         5.89
                         Q3_98  $  13278.50       2260         5.87
                                $  57215.00       9790

                1999     Q1_99  $  16006.50       2752         5.81
                         Q2_99  $  17382.00       2882         6.03
                         Q3_99  $  15299.50       2624         5.83
                         Q4_99  $  17457.00       2702         6.46
                                $  66145.00      10960
                                $ 123360.00      20750
1998-1999 Sales in S. J. by Brand 10/28/97 Pg: 3
Product          Year      Qtr      Sales       Quantity    Unit Cost
=======          ====      ===      =====       ========    =========
Lotta Latte      1998    Q1_98  $  24633.00       3258         7.56
                         Q2_98  $  17433.00       2266         7.69
                         Q3_98  $  17842.00       2482         7.18
                         Q4_98  $  18047.00       2528         7.13
                                $  77955.00      10534

                 1999    Q1_99  $  18753.00       2650         7.07
                         Q2_99  $  24777.00       3310         7.48
                         Q3_99  $  17091.00       2306         7.41
                         Q4_99  $  14935.00       2096         7.12
                                $  75556.00      10362
                                $ 153511.00      20896
                                $ 400560.00      61790
```

Chapter

Command Reference

4

In This Chapter 4-3

Commands 4-3
! (shell escape) 4-6
CONNECT 4-7
EDIT 4-9
QUERY 4-11
QUERY COLUMN 4-12
QUIT or EXIT 4-13
RESET 4-14
RESET COLUMN 4-15
RUN 4-18
SAVE 4-19
SET COLUMN 4-20
SET COLUMN column FILL 4-24
SET COLUMN column FOLD LINE 4-26
SET COLUMN column FORMAT 4-28
SET COLUMN column IF NULL. 4-30
SET COLUMN column JUSTIFY 4-31
SET COLUMN column NO PRINT 4-33
SET COLUMN column NO REPEAT 4-35
SET COLUMN column POSITIVE SIGN 4-37
SET COLUMN column SIGN 4-39
SET COLUMN column SKIP 4-41
SET COLUMN column SPACE 4-43
SET COLUMN column TITLE. 4-46
SET COLUMN column UNDERLINE 4-48
SET COLUMN column WIDTH 4-50

SET ECHO 4-52
SET EDITOR 4-54
SET ERROR OUTPUT 4-55
SET OUTPUT 4-56
SET PAGE LENGTH 4-58
SET PROMPT 4-60
SET ROW DATA 4-62
SET TITLE 4-64

In This Chapter

This chapter contains reference pages in alphabetical order that describe RISQL Entry Tool and RISQL Reporter commands. These commands are not case sensitive.

Many of the commands documented in this chapter are available only with RISQL Reporter, and some commands have options that apply only to RISQL Reporter. These restrictions are noted in the right column of the following table and at the top of each applicable reference page.

Commands

Command	Description	RISQL Reporter Only
!	Starts a system shell session or executes a single shell command (shell escape).	
CONNECT	Connects to a database with the current username, an alternative username, or a DSN.	
EDIT	Edits the current command buffer or a specified file.	
QUERY	Displays current RISQL Entry Tool and RISQL Reporter settings.	✔ *(some options)*
QUERY COLUMN	Displays RISQL Reporter settings for the SET COLUMN command and options.	✔
QUIT EXIT	Ends the current tool session.	

(1 of 3)

Command	Description	RISQL Reporter Only
RESET	Restores current RISQL Entry Tool and RISQL Reporter settings to default values.	✔ *(some options)*
RESET COLUMN	Removes column display options.	✔
RUN	Executes the current command buffer or a specified file.	
SAVE	Saves all current RISQL Reporter settings to a specified file.	✔
SET COLUMN	Sets display options for a result column.	✔
SET COLUMN *column* FILL	Sets the left-fill character for data in a column.	✔
SET COLUMN *column* FOLD LINE	Breaks a query row into two or more print lines.	✔
SET COLUMN *column* FORMAT	Sets the format of numeric data (DECIMAL, INTEGER, EXPONENTIAL).	✔
SET COLUMN *column* IF NULL	Defines the string that is displayed when nulls are returned.	✔
SET COLUMN *column* JUSTIFY	Sets the justification of data in a column.	✔
SET COLUMN *column* NO PRINT	Prevents a column from being displayed in the result.	✔
SET COLUMN *column* NO REPEAT	Suppresses the display of duplicate values in the result column.	✔
SET COLUMN *column* POSITIVE SIGN	Controls the display of the plus sign (+) for integer and decimal data.	✔
SET COLUMN *column* SIGN	Controls the placement of the displayed signs (+ or –) for integer and decimal data.	✔
SET COLUMN *column* SKIP	Skips a line or starts a new page when a value in a column changes.	✔
SET COLUMN *column* SPACE	Sets the lead spacing before a column.	✔
SET COLUMN *column* TITLE	Sets a column title.	✔

(2 of 3)

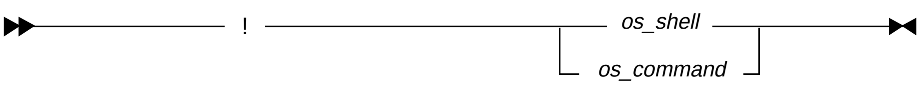
Command	**Description**	**RISQL Reporter Only**
SET COLUMN *column* UNDERLINE	Displays a string of repeating characters underneath the column title.	✔
SET COLUMN *column* WIDTH	Sets the width of each column.	✔
SET ECHO	Turns command echoing on or off.	
SET EDITOR	Specifies the system editor invoked by the EDIT command.	
SET ERROR OUTPUT	Sends error messages to a file or discards them.	
SET OUTPUT	Sends output to a file or discards it.	
SET PAGE LENGTH	Sets the number of lines on each report page.	✔
SET PROMPT	Defines the prompt strings displayed by the RISQL Entry Tool or RISQL Reporter.	
SET ROW DATA	Suppresses output of column headings and any other nondata row information in result set.	
SET TITLE	Creates a report title.	✔

(3 of 3)

Important: *All user-defined object names (except filenames and passwords) specified in RISQL Entry Tool or RISQL Reporter commands can contain single-byte or multi-byte characters.*

! (shell escape)

Suspends the tool session and starts a system shell session or executes a single shell command and reactivates the tool session.



Interactive Shell Session

When a shell process is initialized, your RISQL Entry Tool or RISQL Reporter session is suspended and you can execute one or more shell commands. Unlike RISQL Entry Tool or RISQL Reporter commands, system commands executed from the shell are always case sensitive. The exit command ends the shell session and reactivates the tool session.

Always remember to terminate the interactive shell session and return to your RISQL Entry Tool or RISQL Reporter session. Otherwise, you might inadvertently leave yourself connected to a database, which might interfere with the work of other users.

Single Shell-Command Execution

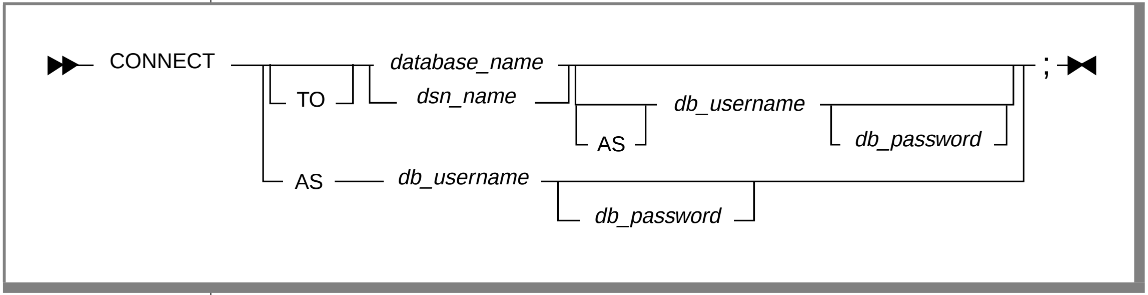
When used with a system command, the ! command suspends the current tool session, starts a new shell process, executes the specified command, and returns to the tool prompt. You can issue only one shell command at a time with this method and the operating-system command controls the output.

Example

```
RISQL>!ls -l *.risql
-rw-rw-rw-   1 curly    pubs    13 Oct 16 13:23 cume.risql
-rw-rw-rw-   1 curly    pubs    13 Oct 16 13:24 mvgavg.risql
-rw-rw-rw-   1 curly    pubs    13 Oct 16 13:23 ratio.risql
-rw-rw-rw-   1 curly    pubs    13 Oct 16 13:24 share.risql
RISQL>
```

CONNECT

Connects to a database with the current username or an alternative username.



Database Connection

When you issue the CONNECT command during a session, any current database connection is automatically disconnected. If the new connection fails, the tool session remains active, but your previous database connection is lost. You cannot access a database until you make a valid connection. You can connect to the same database under the new *db_username* by using the CONNECT AS command.

You can connect to a database on the local server with a DSN or the CONNECT command. However, to connect to a database on a remote server, you must use a DSN.

Prompting for Username and Password

When input comes from a terminal:

- If you omit your username, the tool prompts for *db_username* and *db_password*.
- If you omit your password, the tool prompts for *db_password* alone.

When you redirect input from a file, you must provide your *db_username* and *db_password* in the file.

Example

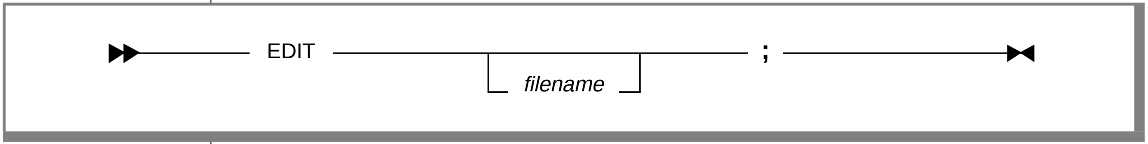
```
RISQL> connect to aroma as curly ;
Please enter password: xxxxxxxx
RISQL>
RISQL> connect to aroma_DSN;
Please enter password: xxxxxxxx
RISQL>
```

See Also

Appendix A, “risql and risqlrpt Command Reference.”

EDIT

Edits the current command buffer or a specified file.



Editing a File

Issue the EDIT command with a *filename* to start an editor and open *filename*. After you terminate the editing session, you can execute the contents of *filename* with the RUN command. The default editor on UNIX platforms is `vi` and the default editor on Windows platforms is *Notepad*.

Editing the Buffer

When you omit *filename*, the contents of the command buffer are placed in a temporary file for editing. The command buffer contains the most recently executed SQL statement or tool command, with the exception of the EDIT and RUN commands. EDIT and RUN declarations cannot be edited in the command buffer. You can save any changes you make to the temporary file by writing them to the original file with an editor command. Otherwise, any changes are made only in the temporary copy and are not retained.

Displaying the Current Editor

QUERY EDITOR displays the current editor.

Example

```
RISQL> edit store_rank ;
select store_name, sum(dollars) as total_sales,
    rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
    join sales s on s.storekey = r.storekey
    join period p on s.perkey = p.perkey
where year = 1999
    and month = 'MAR'
    and region = 'West'
group by store_name;
~
~
~
~
"store_rank" 10 lines, 292 characters
```

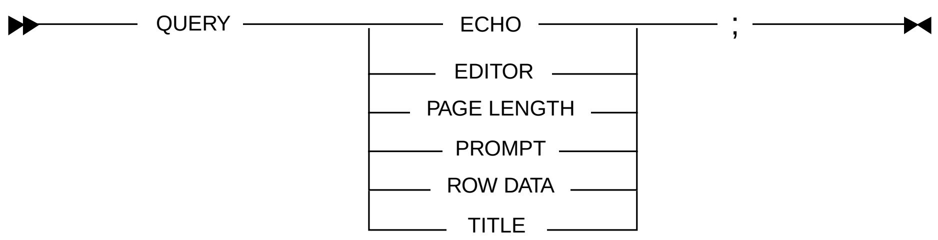
See Also

RUN
SET EDITOR
SET OUTPUT

QUERY

(Some options RISQL Reporter only)

Displays settings for various RISQL Entry Tool and RISQL Reporter commands.



Example

