IBM Cognos Application Development Tools
IBM Cognos PowerHouse 4GL

Version 8.4G

QUIZ Reference

Information Management software

Product Information

This document applies to IBM Cognos PowerHouse 4GL 8.4G and may also apply to subsequent releases. To check for newer versions of this document, visit the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp).

Copyright

Table of Contents

About this Document

Overview

This document is intended for experienced IBM® Cognos® PowerHouse® 4GL users who require a concise summary of QUIZ statements.

Chapter 1, "Introducing QUIZ", introduces QUIZ and the other PowerHouse 4GL components and utilities.

Chapter 2, "Processing Phases of QUIZ", provides information about the sequence of events in the phases of QUIZ processing.

Chapter 3, "QUIZ Statements", provides concise summaries and detailed information about QUIZ statements. Syntax summaries, detailed syntax discussions, and examples are provided for each QUIZ statement, where applicable.

Conventions Used in this Document

This document is for use with the OpenVMS, UNIX®, Linux®, and Microsoft® Windows® operating systems. Any differences in procedures, commands, or examples are clearly labeled. Unless otherwise indicated, references to UNIX also apply to Linux.

In this document, words shown in uppercase type are keywords (for example, SAVE). Words shown in lowercase type are general terms that describe what you should enter (for example, filespec). When you enter code in PowerHouse 4GL components, however, you may use uppercase, lowercase, or mixed case type.

Getting Help

For more information about using this product or for technical assistance, go to http://www.ibm.com/support. Under **Choose support type**, select **Information management,** then under **Choose a product**, select **Cognos Application Development Tools**. Under **Cognos Application Development Tools support**, click **Documentation**.

IBM Cognos PowerHouse 4GL Documentation Set

PowerHouse 4GL documentation, available on the IBM Cognos PowerHouse 4GL Books CD, includes planning and configuration advice, detailed information about statements and procedures, installation instructions, and last minute product information.

Objective	Document
Install PowerHouse 4GL	The *IBM Cognos PowerHouse 4GL Getting Started* document provides step-by-step instructions on installing PowerHouse 4GL.
Review changes and new features	The *IBM Cognos PowerHouse 4GL Release Notes* document provides information on supported environments, changes, and new features for the current version.

Objective	Document
Get an introduction to PowerHouse 4GL	The *IBM Cognos PowerHouse 4GL Primer* document provides an overview of the PowerHouse language and a hands-on demonstration of how to use PowerHouse.
Get detailed reference information for PowerHouse 4GL	The IBM Cognos PowerHouse 4GL Reference documents provide detailed information about the PowerHouse language and each PowerHouse component. The documents are • *IBM Cognos PowerHouse 4GL PowerHouse Rules* • *IBM Cognos PowerHouse 4GL PDL and Utilities Reference* • *IBM Cognos PowerHouse 4GL PHD Reference (OpenVMS)* • *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* • *IBM Cognos PowerHouse 4GL QDESIGN Reference* • *IBM Cognos PowerHouse 4GL QUIZ Reference* • *IBM Cognos PowerHouse 4GL QTP Reference*

IBM Cognos PowerHouse Web Documentation Set

PowerHouse Web documentation, available from the IBM Cognos PowerHouse Web Administrator CD, includes planning and configuration advice, detailed information about statements and procedures, installation instructions, and last minute product information.

Objective	Document
Start using PowerHouse Web	The *IBM Cognos PowerHouse Web Planning and Configuration* document introduces PowerHouse Web, provides planning information and explains how to configure the PowerHouse Web components. **Important:** This document should be the starting point for all PowerHouse Web users.
Install PowerHouse Web	The *IBM Cognos PowerHouse Web Getting Started* document provides step-by-step instructions on installing PowerHouse Web.
Review changes and new features	The *IBM Cognos PowerHouse Web Release Notes* document provides information on supported environments, changes, and new features for the current version.
Get detailed information for developing PowerHouse Web applications	The *IBM Cognos PowerHouse Web Developer's Guide* document provides detailed reference material for application developers.
Administer PowerHouse Web	The *PowerHouse Web Administrator Online Help*, available from within the PowerHouse Web Administrator, provides detailed reference material to help you during PowerHouse Web configuration.

IBM Cognos Axiant 4GL Documentation Set

Axiant 4GL documentation, available from the IBM Cognos Axiant® 4GL CD, includes planning and configuration advice, detailed information about statements and procedures, installation instructions, and last minute product information.

Objective	Document
Install Axiant 4GL	The *IBM Cognos Axiant 4GL Web Getting Started* document provides step-by-step instructions on installing Axiant 4GL.
Review changes and new features	The *IBM Cognos Axiant 4GL Release Notes* document provides information on supported environments, changes, and new features for the current version.
Get an introduction to Axiant 4GL	The *A Guided Tour of Axiant 4GL* document contains hands-on tutorials that introduce the Axiant 4GL migration process and screen customization.
Get detailed reference information on Axiant 4GL	The *Axiant 4GL Online Help*, available from within Axiant 4GL, provides a detailed reference guide to Axiant 4GL.

For More Information

For information on the supported environments for your specific platform, as well as last-minute product information or corrections to the documentation, refer to the *IBM Cognos PowerHouse 4GL Release Notes*.

Chapter 1: Introducing QUIZ

Overview

This chapter introduces QUIZ, the PowerHouse 4GL report writer. It also provides overview information about the other PowerHouse 4GL components and utilities.

About IBM Cognos PowerHouse 4GL

IBM CognosPowerHouse 4GL is an application development environment that allows you to create business applications quickly and easily.

Components

PowerHouse 4GL is divided into the following separate, yet integrated components:

PowerHouse Dictionary

The PowerHouse dictionary is the foundation of PowerHouse 4GL applications. As the backbone of all PowerHouse 4GL systems, the PowerHouse dictionary stores definitions of the data used by your PowerHouse 4GL applications.

There are two dictionary types—PDC and PHD. PDC dictionaries exist as a single file with a .pdc extension (**OpenVMS, UNIX, Windows**). PHD dictionaries exist as five indexed files and have a .phd extension. PHD dictionaries are OpenVMS-specific.

For more information about the PHD dictionary, see the *IBM Cognos PowerHouse 4GL PHD Reference* and the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document. See also the section, "PowerHouse Dictionary on OpenVMS", in Chapter 1, "Introducing the PowerHouse Dictionary", in the *IBM Cognos PowerHouse 4GL PDL and Utilities Reference*.

PDL

The PowerHouse Definition Language (PDL) allows you to create and maintain a PowerHouse dictionary.

PDL source code can be compiled in either the PDL or PHDPDL (**OpenVMS**) compiler.

PDL Compiler

PDL compiler is the component that compiles PDL source statements to a PowerHouse dictionary. Dictionaries generated with the PDL compiler have a .pdc extension (**OpenVMS, UNIX, Windows**).

PHDPDL Compiler (OpenVMS)

PHDPDL is an OpenVMS-specific component that compiles PDL source statements to a PowerHouse dictionary. Dictionaries generated with PHDPDL have a .phd extension.

PHD Screen System (OpenVMS)

PHD is a screen interface to PHD dictionaries. You can initiate PHD with the POWERHOUSE or POW command.

For more information about running PHD, see Chapter 1, "Running PowerHouse 4GL", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

QDESIGN and QUICK

QUICK is an interactive screen processor with a powerful development tool: QDESIGN. As a screen designer, you use QDESIGN to build data entry and retrieval screen systems. QUICK screens are used by data-entry operators and other end-users to process data quickly or to browse effortlessly through their files.

QUICK includes an interactive debugger that lets you analyze and control QUICK screens as they run.

QUIZ

QUIZ is the PowerHouse 4GL report writer. It takes the information you request and gives it a structure. Your information is automatically displayed in columns with headings. The key to the simplicity of QUIZ lies in its relationship with the data dictionary. QUIZ references the rules and standards defined in the data dictionary by the application designer when it formats your report.

QTP

QTP is a high-volume transaction processor. It gives you the power to change the data in your files in one sweep. Because QTP is easy to use and designed for fast, high-volume file updating, it should be used by someone who is familiar with the implications of updating active files.

QTP includes a trace facility that lets you debug QTP requests.

Utilities

PowerHouse 4GL also contains the following data dictionary utilities:

QSHOW

QSHOW is the data dictionary reporting program. It allows you to view and obtain cross-reference information about the contents of your PowerHouse dictionaries. It also allows you to generate PDL source for a PowerHouse dictionary.

QUTIL

QUTIL is a utility that creates and deletes non-relational files and databases.

ETOP (UNIX, Windows)

ETOP is an Eloquence to PDL conversion utility that generates PDL statements directly from an existing Eloquence database.

PHDMAINTENANCE (OpenVMS)

PHDMAINTENANCE creates and manages PHD dictionaries. It is also referred to as PHDMAINT.

PHDADMIN (OpenVMS)

PHDADMIN is a run-time utility for administering security classes in PHD dictionaries.

Chapter 2: Processing Phases of QUIZ

Overview

This chapter discusses the processing phases of QUIZ. It includes information about the Parser and Reporter processes in QUIZ.

About the Processing Phases

Understanding the phases of QUIZ processing can help you determine the source of problems that QUIZ encounters during processing.

QUIZ is logically divided into two distinct processes:

Process	Description
Parser	Interprets QUIZ statements and builds an internal description of a report.
Reporter	Produces the report as specified by the parser.

The Parser

The parser process builds an internal definition of a report based on the statements that you enter. The entered statements control all actions performed by the parser. For example, the ACCESS, REPORT, SELECT, and GO statements initiate the following actions in the parser process:

Statement	Instructs the parser to ...
ACCESS	Construct tables that describe how records are to be linked.
REPORT	Build a report-group description.
SELECT	Store the logical expression to be used for record selection.
GO	Initiate the reporter process.

The Reporter

The reporter process produces a report as specified by the information collected by the parser. Once the reporter produces the specified report, control returns to the parser.

The following steps outline how the reporter works:

1. Finalize all internal tables.

   First, QUIZ completes certain tables that the parser can't complete (either for efficiency's sake or because the information isn't available). During this step, the reporter finalizes all report-groups. A TAB value (and SKIP, if necessary) is assigned to each report item, and the default PAGE HEADING report-group is built (if required). QUIZ opens all files during this step (unless they were already opened).
2. Perform SORT (if required).

If a SORT statement is specified, QUIZ builds all the record complexes at this point, and passes them to the sort utility provided with the operating system. In building record complexes, QUIZ applies the selection criteria specified by the SELECT statement, and then performs the calculations specified by the DEFINE statements. Only the items needed for sorting and reporting are actually sorted. For more information about record complexes, see the ACCESS statement on (p. 16), the CHOOSE statement on (p. 35), the EDIT statement on (p. 64), and the SELECT statement on (p. 113).

3. Retrieve the first record complex.

   If a sort was performed in Step 2, QUIZ retrieves the record complex from the sort output. If a sort was not performed, the record complex is built from the actual data records of the input record-structures, and DEFINE calculations are performed.

4. Initialize break-level to highest level.

   QUIZ sets the break-level, an internal counter, at the highest possible control break-level to ensure that control headings are produced before reporting the first record complex. Each item named in a SORT or SORTED statement is assigned a number based on its position in the statement. The number that's assigned is the control break-level for that item.

5. Report INITIAL HEADING report-group (if required).

   If there is an INITIAL HEADING statement, QUIZ processes the associated report-group.

6. Report record complexes.

   QUIZ repeats the following for each record complex until it encounters end-of-data:

Produce control headings	QUIZ processes all control heading report-groups for reporting, starting at break-level and working down.
Produce detail lines	QUIZ processes the report-group associated with the REPORT statement.
Retrieve next record complex	As in Step 3, if a sort was performed, QUIZ retrieves the record complex from the sort output; otherwise, the record complex is built from the input files specified in the ACCESS statement. If no sort was performed, QUIZ performs record selection and DEFINE calculations.
Calculate the current break-level	As described in Step 4, QUIZ assigns each level of the sort a level number. QUIZ compares the current record complex (just read) to the previous record complex to determine the current level. On an end-of-data condition, QUIZ sets the current break-level to the highest level.
Produce control footings (if required)	QUIZ processes the control footing report-groups associated with control break-levels from the lowest level to break-level. For example, on the last pass (when end-of-data is reached), all footing report-groups are printed, starting at the lowest control break-level and working up.

7. Report FINAL FOOTING report-group (if required).

   QUIZ processes report-groups that are associated with a FINAL FOOTING statement (if any).

   Note that if a report-group cannot fit on the current page, QUIZ, by default, skips to a new page. For example, a five-line report-group will not start printing if there are only four lines left on the page. Skipping also occurs if there is a SKIP PAGE statement in the report-group.