```
    RISQL> query echo ;
    SET ECHO ON ;
```

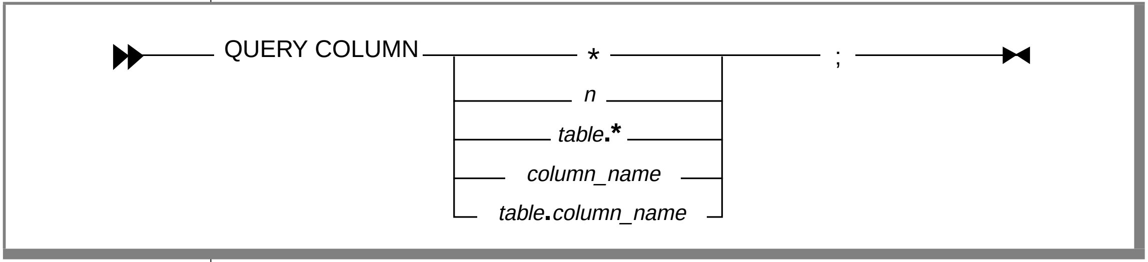
See Also

SET ECHO
SET EDITOR
SET PAGE LENGTH
SET PROMPT
SET ROW DATA
SET TITLE

QUERY COLUMN

(RISQL Reporter only)

Displays RISQL Reporter settings for the SET COLUMN command and its display options.



Column Declaration

The following table describes valid column variables:

Column Variable	Description
*	Global column list
n	Relative column number (3)
*table.**	Table column list (sales.*)
column_name	Unqualified column name (dollars)
table.column_name	Table-qualified column name (sales.dollars)

Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the *SQL Reference Guide*.

Example

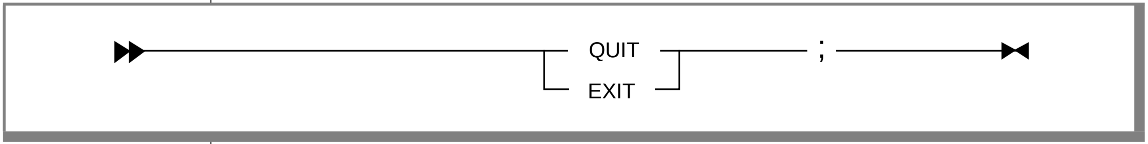
```
RISQL> query column dollars ;
COLUMN DOLLARS WIDTH 8
COLUMN DOLLARS FORMAT DECIMAL
```

See Also

RESET COLUMN
SET COLUMN

QUIT or EXIT

Closes the current tool session and returns to the system prompt.



Example

```
RISQL> quit ;
$
```

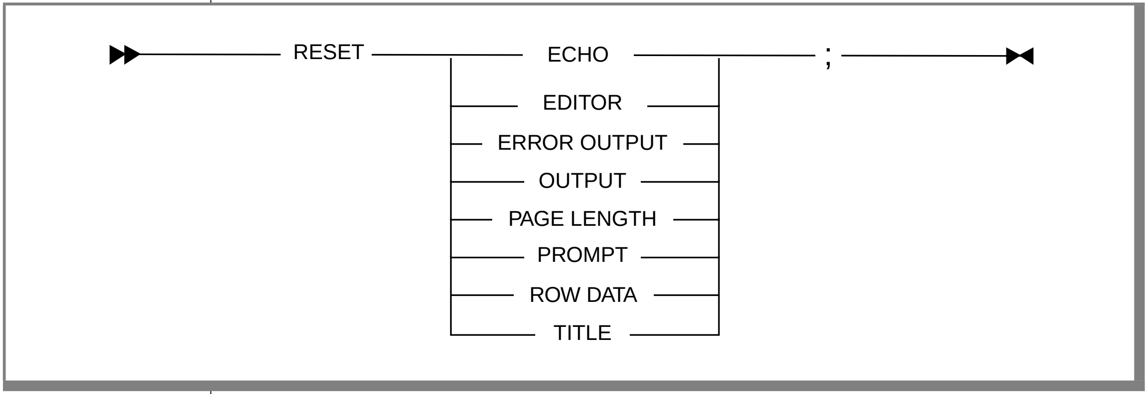
See Also

None.

RESET

(Some options RISQL Reporter only)

Restores settings for various RISQL Entry Tool and RISQL Reporter commands to default values.



Example

```
    RISQL> reset editor ;
```

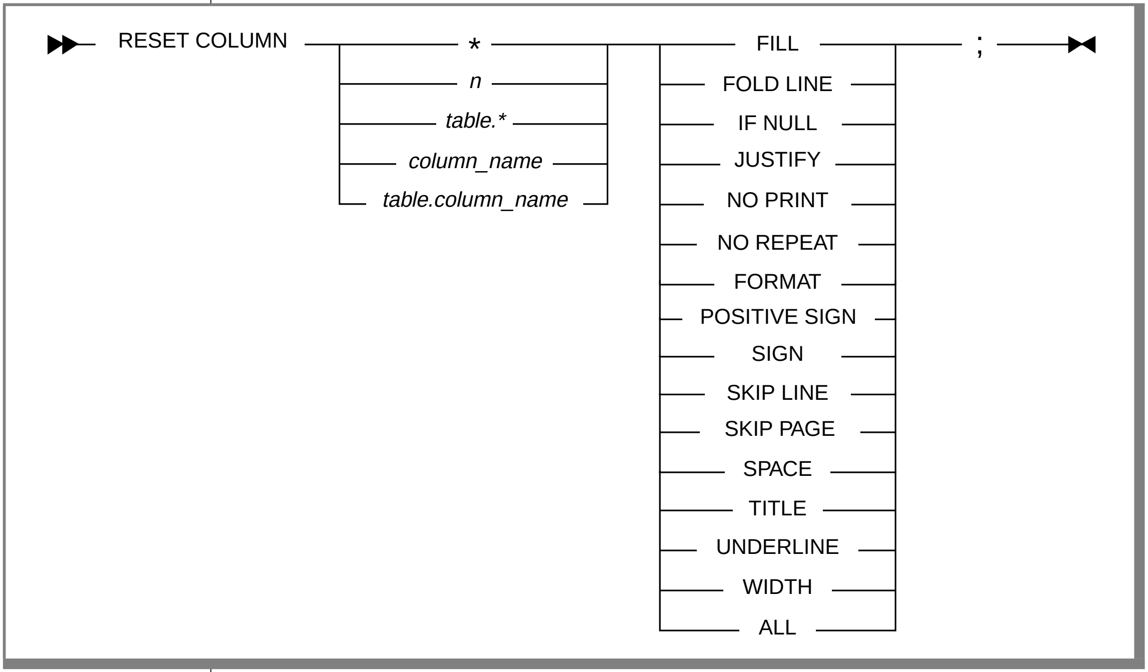
See Also

SET ECHO
SET EDITOR
SET ERROR OUTPUT
SET OUTPUT
SET PAGE LENGTH
SET PROMPT
SET ROW DATA
SET TITLE

RESET COLUMN

(RISQL Reporter only)

Removes column display settings and restores options to default values.



Column Declaration

The following table describes valid column variables:

Column Variable	Description
*	Global column list
n	Relative column number (3)
*table.**	Table column list (sales.*)
column_name	Unqualified column name (dollars)
table.column_name	Table-qualified column name (sales.dollars)

When you reset an option for a single column, options on other columns in the result are left intact. When you reset an option on a column list, only the specified option that was set with that particular column-list variable is reset. Resetting global options resets only those options that were set with the global column wildcard character (*), not all column settings.

Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the *SQL Reference Guide*.

Option Declaration

The following table briefly describes each valid column display option. Format options are described in detail on individual SET COLUMN reference pages.

Format Option	Description
FILL	Sets the left-fill character for data in a column.
FOLD LINE	Breaks a data row into two or more lines.
IF NULL	Defines the string that is displayed when null is returned.
JUSTIFY	Sets the justification of data in a column.
NO PRINT	Prevents a column from being displayed in the result.
NO REPEAT	Suppresses the display of duplicate values in a column.
FORMAT	Sets the format of numeric data in a column (DECIMAL, INTEGER, EXPONENTIAL).
POSITIVE SIGN	Controls the display of the plus sign (+) for integer and decimal data.
SIGN	Controls the placement of the plus and minus signs (+ and –) for integer and decimal data.
SKIP LINE	Inserts a blank row between data rows when a value in a column changes.
SKIP PAGE	Starts a new page when a value in a column changes.
SPACE	Sets lead spacing for data in a column.

(1 of 2)

Format Option	Description
TITLE	Defines a column title.
UNDERLINE	Displays a string of repeating characters between column headings and data.
WIDTH	Sets the width of a column.
ALL	Resets options for all columns to their default values.

(2 of 2)

Examples

To reset an option, you must use the same type of column variable and option argument that you used when setting the option. For example, the following command sets the width of the perkey column in the sales table:

```
RISQL> set column 1 width 1;
```

To reset the default column width for the quantity column, enter:

```
RISQL> reset column 1 width;
```

However, it does not reset the column width if it was set with the following command:

```
RISQL> set column perkey width 1;
```

The following command restores the JUSTIFY option for the sales.dollars column to its default value (RIGHT, in this case):

```
RISQL> reset column sales.dollars justify;
```

The following RESET command restores all display options set with the `sales.*` column variable to their default values:

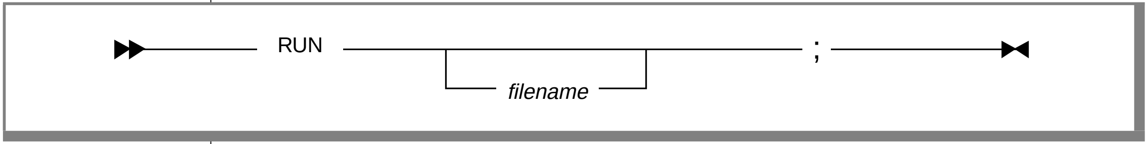
```
RISQL> reset column sales.* all;
```

See Also

QUERY COLUMN
SET COLUMN

RUN

Executes the current command buffer or a specified file.



When you specify *filename*, the RUN command reads and executes the commands in the file.

When you omit *filename*, the tool reads the temporary buffer in the following order of precedence:

- If the most recent tool command was EDIT, issuing the RUN command without *filename* executes the previously edited file.
- If the most recent tool command was not EDIT, issuing the RUN command without *filename* executes the command buffer, which contains the last executed command.

Example

```
RISQL> set echo on ;
RISQL> select store_name, sum(dollars) as total_sales,
>               rank(sum(dollars)) as sales_rank
>  from market m join store r on m.mktkey = r.mktkey
>               join sales s on s.storekey = r.storekey
>               join period p on s.perkey = p.perkey
>  where year = 1999
>               and month = 'MAR'
>               and region = 'West'
>  group by store_name;
STORE_NAME                       TOTAL_SALES     SALES_RANK
Cupertino Coffee Supply             18801.50              1
San Jose Roasting Company           18346.90              2
Beaches Brew                        18282.05              3
Java Judy's                         17826.25              4
Instant Coffee                      15650.50              5
Roasters, Los Gatos                 12694.50              6
```

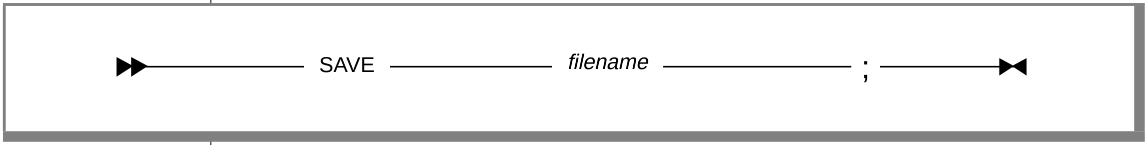
See Also

EDIT
SET ECHO

SAVE

(RISQL Reporter only)

Saves current RISQL Reporter format settings in a specified file.



Valid Commands

The following commands and options can be saved in a file with the SAVE command:

- SET PAGE LENGTH
- SET TITLE
- SET COLUMN (all options)
- Changing Default Settings

You can use the RUN command to execute saved format settings in a file, which lets you develop reusable templates for commonly used report formats. You can also copy the contents of a saved file to the RISQL Reporter initialization files (*.rbretrc*) to set customized default settings for all subsequent RISQL Reporter sessions.

Example

```
    RISQL> save set_commands ;
```

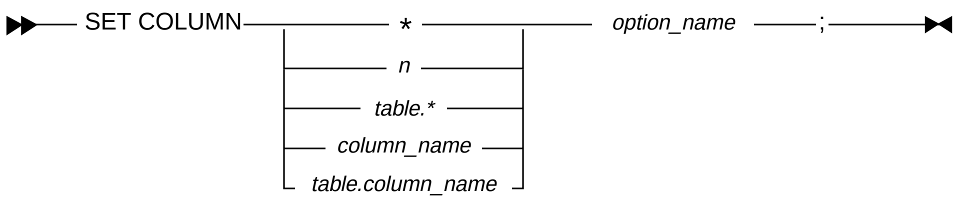
See Also

RUN
Appendix B, ".rbretc Files."

SET COLUMN

(RISQL Reporter only)

Sets column display options.



Column Declaration

The following table describes valid column variables:

Column Variable	Description
*	Global column list
n	Relative column number
*table.**	Table column list (sales.*)
column_name	Unqualified column name (dollars)
table.column_name	Table-qualified column name (sales.dollars)

Column Variable Precedence

The following list describes each valid column specification and gives its level of precedence when more than one option is specified:

- Options specified for the global column list apply to all columns retrieved unless overwritten by a more specific column option specified later in the command syntax.
- Options specified for a relative column number are applied to the column in that position in the query. Relative column number options modify the characteristics of columns that are literals, subqueries, expressions, or display functions. Relative column numbers are retained even when columns are suppressed with the NO PRINT option of the SET COLUMN command. Relative column number options override global options.
- Options specified for a table column list are applied to all columns in that table. Table column list options override global and relative column number options.
- Options specified for unqualified column names are applied to all columns with that name, regardless of the table to which the column belongs. Column aliases defined in a SELECT statement can also be used as unqualified column names. Unqualified column options override global column lists, table column lists, and relative column number options.

Important: *Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the "SQL Reference Guide."*

- Options specified for table-qualified column names are applied to only the specific column. Column aliases defined in a SELECT statement cannot be used in table-qualified names. Table-qualified column name options take precedence over any other option specified for the column.

Option Declaration

The following table briefly describes each valid column display option:

Format Option	Description
FILL	Sets the left-fill character for data in a column.
FOLD LINE	Breaks a data row into two or more lines.
IF NULL	Defines the string that is displayed when null is returned.
JUSTIFY	Sets the justification of data in a column.
NO PRINT	Prevents a column from being displayed in the result.
NO REPEAT	Suppresses the display of duplicate values in a column.
FORMAT	Sets the format of numeric data in a column (DECIMAL, INTEGER, EXPONENTIAL).
POSITIVE SIGN	Controls the display of the plus sign (+) for integer and decimal data.
SIGN	Controls the placement of the plus and minus signs (+ and –) for integer and decimal data.
SKIP LINE	Inserts a blank row between data rows when a value in a column changes.
SKIP PAGE	Starts a new page when a value in a column changes.
SPACE	Sets lead spacing for data in a column.
TITLE	Defines a column title.
UNDERLINE	Displays a string of repeating characters between column headings and data.
WIDTH	Sets the width of a column.

SET COLUMN display options are described in detail in the topics listed under "See Also" below.

Displaying Current Settings

Use the QUERY COLUMN command to display current column settings.

Restoring Defaults

Use the RESET COLUMN command to return column settings to their default values.

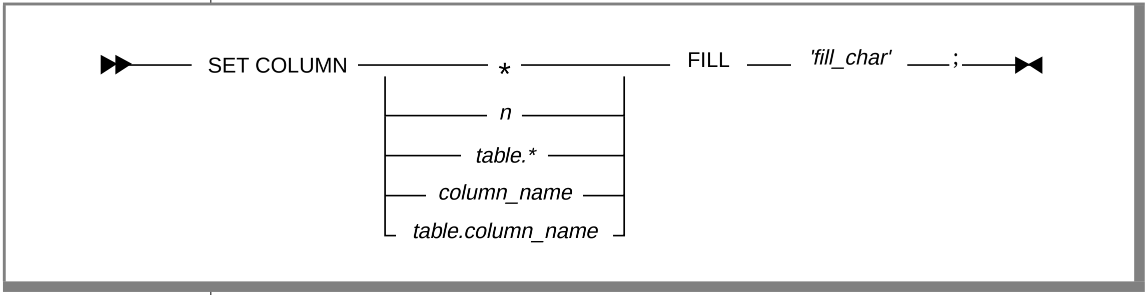
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN column FILL
SET COLUMN column FOLD LINE
SET COLUMN column FORMAT
SET COLUMN column IF NULL
SET COLUMN column JUSTIFY
SET COLUMN column NO PRINT
SET COLUMN column NO REPEAT
SET COLUMN column POSITIVE SIGN
SET COLUMN column SIGN
SET COLUMN column SKIP
SET COLUMN column SPACE
SET COLUMN column TITLE
SET COLUMN column UNDERLINE
SET COLUMN column WIDTH

SET COLUMN column FILL

(RISQL Reporter only)

Defines a left-fill character for a result column.



Using FILL with the Justify Option

When the FILL option is in use, the JUSTIFY LEFT option has no effect on data in the column because every character space in a column field is filled.

Using FILL with Negative Numeric Data

The RISQL Reporter makes a special provision for negative numbers when you format integer or decimal data with leading zero characters. In this case, the leading minus sign (–) indicating a negative number is placed in the leftmost character position. When you fill with any other character, the minus sign appears immediately to the left of the leftmost numeric data character.

Using FILL with Positive Numeric Data

If you format integer or decimal data with leading zero characters and display the plus sign, the leading plus sign (+) indicating a positive number is placed in the leftmost character position. However, if you fill a column containing integer or decimal data with nonzero characters, the plus sign in a leading position is not displayed.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on the column.

Restoring Default

RESET COLUMN *column* FILL returns the FILL option to its default state.

Example

```
RISQL> set column emp_no fill '0' ;
RISQL> select emp_no from employees;
```

Data in the Emp_No column is displayed in the result as follows:

```
EMP_NO
00012
00036
00037
00052
00123
00320
```

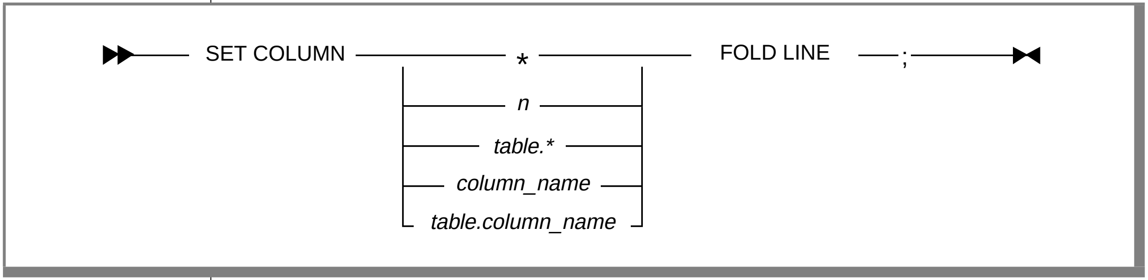
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN *column* JUSTIFY
SET COLUMN *column* SIGN

SET COLUMN column FOLD LINE

(RISQL Reporter only)

Breaks a query row into two or more lines.



\n Character

The FOLD LINE option of the SET COLUMN command causes the RISQL Reporter to insert a new-line (\n) character before displaying the specified column. This option lets you break a long row into several rows, so the total width of the line does not exceed the width of your anticipated output device (such as a small terminal screen or a printer).

Multiple Columns

Multiple FOLD LINE columns are permitted.

Stacking Data

Placing FOLD LINE on every column (set column * fold line) stacks data, displaying a column-oriented report rather than a row-oriented report. Using the FOLD LINE option in this manner is particularly useful with the LABEL option. Including a SET ROW DATA ONLY statement prevents stacked column headings from being displayed.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* FOLD LINE returns the FOLD LINE option to its default state.

Example

```
RISQL> set column city fold line;
RISQL> select prod_name, city, quantity,
>  cume(quantity) as running_total
>  from market ma join store st
>         on ma.mktkey = st.mktkey
>         join sales sa on st.storekey = sa.storekey
>         join product pr on pr.prodkey = sa.prodkey
>         join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>     or prod_name like 'Cafe%')
>  and city like 'New O%'
>  and week = 32
>  and year = 1999
>  order by prod_name, quantity asc;



PROD_NAME
 CITY                    QUANTITY    RUNNING_TOTAL
Cafe Au Lait
 New Orleans                    5                  5
Cafe Au Lait
 New Orleans                    5                 10
Cafe Au Lait
 New Orleans                   14                 24
Cafe Au Lait
 New Orleans                   14                 38
Expresso XO
 New Orleans                   19                 57
Expresso XO
 New Orleans                   19                 76
```

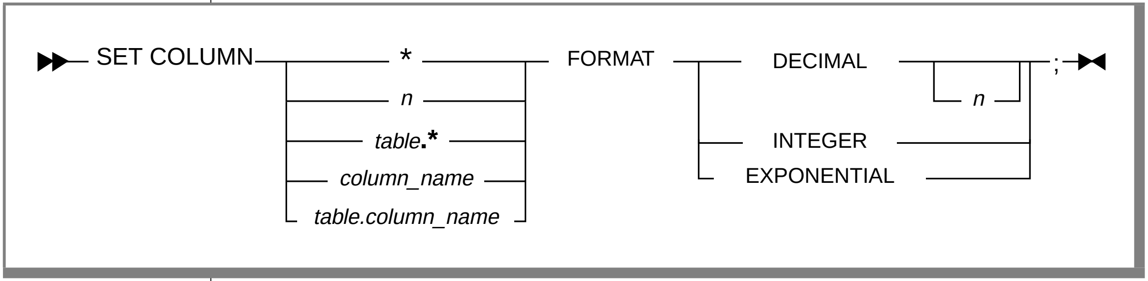
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN *column* SPACE
SET ROW DATA ONLY

SET COLUMN column FORMAT

(RISQL Reporter only)

Changes the numeric format of data in a result column.



Default

Data is displayed in its original datatype by default.

Displayed Asterisks

If the value exceeds the width or format limitation of the column, asterisks (*) are printed in the field. Asterisks occur most often in columns that are formatted as decimal or integer. However, if exponential format is specified and the column width is less than 9, asterisks are displayed.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* FORMAT returns the FORMAT option to its default state.

Example

The following example rounds off decimal values and displays all numbers as integers in column 3:

```
RISQL> set column 3 format integer;
RISQL> select month, sum(dollars) as dols,
>  cume(sum(dollars)) as run_dollars,
>  sum(quantity) as qty,
>  cume(sum(quantity)) as run_quantity
>  from market ma join store st
>  on ma.mktkey = st.mktkey
>  join sales sa on st.storekey = sa.storekey
>  join product pr on pr.prodkey = sa.prodkey
>  join period pe on pe.perkey = sa.perkey
>  and qtr = 'Q2_99'
>  and prod_name = 'Colombiano'
>  and city = 'Los Angeles'
>  group by month;
MONTH DOLS       RUN_DOLL QTY    RUN_QUANTITY
JUN      4011.50  4011.50    660          660
MAY      2853.50  6865.00    482         1142
APR      2736.50  9601.50    452         1594
```

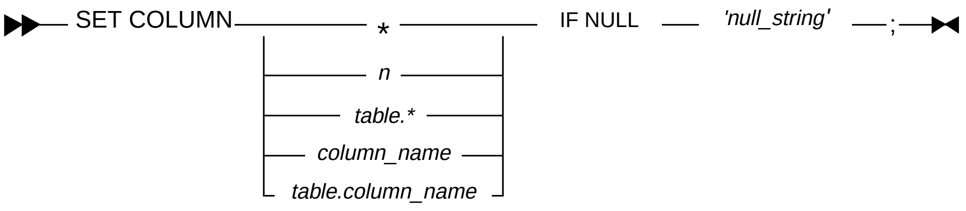
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN *column* WIDTH

SET COLUMN column IF NULL

(RISQL Reporter only)

Defines a string that is displayed when nulls are returned from the database.



Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* IF NULL returns the IF NULL option to its default setting.

Example

```
RISQL> set column mgrno if null 'N/A' ;
RISQL> select name, department, title, mgrno, empno
> from dept natural join employees
> where department like 'Sales%' ;
NAME           DEPARTMENT   TITLE     MGRNO   EMPNO
Blake, Joe     Sales        AcctRep   N/A     1874
Peach, Etta    Sales        AcctRep   N/A     1051
Slo, Monica    Sales        Mgr       089     0498
Tepio, Curtis  Sales        AcctRep   N/A     1179
…
```

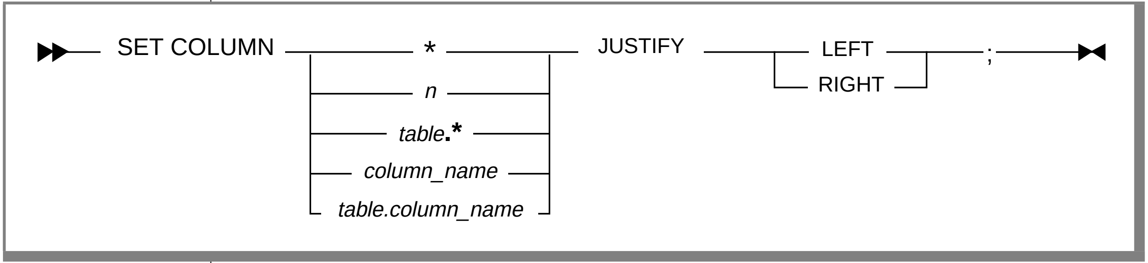
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN

SET COLUMN column JUSTIFY

(RISQL Reporter only)

Justifies data, either left or right, in a result column.



Important: *When the FILL option is in use, the JUSTIFY LEFT option has no effect on data in the column because every character space in a column field is filled.*

Defaults

Default justification is determined by datatype; numeric data is right-aligned and character data is left-aligned. Use the RESET COLUMN statement to reset options to default values.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* JUSTIFY returns the JUSTIFY option to its default state.

Example

In the following example, the employee number column (Emp_No) is of character datatype, and is therefore left-justified by default. You can right-justify employee numbers with the following command:

```
RISQL> set column emp_no justify right
RISQL> select emp_no from employees;
EMP_NO
       12
       36
       37
       52
       123
       320
```

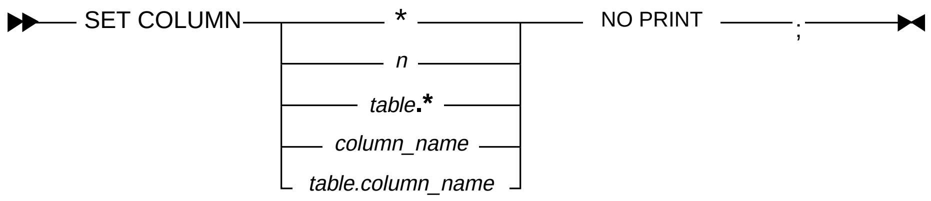
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN

SET COLUMN column NO PRINT

(RISQL Reporter only)

Suppresses the display of a result column.



Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* NO PRINT returns the NO PRINT option to its default state.

Example

The results of the following query apply only to stores of type Large, but the Store_Type column of data (as well as the State column) is hidden in the report:

```
RISQL> set column state no print;
RISQL> set column store_type no print;
RISQL> select store_name, store_type, state,
>              sum(dollars) as sales
>  from store natural join sales
>  where store_type = 'Large'
>  group by store_name, store_type, state;
STORE_NAME                     SALES
Miami Espresso                       507022.35
San Jose Roasting Company            509819.15
Beaches Brew                         503493.10
Olympic Coffee Company               514830.00
```

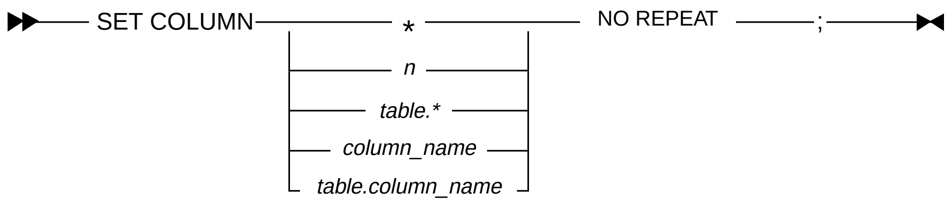
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN

SET COLUMN column NO REPEAT

(RISQL Reporter only)

Suppresses the display of duplicate values in a result column.



Displaying Data Hierarchy

When you specify NO REPEAT on a column, the RISQL Reporter displays duplicate data values as blanks in the result. This option is useful for reports that order data by a column with values that are not unique.

Multiple Columns

When you specify multiple NO REPEAT columns, the RISQL Reporter compares values for each NO REPEAT column starting with the leftmost column. The first column that changes value ends further checking for repeated values in remaining columns to the right in that row.

New Page

If a page break occurs, hidden values are displayed at the top of the new page. Subsequent repeating values are suppressed until another page break occurs.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Defaults

RESET COLUMN *column* NO REPEAT returns the NO REPEAT option to its default state.

Example

```
RISQL> set column state no repeat;
RISQL> set column store_type no repeat;
RISQL> select store_name, state, store_type,
>  sum(dollars) as sales
>  from store natural join sales
>   where store_type = 'Large'
>   group by store_name, store_type, state
> order by state;
STORE_NAME                 STATE STORE_TYPE   SALES
Beaches Brew               CA    Large         503493.10
San Jose Roasting Company                      509819.15
Miami Espresso             FL    Large         507022.35
Olympic Coffee Company     GA    Large         514830.00
```

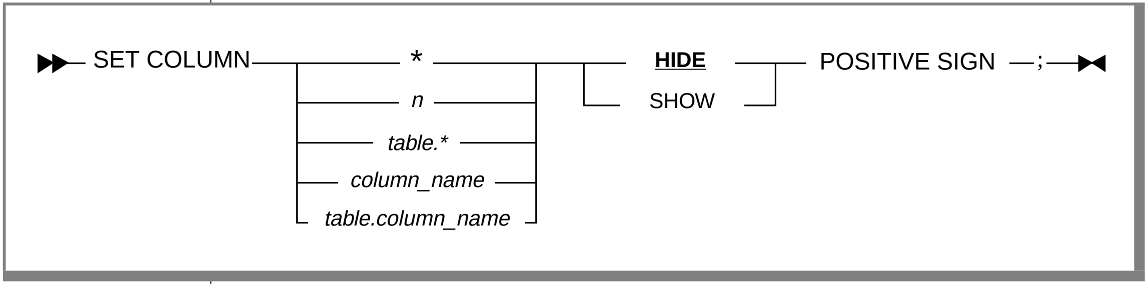
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN

SET COLUMN column POSITIVE SIGN

(RISQL Reporter only)

Specifies whether to display or hide plus signs (+) for positive numbers.



Using with SET COLUMN column FILL

The RISQL Reporter makes a special provision for positive numbers when you format integer or decimal data with leading zero characters and show the positive sign. In this case, the leading plus sign (+) indicating a positive number is placed in the leftmost character position.

If you fill a column containing integer or decimal data with nonzero characters, leading plus signs are not displayed, regardless of the POSITIVE SIGN option. Trailing plus signs, however, are displayed when the POSITIVE SIGN option is set to SHOW.

Using with SET COLUMN column SIGN

The SET COLUMN *column* POSITIVE SIGN command can be used with the SET COLUMN *column* SIGN TRAILING command to display the plus sign (+) at the rightmost character position for the columns.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* POSITIVE SIGN returns the POSITIVE SIGN option to its default state, HIDE.

Example

To display the plus sign, specify the SHOW keyword in the SET COLUMN command, as follows:

```
RISQL> set column dollars show positive sign;
RISQL> select dollars from sales_new;

     DOLLARS
     -132.13
     +566.00
     -212.98
```

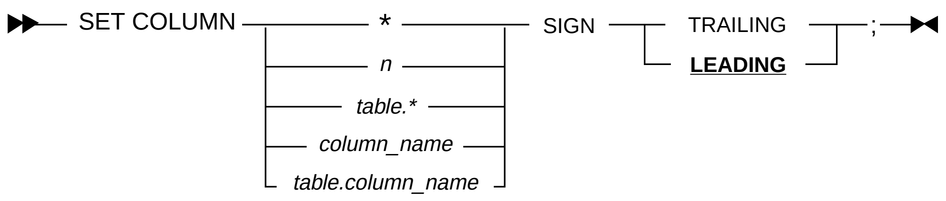
See Also

SET COLUMN column SIGN

SET COLUMN column SIGN

(RISQL Reporter only)

Controls the placement of the signs (+ and –) for integer and decimal data.



Using with SET COLUMN FILL

If you format integer or decimal data with leading zero characters, leading plus and minus signs are placed in the leftmost character position. When you fill with any other character, leading minus signs display immediately to the left of the leftmost numeric data character; and leading plus signs are not displayed, regardless of the POSITIVE SIGN option.

Using with SET COLUMN POSITIVE SIGN

Plus signs (+) for positive numbers are displayed only if specified by the POSITIVE SIGN option.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* SIGN returns the SIGN option to its default state, LEADING.

Example

To move the minus sign to the right of column data, specify the TRAILING keyword in the SET COLUMN command:

```
    RISQL> set column dollars sign trailing;
    RISQL> select dollars from sales_new;
        DOLLARS
        132.13-
        566.00
        212.98-
```

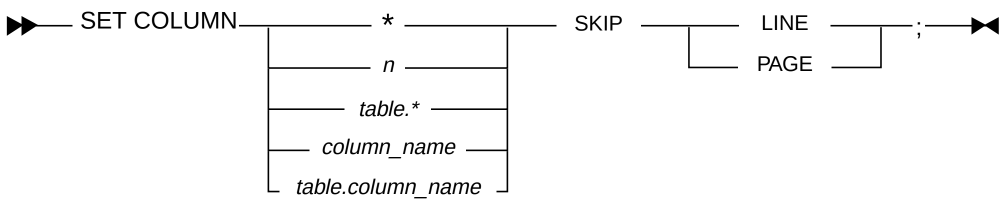
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column FILL
SET COLUMN column JUSTIFY
SET COLUMN column POSITIVE SIGN

SET COLUMN column SKIP

(RISQL Reporter only)

Skips a line or goes forward to a new page when a column value changes in the result.



Multiple Columns

When you use the SKIP option on multiple columns, the RISQL Reporter compares values for each SKIP column starting with the leftmost column in the query. The first column encountered that changes value causes the RISQL Reporter to skip a line or begin a new page, as specified. Remaining SKIP columns in the row are not checked.

If both a SKIP LINE and a SKIP PAGE are used on the same column, SKIP PAGE takes precedence.

Tip: *In general, SKIP PAGE should be used on columns to the left of columns with SKIP LINE settings.*

Using with SET PAGE LENGTH

The RISQL Reporter automatically starts a new page when the number of lines on a page, as specified with SET PAGE LENGTH, is exceeded. If a column value change occurs concurrently with a skip induced by the SET PAGE LENGTH option, only one page is skipped.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* SKIP returns the SKIP option to its default state.

Examples

```
RISQL> set column district skip line;
RISQL> select hq_city, district
> from market
> where region = 'South'
>  or region = 'West'
> order by district ;
HQ_CITY             DISTRICT
Atlanta             Atlanta
Miami               Atlanta

Phoenix             Los Angeles
Los Angeles         Los Angeles

New Orleans         New Orleans
Houston             New Orleans

San Jose            San Francisco
San Francisco       San Francisco
Oakland             San Francisco
```

The following command prints data for each district on a separate page:

```
RISQL> set column district skip page;
```

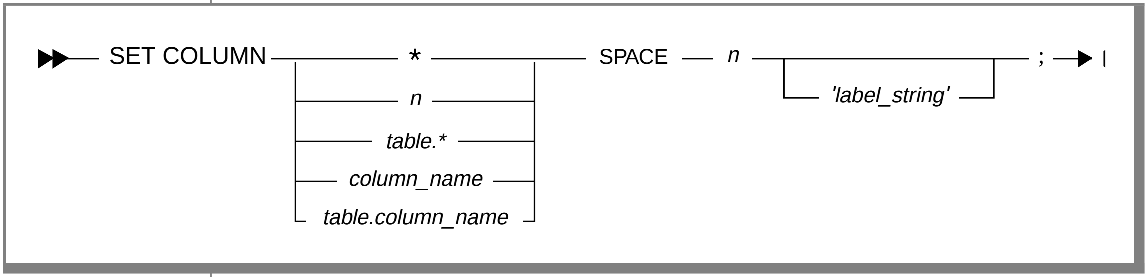
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET PAGE LENGTH

SET COLUMN column SPACE

(RISQL Reporter only)

Defines the lead spacing and data labels for result columns.



Changing Column Margins

The SPACE option changes the space between columns and makes data columns easier to read. The default is one space for all columns except the leftmost column, which has no leading spaces. You can set spacing to zero to simulate concatenated columns.

Column Width

When you use the SPACE option with the WIDTH option of the SET COLUMN command, do not include the length of *label_string* in the width specification; the column expands automatically to include *label_string*.

Wrapping Lines

The *label_string* modifier is often used in conjunction with SET COLUMN * FOLD LINE, which displays each column value on a separate line, as illustrated on page 3-24.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* SPACE returns the SPACE option to its default state.

Examples

To indent the result display, set the leading space on the leftmost column to 5 as follows:

```
RISQL> set column prod_name space 5;
RISQL> select prod_name, city, quantity, dollars
>  from store st
>                join sales sa on st.storekey = sa.storekey
>                join product pr on pr.prodkey = sa.prodkey
>                join period pe on pe.perkey = sa.perkey
>  where (prod_name like 'Expresso%'
>     or prod_name like 'Cafe%')
>               and city like 'New O%'
>                and week = 32
>                and year = 1999
>  order by prod_name, quantity asc;
     PROD_NAME      CITY             QUANTI DOLLARS
     Cafe Au Lait   New Orleans           5     23.75
     Cafe Au Lait   New Orleans           5     23.75
     Cafe Au Lait   New Orleans          14    112.00
     Cafe Au Lait   New Orleans          14    112.00
     Expresso XO    New Orleans          19    114.00
     Expresso XO    New Orleans          19    114.00
```

To precede each value in the Total column with a dollar sign ($), set the label_string as follows:

```
RISQL> set column total width 9;
RISQL> set column total space 1 '$';
RISQL> select district,
> sum(dollars) as total
> from market ma join store st
>   on ma.mktkey = st.mktkey
>   join sales sa on st.storekey = sa.storekey
>   join product pr on pr.prodkey = sa.prodkey
>   join period pe on pe.perkey = sa.perkey
> where prod_name like 'Cafe%'
> and year = 1999
> group by district
> order by total asc;
DISTRICT              TOTAL
Boston               $ 59623.00
Minneapolis          $ 62439.00
New Orleans          $ 62515.50
New York             $ 64072.00
Chicago              $ 66264.00
Atlanta              $ 69294.00
Los Angeles          $ 71858.00
San Francisco        $141803.00
```

Tip: *A label is displayed at the left margin of a column by default. In this example, the column width is modified so the dollar signs are displayed close to the data values.*

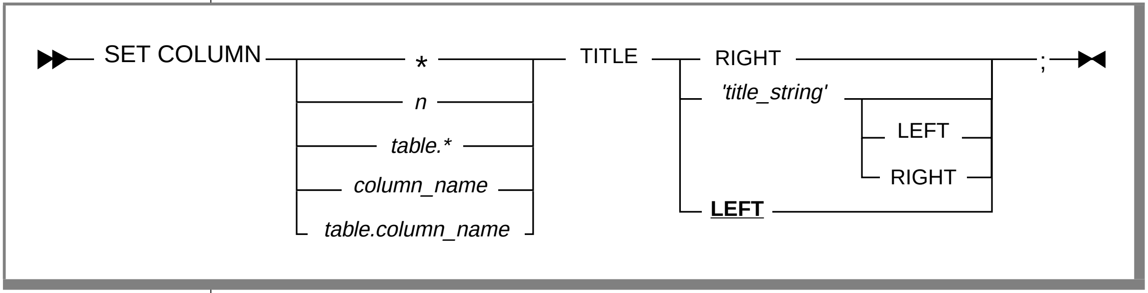
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN *column* FOLD LINE
SET COLUMN *column* WIDTH

SET COLUMN column TITLE

(RISQL Reporter only)

Defines a column title and its position in the result.



Default Title String

The default *title_string* is one of the following:

- Internal name of the column, as stored in the database
- Column alias defined in the SELECT statement
- Literal value of character literals
- Blanks for all other columns

Macros

You cannot use macros in column titles.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* TITLE returns the TITLE option to its default state.

Examples

The following command right-justifies the existing title of the Percent column:

```
RISQL> set column percent title right;
```

The following command changes the title of the Cume (Quantity) column and right-justifies it:

```
> set column 4 title 'RUNNING TOTAL' right;
> select prod_name, city, quantity, cume(quantity)
> from store st
>       join sales sa on st.storekey = sa.storekey
>       join product pr on pr.prodkey = sa.prodkey
>       join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>       or prod_name like 'Cafe%')
>       and city like 'New O%'
>       and week = 32
>       and year = 1999
> order by prod_name, quantity asc;

PROD_NAME       CITY            QUANTITY        RUNNING TOTAL
Cafe Au Lait    New Orleans            5                    5
Cafe Au Lait    New Orleans            5                   10
Cafe Au Lait    New Orleans           14                   24
Cafe Au Lait    New Orleans           14                   38
Expresso XO     New Orleans           19                   57
Expresso XO     New Orleans           19                   76
```

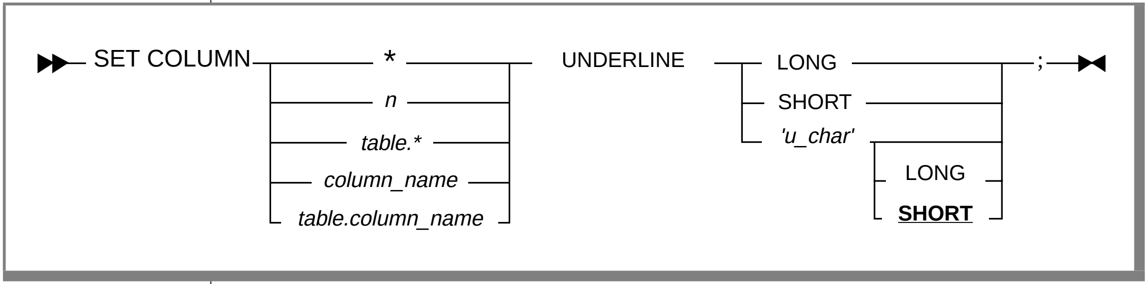
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN

SET COLUMN column UNDERLINE

(RISQL Reporter only)

Displays a horizontal line composed of a specified character string underneath column titles.



Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* UNDERLINE returns the UNDERLINE option to its default state.

Example

```
RISQL> set column * underline '*' short;
RISQL> select prod_name, city, quantity,
>       cume(quantity) as running_total
> from store st
>       join sales sa on st.storekey = sa.storekey
>       join product pr on pr.prodkey = sa.prodkey
>       join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>    or prod_name like 'Cafe%')
>       and city like 'New O%'
>       and week = 32
>       and year = 1999
> order by prod_name, quantity asc;



PROD_NAME       CITY            QUANTITY    RUNNING_TOTAL
*********       ****            ********    *************
Cafe Au Lait    New Orleans            5                5
Cafe Au Lait    New Orleans            5               10
Cafe Au Lait    New Orleans           14               24
Cafe Au Lait    New Orleans           14               38
Expresso XO     New Orleans           19               57
Expresso XO     New Orleans           19               76
```

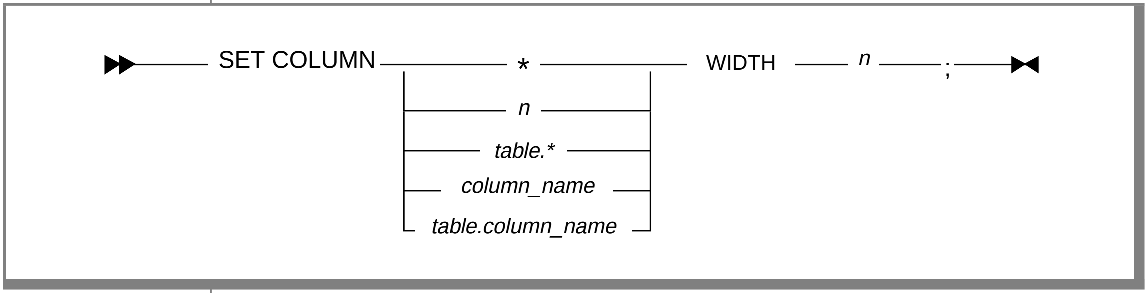
See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN

SET COLUMN column WIDTH

(RISQL Reporter only)

Defines a display width for a result column.



Default Width

The default width of a column field is the datatype width that was defined in the table definition statement. For example, if a column is defined as character(10), the column is displayed as 10 spaces wide in the result. Titles are truncated when they are longer than the defined width.

Default Justification

By default, numeric data is right-justified and character data is left-justified.

Displaying Current Setting

QUERY COLUMN *column* displays all RISQL Reporter options set on *column*.

Restoring Default

RESET COLUMN *column* WIDTH returns the WIDTH option to its default state.

Example

The following command sets the width of the dollars column to 9, which provides two leading spaces before the expected 7-digit data.

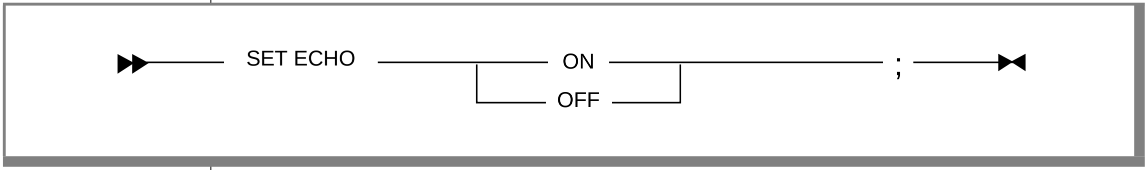
```
RISQL> set column dollars width 9;
```

See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column JUSTIFY
SET COLUMN column TITLE

SET ECHO

Turns command echoing on and off.



Defaults

Default settings differ depending on standard input and standard output settings. If both standard input and standard output are directed to the terminal, the default for SET ECHO is OFF. Otherwise, the default is ON. The following table summarizes the default state for command echoing:

Destination	Input from Terminal	Input from File
Output to Terminal	ECHO off	ECHO on
Output to File	ECHO on	ECHO on

Displaying Current Setting

The QUERY ECHO command displays the current status of the SET ECHO command.

Restoring Default

The RESET ECHO command returns the SET ECHO command to its default state.

Example

If you use the RUN command to execute the contents of the command buffer or a file and ECHO is set to ON, then the contents of the input file are displayed before the output. To submit a query stored in a file named *lotta_latte_LA_99*, enter:

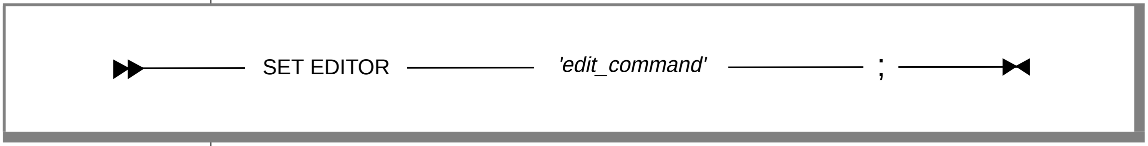
```
RISQL> set echo on;
RISQL> run lotta_latte_LA_99;
RISQL> select count(*) as num_items,
>   string(avg(dollars), 7, 2) as avg_sale,
>   sum(dollars) as total_sales
> from store st
>   join sales sa on st.storekey = sa.storekey
>   join product pr on pr.prodkey = sa.prodkey
>   join period pe on pe.perkey = sa.perkey
> where prod_name = 'Lotta Latte'
>   and year = 1999
>   and city = 'Los Angeles';
NUM_ITEMS   AVG_SALE        TOTAL_SALES
        360    106.69             38405.00
```

See Also

QUERY
RESET
RUN

SET EDITOR

Specifies the system editor, which is invoked by the EDIT command.



Displaying Current Setting

The QUERY EDITOR command displays the current status of the SET EDITOR command.

Restoring Default

The RESET EDITOR command returns the SET EDITOR command to its default state.

Examples

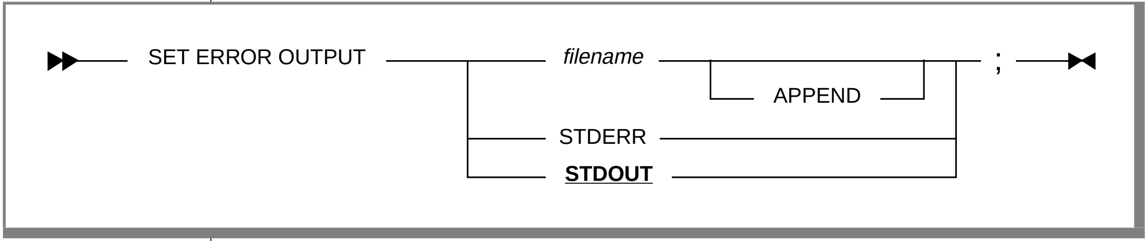
```
RISQL> query editor;
EDITOR 'vi'
RISQL> set editor 'ed';
RISQL>
RISQL> set editor 'ed -s';
RISQL> query editor;
EDITOR 'ed -s'
```

See Also

EDIT
QUERY
RESET

SET ERROR OUTPUT

Sends all error, warning, and informational messages to a designated file.



Using SET ERROR OUTPUT with SET STATS

The error output destination that you specify with the SET ERROR OUTPUT command is also the destination for any statistics output. You obtain statistics output by issuing the SET STATS ON or SET STATS INFO command, as described in the *SQL Reference Guide*.

Restoring Default

The RESET ERROR OUTPUT command resets message output to *stdout*, the default setting.

Example

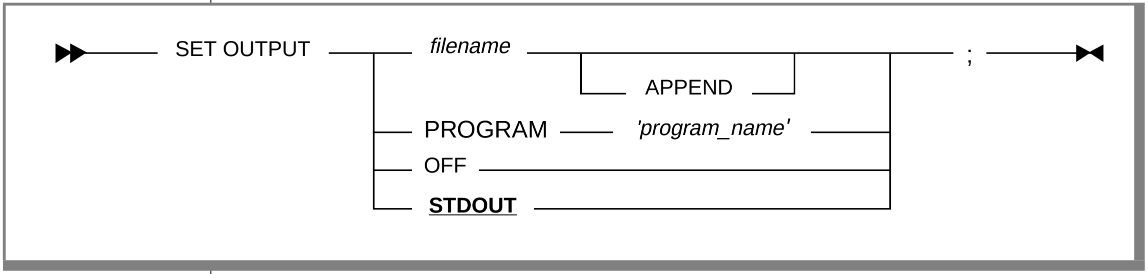
```
RISQL> set error output error_messages;
```

See Also

RESET

SET OUTPUT

Sends query output to a designated file.



filename	Name of a file to which output is sent. The file can be a full or relative pathname. If the specified file exists when the command is issued or the tool starts, its contents are overwritten. Filenames must consist of single-byte characters.
APPEND	Appends output to the specified file.
PROGRAM	Specifies the name of a program that processes output displayed on a terminal. When the PROGRAM option is active, all output lines are sent to the program's standard input.
program_name	Name of output program. The program name can include parameters and must be enclosed in single quotation marks ('). The program name can consist of single-byte or multibyte characters. Typical system output programs include *more* and *cat*. There is no default *program_name*.
OFF	Discards all query output. When you discard output, query statistics are retained and can be displayed with the SET STATS command.
STDOUT	Directs all query output to standard output. Depending on whether output is redirected at the system level, standard output is either the terminal or a file. STDOUT is the default setting.

Restoring Default

The RESET OUTPUT command returns the SET OUTPUT command to its default state.

Examples

UNIX