8. Return control to the parser.

Chapter 3: QUIZ Statements

Overview

This chapter describes each QUIZ statement in detail. For each statement, you'll find

- detailed syntax descriptions
- detailed statement discussions
- examples

Summary of QUIZ Statements

Statement	Purpose
ACCESS	Specifies the input record-structures and their logical relationships in a report.
AND SELECT	Applies selection conditions to record-structures in the ACCESS statement when building a record complex. See the SELECT statement on (p. 113).
BUILD	Compiles and saves a QUIZ report.
CANCEL	Cancels the current QUIZ report specifications.
CHOOSE	Extracts data from an indexed file, relational table or view, or SQL cursor by item value.
[SQL] DECLARE CURSOR (query-specification)	Defines a set of data as a run-time view.
[SQL] DECLARE CURSOR (stored procedure)	Calls a stored procedure in an IBM® DB2®, ODBC, or Sybase database or a stored function in Oracle.
DEFINE	Assigns a name to an expression or prompts for values at execution-time.
DISPLAY	Displays a message.
EDIT	Validates entries made in response to a PARM prompt.
EXECUTE	Executes a compiled QUIZ report.
EXIT	Ends a QUIZ session.
FINAL FOOTING	Sets the content and format of the footing at the end of the report.
FOOTING AT	Sets the content and format of control break footings.
GO	Initiates execution of a QUIZ report.
HEADING AT	Sets the content and format of control break headings.

Statement	Purpose
INITIAL HEADING	Sets the content and format of the heading at the beginning of the report.
NOREPORT	Specifies what to print if no record complexes are selected.
PAGE FOOTING	Sets the content and format of a footing at the end of the page.
PAGE HEADING	Sets the content and format of a heading at the beginning of the page.
QSHOW	Runs QSHOW from QUIZ.
query-specification (SELECT)	Defines a collection of rows that are accessible when the cursor is opened.
QUIT	Ends a QUIZ session.
REPORT	Sets the content and format of report detail lines.
report-group[1]	Determines the content and format of detail lines, headings, and footings.
report-item[1]	Specifies what's to be reported, its position and format, and whether it's to be printed on every line or only at a control break.
REVISE	Edits the current temporary save file or a specified file.
SAVE	Saves the current QUIZ source statements in a file.
SELECT	Applies selection conditions to record-structures in the ACCESS statement when building a record complex.
SET	Overrides default QUIZ settings.
SET PAGE	Changes the default page-control settings.
SET REPORT	Changes the default report-control settings.
SET SUBFILE	Directs output to a subfile and changes the default subfile settings.
SHOW	Displays all record-structures or items for the report.
SORT	Sorts record complexes in a desired sequence and defines control breaks.
SORTED	Defines control breaks for records known to be in sorted order.
USE	Processes QUIZ source statements that are contained in a file.

[1]*Report-group and report-item aren't statements; they are constructs used in a statement.*

QUIZ Run Time Syntax

Statement	Options
DISPLAY	

Statement	Options
EXECUTE	GO|NOGO
EXIT	
GO	OMNIREUSE
QUIT	
SAVE	CLEAR
SET	CLOSE|NOCLOSE FORMFEED|NOFORMFEED DEVICE CONTROL JOB|NOJOB PAGE IMAGES|LENGTH|NUMBER|WIDTH REPORT COPIES REPORT DEVICE REPORT FORMS REPORT LIMIT REPORT NAME REPORT PRIORITY REPORT SPACING STACKSIZE WAIT|NOWAIT
SHOW	ACTIVITY|COMMANDS|STATUS
USE	DETAIL|NODETAIL LIST|NOLIST

ACCESS

Specifies the input record-structures and their logical relationships in a report.

Syntax

ACCESS primary-data-structure [ALIAS name]
[LINK {direct-linkage|indexed-linkage|TO cursor
[sql-substitution...]} [ALIAS name] [OPTIONAL]
[{AND|LINK}
{direct-linkage|indexed-linkage|TO cursor
[sql-substitution...]} [ALIAS name] [OPTIONAL]...]

primary-data-structure

Names the primary data structure which can be

- a cursor defined in a DECLARE CURSOR statement
- a record-structure named in the dictionary
- a subfile
- a table or view defined in a relational database

cursor [sql-substitution...]

A cursor is the name of a set of data defined by the PowerHouse 4GL SQL DECLARE CURSOR statement.

An sql-substitution can be specified for any substitution variable defined on the DECLARE CURSOR statement. Two default sql-substitutions, WHERE and ORDERBY, will be inserted in generated SQL statements even if the corresponding substitution-variables do not exist on a DECLARE CURSOR statement.

The syntax for an sql-substitution is:

substitution-variable (**text**)

For more information about substitutions, see Chapter 1, "About PowerHouse 4GL and Relational Databases", in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

Limit: Any sql-substitutions must appear before any other options.

record-structure [IN file]

Names a record-structure and, optionally, the name of the file to which it belongs. Both must be declared in the data dictionary. A file name adds clarity if the file name differs from the record-structure name, as in coded record-structures.

*subfilespec

Specifies the name of an existing subfile, prefixed by an asterisk (*).

Subfilespec may be a logical name (**OpenVMS**) or an environment variable name (**UNIX, Windows**).

For more information about subfiles, see (p. 142).

table [IN database]

Names the table or view in a relational database and, optionally, the name of the database to which it belongs. The database must be attached to the current data dictionary.

If an IN database qualifier is not used, QUIZ first assumes that the primary record is a record-structure defined in the data dictionary, or a cursor defined in a DECLARE CURSOR statement.

If the record-structure exists, QUIZ uses the first file containing the record-structure. If the first assumption fails and you are running QUIZ using the SEARCH option of the **subdictionary** program parameter, QUIZ searches every attached relational database for tables or views with the specified name. If there is one table or view with the specified name, QUIZ uses it. If there is more than one, or there is no table or view with the name, QUIZ issues an error message.

For more information about the **subdictionary** program parameter, see Chapter 2, "Program Parameters", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

ALIAS name

Assigns an alternative name to a data structure. Once the alias is assigned, subsequent references to the data structure must use this name.

LINK

Specifies that the linkage to the subordinate data structure is hierarchical. The first related data structure must be linked hierarchically. If more than one data record exists for the subordinate data structure, a record complex is generated for each data record in the subordinate data structure. The data for higher-level record-structures is repeated in each of the record complexes.

direct-linkage

Specifies the linkage to a particular data record of the subordinate data structure, which can be

- a record-structure named in the dictionary
- a subfile

Direct-linkage has the form:

TO RECORD numeric-expression OF {record [IN file]|*subfilespec}

numeric-expression

A numeric-expression that produces a value that corresponds to a record number in the subordinate record-structure. The record number isn't verified until execution time.

Limits:

UNIX, Windows:	Must be a valid record number greater than or equal to 0. The first record is record number 0.
OpenVMS:	Must be a valid record number greater than or equal to 1. The first record is record number 1.

OF {record [IN file]|*subfilespec}

Direct-linkage is valid only for record-structures in direct files, relative files, or subfiles. A subfilespec must be the qualified or unqualified name of an existing subfile or a portable subfile dictionary, and it must be prefixed by an asterisk.

indexed-linkage

Specifies the linkage via an item or column, or items or columns either to

- the subordinate record-structure for hierarchical linkage, or
- the related record-structure for parallel linkage

Indexed-linkage in QUIZ has the form:

[item [,item]...]|[(expression[,expression]...)]
[VIAINDEX indexname]
TO [linkitem[,linkitem]... OF] {record [IN file]|*subfilespec|table
[IN database]}

item[,item]...

Specifies an item or items. QUIZ uses the value of the item to find the associated data record in the linked record structure. Items in the item list must exist in record structures that have previously been declared in the ACCESS statement list.

(expression[,expression]...)

Specifies an expression. QUIZ uses the result of the expression to find associated data records in the linked record-structure. Each expression must result in the value for a single item. The item must exist in a record-structure that has been declared in an ACCESS statement list.

Expressions must be enclosed in parentheses.

Limit: Up to 255 expressions can be specified.

Limit: The expression must be a simple expression and cannot be a case or conditional expression.

VIAINDEX indexname

Specifies either

- the name of an index of the subordinate or related record-structure for an indexed file
- the name of an index that has been defined for the subordinate or related relational table or view

Using indexname forces data records to be retrieved in index order; otherwise, the order is determined by the file system.

UNIX, Windows: VIAINDEX is ignored for Eloquence, except when the index specified is an TPI or a B-Tree index.

linkitem[,linkitem]... OF

If the subordinate or related record-structure is in an indexed or Eloquence (**UNIX, Windows**) file, the first linkitem must be the first segment of an index in that record-structure.

If the first linkitem is also the initial segment of a multiple-segment index, then subsequent linkitems can be used to specify additional segments of the multiple-segment index. The number of linkitems must be less than or equal to the number of segments in the index.

If the subordinate or related record-structure refers to a table or view, any item (column) in the table or view can be a linkitem.

Limit: There can be up to 255 linkitems. PowerHouse 4GL (**UNIX, Windows**) does not support linkage via an initial subset of the segments of a multi-segment index in Eloquence unless the index is a B-Tree or TPI index. All segments of the index must be specified.

If you specify both the VIAINDEX option and linkitems, the linkitems must match consecutive initial segments of the index. There can be additional segments in the index that aren't used in the linkage; in this case, linkage is by means of only the specified segments instead of the entire index, but retrieval is in index order.

The item list or expression list specifies which items and/or columns from previous record-structures in the ACCESS statement list will be used in the link. The linkitem list or the indexname specifies which items or columns in the subordinate or related record-structure will be used in the link. If you explicitly specify both sides of the link, the two sides must match in the number of values, and in the type and size of the values.

TO cursor-name [sql-substitution...]

[**AND TO cursor-name** [**sql-substitution...**]

When the linkage refers to a cursor, only the "LINK TO" or "AND TO" syntax is allowed. Linkage criteria can be specified within the cursor declaration, or by using sql-substitutions.

The syntax for an sql-substitution is:

substitution-variable (**text**)

For more information about sql-substitutions and substitution-variables, see Chapter 1, "About PowerHouse 4GL and Relational Databases", in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

AND

Specifies that the record-structures on either side of the AND keyword are in a parallel relationship and are on the same level in the linkage hierarchy.

When a data record from one record-structure is read, a data record from the other record-structure is also read. If one record-structure has fewer data records than the other, a dummy record for that record-structure is included in the record complex. The content of this dummy record depends on whether the record is from a relational database table and whether null value support is enabled in the PowerHouse dictionary. If the record is from a relational database table and null value support is enabled, then the data record is initialized to null values. If null value support is disabled or the record is not from a relational database table, then the data record is initialized to zeroes, spaces, and dictionary initial values. Regardless, the linkitem is initialized to the value used for linkage.

If the linkage to one record-structure is by unique index or record number, the data record, if it exists, is assumed to exist for all related data records in the parallel record-structure and the data is repeated in subsequent record complexes.

OPTIONAL

Specifies that the construction of the record complex continues even if a related data record isn't found in the subordinate record-structure. A dummy record-structure is added to the record complex. The content of this dummy record depends on whether the record is from a relational database table and whether null value support is enabled in the PowerHouse dictionary. If the record is from a relational database table and null value support is enabled, then the data record is initialized to null values. If null value support is disabled or the record is not from a relational database table, then the data record is initialized to zeroes, spaces, and dictionary initial values. Regardless, the linkitem is initialized to the value used for linkage.

For parallel relationships, specify OPTIONAL for the record-structure you want if construction of the record complex is to continue when there are no related data records in any of the parallel record-structures. When any record-structure in a parallel linkage is assigned as OPTIONAL, all parallel record-structures in the related group are treated as optional.

Limit: Cannot be used with the primary record-structure in the ACCESS statement.

Discussion

Each QUIZ report requires an ACCESS statement. The ACCESS statement declares

- the record-structures that are read
- the order in which the record-structures are read
- optionally, how the linkages between record-structures are constructed

The ACCESS statement causes QUIZ to open the files that contain the record-structures named in the ACCESS statement.

All files opened by a previous ACCESS statement in the current QUIZ session are closed when another ACCESS statement is encountered.

Limit: You can access a maximum of 31 record-structures, including subfiles. There can be a maximum of 1023 items per record-structure.

Limit: QUIZ has a 32,767 byte buffer limit that includes all record-structures in the ACCESS statement, all defined items, and space for copies of values used in linkage.

This list describes terms used in the rest of this discussion:

Term	Definition
compound record	Consists of a data record from the primary record-structure and one data record from each of the related "linked to" record-structures.

Term	Definition
data record	A single occurrence of a record in a record- structure. QUIZ retrieves one or more data records from the subordinate record-structure for each data record in the primary record-structure.
dummy record	A record with zeros, spaces, and dictionary initial values.
initial subset	Either the first segment alone or the first segment followed by one or more consecutive segments.
linkage	Describes the different types of relationships that can occur between two or more record-structures.
primary record-structure	The first record-structure named in the ACCESS statement.
record complex	Consists of a data record from the primary record structure and one data record from each of the related "linked to" record-structures and all defined items. QUIZ builds a series of record complexes for each data record in the primary record-structure.
record-structure	A collection of elements that relate to a particular activity. A record-structure describes the structure of the data records that make up a file. A file can contain one or more record-structures.

Viewing Available Record-structures and Items

To display the record-structures that are available at execution-time during retrieval, enter

```
> SHOW FILES
```

To display the items that are available in the accessed record-structures at execution-time during retrieval, enter

```
> SHOW ITEMS
```

Default Linkage

If you don't specify both sides of the linkage, QUIZ can usually construct a default linkage. The following table shows how QUIZ attempts to construct a default linkage:

"From" side of link	"To" side of link	What QUIZ does
item(s), expression(s)	---	This combination isn't allowed if the subordinate or related record-structure has more than one index. If it has one index, QUIZ attempts to match that index's segments, in order, to the item(s) and expression(s) specified. There must be at least as many segments as items and expressions. If there are more segments than items and expressions, QUIZ ignores the extra segments.
---	linkitem(s)	QUIZ attempts to match the linkitems to identically named items in any record-structure that has previously been declared in the ACCESS statement list. Usually QUIZ searches the record-structures in the order that they were declared. However, if parallel linkage has already been established between a parallel driver record-structure and one or more related record-structures, then QUIZ looks in the parallel driver record-structure first.
---	indexname	QUIZ uses all the segments of the index as linkitems and attempts to construct a linkage using the procedure described above for linkitems. QUIZ uses the index to determine the order of data record retrieval.

"From" side of link	"To" side of link	What QUIZ does
---	indexname and linkitem(s)	QUIZ uses the specified linkitems (which aren't necessarily all the items in the index, as long as they are consecutive initial segments of the index) and attempts to construct a linkage as described above for linkitems. QUIZ uses the index to determine the order of data record retrieval.
---	---	There must be at least one index defined for the subordinate or related record-structure. For hierarchical linkage, QUIZ examines each index in the subordinate or related record-structure, in the order that the indexes were defined (the same order in which QSHOW reports them), to find one whose segments all match, by name, items in previous record-structures in the ACCESS statement list. The previous record-structures are checked, in the order in which they were declared. If this fails, QUIZ uses the first index whose first segment matches an item in a previous record-structure, and constructs the linkage from that first item, plus as many consecutive items as possible. The matching criteria are the same for parallel linkage as for hierarchical linkage, but QUIZ examines the indexes of the related record-structure in a more particular order. If a linkage has already been established between the parallel driver record-structure and one or more related record-structures in parallel, QUIZ tries to use the same item from the parallel driver record-structure again to match an index in the new related record-structure. If this doesn't work, QUIZ tries to match an index in the new related record-structure to any item in the driver record-structure. If that also fails, QUIZ tries to establish linkage by means of the procedure described above for hierarchical linkage.

QUIZ links data records in indexed files by matching values for identically-named items in the record-structures. In the following ACCESS statement, default linkage is by the item EMPLOYEE which is an item in both the STOCKS and EMPLOYEES record-structures:

```
> ACCESS STOCKS LINK TO EMPLOYEES
```

The item EMPLOYEE in the record-structure EMPLOYEES (the record-structure being linked to) must be defined as a segment in an index. The item EMPLOYEE in the record-structure STOCKS (the record-structure before the keyword TO) does not have to be a segment in an index.

How Linkage Works

Each time you retrieve a primary data record in a linkage, you retrieve all related data records from the linked record-structures.

Each record-structure named in the ACCESS statement can be linked to one or more record-structures. Each pair of record-structures is linked by a specific item and value. You can use any item in the primary record-structure to link to another record-structure. However, in the record-structure being linked to, you must use a segment that's defined in an index unless you are specifying linkage using record numbers.

Types of Relationships

When linking any two record-structures, there are two types of data relationships:

- one-to-one
- one-to-many

One-to-One Relationships

A one-to-one relationship between two record-structures occurs when one or more linkitems from a primary record-structure is in a unique index in another. QUIZ only treats an index as unique if all segments of the index are used in the linkage. If you only link to the first few segments of a unique index, QUIZ assumes a repeating link and therefore a one-to-many linkage.

For example, assume that an application contains a file called EMPLOYEES and another called POSITIONS, with the following record-structures:

EMPLOYEES

EMPLOYEE	LASTNAME	FIRSTNAME	**POSITIONCODE**	CITYBRANCH
1000	Abra	Margaret	**pgm III**	Toronto East

In the example below, the item POSITIONCODE in POSITIONS is defined as a segment in a unique index.

POSITIONS

POSITIONCODE	POSITIONTITLE	SALARY	CHARGEOUT
pgm III	Sr Programmer	4200.00	110.00

If you enter

```
> ACCESS EMPLOYEES LINK TO POSITIONS
```

you define a linkage that uses the item POSITIONCODE in EMPLOYEES to link to the segment POSITIONCODE in POSITIONS. Because POSITIONCODE is in a unique index in POSITIONS, there is only one matching record in POSITIONS for each record in EMPLOYEES.

If a segment is defined in an index of type UNIQUE in the data dictionary, QUIZ only retrieves one data record for each segment value.

One-to-Many Relationships

A one-to-many relationship between two record-structures occurs when one or more linkitems from one record-structure is in a repeating (non-unique) index of another.

For example, assume that an application contains a file called EMPLOYEES and another called BILLINGS, with the record-structures shown below.

Here, EMPLOYEE in EMPLOYEES is defined as a segment in a unique index:

EMPLOYEES

EMPLOYEE	LASTNAME	FIRSTNAME	POSITIONCODE	CITYBRANCH
1000	Abra	Margaret	pgm III	Toronto East

Here, EMPLOYEE in BILLINGS is defined as a segment in a repeating index:

BILLINGS

EMPLOYEE	PROJECT	HOURS
1000	5000	20.00
1000	5002	20.00

If you enter

```
> ACCESS EMPLOYEES LINK TO BILLINGS
```

you define a linkage that uses the item EMPLOYEE in EMPLOYEES to link to the segment EMPLOYEE in BILLINGS. Because EMPLOYEE is in a repeating index in BILLINGS, there may be more than one matching record in BILLINGS for each record in EMPLOYEES.

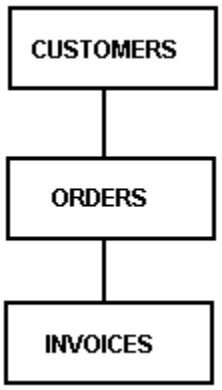
Specifying Linkage in a One-to-Many Relationship

In PowerHouse 4GL, there are two ways to specify linkage in a one-to-many relationship:

- hierarchical
- parallel

Hierarchical Linkage

A hierarchical linkage can be shown as a chain of record-structures:



In this example, CUSTOMERS can have one or many related ORDERS and each ORDER can have one or more related INVOICES. The relationship between CUSTOMERS and INVOICES exists only through ORDERS.

Defining a Hierarchical Linkage

You can define a hierarchical linkage with the LINK TO option. When you define this type of linkage, each record-structure is related to a record-structure that precedes it in the ACCESS statement.

For example:

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERS &
>   LINK TO INVOICES
```

defines a linkage between CUSTOMERS and ORDERS, and between ORDERS and INVOICES.

In this example, the relationship between CUSTOMERS and INVOICES only exists through ORDERS.

CUSTOMERS

ACCOUNTNUMBER	CUSTOMER
1010	Kane Contractors

ACCOUNTNUMBER is a segment in a repeating index in ORDERS.

ORDERS

ORDERNUMBER	ACCOUNTNUMBER	PARTNUMBER
5001	1010	1001
5002	1010	1002

ORDERNUMBER is a segment in a repeating index in INVOICES.

INVOICES

INVOICENUMBER	ORDERNUMBER
2001	5001
2002	5001
2003	5002

How Hierarchical Linkage Works

When you execute the report, QUIZ does the following:

1. Reads the first record from CUSTOMERS.
2. Reads the first record from ORDERS that has a value for ACCOUNTNUMBER equal to the value of ACCOUNTNUMBER in the CUSTOMERS record.
3. Reads the first record from INVOICES that has a value for ORDERNUMBER equal to the value of ORDERNUMBER in the ORDERS record.
4. Reports the record complex.

After reporting the record complex, QUIZ does the following:

1. Reads the next data record from INVOICES when the value for ORDERNUMBER equals the value of ORDERNUMBER in ORDERS.
2. Reports the new record complex.
3. Continues to report all records from INVOICES when the value for ORDERNUMBER is equal to that of the ORDERS record.

When no more INVOICES records exist for the current ORDERS record, QUIZ does the following:

1. Reads the next ORDERS record when the value for ACCOUNTNUMBER is equal to that of the CUSTOMERS record.
2. Reads a new INVOICES record with an ORDERNUMBER equal to that of the ORDERS record.
3. Reports the record complex.

4. Continues to report all records from INVOICES when the value for ORDERNUMBER is equal to that of the ORDERS record.

When no more ORDERS records exist for the current CUSTOMERS record, QUIZ does the following:

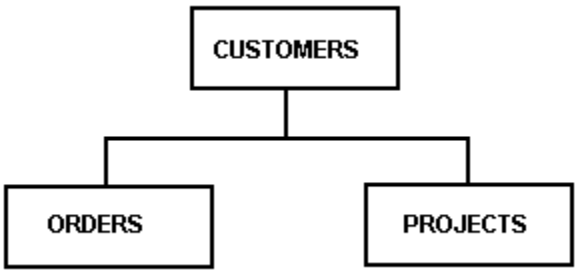
1. Reads the next CUSTOMERS record.
2. Repeats the whole process.

Parallel Linkage

The first record-structure in a parallel linkage is linked using a standard hierarchical linkage, as in

```
> ACCESS CUSTOMERS LINK TO ORDERS
```

A parallel linkage is a branch:



Defining Parallel Linkage

A parallel relationship defines a linkage branch in which more than one record-structure is linked to a common record-structure. Record-structures that are joined by the AND TO option aren't directly related to one another. The record-structures are instead directly related to a common record-structure. A single item in the common record-structure is used to link related data records in the other two record-structures.

Define a parallel linkage with the AND TO option. When you define this type of linkage, more than one record-structure is linked to a common record-structure to form a branch. The relationships that exist aren't directly related to one another.

For example:

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERS &
>   AND TO PROJECTS
```

This defines a linkage between CUSTOMERS and ORDERS and between CUSTOMERS and PROJECTS.

CUSTOMERS

ACCOUNTNUMBER	CUSTOMER
1010	Kane Contractors

ORDERS

ORDERNUMBER	**ACCOUNTNUMBER**	PARTNUMBER
5001	**1010**	1001
5002	**1010**	1002

PROJECTS

PROJECT	CUSTOMER	TITLE	ACCOUNTNUMBER
6001	Kane	MIS	**1010**
6002	Kane	Support	**1010**

How Parallel Linkage Works

When you execute the report, QUIZ does the following:

1. Reads the first data record from CUSTOMERS.
2. Reads the first data record from ORDERS that has a value for ACCOUNTNUMBER equal to the value of ACCOUNTNUMBER in CUSTOMERS.
3. Reads the first data record from PROJECTS that has a value for ACCOUNTNUMBER equal to the value of ACCOUNTNUMBER in CUSTOMERS.
4. Reports the record complex.

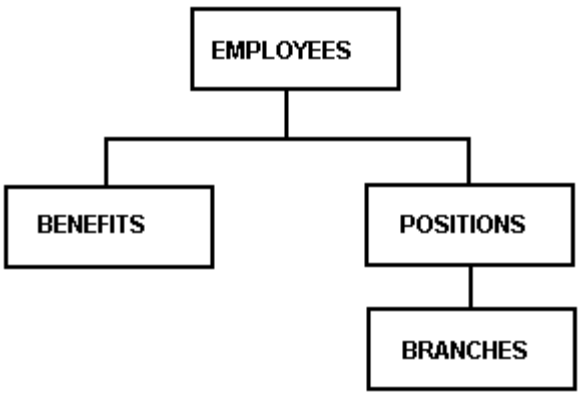
After reporting the record complex, QUIZ does the following:

1. Reads the next record from both ORDERS and PROJECTS when the value for ACCOUNTNUMBER equals that of the CUSTOMERS record-structure.
2. Reports the new record complex. QUIZ continues, reading from both ORDERS and PROJECTS for each new record complex.