```
    RISQL> set output /home/curly/test.output append;

    RISQL> set output program 'more';
```

♦

WIN NT

```
    RISQL> set output e: /home/curly/test.output;
```

♦

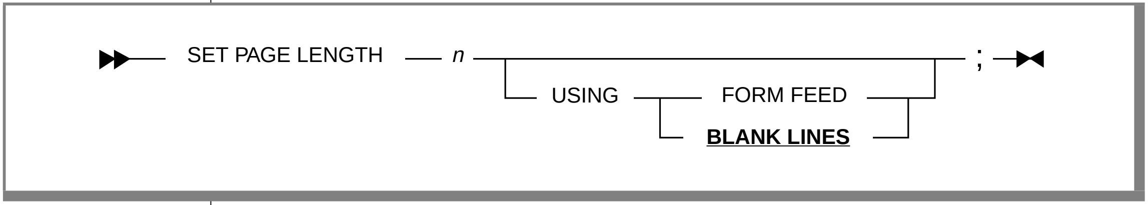
See Also

RESET
RUN
SET ECHO
SET ERROR OUPUT

SET PAGE LENGTH

(RISQL Reporter only)

Sets the number of lines printed on each report page.



Report Heading Options

The following lines can be displayed at the top of each report page:

- Report title followed by a blank line
- Column title
- Column underline

Determining Page Length

The RISQL Reporter prints lines of data until either a page break is indicated by a SKIP PAGE option or the maximum number of report lines, as defined by the SET PAGE LENGTH command, is reached.

The number of report lines displayed on a page is determined by subtracting the heading option lines (report title lines, the column title line, and the column underscore line) and the top and bottom margin allowance lines (4) from *n*. Depending on the various SET options, some or all of these lines might not be displayed and, therefore, might not be needed in *n*.

Setting *n* to 0 creates a page of infinite length or, in other words, one page. This is the default page length.

You must set *n* to at least one line greater than the total of the report title line, the column title line, and the column underscore line. Otherwise, the RISQL Reporter will automatically adjust *n* to 0.

Using with SET COLUMN SKIP

If the SET COLUMN SKIP option is specified when *n* is 0, page skipping is ignored.

When SET COLUMN SKIP is in use:

- If USING BLANK LINES is specified and a page skip is required, the tool completes a partial page by printing the number of blank lines required to fill the page. USING BLANK LINES is the default setting.
- If USING FORM FEED is specified and a page skip is required, a form-feed character is generated to skip to the next page. This character is an ASCII decimal 12 (equivalent to Control-L).

Using with SET COLUMN FOLD LINE

If the SET COLUMN *column* FOLD LINE option is used and results in multiple report lines per row, report lines are printed without breaking a row across two pages. If the number of report lines required for a single row is greater than the number of report lines available on a page, the RISQL Reporter automatically adjusts *n* to 0.

Displaying Current Setting

The QUERY PAGE LENGTH command displays the current status of the SET PAGE LENGTH command.

Restoring Default

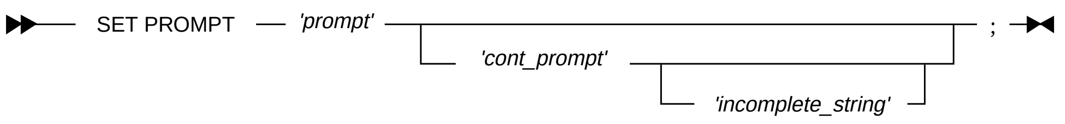
The RESET PAGE LENGTH command returns the SET PAGE LENGTH command to its default state.

See Also

QUERY
RESET
SET COLUMN *column* FOLD LINE
SET COLUMN *column* SKIP PAGE

SET PROMPT

Defines the first-line, continuation, and incomplete string prompts displayed by the RISQL Entry Tool or RISQL Reporter.



Important: *All prompt strings must be enclosed in single quotation marks (').*

Changing Prompts

You can change the first-line prompt without changing the continuation or incomplete string prompts. However, you must change the first-line and continuation prompts, or restate their default values, if you want to change subsequent prompts.

Displaying Current Setting

The QUERY PROMPT command displays the current status of the SET PROMPT command.

Restoring Default

The RESET PROMPT command returns the SET PROMPT command to its default state.

Example

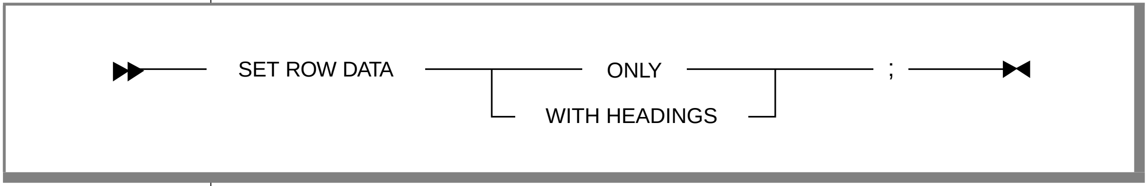
```
RISQL> set prompt '>>> ' '>> ' 'UNMATCHED QUOTE> ';
>>> select * from store
>> where city = 'Los Angeles ;
UNMATCHED QUOTE>
```

See Also

QUERY
RESET

SET ROW DATA

Limits query results to the display of row data only (no column headings, blank lines, or other formatting).



Default Settings

WITH HEADINGS is the default setting in regular mode. ONLY is the default setting in quiet mode. For more information about quiet mode, refer to page 4-63.

Using Row-Only Data

The SET ROW DATA command can be used to extract selected data from a database and then to place that data into a file for transfer to another program. This command can also be used to redirect row data output directly from the RISQL Entry Tool or RISQL Reporter to another program, such as the Table Management Utility (TMU).

Redirecting Extraction Results from Within the Tool

When used together, the following commands capture query results in a file for transfer to another program during a RISQL Entry Tool or RISQL Reporter session:

- SET ROW DATA ONLY causes the query result columns to be returned without column headings of other nondata text.
- SET ECHO OFF suppresses the RISQL prompt and input commands, as described in "Including Input in Output" on page 2-21 and "SET ECHO" on page 4-52.

- SET OUTPUT *filename* sends row data output to a file rather than to the terminal (standard output), as described in "Saving or Discarding Output" on page 2-23.

Operating in Quiet Mode

You can perform data-only extractions at the operating-system level directly from the UNIX shell by specifying quiet mode (-q) in the tool startup command. For more information about quiet mode, refer to "Redirecting Extraction Results from the Shell" on page 2-39 and Appendix B, ".rbretc Files."

Example

To send only the San Francisco Aroma Roma sales figures for the first quarter of 1999 to a file named *qtr_extract*, enter:

```
RISQL> set row data only;
RISQL> set echo off;
RISQL> set output qtr_extract;
RISQL> select * from store st
>   join sales sa on st.storekey = sa.storekey
>   join product pr on pr.prodkey = sa.prodkey
>   join period pe on pe.perkey = sa.perkey
> where year = 1999
>   and qtr = 'Q1_99'
>   and city = 'San Jose'
>   and prod_name = 'Aroma Roma';
```

The result of this query is sent directly to the *qrt_extract* file.

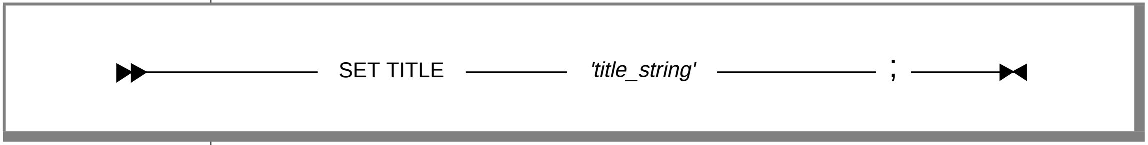
See Also

RESET ROW DATA
SET ECHO OFF
SET ERROR OUTPUT
SET OUTPUT

SET TITLE

(RISQL Reporter only)

Defines a report title.



Title Variables

The *title_string* can contain one or more of the following predefined title variables:

- $DATE displays the date. The datetime data is formatted in accordance with the current client locale.
- $TIME displays the time. The timestamp data is formatted in accordance with the current client locale.
- $PAGE displays the page number.

Title variables are embedded in text strings and are case sensitive. A blank line is displayed under the title and before column titles and data.

Displaying Current Setting

The QUERY TITLE command displays the current title.

Restoring Default

The RESET TITLE command deletes the current title.

Example

```
RISQL> set title 'SALES RANKING IN THE WEST REGION Report Run:
$DATE at $TIME';
RISQL> select store_name, sum(dollars) as total_sales,
                rank(sum(dollars)) as sales_rank
>  from market m join store r on m.mktkey = r.mktkey
>                 join sales s on s.storekey = r.storekey
                join period p on s.perkey = p.perkey
>  where year = 1999
>                 and month = 'MAR'
                and region = 'West'
> >  group by store_name;

> > SALES RANKING IN THE WEST REGION Report Run: 07/02/99 at
09:34:33

STORE_NAME                     TOTAL_SALES     SALES_RANK
Cupertino Coffee Supply               18801.50            1
San Jose Roasting Company             18346.90            2
Beaches Brew                          18282.05            3
Java Judy's                           17826.25            4
Instant Coffee                        15650.50            5
Roasters, Los Gatos                   12694.50            6
```

See Also

QUERY
RESET
SET PAGE LENGTH

Appendix

A

risql and risqlrpt Command Reference

This appendix describes the *risql* and *risqlrpt* commands and their command-line options and environment variables.

risql and risqlrpt

The following syntax defines the command-line use of the *risql* and *risqlrpt* commands to invoke the RISQL Entry Tool and RISQL Reporter:

```
risql [-q] [-s dsn] [-h host] [-d database] [db_username
[db_password]]
risqlrpt [-q] [-s dsn] [-h host] [-d database] [db_username
[db_password]]
```

where:

risql	Invokes the RISQL Entry Tool.
risqlrpt	Invokes the RISQL Reporter.
-d *database*	Specifies a logical database name, as defined in the file *rbw.config*. Alternatively, a logical database name can be supplied by the environment variable RB_PATH or with an SQL CONNECT command and not specified on the _command line. The -d option overrides RB_PATH.
-q	Causes the RISQL Entry Tool or RISQL Reporter to run in quiet mode. In quiet mode, the copyright notice is neither displayed on the screen nor printed to a file when output is redirected; the default setting for SET ECHO is OFF; and the default setting for SET ROW DATA is ONLY.
-s *dsn*	Data source name (DSN) defined in the *.odbc.ini* file on UNIX and Windows systems. The -s *dsn* option takes precedence over the -d *database* option or pertinent environment variables defined for the session. A DSN can also be specified with the RB_DSN environment variable.
-h *host*	RB_HOST value defined in the *rbw.config* file. This RB_HOST value can also be specified with the RB_HOST environment variable.
db_username	Valid database username.
db_password	Database password associated with the specified database username.

Database Connection

If you neither provide the logical database name on the command line nor specify a database name with the RB_PATH environment variable, the RISQL Entry Tool or RISQL Reporter will start, but it will not be connected to a database. Before a database connection is established, the tool accepts only tool-specific or operating-system commands—not SQL statements.

When the -*d* option is provided, the tool searches for a logical database name definition in the *rbw.config* file.

When the -*s dsn* option is provided, the tool searches for a logical data source name (DSN) definition in the *.odbc.ini* file. This option takes precedence over the -s *database* option and pertinent environment variables defined for the session.

If you enter input from your terminal and you provide the *database* option at startup but fail to provide any login information, the tool prompts for both *db_username* and *db_password*. If you provide *db_username* but not *db_password*, the tool prompts for *db_password*.

If command input is redirected from a file before a database connection has been established, you must include a CONNECT statement on the first line of the input file before any SQL statements. The tool prompts for *db_username* and *db_password* only when input comes from a terminal, so you must include these arguments in your CONNECT statement. If either argument is missing from the source file, the database connection attempt fails.

If a database connection is not made when the tool starts, you can use the CONNECT command at the RISQL prompt to establish one.

When a database connection is requested, the tool reads two initialization files named *.rbretrc* from the directory where the Informix Red Brick Decision Server configuration files are located and from your home directory. Any valid SQL statement, RISQL statement, or tool command can be executed from either *.rbretrc* file. For details, refer to Appendix B, ".rbretc Files."

Quiet Mode

The optional -q parameter causes the RISQL Entry Tool or RISQL Reporter to run in quiet mode. In quiet mode, the copyright notice is neither displayed on the screen nor printed to a file when output is redirected. The default settings of the following SET commands are also changed:

- SET ECHO is set to OFF.
- SET ROW DATA is set to ONLY.

Quiet mode is helpful when you are using the RISQL Entry Tool or RISQL Reporter in data extraction scripts. For more information about operating in quiet mode, refer to "Redirecting Extraction Results from the Shell" on page 2-39.

Environment Variables

The RISQL Entry Tool and RISQL Reporter use the following environment variables:

- RB_CONFIG, a pathname to the directory containing the *rbw.config* file, which is usually the directory that contains the *bin* subdirectory of Informix Red Brick Decision Server executable files. RB_CONFIG is required.

  The following error messages are displayed when the RB_CONFIG environment variable is either not defined or defined incorrectly when you start the RISQL Entry Tool:

  - Undefined RB_CONFIG environment variable:

    ```
    ** FATAL ** (803) RB_CONFIG must be defined.
    ```

  - Incorrectly defined RB_CONFIG environment variable:

    ```
    Error in opening config file; unable to continue.
    ** FATAL ** (10502) Message base lookup failed for
    message 10502, error code 200040.
    ```

  When you start the RISQL Reporter, the tool checks the *rbw.config* file for license verification. If the RB_CONFIG environment variable is not set when the tool is started, the file is not found and the following message is displayed:

  ```
  ** FATAL ** (802) Not licensed for 'RISQL_REPORTER'.
  ```

  For more information about Informix Red Brick Decision Server error messages, refer to the *Messages and Codes Reference Guide*.

- RB_PATH, an environment variable that defines a logical database name in the *rbw.config* file. RB_PATH is optional.
- RB_HOST, an environment variable that identifies the Informix Red Brick Decision Server daemon process. RB_HOST is optional.
- RB_DSN, an environment variable that defines a logical data source name.

Option Precedence

If both a command-line option and an environment variable are specified for any argument, the command-line option takes precedence.

Input/Output Redirection

Standard system input and output redirection is permitted (>, <, >>, <<, |).

Redirection settings affect the default ECHO state, which controls whether input is echoed with output.

Examples

The following example shows the username and password prompts when the RISQL Entry Tool is started with a logical database name.