Mixed Hierarchical and Parallel Linkage

The following example shows a mixed hierarchical and parallel linkage:

```
> ACCESS EMPLOYEES &
>   LINK TO BENEFITS &
>   AND TO POSITIONS &
>   LINK TO BRANCHES
```



Depending on the requirements of your application and the nature of the data records that are stored in your files, you might choose either hierarchical or parallel linkage.

How QUIZ Builds Record Complexes

If the number of related data records for the record-structures in the two parallel files isn't the same, QUIZ substitutes dummy data records for the record-structure that has run out of related records. The content of this dummy record depends on whether the record is from a relational database table and whether null value support is enabled in the PowerHouse dictionary. If the record is from a relational database table and null value support is enabled, then the data record is initialized to null values. If null value support is disabled or the record is not from a relational database table, then the data record is initialized to zeroes, spaces, and dictionary initial values. Regardless, the linkitem is initialized to the value used for linkage.

For example:

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERS &
>   AND TO PROJECTS
```

CUSTOMERS

ACCOUNTNUMBER	CUSTOMER
1010	Kane Contractors

ORDERS

ORDERNUMBER	**ACCOUNTNUMBER**	PARTNUMBER
5001	**1010**	1001

PROJECTS

PROJECT	CUSTOMER	TITLE	**ACCOUNTNUMBER**
6001	Kane	MIS	**1010**
6002	Kane	Support	**1010**

In this example, QUIZ builds two record complexes for ACCOUNTNUMBER 1010 using the following steps:

To build record complex 1, QUIZ does the following:

1. Reads the CUSTOMERS record for ACCOUNTNUMBER 1010.
2. Reads the ORDERS record for ORDERNUMBER 5001.
3. Reads the PROJECTS record for PROJECT 6001.

To build record complex 2, QUIZ does the following:

1. Copies the CUSTOMERS record for ACCOUNTNUMBER 1010.
2. Adds a dummy data record for ORDERS (since there are no more records in ORDERS with ACCOUNTNUMBER 1010, but there are in PROJECTS).
3. Reads the PROJECTS record for PROJECT 6002.

Optional Linkage

In hierarchical linkage, QUIZ doesn't build a record complex if a data record in the primary record-structure has no related data records in a record-structure to which it is being linked. As a result, the primary record isn't processed. This prevents incomplete data from being processed.

The OPTIONAL keyword in the ACCESS statement defines a linkage that is processed even if no related data records are found.

For example, the following statement defines an optional linkage between EMPLOYEES and BILLINGS:

```
> ACCESS EMPLOYEES LINK TO BILLINGS OPTIONAL
```

When this statement is executed, data for employees with or without billings is processed. Without the OPTIONAL keyword, the data for EMPLOYEES without BILLINGS isn't processed at all.

In an optional linkage, when no related data records are found, QUIZ constructs a record complex using a dummy data record for the related record-structure. The content of this dummy record depends on whether the record is from a relational database table and whether null value support is enabled in the PowerHouse dictionary. If the record is from a relational database table and null value support is enabled, then the data record is initialized to null values. If null value support is disabled or the record is not from a relational database table, then the data record is initialized to zeroes, spaces, and dictionary initial values. Regardless, the linkitem is initialized to the value used for linkage.

In a parallel linkage, use the OPTIONAL keyword when you want QUIZ to process data records from the primary record-structure if neither parallel record-structure has any related data records. QUIZ automatically handles the situation in which related data records for one parallel record-structure run out before related data records for another parallel record-structure. You can add the OPTIONAL keyword to either parallel record-structure in the ACCESS statement.

How to Specify Linkage Explicitly

In many reports, getting the exact linkage you want is easier when you define your linkage explicitly. By using an explicit linkage, you identify the item or expression that QUIZ uses to link each pair of record-structures.

You can specify linkage to record-structures by using

- the VIAINDEX option to specify linkage via a particular index
- an initial subset of the index segments
- names of items and segments
- record numbers

Specifying Linkage Using the VIAINDEX Option

You can specify linkage using the viaindex option by

- a single segment index
- a multiple segment index

Single Segment Index

You can specify linkage using the VIAINDEX option of the ACCESS statement to explicitly control linkage to a record-structure via a single segment index.

For example:

```
> ACCESS EMPLOYEES LINK VIAINDEX EMPINDEX TO STOCKS
```

links the EMPLOYEES record-structure to the STOCKS record-structure via the index EMPINDEX of STOCKS.

Multiple Segment Index

You can specify linkage using the VIAINDEX option of the ACCESS statement to explicitly control linkage to a record-structure via a multiple segment index.

For example:

```
> ACCESS DIVISIONS LINK VIAINDEX INDEX1 TO BRANCHES
```

A record-structure and an index are defined in the data dictionary as follows:

```
Record BRANCHES
    Item SEGMENTX
    Item SEGMENTY
    Item SEGMENTZ
Index INDEX1
    Segment SEGMENTX
    Segment SEGMENTY
    Segment SEGMENTZ
```

Linkage between DIVISIONS and BRANCHES is based on a multiple segment index. The linkage can be based on one, two, or three segments, depending on the name matches that QUIZ finds when establishing linkage.

If, in the preceding example, all three segments are used in an index to BRANCHES and DIVISIONS, then linkage is established by SEGMENTX, SEGMENTY, and SEGMENTZ.

Specifying Linkage Using an Initial Subset

You can specify linkage using an initial subset by using

- segments and items
- expressions
- lists of segments, items, or expressions

Segments and Items

You can specify linkage using an initial subset of the index segments. For example, you can specify linkage via the first two segments of INDEX1, as in

```
> ACCESS DIVISIONS LINK VIAINDEX INDEX1 &
>   TO SEGMENTX, SEGMENTY OF BRANCHES
```

Items in the DIVISION record are linked to segments of BRANCHES:

```
Record DIVISIONS
   Item ITEMA
   Item ITEMB
   Item ITEMC
```

You can also specify a list of items and/or segments from previously declared record-structures. For example:

```
> ACCESS DIVISIONS LINK ITEMA, ITEMB &
>      VIAINDEX INDEX1 TO SEGMENTX, SEGMENTY OF BRANCHES
```

If you specify a list of items and/or segments, the number must equal the number of specified segments of the "link to" record-structure.

Lists of Segments, Items, or Expressions

You can specify a list of items or expressions to be linked to the specified segments. In the following example, SEGMENTZ isn't specified:

```
> ACCESS DIVISIONS LINK ("USA", "WEST") &
>   VIAINDEX INDEX1 TO SEGMENTX, SEGMENTY OF BRANCHES
```

If you specify a list of items and expressions, the number of entries must be equal to the number of specified segments of the "link to" record-structure.

You can specify a "from" list without a "to" list when specifying linkage. For example:

```
> ACCESS DIVISIONS LINK ITEMA, ITEMB, ITEMC &
>   VIAINDEX INDEX1 TO BRANCHES
```

Specifying Linkage Using Names

You can specify a linkage even if the linkage items and segments don't have the same names. To do this, specify the items and segments to be matched. QUIZ matches specified items and segments according to how they're declared in the ACCESS statement.

For example, suppose you want to link the DIVISIONS data records to the BRANCHES data records, but the names of the linkage items in these record-structures don't match.

You can specify explicit linkage using

- ITEMA and ITEMB of DIVISIONS
- SEGMENTX and SEGMENTY of BRANCHES

For example:

```
> ACCESS DIVISIONS LINK ITEMA, ITEMB &
>   VIAINDEX INDEX1 TO SEGMENTX, SEGMENTY OF BRANCHES
```

Specifying Linkage Using Record Numbers

You can specify linkage using record numbers (a numeric item, a number, or an expression) to link to record-structures that are stored in direct or relative files, or in subfiles.

For example:

```
> ACCESS EMPLOYEES LINK TO RECORD 2 OF BILLINGS
```

Ascending/Descending Index Support

Segments of an index may be specified in ascending or descending order for databases that support ordering on an index or segment.

When PowerHouse 4GL generates database retrieval requests as a result of an explicit VIAINDEX index, it generates the sorting specification in the request to match the order declared when the index was defined.

For example, if an index called PAYMENTS_CUSTOMER_DATE consists of two segments, CUSTOMER_NUM (ascending) and PAYMENT_DATE (descending), then the PowerHouse 4GL database request generated for the statement

```
> ACCESS VIAINDEX PAYMENTS_CUSTOMER_DATE
```

contains the equivalent of

```
> SORT ON CUSTOMER_NUM A PAYMENT_DATE D
```

to produce data in order of CUSTOMER_NUM with the most recent payments first for each customer.

For example:

```
> SQL DECLARE empskills CURSOR FOR &
> SELECT * FROM employees, skills, branches &
>   WHERE employee = :employee AND maritalstatus= 'M' &
>   AND employees.employee = skills.employee &
>   AND employees.branch = branches.branch &
>   ORDER BY branch, lastname, firstname, skills
>   ACCESS *emplist LINK TO empskills
```

The above statements could also be coded using a substitution in the WHERE clause instead of the explicitly coded WHERE clause:

```
> SQL DECLARE empskills CURSOR FOR &
> SELECT * FROM employees, skills, branches &
>   WHERE ::WHERE(1=1) AND maritalstatus= 'M' &
>   AND employees.employee = skills.employee &
>   AND employees.branch = branches.branch &
>   ORDER BY branch, lastname, firstname, skills
> ACCESS *emplist LINK TO empskills &
>   WHERE(employee=:employee)
```

The link between records and result rows uses the linkitem employee, which only appears in the WHERE clause of the cursor declaration. For every record of the subfile, PowerHouse 4GL will execute the SELECT statement declared in the EMPSKILLS cursor.

The following are valid ACCESS statements using the above cursor:

```
>  ACCESS empskills WHERE(employee=10)
>  ACCESS empskills
```

When QUIZ is reporting, it uses the last entered ACCESS statement.

BUILD

Compiles and saves a QUIZ report.

Syntax

BUILD filespec [option]...

filespec

Specifies the file in which the compiled QUIZ report is saved.

Options

The options are DISPLAY and USERS INCLUDE.

DISPLAY string...

Instructs QUIZ to display information whenever the compiled report is executed. You can enter multiple strings.

Limit: 100 characters per string

USERS INCLUDE option

Restricts execution of the report to the application security classes listed.

The USERS INCLUDE options are ALL and class [,class]....

ALL

Includes all application security classes declared in the data dictionary, including the application security class UNKNOWN.

class [,class]...

An application security class declared in the data dictionary. Only the specified classes are able to execute the report. The application security class UNKNOWN can be listed as a class.

When you compile a report, QUIZ assumes your level of security. If security is changed later, QUIZ has no knowledge of the change. (Security is handled by the parser process, which is bypassed when you use compiled reports.)

Limit: 1 to 64 user classes

Discussion

The BUILD statement compiles the current QUIZ report and saves it in a file in compiled form.

Compiled reports execute more quickly than source statements. To execute a compiled report, use the EXECUTE statement, rather than the USE statement.

The BUILD statement stores the parsed QUIZ syntax in the named file. You can include DISPLAY as an option of the BUILD statement.

In QUIZ, statements with errors are usually ignored, so building the report simply builds without the statements that were in error.

Creating Compiled Reports

A compiled report is based on the current release of QUIZ. If you are using a release of QUIZ that differs from the one used to compile the report, you should recompile the report from the source statements.

The DISPLAY statement, the SAVE statement, and the following options of the SET statement are not saved in a compiled report:

SET options		
DICTIONARY	JOB|NOJOB	LIST|NOLIST
PRINT|NOPRINT	VERIFY|NOVERIFY	

When you execute a compiled report with the **auto** program parameter, the QUIZ session ends on completion of the report. This happens whether or not an EXIT statement was included in the report.

Limit: A compiled report is based on the current definition of files and elements in the data dictionary. Compiled reports must be recompiled when you make changes to the dictionary.

For more information about how QUIZ creates compiled reports, see the section, "Locating Files", in Chapter 1, "Running PowerHouse 4GL", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Example

The following example generates a report of the current inventory, grouped by suppliers. The BUILD statement compiles the report:

```
> ACCESS PARTS &
>   LINK TO PARTSUPPLIERS &
>   LINK TO SUPPLIERS
>
> SORT &
>   ON SUPPLIERKEY OF PARTSUPPLIERS &
>   ON PARTNUMBER OF PARTS &
>   ON PARTVARIANT OF PARTS
>
> REPORT &
>   SUPPLIERNAME OF SUPPLIERS &
>   PRINT AT SUPPLIERKEY OF PARTSUPPLIERS &
>   PARTNUMBER OF PARTS &
>
> PRINT AT PARTNUMBER OF PARTS &
>   PARTVARIANT OF PARTS &
>   QOH OF PARTS &
>   UNITCOST OF PARTS
>
> FOOTING AT &
>   SUPPLIERKEY OF PARTSUPPLIERS &
>   SKIP 2
>
> BUILD &
>   CURRINV &
>   DISPLAY "  Current Inventory by Supplier  " &
>   "  ----------------------------------------  "
```

The DISPLAY option of the BUILD statement is used to display messages at execution-time. Never use the DISPLAY statement to display messages at execution-time in compiled reports. Unlike the DISPLAY option of the BUILD statement, the DISPLAY statement is never compiled into a QUIZ report.

CANCEL

Cancels the current QUIZ report specifications.

Syntax

CANCEL[CLEAR]

CLEAR

Removes any source statements in the temporary save file, QUIZSAVE, once the report specifications are canceled.

Discussion

The CANCEL statement cancels the specifications of the current QUIZ report. The CANCEL statement does not cancel SET statements, except SET PAGE TITLE and SET SUBFILE.

Clearing the Temporary Save File

All QUIZ statements that you enter are automatically stored in the temporary save file, QUIZSAVE.

The CANCEL statement doesn't clear the temporary save file, QUIZSAVE, unless the CLEAR option is specified. By using the CLEAR option, you can ensure that statements with errors aren't retained in the temporary save file. This is important when you save QUIZ statements to permanent files.

Example

This report demonstrates the use of CANCEL CLEAR to correct errors made in the statement entry. In this example

- the report statements typed before CANCEL are ignored when QUIZ processes the report.
- the CLEAR option removes all the report statements from the temporary save file.

```
> ACCESS PARTS &
>   LINK TO PARTSUPPLIERS &
>   LINK TO SUPPLIERS
>
> CHOOSE PARTNUMBER 20
>
> SORT ON SUPPLIERKEY OF PARTSUPPLIERS
>
> REPRRT SUPPLIERNAME OF SUPPLIERS
  ^^^^^^
*E* Expected: ACCESS BUILD CHOOSE DEFINE EDIT DISPLAY EXECUTE EXIT FINAL
FOOTING GO HEADING INITIAL PAGE QSHOW QUIT CANCEL REPORT NOREPORT SAVE SELECT
AND SET SHOW SORT SORTED USE REVISE
>
> CANCEL CLEAR
>
> ACCESS PARTS &
>  LINK TO PARTSUPPLIERS &
>  LINK TO SUPPLIERS
>
> CHOOSE PARTNUMBER 20
>
> SORT ON SUPPLIERKEY OF PARTSUPPLIERS
>
> REPORT SUPPLIERNAME OF SUPPLIERS
>
> PRINT AT CUSTOMERKEY OF PARTSUPPLIERS
>
> GO
```

Use the CLEAR option when you specify CANCEL, especially if you intend to save source code to a permanent file.

If you specify SAVE and you haven't cleared the temporary save file, all the statements that you entered in that QUIZ session are included in your permanent file.

CHOOSE

Extracts data from an indexed file, relational table or view, or SQL cursor by item value.

Syntax

CHOOSE [sql-substitution...|VIAINDEX indexname]
[linkitem [GENERIC|NOGENERIC] [choose-option]
[,linkitem [GENERIC|NOGENERIC] [choose-option]]...]

Options

sql-substitution...

An sql-substitution can be specified for any substitution variable defined on the DECLARE CURSOR statement. Two default sql-substitutions, WHERE and ORDERBY, will be inserted in generated SQL statements even if the corresponding substitution-variables do not exist on a DECLARE CURSOR statement.

The syntax for an sql-substitution is:

substitution-variable (text)

For more information about sql-substitutions and substitution-variables, see Chapter 1, "About PowerHouse 4GL and Relational Databases", in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

Limit: The VIAINDEX and sql-substitution options are mutually exclusive.

Limit: Sql-substitutions are valid only if the primary record structure is a cursor and must appear immediately following the CHOOSE keyword.

Limit: Program variables, using a colon and a PowerHouse 4GL variable name, cannot be used in sql-substitution.

VIAINDEX indexname

The name of an index in the primary record-structure of an indexed file or relational table. Using indexname forces data records to be retrieved in index order; otherwise, the order is determined by the file system. If indexname and linkitems are used together, the linkitems must exist in the index and be consecutive initial segments of that index. If you use indexname alone, without linkitems, you choose all the data records indexed by that index. However, you don't have to specify all the segments that make up the index.

Limit: This option is not valid when the CHOOSE statement applies to a DECLARE CURSOR. The VIAINDEX and sql-substitution options are mutually exclusive.

Limit (**UNIX, Windows**): The VIAINDEX option is required when the CHOOSE statement is used with a B-Tree or TPI index on an Eloquence dataset.

linkitem

For a table or view in a relational database (if VIAINDEX is not specified), any column in the table can be a linkitem. For a record-structure in an indexed file, the first linkitem must either be the segment of a single segment index or the first segment of a multiple-segment index. In the case of a multiple-segment index, the subsequent linkitems from the same index may be specified, but the linkitems must be specified in the same order in which the segments are defined in the dictionary.

Limit: 255 linkitems. All segments must be specified for Eloquence indexes unless they are B-Tree or TPI indexes.

GENERIC|NOGENERIC

GENERIC allows users to choose data records using partial index values. NOGENERIC prevents users from choosing records using partial values. For information about generic retrieval, see (p. 40).

Limit: Not valid for Eloquence indexes, unless they are B-Tree or TPI indexes.

Limit: Applies to character segments only.

Default: GENERIC

Choose-Options

The options are case-expression-set, conditional-expression-set, PARM, SYSTEMVALUE, and value-set.

case-expression-set

Compares the value of an item against a value or range of values and selects one expression-set to be used to determine the values for the linkitem.

The general form is:

(CASE [OF] item
WHEN value-set|EXISTS|NULL|MISSING
{THEN|:} expression-set|NULL|MISSING
[WHEN value-set|EXISTS|NULL|MISSING
{THEN|:} expression-set|NULL|MISSING]...
[DEFAULT expression-set|NULL|MISSING])

If there is no match, the specified default is used. If no default is specified, no records are chosen.

Items that occur, also known as arrays, cannot be used.

The general form of value-set used within the case-expression-set is the same as that used for the value-set choose-option. For more information, see (p. 40).

expression-set

The general form of an expression-set is:

expression|(conditional-expression)
[TO expression|(conditional-expression)]
[,expression|(conditional-expression)
[TO expression|(conditional-expression)]]...

A conditional-expression differs from a conditional-expression-set. For more information about conditional-expressions, see the section, "Conditional Expressions", in Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

If specific values are used exclusively in the expression-set, the form is the same as that used for the value-set choose-option.

If an evaluated expression contains a generic retrieval character (by default, @ or @@) as the last non-blank character, and the TO and NOGENERIC options are not used, QUIZ will use generic retrieval.

Limits:

- The resulting value of an expression must be compatible in type and size to that of the linkitem.
- Record items, table columns, and cursor columns may not be used in expressions as they are available only after the record complex is built (the CHOOSE is used to build the record complex).
- If an expression contains defined items, the defined items must not contain record items. Defined items used in expressions must be coded before the CHOOSE statement.

conditional-expression-set

A conditional-expression-set, when evaluated, selects one expression-set to be used to determine the values for the linkitem.

The general form is:

(expression-set [IF condition]
[ELSE expression-set IF condition]...
[ELSE expression-set])

When a conditional-expression-set doesn't end with an unqualified ELSE option and none of the conditions have been satisfied, no records are chosen.

The conditional-expression-set must be enclosed within parentheses to avoid ambiguity between the linkitem and defined items within the expressions. These parentheses are required. At run-time, each form of expression evaluates to a value-set.

PARM [parm-option]...

Prompts for linkitem values at execution-time. Partial linkitem values using the generic retrieval character (by default, @ and @@) are allowed in response to a PARM prompt.

Limit: Only one value per input line can be entered. The PARM option cannot be used with any other CHOOSE option.

parm-options		
ALL|NOTALL	DOWNSHIFT|UPSHIFT	FORCE|NOFORCE CENTURY
FORMAT	ON ERRORS	PROMPT
RANGE|NORANGE	SEPARATOR	

ALL|NOTALL

ALL allows users to choose all the data records in a record structure either by entering the generic retrieval character (by default, @) or by making a null entry in response to the first prompt. NOTALL prevents users from choosing all values using the generic retrieval character or by making a null entry.

Default: ALL

DOWNSHIFT|UPSHIFT

Shifts the entered value of a character item to either lowercase or uppercase. Stored values aren't changed.

Limit: Valid only for character items. Non-alphabetic characters within a character item aren't affected.

FORCE|NOFORCE CENTURY

FORCE CENTURY specifies that the user must enter a century on all century-included date fields. This option applies to century-included dates with two or four-digit year formats.

Default: To find out the active value of the option, you must look at the ELEMENT, the USAGE, and the SYSTEM OPTIONS statements. If the option is unspecified on the CHOOSE statement, the active value is taken from the ELEMENT. If the option is unspecified on the ELEMENT statement or a related USAGE, the active value is taken from the SYSTEM OPTIONS.

Limit: Valid only for century-included dates.

FORMAT date-format

Specifies the format for entering date item values. Date values can be entered either with or without separator characters. A date-format can be one of the following:

Date-format	Example	Date-format	Example
YYMMDD	01/05/23	YYMMMDD	01/MAY/23
YYYYMMDD	2001/05/23	YYYYMMMDD	2001/MAY/23
YYMM	01/05	YYMMM	01/MAY
YYYYMM	2001/05	YYYYMMM	2001/MAY
YYDDD	01/125	YYYYDDD	2001/125

Date-format	Example	Date-format	Example
MMDDYY	05/23/01	MMMDDYY	MAY/23/01
MMDDYYYY	05/23/2001	MMMDDYYYY	MAY/23/2001
MMYY	05/01	MMMYY	MAY/01
MMYYYY	05/2001	MMMYYYY	MAY/2001
MMDD	05/23	MMMDD	MAY/23
DDMMYY	23/05/01	DDMMMYY	23/MAY/01
DDMMYYYY	23/05/2001	DDMMMYYYY	23/MAY/2001
DDMM	23/05	DDMMM	23/MAY
DDDYY	125/01	DDDYYYY	125/2001

YYYY - four digit year (e.g., 2001)

MM - two digit month (e.g., 05)

MMM - three character month name (e.g., MAY)

DD - two digit day for a month (e.g., 23)

DDD - three digit day for a year (e.g., 365)

Regardless of the output order of the date, the internal working format is YYMMDD (for dates without centuries), YYYYMMDD (for dates with centuries).

The FORMAT option governs data entry by determining the way you can enter date values. Dates can always be entered in the format specified in the FORMAT option, with or without the established separator character and with either the MM or MMM month format.

If the FORMAT option is used but the SEPARATOR option isn't, the only separator character that QUIZ accepts is the separator character specified by System Options, or if it isn't specified, a slash (/).

If a two-digit year is specified in the date format, applications won't accept a four-digit year. A two-digit year is represented by YY (for example, 01).

If a four-digit year is specified in the date format, you can only enter a two-digit year if you enter a separator character between the year and any adjacent numeric component of the date. The default century is added automatically.

Single-digit day and month entries are accepted if the user enters the separator character, as in 4/8/2001. An entry of 4AUG2001 is also allowed, because PowerHouse 4GL accepts a single-digit day entry if the middle value is a three-character month.

A three-digit day of the year from 1 to 366 is represented by DDD.

Although values for date items can be entered in a variety of formats, the values are always stored in either YYMMDD or YYYYMMDD form.

Limit: Valid only for date items. This option only affects the entry format; the display format isn't affected.

Default: If the DATE PARM INHERIT system option is set, the date format is taken from the element corresponding to the linkitem. If there is no date format specified on the element,then the date format is taken from the DATE FORMAT system option. If the DATE FORMAT system option is not set, then YYYYMMDD is used for eight-digit dates and YYMMDD is used for six-digit dates. If DATE PARM INHERIT is not set, then YYYYMMDD is used for eight-digit dates and YYMMDD is used for six-digit dates.

ON ERRORS REPROMPT [n [TIMES]]

Specifies how many times QUIZ reprompts the user if entered values fail editing. The report terminates if edit errors aren't resolved within the specified number (n) of reprompts.

When operating QUIZ in batch mode, REPROMPT is ignored because QUIZ prompts only once. TIMES is used only for documentation.

Limit: 99 reprompts for interactive sessions; 1 prompt for batch jobs.

Default: Without a specification, QUIZ reprompts indefinitely.

PROMPT [string] [n TIMES]

Displays a prompting message at execution-time.

string

Specifies the prompt message displayed at execution-time. If no prompt string is used, QUIZ uses the label assigned to the element in the data dictionary as a prompt string. If no label exists, QUIZ uses the name of the segment.

TIMES

Limits the number of accepted values that users can enter. Invalid entries aren't counted. If omitted, QUIZ prompts until a null entry.

Limit: The number specified must be between 1 and 500.

RANGE [TOPROMPT string]}|NORANGE

RANGE specifies that the user can enter a range of PARM values. NORANGE prevents the entry of range values.

Limit: Not valid for Eloquence indexes, unless they are B-Tree or TPI indexes.

Default: NORANGE

TOPROMPT string

Specifies the string used to prompt for the ending or "to" value index at execution-time.

Limit: Valid when both PARM and RANGE have been specified.

Default: If RANGE is specified and TOPROMPT is omitted, the upper limit of the range is prompted for with the string "Up to:". If no upper limit is entered, the CHOOSE statement is executed using an exact match of the value entered for the lower limit.

SEPARATOR char

Specifies the character that overrides the separator character specified for dates, by default, the System Options separator or slash (/). The separator character separates the day, month, and year portions of a date element when it is displayed. For example, the separator character "-" produces dates that are displayed like this: "01-05-25".

If SEPARATOR is used and FORMAT isn't used, the specified separator character is used with the default date format. Date values can be entered either with or without separator characters.

Limit: Valid for date items only.

Default: If the DATE PARM INHERIT system option is set, then the date separator is taken from the element corresponding to the linkitem. If there is no date separator specified on the element, then the date separator is taken from the DATE SEPARATOR system option. If the DATE PARM INHERIT system option is not specified, then the date separator is taken from the DATE SEPARATOR system option. In either case, if the DATE SEPARATOR system option is not set, then a slash (/) is used.

SYSTEMVALUE [LOGICAL|SYMBOL] string-expression [RANGE|NORANGE]

Gives the ability to extract the values defined at the operating system level. Permits the use of values set at the operating system level.

Limit: The SYSTEMVALUE option cannot be used with any other CHOOSE option. If the SYSTEMVALUE cannot be translated, no records will be retrieved.

string-expression

Specifies the name of the environment variable.

LOGICAL|SYMBOL (OpenVMS)

LOGICAL specifies the retrieval of an OpenVMS logical name. If the keyword LOGICAL is not explicitly used, a logical name is assumed.

SYMBOL specifies retrieval of a DCL symbol. Local symbols are retrieved before global symbols.

The standard OpenVMS logical name table search order, set by LNM$FILE_DEV, is used. Typically, the process table (LNM$PROCESS) is searched first, followed by the job table (LNM$JOB), the group table (LNM$GROUP) and finally the system table (LNM$SYSTEM).

RANGE|NORANGE

Specifies whether a sequence of values is interpreted as a range or a list.

Defaults: NORANGE (**UNIX, Windows**) or LOGICAL NORANGE (**OpenVMS**).

value-set

Specifies a value, a series of values, or a range of values. The general form is:

value [TO value][[,] value[TO value]]...

If there are multiple records associated with a particular value, all such records are retrieved for processing.

If there are multiple values or value ranges, the records are retrieved in the order listed. For example, `CHOOSE BRANCH_CODE "M@"` will retrieve `"MEL"` then `"MON"` due to the ascending index, but `CHOOSE BRANCH_CODE "MON", "MEL"` will retrieve `"MON"` then `"MEL"`.

Limit: Not valid for Eloquence indexes, unless they are B-Tree or TPI indexes. The maximum number of values or series of value ranges is 500. The value-set option cannot be used with any other CHOOSE option.

Discussion

When the CHOOSE statement is used, QUIZ retrieves primary record-structure data records for the declared linkitem.

For a given linkitem, the CHOOSE options (case-expression-set, conditional-expression-set, PARM, SYSTEMVALUE, and value-set) are mutually exclusive.

The Difference Between the CHOOSE and SELECT Statements

The SELECT statement can also be used to retrieve a set of data records. However, the CHOOSE and SELECT statements aren't mutually exclusive. The CHOOSE statement always retrieves records by index value; the SELECT statement always reads records retrieved.

For more efficient performance, use the CHOOSE statement instead of the SELECT statement when possible. Instead of reading the primary record-structure sequentially, QUIZ reads records by index value only. This reduces processing time and increases efficiency.

If CHOOSE and SELECT are used in the same report, the CHOOSE statement is performed before the SELECT statement. Entering the CHOOSE statement by itself cancels any previous CHOOSE statements and associated EDIT statements.

Using Generic Retrieval

Include the partial segment value and the generic retrieval character (by default, @) in the statement within quotation marks. For example:

```
> CHOOSE LASTNAME "M@"
```

or

```
> CHOOSE LASTNAME "M@@"
```

The entry "M@" chooses all last names that start with the letter M. For example, M@ would match "Moffat" and "Morrissey" but not "Smith". The entry "M@@" chooses all last names beginning with the letter M through to the end of the file. M@@ would successfully match "Moffat", "Morrissey", and "Smith" and every other last name beginning with a letter greater than "M".

Retrieval by partial values isn't valid when you specify a series of values, or a range of values using more than one generic retrieval character, as in

```
> CHOOSE LASTNAME "A@@" TO "B@@"
```

However, the entry

```
> CHOOSE LASTNAME "A" TO "B"
```

has a generic-like effect. This statement chooses names from the lowest value name beginning with A to the highest value name beginning with B.

Preventing Generic Retrieval

To prevent a user from selecting records using partial segment values and the generic retrieval character, include the NOGENERIC option. For example:

```
> CHOOSE LASTNAME NOGENERIC PARM
```

Now, an entry of "M@" causes QUIZ to look for an index value of exactly M@.

NOGENERIC is used in the case of a multi-segment index or a substructured segment where the last non-blank character corresponds to the bit pattern of the generic retrieval character. Unless NOGENERIC is specified, QUIZ will assume that generic retrieval is required and potentially retrieve many records rather than a specific matched value.

Responding to PARM Prompts

QUIZ waits three minutes for the user to respond to any prompt; otherwise, an exception error occurs and the report is terminated.

The user can enter only one value per input line.

A null entry (pressing [Return]) in response to the first "from" prompt tells QUIZ to select the whole record-structure and continue processing, unless the NOTALL option is specified.

When the user runs QUIZ interactively, a null entry in response to the first CHOOSE prompt causes QUIZ to issue a message asking if the user wants to choose all the records.

If the user enters "y" or "yes", then QUIZ chooses all the records for that record-structure. If the user enters "n" or "no" or a null entry, then QUIZ doesn't choose any records.

No query is made when QUIZ is running in batch mode.

Interrupting Prompting

The user can interrupt prompting by pressing the following in response to a prompt: [Ctrl-C] (**OpenVMS, UNIX, Windows**).

When the user does this, QUIZ issues a message asking if the user wants to continue processing:

- If the user responds "yes", QUIZ restarts processing of the report from the point of interruption.
- If the user responds "no", QUIZ terminates the execution of the current report.

Using Generic Retrieval in Response to PARM Prompts

For record-structures in indexed files and relational tables, the user can enter part of a value, along with the generic retrieval character (by default, @) in response to a PARM prompt. Generic retrieval is available only for character-type items.

Generic retrieval can be specified in two ways:

- a partial string followed by a single generic retrieval character selects a value beginning with that string
- a partial string followed by two generic retrieval characters selects from the string to high values

For example, if the user enters "M@" in response to the following prompt:

```
LASTNAME:
```

QUIZ retrieves all last names that begin with the letter M. The generic retrieval character instructs QUIZ to match the entry with all other values beginning with the specified letter (M). Likewise, the user could choose all last names beginning with MO, by entering "MO@", or all last names beginning with the letter M through to the end of the file, by entering "M@@".

Sequence of Prompts

Prompting by the CHOOSE statement occurs first, followed by prompting by the DEFINE statement. For information about entering execution-time values for defined items, see (p. 51).

How QUIZ Selects Values

QUIZ selects values in one of two ways:

- If you specify a single value, QUIZ selects data records matching that exact value.
- If you specify a range, QUIZ selects any values within that range including the specified lower and upper range limits.

Editing Execution-Time Parameter Values

You can use the EDIT statement to

- validate entered values
- specify both a pattern and acceptable values for execution-time parameters

When you use the EDIT statement with execution-time parameters, QUIZ checks values as they're entered:

- If an editing error occurs, QUIZ issues a message and repeats the prompt.
- If editing errors aren't resolved, QUIZ reprompts indefinitely unless the ON ERRORS REPROMPT option of the EDIT statement is used. If the ON ERRORS REPROMPT is used and the edit errors are not resolved within the specified number of reprompts, the QUIZ report is terminated.

When you enter a subsequent CHOOSE statement, both the EDIT and previous CHOOSE statements are canceled.

Ascending/Descending Indexes

Although a record-structure may be indexed in ascending or descending order, you must specify the logical range from low values to high values. QUIZ retrieves the data records, processing them in ascending or descending order depending on how the record-structure is indexed.

If a record-structure is indexed in ascending order, the chosen records are returned from the lowest value to the highest value. If a record-structure is indexed in descending order, the same data is displayed in descending order.

A change in the collating sequence in the data dictionary doesn't affect the CHOOSE statement.

The CHOOSE statement can only be used for indexes; it uses the record-structure system's index information to determine which data records to process. If the primary record-structure is a relational table, any item in the record-structure can be used on the CHOOSE statement.

Retrieving Data Records

The CHOOSE statement ignores trailing blanks in non-range selection values. This means that the CHOOSE statement values "1000 " and "1000" are treated as the same value for retrieval.

The CHOOSE statement searches for and retrieves stored values, not displayed values. If you want to choose the data records with the value "$483.27", you must enter

```
> CHOOSE AMOUNT 48327
```

not

```
> CHOOSE AMOUNT 483.27
```

since the value is stored without any decimal point, leading sign, or trailing sign.

Limit: The CHOOSE statement can only be applied to indexes that are defined for the primary record-structure being accessed. You can use only one CHOOSE statement in a single report.

CHOOSE with Expressions

Expressions must be completely evaluated into a single value prior to choosing any records. Because the CHOOSE statement sets the retrieval criteria for the PRIMARY file and no records have been read when the CHOOSE is evaluated, record items cannot be used in the expressions.

At compile-time, PowerHouse 4GL creates all the combinations that could be used at run-time and the syntax is checked. You can see the text of the generated statements by using SET LIST SQL.

Examples

Using Ranges with the PARM Option

This example demonstrates how to use ranges with the PARM option:

```
> ACCESS CUSTOMERS
> CHOOSE CUSTOMERNAME PARM RANGE &
>   PROMPT "Enter the initial value for CUSTOMERNAME: "&
>   TOPROMPT "Enter the ending value for CUSTOMERNAME: "
> GO
```

QUIZ displays these prompts at execution-time:

```
Enter the initial value for CUSTOMERNAME:
Enter the ending value for CUSTOMERNAME:
```

Entering a Sequence of Values

If you enter a sequence of values in response to the following prompts:

```
Enter the initial value for CUSTOMERNAME: Windsor
Enter the ending value for CUSTOMERNAME: <CR>
Enter the initial value for CUSTOMERNAME: Bennet
Enter the ending value for CUSTOMERNAME: Cuthbert
Enter the initial value for CUSTOMERNAME: Truman
Enter the ending value for CUSTOMERNAME: <CR>
Enter the initial value for CUSTOMERNAME: <CR>
```

QUIZ selects data records

- with only the names "Windsor" or "Truman", and
- that fall between the names "Bennet" and "Cuthbert", inclusive

However, if there is a data record with a CUSTOMERNAME of "Cuthbertson", QUIZ selects it as well. To avoid selecting "Cuthbertson" along with "Cuthbert", you must include a space at the end of "Cuthbert" when you respond to the "to" prompt.

Default Prompting by QUIZ

In the next example, only the PROMPT option is specified:

```
> ACCESS CUSTOMERS
>
> CHOOSE CUSTOMERNAME PARM RANGE &
>   PROMPT "Enter the initial customer name: "
Choose viaindex CUSTOMERNAME
> GO
```

QUIZ must supply the prompt for the upper range limit at execution-time:

```
Enter the initial customer name:
Up to:
```

In the next example, neither the PROMPT nor the TOPROMPT option is specified:

```
> ACCESS CUSTOMERS
> CHOOSE CUSTOMERNAME PARM RANGE
Choose viaindex CUSTOMERNAME
> GO
```

QUIZ supplies the prompts at execution-time:

```
Customer Name:
Up to:
```

Using the TIMES Option

To indicate how many values can be entered, enter a number immediately following the prompt string (if you specified a string) or immediately following the prompt keyword (if you didn't specify a string). The number must be followed by the keyword TIMES.

This example

```
> CHOOSE CUSTOMERNAME &
>   PARM PROMPT "Enter a customer name: " 3 TIMES
```

tells QUIZ to prompt for a last name a maximum of three times.

Using the EDIT Statement

To control the values that users can enter when responding to a CHOOSE statement, use the EDIT statement. This example

```
> CHOOSE ACCOUNTNUMBER &
>   PARM PROMPT "Enter an Account Number: "
Choose viaindex ACCOUNTNUMBER
>
> EDIT ACCOUNTNUMBER VALUES 1000 TO 5000
```

restricts the user to entering account numbers that fall within the range of one thousand to five thousand.

Using Expressions

In the following example, the user can choose one of three ranges of employee id-numbers by entering different values at the prompt. If no values are entered, the default range in the case expression is used. The value-sets after the THEN keyword can also be expressions that are evaluated when the report is run.