```
$ risql -d aroma
(C) Copyright 1991-1999 Informix Software, Inc.
All rights reserved.
RISQL Entry Tool Version 6.0.1(0)
Please type username:
Please enter username: curly
Please type password: xxxxxxxx
RISQL>
```

The following example shows the username and password prompts when the RISQL Reporter is started with a logical database name.

```
$ risqlrpt -d aroma
(C) Copyright 1991-1999 Informix Software, Inc.
All rights reserved.
RISQL Reporter Version 6.0.1(0)
Please type username: curly
Please type password: xxxxxxxx
RISQL>
```

See Also

“Using File Redirection” on page 2-19
“Creating Data Extraction Applications” on page 2-38
“CONNECT” on page 4-7
“SET ECHO” on page 4-52
Appendix B, “.rbretc Files”

.rbretrc Files

This appendix describes the RISQL Entry Tool and RISQL Reporter initialization files (*.rbretrc* files) and provides an example to illustrate the file format.

The appendix is divided into the following sections:

- Tool Initialization
- Sample Initialization File

Tool Initialization

On startup, the RISQL Entry Tool and RISQL Reporter read an initialization file named *.rbretrc*. Any valid SQL or RISQL statement or tool command can be executed from *.rbretrc*. Only one command is permitted per line. Settings made in the *.rbretrc* file during a tool session (for example, during a shell-escape sequence) are not applied until the tool is restarted.

There is no default *.rbretrc* file; database administrators and users must create initialization files with an operating-system editor.

An *.rbretrc* file can exist in one or both of the following locations:

- Global *.rbretrc* file, which is located in the directory that contains the Informix Red Brick Decision Server configuration file *rbw.config* and affects all sessions. This file is usually created by the database administrator and must have global read permission set so that all users can access it.
- Local *.rbretrc* file, which is located in a user's home directory and overrides the settings of the global file when both exist. This file is usually created by individual users.

If both global and local *.rbretrc* files exist, the tool reads the global file first and then reads the local file. Therefore, statements and commands in local *.rbretrc* files take precedence over statements and commands in the global *.rbretrc* file. Any statements or commands in a global *.rbretrc* file that are not present in a local *.rbretrc* file are retained.

Important: *On Windows platforms, the $HOME environment variable must be set to an existing directory that contains the* ***.rbretrc*** *file.*

Sample Initialization File

The following commands provide a customized user environment for either the RISQL Entry Tool or the RISQL Reporter; therefore, they are good candidates for a *.rbretrc* file:

```
set prompt 'Enter Statement Here>' ;
set editor 'ed' ;
set echo on ;
set output program 'more -c' ;
```

The following commands are suitable for a default report format when you use the RISQL Reporter for report writing:

```
set page length 66 ;
set output /home/user/curly/report.test ;
set column * underline '=' ;
set column dollars decimal ;
set column dollars space 2 '$' ;
set column if null 'N/A' ;
```

Locales

This appendix explains how to override the locale for Red Brick client tools and identifies the languages, territories, code pages, and collation sequences supported by the Red Brick products. The table that starts on page C-3 lists the *logical* combinations of these locale components; however, any combination can be used in a locale specification.

For more information on locales, see the *Administrator's Guide*.

Overriding the Locale

RISQL Entry Tool and RISQL Reporter users can override the locale by setting the RB_NLS_LOCALE environment variable.

For example, from the UNIX C shell prompt:

```
% setenv RB_NLS_LOCALE German_Austria.Latin1@Default
```

From the Windows MS-DOS shell:

```
C:\>set RB_NLS_LOCALE=German_Austria.Latin1@Default
```

where

German = language

Austria = territory

Latin1 = codepage

Default = collation sequence

It is not necessary to specify all four parts of a locale; however, Informix recommends that if the locale is not fully specified, the language should be one of the specified components. Otherwise, the unspecified components might default to incompatible values.

The following rules govern default values for unspecified locale components:

- If only the language is specified, omitted components are set to the default values for that language. For example, if the locale is set to

  ```
  Japanese
  ```

  the complete locale specification will be:

  ```
  Japanese_Japan.JapanEUC@Binary
  ```

  The table that starts on page C-3 lists default components for each language.

- If only territory is specified, the language defaults to English, the codepage to US-ASCII, and the sort to Binary. For example, if the locale is set to:

  ```
  _Japan
  ```

  the complete, but *impractical*, locale specification will be:

  ```
  English_Japan.US-ASCII@Binary
  ```

- Similarly, if only the codepage is specified, the language defaults to English, the territory defaults to UnitedStates, and the sort component defaults to Binary.
- Finally, if only the sort component is specified, the language defaults to English, the territory defaults to UnitedStates, and the codepage defaults to US-ASCII.

It is not necessary to specify all the separator characters (the underscore, the period, and the @ character) in a partial locale specification. Only the character that immediately precedes the component is required, such as the underscore character (_) in the previous territory example.

For more information about locales, refer to the *Administrator's Guide*.

Defined Locale Components

The following table identifies the languages, territories, code sets, and sort components of a locale and lists the *logical* combinations of these components; however, any combination can be used in a locale specification.

The values shown in boldface are the default values for the corresponding language when an incomplete locale is specified. For example, the default codepage for German is **Latin1**.

Language	Territory	Code Page (Character Set)	Sort
English	**UnitedStates** Australia Canada South Africa UnitedKingdom	**US-ASCII** Latin1 MS1252 UTF-8 IBM037 IBM285	**Default** Binary
German	Germany Austria German-Switzerland	**Latin1** MS1252 ISO-8859-9 UTF-8 IBM273	**Default** Binary

(1 of 6)

Language	Territory	Code Page (Character Set)	Sort
French	France French-Belgium French-Canada French-Switzerland	**Latin1** MS1252 ISO-8859-9 UTF-8 IBM297	**Default** Binary
Spanish	Spain Argentina Chile Mexico	**Latin1** MS1252 ISO-8859-9 UTF-8	**Spanish** TraditionalSpanish Binary
Italian	Italy Italian-Switzerland	**Latin1** MS1252 ISO-8859-9 UTF-8 IBM280	**Default** Binary
Portuguese	Portugal Brazil	**Latin1** MS1252 ISO-8859-9 UTF-8 IBM037	**Default** Binary
Norwegian	Norway	**Latin1** MS1252 ISO-8859-9 UTF-8	**Danish** Binary
Swedish	Sweden	**Latin1** MS1252 ISO-8859-9 UTF-8	**Swedish** Binary
Danish	Denmark	**Latin1** MS1252 ISO-8859-9 UTF-8	**Danish** Binary
Finnish	Finland	**Latin1** MS1252 ISO-8859-9 UTF-8	**Finnish** Binary

(2 of 6)

Language	Territory	Code Page (Character Set)	Sort
Japanese	Japan	**JapanEUC** MS932 UTF-8 IBM930	**Binary**
CanadianFrench	French-Canada	**Latin1** MS1252 ISO-8859-9 UTF-8 IBM297	**Default** Binary
TraditionalChinese	Taiwan	**EUC-TW** MS950 UTF-8 IBM937	Binary
SimplifiedChinese	China	**MS936** UTF-8 IBM935	Binary
Albanian	Albania	**ISO-8859-2** MS1250 Latin1 MS1252 ISO-8859-9 UTF-8	Default Binary
Arabic	SaudiArabia	**ISO-8859-6** MS1256 UTF-8	Default Binary
Bulgarian	Bulgaria	**ISO-8859-5** MS1251 UTF-8	Default Binary
Byelorussian	Belarus	**ISO-8859-5** MS1251 UTF-8	Default Binary
Catalan	Catalonia	**Latin1** MS1252 ISO-8859-9 UTF-8	Default Binary

(3 of 6)

Language	Territory	Code Page (Character Set)	Sort
Croatian	Croatia	**ISO-8859-2** MS1250 UTF-8	Croatian Binary
Czech	CzechRepublic	**ISO-8859-2** MS1250 UTF-8	Czech Binary
Dutch	Netherlands Dutch-Belgium	**Latin1** MS1252 ISO-8859-9 UTF-8 IBM037	Default Binary
Estonian	Estonia	**ISO-8859-4** ISO-8859-10 MS1257 UTF-8	Estonian Binary
Farsi	Iran	**ISO-8859-6** UTF-8	Default Binary
Greek	Greece	**ISO-8859-7** MS1253 UTF-8	Default Binary
Hebrew	Israel	**ISO-8859-8** MS1255 UTF-8	Default Binary
Hungarian	Hungary	**ISO-8859-2** MS1250 UTF-8	Hungarian Binary
Korean	Korea	**MS949** MS1361 UTF-8	Binary
Latvian	Latvia	**ISO-8859-4** ISO-8859-10 MS1257 UTF-8	Latvian Binary

(4 of 6)

Language	Territory	Code Page (Character Set)	Sort
Lithuanian	Lithuania	**ISO-8859-4** ISO-8859-10 MS1257 UTF-8	Lithuanian Binary
Macedonian	Macedonia	**ISO-8859-5** MS1251 UTF-8	Default Binary
Romanian	Romania	**ISO-8859-2** MS1250 UTF-8	Romanian Binary
Russian	Russia	**ISO-8859-5** MS1251 UTF-8	Default Binary
Serbian	Yugoslavia	**ISO-8859-2** MS1250 UTF-8	Default Binary
CyrillicSerbian	Yugoslavia	**ISO-8859-5** MS1251 UTF-8	Default Binary
Slovak	Slovakia	**ISO-8859-2** MS1250 UTF-8	Slovak Binary
Slovenian	Slovenia	**ISO-8859-2** MS1250 UTF-8	Slovenian Binary
Thai	Thailand	**MS874** UTF-8	Thai Binary

(5 of 6)

Language	Territory	Code Page (Character Set)	Sort
Turkish	Turkey	**ISO-8859-9** ISO-8859-3 MS1254 UTF-8	Turkish Binary
Ukrainian	Ukraine	**ISO-8859-5** MS1251 UTF-8	Ukrainian Binary
Vietnamese	VietNam	**MS1258** UTF-8	Vietnamese Binary

(6 of 6)

Notes on the Locale Table

- The component strings in this table must be spelled exactly as shown, but they are not case sensitive.
- In the Sort column, any value that is not *Binary* is a linguistic sort definition. *Default* refers to the sort definition specified by the CAN/CSA Z243.4.1 Canadian ordering standard, which covers English and several Western European languages.
- All codepages include US-ASCII as a subset, so any of the listed codepages can be used when the language is English; however, the codepages listed for each language are the most appropriate choices.
- Code page conversions can reliably be performed between any two codepages listed for a given language. Conversions outside the scope of each language row in the table are not supported. For example, characters can be converted from Latin1 to MS1252 but not from Latin1 to JapanEUC.
- The MS932 code page, listed for Japanese, is a superset of Shift-JIS, excluding Shift-JIS codes 0x8160 0x8161 0x817c 0x8191 0x8192 0x81ca. A Shift-JIS character set is also available.
- Unicode is not a supported codepage, nor are any shifted encoding schemes.

Index

A

access, database 2-36
administration, database 2-36
asterisks(*)
  column variable 4-16
  displayed in column 3-20, 4-28
authorization, databases 2-4, 2-37

C

cases, tracked by technical support Intro-12
character data, sort methods C-8
character set, conversions C-8
column titles
  changing 3-12
  defining 4-46
  underlining 3-13, 4-48
  width 3-16
columns
  changing width 3-16, 4-50
  hiding 3-18, 4-33
  left-fill 3-19
  setting report format 4-20 to 4-51
  specification precedence 3-5, 4-21
  widening space between 3-17, 4-43
  See also format settings, displaying *and* RESET COLUMN
commands, RISQL Entry Tool and RISQL Reporter
  command buffer 2-29
  SAVE syntax 4-19
  saving 3-4, 3-8
comment icons Intro-10
comments
  C-style 2-9, 2-32
  inserting 2-32
  SQL-style 2-9, 2-32
CONNECT command, SQL 4-7
contact information Intro-17
conventions
  syntax diagrams Intro-7
  syntax notation Intro-6
csh prompt 2-4
currency symbol 3-25, 4-45
Customer Support Center Intro-11

D

-d option 2-12, A-2
Data 2-6, A-2
data
  format, changing 3-20
  hiding 3-18, 4-33
  justifying 3-15, 4-31
  labeling 3-25
  left-fill 3-19, 4-24
  numeric format 3-20, 4-28
  numeric rounding 3-20
  suppressing duplicate 3-22, 4-35
  See also format settings, displaying *and* RESET COLUMN command
databases
  access to 2-4, 2-5
  administering 2-36
  authorizations for 2-4, 2-37
  changing information in 2-38
  connecting to 2-10 to 2-14, 4-7, A-3
  disconnecting from 2-13

db_password 2-6, A-2
db_username 2-6, A-2
DECIMAL numeric format, SET COLUMN command 3-20, 4-28
default editor 2-26, 4-54
default report format, *See* initialization file
default report format, setting 4-12, 4-20 to 4-51
defaults, setting 4-14, 4-15
demonstration database
  script to install Intro-4
dependencies, software Intro-4
documentation
  list for Red Brick Decision Server Intro-14
documentation, types of
  online manuals Intro-16
  printed manuals Intro-17
double-spaced reports 3-11, 4-41
dsn
  RB_DSN environment variable 2-14
  -s option precedence over other options and environment variables 2-12
  the -s option 2-6
duplicate data, suppressing 3-22

E

EDIT command, RISQL Entry Tool and RISQL Reporter 2-27, 4-9
editors
  changing default 2-26, 4-54
  ending a session 2-27
  starting 2-27, 4-9
environment variables
  RB_CONFIG 2-4
  RB_HOST A-5
  RB_PATH 2-10, 2-11, A-3
error messages
  discarding 2-25, 4-55
  identifying 2-10
  saving 2-25, 4-55
execution, interrupting 2-17
EXIT command, RISQL Entry Tool and RISQL Reporter 2-15, 4-13
exit status codes 2-40
EXPONENTIAL numeric format, SET COLUMN command 3-20, 4-28

F

feature icons Intro-10
features of this product, new Intro-5
file redirection, *See* redirection, file
file, saving output 2-19, 4-56
FILL option, SET COLUMN command 3-19, 4-24, 4-37, 4-39
filter, specifying 2-24, 4-56
FOLD LINE option, SET COLUMN command 3-10, 3-23, 4-26
FORMAT option, SET COLUMN command 3-20, 4-28
format settings
  displaying 3-27, 4-11
  saving 3-8, 4-19

G

GRANT command 2-36

H

headings, suppressing in output 4-62
hiding data 4-33
  See also format settings, displaying *and* RESET COLUMN

I

icons
  feature Intro-10
  platform Intro-10
  tip Intro-10
  warning Intro-10
IF NULL option, SET COLUMN command 3-27, 4-30
important paragraphs, icon for icons
  important Intro-10
INCOMPLETE STRING> prompt 2-30, 2-35, 4-60
initialization file 2-10, B-1
input
  including in output 2-21, 4-52
  reading 2-20
  valid 2-8
INTEGER numeric format, SET COLUMN command 3-20, 4-28
invoking tool, *See* starting tool

J

JUSTIFY option, SET COLUMN command 3-15, 4-31

K

keywords
  in syntax diagrams Intro-9
ksh prompt 2-4

L

layout, report 3-4
literals in SQL syntax 2-9
locales
  default C-2
  defining 2-5
  list of supported C-3
  overriding the warehouse locale C-2
logging in 2-7
logging out 2-15
logical database names 2-10

M

minus sign, placement 3-22
more output program 2-24
multibyte characters 4-5

N

new features of this product Intro-5
new-line character 2-9, 4-26
NO PRINT option, SET COLUMN command 3-18, 4-33
NO REPEAT option, SET COLUMN command 3-22, 4-35
notation conventions Intro-5
NULL, displaying 3-27, 4-30
numeric format 3-20, 4-28

O

object names 4-5
online manuals Intro-16
output
  appending 2-23, 4-56
  discarding 2-23, 4-56
  including input 2-21, 4-52
  restoring default 2-24
  saving 2-23, 4-56
output program, specifying 2-24, 4-56

P

page breaks, in reports 3-10
page layout
  default, for reports 3-10
  formatting reports 3-10, 4-58
passwords
  changing 2-36
  hiding 2-19
platform icons Intro-10
plus sign
  displaying 4-37
  placement 3-22
POSITIVE SIGN option, SET COLUMN command 4-37, 4-39
precedence, in column settings 3-5, 4-21
printed manuals Intro-17
PROGRAM option, SET OUTPUT command 4-56
prompts
  changing 2-35, 4-60
  INCOMPLETE STRING> 2-30

Q

-q option 2-8, 2-20, 2-22, A-2, A-4
queries
  canceling 2-17
  executing 2-15, 2-17, 4-18
QUERY COLUMN command, RISQL Reporter 3-27, 4-12
QUERY command, RISQL Entry Tool and RISQL Reporter 4-11
quiet mode
  and SET ECHO 2-22
  description 2-8, A-4
  in data extraction applications 2-39
  with redirection 2-20
QUIT command, RISQL Entry Tool and RISQL Reporter 2-15, 4-13
quoted strings in SQL syntax 2-9, 2-30

R

RB 2-6
.rbretrc file 2-10, B-1
RB_CONFIG environment variable 2-4, A-4
RB_DSN environment variable A-5
RB_HOST environment variable A-5
RB_PATH environment variable 2-10, 2-11, A-3
redirection, file
  overview 2-19
  system-level 2-19
  tool commands 2-19, 2-20
report format
  building templates 3-8
  column title 3-12, 4-46
  creating templates 4-19
  displaying settings 3-27, 4-11, 4-12
  displaying + sign 4-37
  laying out 3-4
  overview 3-4
  sample reports 3-29
  saving defaults 3-8, 4-19
  setting defaults 3-28
  settings 4-20 to 4-51
  sign placement 4-39
  skipping lines 3-11, 4-41
  wrapping lines 3-23, 4-26
report format, plus sign placement 4-37
report titles 3-9, 4-64
RESET COLUMN command, RISQL Reporter 3-28, 4-15
RESET command, RISQL Entry Tool and RISQL Reporter 4-14
risql and risqlrpt programs, in search path 2-7
RISQL Entry Tool
  description 1-4, 2-3
  displaying settings 4-11
  identifying version 2-7
  initialization file B-1
  prompt 2-35, 4-60
  restoring default settings 4-14
  starting 2-7
  suspending session 4-6
  valid input 2-8
RISQL Entry Tool commands
  CONNECT 2-13, 4-7
  EDIT 2-27, 4-9
  EXIT 2-15, 4-13
  QUERY 4-11
  QUIT 2-15, 4-13
  RESET 4-14
  RUN 2-20
  SET ECHO 2-21, 4-52
  SET EDITOR 2-26, 4-54
  SET ERROR OUTPUT 2-20, 2-25, 4-55
  SET OUTPUT 2-20, 2-23, 4-56
  SET PROMPT 2-35, 4-60
  SET STATS 2-17
  summary 4-3
  syntax 2-9
RISQL Reporter
  description 1-4, 2-3
  displaying format settings 4-12
  displaying settings 4-11
  identifying version 2-7
  initialization file B-1
  prompt 2-35, 4-60
  restoring default settings 4-14
  starting 2-6

supported features 1-4
suspending session 4-6
terminating session 4-13
valid input 2-8
RISQL Reporter commands
CONNECT 2-13, 4-7
EDIT 2-27, 4-9
EXIT 2-15, 4-13
QUERY 4-11
QUERY COLUMN 3-27, 4-12
QUIT 2-15, 4-13
RESET 4-14
RESET COLUMN 3-28, 4-15
RUN 2-20
SET COLUMN 4-20 to 4-51
SET ECHO 2-21, 4-52
SET EDITOR 2-26, 4-54
SET ERROR OUTPUT 2-20, 2-25, 4-55
SET OUTPUT 2-20, 2-23, 4-56
SET PAGE LENGTH 3-10, 4-58
SET PROMPT 2-35, 4-60
SET STATS 2-17
SET TITLE 3-9
summary 4-3
syntax 2-9
risqlrpt, command syntax 2-7, A-2
risql, command syntax 2-6, 2-7, A-2
RISQL> prompt 2-35
RUN command, RISQL Entry Tool and RISQL Reporter 2-20, 2-29, 4-18

S

sample reports 3-29
SAVE command, RISQL Entry Tool and RISQL Reporter 3-4, 3-8, 4-19
search path, modifying 2-7
semicolon, in SQL syntax 2-9
session, ending 2-15
SET COLUMN command, RISQL Reporter 4-20 to 4-51
FILL option 3-19, 4-24
FOLD LINE option 3-10, 3-23, 4-26
FORMAT option 3-20, 4-28
IF NULL option 3-27, 4-30
JUSTIFY option 3-15, 4-31
NO PRINT option 3-18, 4-33
NO REPEAT option 3-22, 4-35
POSITIVE SIGN option 4-37
SIGN option 4-39
SKIP LINE option 3-11, 4-41
SKIP PAGE option 3-10
SPACE option 3-17, 3-25, 4-43
TITLE option 3-12, 4-46
UNDERLINE option 3-13, 4-48
usage 3-4
WIDTH option 3-16, 4-50
SET commands, RISQL Entry Tool and RISQL Reporter
ECHO 2-21, 2-39, 4-52
EDITOR 2-26, 4-54
ERROR OUTPUT 2-20, 2-25, 4-55
OUTPUT 2-20, 2-23, 2-39, 4-56
PAGE LENGTH 3-10, 4-58
PROGRAM option, OUTPUT 4-56
PROMPT 2-35, 4-60
ROW DATA ONLY 2-39, 4-62
TITLE 3-9
SET STATS command 2-17
sh prompt 2-4
shell
escape command 2-33, 4-6
interactive session 2-34
prompt 2-4
suspending 2-33, 4-6
termination 2-9
shell,escaping to 2-33
SIGN option, SET COLUMN command 3-22, 4-37, 4-39
single-byte characters 4-5
SKIP LINE option, SET COLUMN command 3-11, 4-41
SKIP PAGE option, SET COLUMN command 3-10
software dependencies Intro-4
SPACE option, SET COLUMN command 3-17, 3-25, 4-43
starting tool
See initialization file
statistics
reporting 2-17
turning off 2-18
status codes, exiting 2-40
support
technical Intro-11
suspending
shell 2-33, 4-6
tool session 4-6
syntax diagrams
conventions for Intro-7
keywords in Intro-9
variables in Intro-9
syntax notation Intro-5, 2-9
system commands operation, running 2-19, 2-33
system prompt 2-4
system requirements
database Intro-4
software Intro-4

T

technical support Intro-11
templates, report 3-8, B-3
termination, SQL statement 2-9
tip icons Intro-10
TITLE option, SET COLUMN command 3-12, 3-16, 4-46
titles, report 3-9, 4-64
tool session, ending 2-15
tool session, starting 2-15
tool session,ending 4-13
tool session,starting 4-13
troubleshooting
general problems Intro-13

U

UNDERLINE option, SET
COLUMN command 3-13, 4-48
UNIX prompt 2-4
user access 2-36
user name, hiding 2-19
users, types of Intro-3

V

variables, in syntax
diagrams Intro-9

W

warning icons Intro-10
widening space between
columns 4-43
WIDTH option, SET COLUMN
command 3-16, 4-50
Windows NT prompt 2-4

Symbols

!, shell escape command 2-33, 4-6
$DATE 3-9, 4-64
$PAGE 3-9, 4-64
$TIME 3-9, 4-64
*, column variable wildcard 4-16
*, displayed in column 3-20, 4-28
+, displaying 4-37
+, placement 3-22, 4-39
–, placement 3-22, 4-39
.rbretrc file 2-10
;, in SQL syntax 2-9
>, <, file redirection symbols 2-19
\n, new-line character 2-9, 4-26
|, file redirection symbol 2-19