```
> ACCESS EMPLOYEES
> DEFINE X INTEGER = PARM &
>   PROMPT "Employee Group:   "
> CHOOSE EMPLOYEE &
>  (CASE OF X &
>   WHEN 1 THEN 0001 TO 1999 &
>   WHEN 2 THEN 2000 TO 2999 &
>   WHEN 3 THEN 3000 to 3999 &
>   DEFAULT 0001 TO 3999)
```

In the next example, one of the two value-sets is used depending on the value of X. If the value 1 is entered at the prompt, employees 1001, 1002, are 1003 are chosen. The IF condition applies to the complete list of employee numbers preceding it.

```
> ACCESS EMPLOYEES
> DEFINE X INTEGER = PARM PROMPT "X: "
> CHOOSE EMPLOYEE (1001, 1002, 1003 IF X = 1 &
>   ELSE 1004, 1005, 1006)
```

To apply the IF condition to a single value or expression, use parentheses. The following example chooses employee numbers 1001,1002, and 1003 if X = 1. If X is not equal to one, employee numbers 1001 and 1002 are chosen.

```
>  ACCESS EMPLOYEES
>  DEFINE X INTEGER = PARM PROMPT "X: "
>  CHOOSE EMPLOYEE (1001, 1002, (1003 IF X = 1))
```

The "STARTDATE TO ENDDATE" expression set is evaluated when the report is run and values are available from the prompts.

```
>  ACCESS EMPLOYEES
>  DEFINE STARTDATE DATE = PARM &
>      PROMPT "START DATE(YY/MM/DD): " FORMAT YYMMDD
>  DEFINE ENDDATE DATE = PARM &
>      PROMPT "END DATE(YY/MM/DD): " FORMAT YYMMDD
```

```
>  CHOOSE DATEJOINED &
>     (STARTDATE TO ENDDATE &
>     IF STARTDATE <> 0 AND ENDDATE <> 0)
```

The following two examples simplify entry of selection criteria. The CHOOSE statements manipulate what was entered to match what is stored in the files.

```
>  ACCESS BILLINGS
>  DEFINE INVOICENO NUMERIC = PARM PROMPT "Invoice #: "
>  CHOOSE INVOICEKEY("PR" + ASCII(INVOICENO,8))
.
.
.

>  ACCESS EMPLOYEES
>  DEFINE FNAME CHARACTER*20 = PARM PROMPT "First Name: "
>  DEFINE MNAME CHARACTER*20 = PARM PROMPT "Middle Name: "
>  DEFINE LNAME CHARACTER*20 = PARM PROMPT "Last Name: "
>  CHOOSE CHOOSENAME(PACK(FNAME + MNAME[1:1] + LNAME))
.
.
.
```

Using the SYSTEMVALUE Option

One or more values may be specified using the SYSTEMVALUE option. Values are separated by commas. If you need to include a comma as part of the value, prefix that comma with a backslash (\). For example, if you want to choose the value "SMITH, JOHN", define the system value, CHOOSEVALUES, as follows:

OpenVMS:	`define CHOOSEVALUES "SMITH\,JOHN"`
UNIX:	`setenv CHOOSEVALUES "SMITH\,JOHN"`
Windows:	`set CHOOSEVALUES="SMITH\,JOHN"`

To define a single value, you can use a statement such as the following:

OpenVMS:	`define LOG2 17`
UNIX:	`setenv LOG2 17`
Windows:	`set LOG2=17`

To define a list of values, the values must be expressed as a string, with commas separating values:

OpenVMS:	`define LOG3 "1,3,4"`
UNIX:	`setenv LOG3 "1,3,4"`
Windows:	`set LOG3="1,3,4"`

These values would be used as:

```
> CHOOSE PROJECTNO SYSTEMVALUE "LOG3" NORANGE
```

which is equivalent to:

```
> CHOOSE PROJECTNO PARM
> GO
Projectno: 1
Projectno: 3
Projectno: 4
Projectno: <CR>
```

Using Ranges with the SYSTEMVALUE Option

When using the RANGE option, both single values and ranges may be used. To indicate a single value, specify the value followed by two commas. The following specifies

- the single value 1
- the range 3 to 5
- the single value 7
- the range 9 to 15

OpenVMS:	`define log4 "1,,3,5,7,,9,15"`
UNIX:	`setenv log4 "1,,3,5,7,,9,15"`
Windows:	`set log4="1,,3,5,7,,9,15"`

The SYSTEMVALUE option

```
> CHOOSE PROJECTNO SYSTEMVALUE "LOG4" RANGE
```

is equivalent to:

```
> CHOOSE PROJECTNO PARM RANGE
> GO
Projectno: 1
Up to: <cr>
Projectno: 3
Up to: 5
Projectno: 7
Up to: <cr>
Projectno: 9
Up to: 15
Projectno: <cr>
```

To choose all values for a segment, enter the value @ or @@.

[SQL] DECLARE CURSOR (query-specification)

Defines a set of data as a run-time view.

Syntax

[SQL[IN database]]
DECLARE name CURSOR FOR
query-specification [UNION [ALL] query-specification...]
[ORDER BY sort-specification]

IN database

Specifies the name PowerHouse 4GL uses to attach to the database. This is the name used to declare the database in PDL.

name

Defines a logical name used to identify the set of data resulting from the query.

Limit: The name must be unique within the scope of the cursor. For a description of a cursor's scope, see (p. 47).

query-specification [UNION [ALL] query-specification...]

The query-specification defines a collection of rows that will be accessible when the cursor is opened.

The ALL option on the UNION option indicates that redundant duplicate rows are retained; otherwise, they are eliminated.

Parentheses are used in a union of three or more query specifications to enforce precedence in eliminating duplicate rows in the unioned sets. For example, a union of the three query-specifications X, Y, and Z, must be written as:

(X UNION Y) UNION Z

or

X UNION (Y UNION Z)

For more information, see (p. 94).

[ORDER BY sort-specification]

The sort-specification syntax is:

{columnspec|n} [ASC|DESC][,{columnspec|n} [ASC|DESC]]...]

The columnspec must identify a column of the project-list. The default sort order is ascending.

The integer refers to the position of the column in the project-list. In the following example, the integer 2 refers to the derived column of averages.

```
> SQL DECLARE Y CURSOR &
>   FOR SELECT SP.PNO, AVG(SP.QTY) &
>   FROM SP &
>   GROUP BY SP.PNO &
>   ORDER BY 2
```

If the cursor definition involves a UNION, the sort specification may refer to column names if the corresponding column names of each query specification are identical; otherwise, the sort specification must reference an integer.

The Scope of a Cursor

A cursor is valid until a BUILD or CANCEL statement is encountered.

Example

The code from the initial SELECT up to and including the GROUP BY option is known as a query-specification.

```
> SQL IN EMPLOYEESDATABASE &
>   DECLARE EMPSKILLS CURSOR FOR &
>   SELECT EMPLOYEES.ID, EMPLOYEES.FIRSTNAME, &
>   EMPLOYEES.LASTNAME, S.SKILL, &
>   FROM EMPLOYEES, SKILLS S &
>   WHERE EMPLOYEES.ID = S.ID &
>   AND EMPLOYEES.ID IN &
>   (SELECT ID FROM SELECTEDEMPLOYEES)
```

For more information about using cursors, see the section, "SQL Overview", in Chapter 1, "About PowerHouse 4GL and Relational Databases", in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

[SQL] DECLARE CURSOR (stored procedure)

Calls a stored procedure or stored function from the specified database.

Syntax

```
    [SQL [IN database]]
DECLARE name CURSOR FOR
CALL stored-procedure|stored-function
    [([ITEM] item [IN[OUT]]|[OUT]
        [,[ITEM] item [IN[OUT]]|[OUT]]...)]
    [ON ERROR CONTINUE|TERMINATE]
    [RETURNING return-parameter]
    [[RESULT] SET item [,item]...]
```

IN database

Specifies the database against which the stored procedure is executed.

Limit: Stored procedure calls are valid for DB2, ODBC, Oracle, Oracle Rdb (declared as TYPE RDB in the dictionary), and Sybase databases. Stored function calls are valid only for Oracle databases.

DECLARE name

Defines a logical name used to identify the set of data resulting from the stored procedure.

CALL stored-procedure|stored-function

The name of a stored procedure or stored function in the database.

The syntax for a procedure name varies with the RDBMS. For information on a specific database system, see "Stored Procedures" in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

([ITEM] item [IN[OUT]]|[OUT] [,[ITEM] item [IN[OUT]|[OUT]]...)

Items that are passed to the stored procedure or Oracle stored function.

Limit: Input parameters can only be record items or columns. Blob items may also be used for input parameters when calling an Oracle stored procedure or stored function.

IN

Specifies that the item is an input parameter.

IN OUT

Ignored in QUIZ.

OUT

Ignored in QUIZ.

Default: IN

ON ERROR CONTINUE|TERMINATE

Specifies the action to be taken if an SQL statement fails. If the TERMINATE option is in effect, processing ends. If CONTINUE is specified, the SQL error is ignored and the processing continues as if the error had not occurred.

Default: TERMINATE

RETURNING return-parameter

Ignored in QUIZ.

[RESULT] SET item [,item]...

The description of the result set returned from the stored procedure. Each item is defined using a name, datatype, and, optionally, its size.

Limit: This statement is valid only for DB2, ODBC, Oracle, and Sybase databases.

name sql-datatype [(n)]

The syntax must use the stored procedure datatypes specified in the first column of the following table. The Sybase equivalents are shown in the second column.

Limit: This option is valid only for DB2, ODBC, Oracle, and Sybase. Only one result set can be returned from a stored procedure.

Discussion

Stored procedures and stored functions are collections of SQL statements and logic that are stored in a database. Calls to stored procedures can take input parameters from a calling program, and return values for output parameters to a calling program. A stored procedure in DB2, ODBC, Oracle, or Sybase may also return result sets. PowerHouse 4GL supports a single result set per execution of a stored procedure.

For information on stored procedures of specific database systems, see "Stored Procedures" in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

Examples

In the following example, the DECLARE CURSOR statement declares a cursor, EMPSKILLS, for the stored procedure, SPEMPLOYEESKILLS, that returns a result set consisting of five items (ID, FIRSTNAME, LASTNAME, SKILL, and SKILLLEVEL).

```
> SQL IN EMPLOYEESDATABASE &
> DECLARE EMPSKILLS CURSOR FOR &
>   CALL SPEMPLOYEESKILLS(EMPLOYEEID IN) &
>   RESULT SET ID DECIMAL, &
>        FIRSTNAME VARCHAR(20), &
>        LASTNAME VARCHAR(20), &
>        SKILL CHARACTER(10), &
>        SKILLLEVEL FLOAT
```

DEFINE

Assigns a name to an expression or prompts for values at execution-time.

Syntax

DEFINE name [type[*n] [type-option]] [format-option] =
{conditional-expression|case-processing|parm-processing}

name

Names the defined item.

Limit: 64 characters; must begin with a letter. You cannot give a defined item the same name as a record item in a record-structure in the ACCESS statement list.

type[*n]

Establishes the physical format of the defined item.

type

Specifies the datatype for the defined item.

For more information about items, datatypes, and sizes, see Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

*n

Specifies the size of the defined item.

For character items, *n specifies the maximum number of alphanumeric characters that the item can contain. For all datatypes that store numeric values, *n specifies the maximum number of digits that the numeric item can display.

Limit: You can't specify a size (*n) when you define a date datatype (DATE, DATETIME, INTERVAL, PHDATE, JDATE, VMSDATE, or ZDATE). The size of a DATE item is controlled strictly by the CENTURY INCLUDED | EXCLUDED option.

Default: NUMERIC*6

type-option

An option that assigns special attributes to numeric or date items.

The type-options are CENTURY, NUMERIC, SIGNED, UNSIGNED, and SIZE.

CENTURY INCLUDED|EXCLUDED

Specifies whether the year component of a date item includes a century prefix.

Limit: Valid only for DATE, JDATE, and PHDATE datatypes; must immediately follow the type in the statement.

Default: Determined by the dictionary.

NUMERIC

Indicates that the datatype is to have a type of ZONED NUMERIC rather than RIGHT OVERPUNCHED NUMERIC.

Limit: Valid only for ZONED datatypes.

SIGNED|{UNSIGNED [WHEN POSITIVE]}

Sets the sign for INTEGER, PACKED, and ZONED. For datatypes PACKED and ZONED, unsigned items can store both positive and negative values. For datatype INTEGER, an unsigned item can store positive values only.

The following is a list of qualifications and exceptions for this option:

- This option is valid only for types INTEGER, PACKED, and ZONED and must immediately follow the type specification.
- For INTEGER, this option specifies whether PowerHouse 4GL interprets the number as a two's complement binary number (SIGNED) or as an absolute binary number (UNSIGNED).
- For PACKED and ZONED, this option specifies whether the item includes a sign (SIGNED) if positive or negative, or only when negative (UNSIGNED).
- The WHEN POSITIVE option is valid only for PACKED UNSIGNED or ZONED UNSIGNED, and is used only for documentation.

Default: UNSIGNED for ZONED; SIGNED for INTEGER and PACKED.

SIZE m [BYTES]

Specifies a storage size in bytes. Use SIZE m BYTES if specifying *n leads to an undesired default storage size.

If you use both *n and SIZE m, you must ensure that they don't conflict with each other.

BYTES is used only for documentation.

Limit: SIZE isn't valid for date datatypes (DATE, DATETIME, INTERVAL, PHDATE, JDATE, VMSDATE, or ZDATE) or the item types NUMERIC and INTERVAL.

conditional-expression

A means of evaluating a series of expressions based on conditions. A conditional-expression, when evaluated, results in the value of the defined item. The expression is calculated once when the record complex is read. The general form is:

expression [IF condition1
[ELSE expression IF condition2]...
[ELSE expression]]

When the IF option is used alone without ELSE and the condition isn't met, then numeric and date defined items are set to zero and character defined items are set to spaces.

For more information about conditional expressions, see Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Limit: Conditional-expressions, case-processing, and parm-processing can't be used together in any combination.

case-processing

Compares the value of an item against a known value or series of values and performs actions based on the outcome of the comparison. The comparison is calculated once for every record complex when the data to be evaluated is available. If there is a match, the resulting value is assigned to the defined item. If there is no match, the specified default is assigned. If no default is specified, zeros or spaces are assigned. The general form is:

CASE [OF] item
WHEN value-set|EXISTS|NULL|MISSING
{THEN|:} value|NULL|MISSING
[WHEN value-set|EXISTS|NULL|MISSING
{THEN|:} value|NULL|MISSING]...
[DEFAULT value|NULL|MISSING]

When the defined item value is calculated based on the value of only one item, and those values are known, case-processing is more efficient than a conditional expression.

The comparison is calculated once for every record complex when the data to be evaluated is available.

OF is used only for documentation. A colon (:) can be substituted for the THEN keyword.

value-set

Specifies a value, a series of values, or a range of values. The general form is:

value [TO value][[,] value[TO value]]...

The values assigned to the defined item by case must be of the same type as the defined item. For example, if you create the defined item PROJECTNAME and specify that it is a character-type item, you must assign a string to the item PROJECTNAME:

```
> DEFINE PROJECTNAME CHARACTER*20 = &
>   CASE OF PROJECTCODE &
>     WHEN 1001 THEN "PRODUCTION" &
>     WHEN 1002 THEN "PROMOTIONS"
```

Limit: Case-processing, conditional-expressions, and parm-processing can't be used together in any combination.

parm-processing

Prompts for defined item values at execution-time. The general form is:

PARM [parm-option]...

Limit: Only one value per input line can be entered. Parm-processing, case-processing, and conditional-expressions can't be used together in any combination:

parm-options		
DOWNSHIFT|UPSHIFT	FORCE|NOFORCE	FORMAT
ON ERRORS	PROMPT	SEPARATOR

DOWNSHIFT|UPSHIFT

Shifts alphabetic letters to either lowercase or uppercase.

Limit: Valid for alphabetic letters only.

FORCE|NOFORCE CENTURY

FORCE CENTURY specifies that the user must enter a century on all century-included date fields. The option applies to century-included dates with two or four-digit year formats.

Default: The default depends on what is specified for the same option of the SYSTEM OPTIONS statement in PDL.

Limit: Valid only for century-included dates.

FORMAT date-format

Specifies the format for entering date item values. Date values can be entered either with or without separator characters. A date-format can be one of the following:

Date-format	Example	Date-format	Example
YYMMDD	01/05/23	YYMMMDD	01/MAY/23
YYYYMMDD	2001/05/23	YYYYMMMDD	2001/MAY/23
YYMM	01/05	YYMMM	01/MAY
YYYYMM	2001/05	YYYYMMM	2001/MAY
YYDDD	01/125	YYYYDDD	2001/125
MMDDYY	05/23/01	MMMDDYY	MAY/23/01
MMDDYYYY	05/23/2001	MMMDDYYYY	MAY/23/2001
MMYY	05/01	MMMYY	MAY/01
MMYYYY	05/2001	MMMYYYY	MAY/2001
MMDD	05/23	MMMDD	MAY/23

Date-format	Example	Date-format	Example
DDMMYY	23/05/01	DDMMMYY	23/MAY/01
DDMMYYYY	23/05/2001	DDMMMYYYY	23/MAY/2001
DDMM	23/05	DDMMM	23/MAY
DDDYY	125/01	DDDYYYY	125/2001

YYYY - four digit year (e.g., 2001)

MM - two digit month (e.g., 05)

MMM - three character month name (e.g., MAY)

DD - two digit day for a month (e.g., 23)

DDD - three digit day for a year (e.g., 365)

Regardless of the output order of the date, the internal working format is YYMMDD (for dates without centuries), YYYYMMDD (for dates with centuries).

The FORMAT option governs data entry by determining the way you can enter date values. Dates can always be entered in the format specified in the FORMAT option, with or without the established separator character and with either the MM or MMM month format.

If the FORMAT option is used but the SEPARATOR option isn't, the only separator character that QUIZ accepts is the separator character specified by System Options, or if it isn't specified, a slash (/).

If a two-digit year is specified in the date format, applications won't accept a four-digit year. A two-digit year is represented by YY (for example, 01).

If a four-digit year is specified in the date format, you can only enter a two-digit year if you enter a separator character between the year and any adjacent numeric component of the date. The default century is added automatically.

Single-digit day and month entries are accepted if the user enters the separator character, as in 4/8/2001. An entry of 4AUG2001 is also allowed, because PowerHouse 4GL accepts a single-digit day entry if the middle value is a three-character month.

A three-digit day of the year from 1 to 366 is represented by DDD.

Although values for date items can be entered in a variety of formats, the values are always stored in either YYMMDD or YYYYMMDD form.

Limit: Valid only for date items. This option only affects the entry format; the display format isn't affected.

Default: If the DATE PARM INHERIT system option is set, the date format is taken from the DATE FORMAT system option. If the DATE FORMAT system option is not set, then YYYYMMDD is used for eight-digit dates and YYMMDD is used for six-digit dates. If DATE PARM INHERIT is not set, then YYYYMMDD is used for eight-digit dates and YYMMDD is used for six-digit dates.

ON ERRORS REPROMPT [n [TIMES]]

Specifies how many times QUIZ reprompts if the entered values fail editing. The report terminates if edit errors aren't resolved within the specified number (n) of reprompts. When operating QUIZ in batch mode, REPROMPT is ignored because QUIZ prompts only once.

TIMES is used only for documentation.

Limit: 99 reprompts for interactive sessions.

Default: Without a specification, QUIZ reprompts indefinitely.

PROMPT string

Displays a prompting message at execution time.

Default: The name of the defined item.

SEPARATOR char

Specifies the character that overrides the separator character specified for dates, by default, the System Options separator or slash (/). The separator character separates the day, month, and year portions of a date item when it is entered.

If the SEPARATOR option is used but the FORMAT option isn't used, then the specified separator character is used with the default date format. Date values can be entered either with or without separator characters.

Limit: Valid only for date items.

Default: The date separator is taken from the DATE SEPARATOR system option. If the DATE SEPARATOR system option is not set, then a slash (/) is used.

Format-Option

Format-Option		
BWZ|NOBWZ	FILL	FLOAT
FORMAT	HEADING	LEADING
NULLSEPARATOR	OUTPUT	PICTURE
SEPARATOR	SIGNIFICANCE	TRAILING

BWZ|NOBWZ

If BWZ (blank when zero) is specified and the item value equals zero, the value is displayed as a blank. NOBWZ displays one or more zeros, depending on the significance of the item.

Limit: Valid for numeric values only.

Default: NOBWZ

FILL char

Specifies the character used to fill unused space to the left of the most significant digit, float character, or leading sign in the picture.

The fill character also replaces any unnecessary leading nonsubstitution characters, including commas and leading spaces.

For example, the following attributes work together to display a dollar amount that is preceded by leading asterisks:

Fill	Float	Picture	Value	Display
*	$	"^,^^^,^^^.^^"	123456	***$1,234.56

Limit: Valid for numeric values only.

Default: A space.

FLOAT char

Specifies the float character. The float character is inserted immediately to the left of the most significant digit. To ensure that there is enough room for the float character, add either a space or an extra substitution character (^) to the left side of the picture.

For example, the following element attributes work together to display a dollar sign in front of an item value:

Stored value	Picture	Float	Display
1234	"^^^^^"	$	$1234
56789	" ^^^.^^"	$	$567.89

Limit: Valid for numeric values only.

Default: A space.

FORMAT date-format

FORMAT date-format

Specifies the format for displaying date item values. A date-format can be one of the following:

Date-format	Example	Date-format	Example
YYMMDD	01/05/23	YYMMMDD	01/MAY/23
YYYYMMDD	2001/05/23	YYYYMMMDD	2001/MAY/23
YYMM	01/05	YYMMM	01/MAY
YYYYMM	2001/05	YYYYMMM	2001/MAY
YYDDD	01/125	YYYYDDD	2001/125
MMDDYY	05/23/01	MMMDDYY	MAY/23/01
MMDDYYYY	05/23/2001	MMMDDYYYY	MAY/23/2001
MMYY	05/01	MMMYY	MAY/01
MMYYYY	05/2001	MMMYYYY	MAY/2001
MMDD	05/23	MMMDD	MAY/23
DDMMYY	23/05/01	DDMMMYY	23/MAY/01
DDMMYYYY	23/05/2001	DDMMMYYYY	23/MAY/2001
DDMM	23/05	DDMMM	23/MAY
DDDYY	125/01	DDDYYYY	125/2001

YYYY - four digit year (e.g., 2001)

MM - two digit month (e.g., 05)

MMM - three character month name (e.g., MAY)

DD - two digit day for a month (e.g., 23)

DDD - three digit day for a year (e.g., 365)

Regardless of the output order of the date, the internal working format is YYMMDD (for dates without centuries), YYYYMMDD (for dates with centuries).

Limit: Valid only for date items. This option only affects the display format; the entry format isn't affected.

Default: If DATE FORMAT is not specified, the system option is used. If the system option is not specified, the default is YYYYMMDD.

HEADING string

Provides a default column heading for QUIZ reports. The HEADING option customizes column headings for any report-items or defined items included in the REPORT statement.

To produce multi-line headings, embed a multi-line heading character (^) in the heading string. For example, the heading "Employee^Number" displays a two-line heading in a QUIZ report:

```
Employee
Number
```

If you do not specify a multiline heading, when the column heading is much longer than the column data width, PowerHouse 4GL wraps the heading automatically. PDC dictionaries determine the heading at dictionary compile time; PHD dictionaries determine the heading at component runtime. The algorithms used to determine where to split the text and the resultant column width are different for PDC and PHD dictionaries. If you are switching between dictionary types, you may have to reformat your reports or specify your own wrapped headings.

Limit: 60 characters per string

Default: Based on the defined item name and size.

LEADING [SIGN] char

Determines the character that indicates a negative number. The leading sign is displayed to the left of the most significant digit and the float character. To ensure that there is enough room for the leading sign, add sufficient substitution characters (^) or nonsubstitution characters to the picture.

If the picture is too small, overflow occurs and QUIZ displays crosshatches (#).

SIGN is used only for documentation.

Limit: Valid for numeric values only.

Default: A negative sign (-)

NULLSEPARATOR|NONULLSEPARATOR

NULLSEPARATOR specifies that all dates are to be displayed without a separator. This allows display of century-included dates in the same space as century-excluded dates.

The DATE SEPARATOR is used for display formatting if NULLSEPARATOR is not used, or is canceled by the NONULLSEPARATOR option.

The DATE SEPARATOR may be used during input. If NULLSEPARATOR is specified, the value is redisplayed after formatting without the separator.

Default: NONULLSEPARATOR

OUTPUT [SCALE] n

Establishes the output scaling factor. Before it's displayed, the stored value of the item is multiplied by 10 and raised to the power of the output scale value (that is, 10^n). The result is rounded after scaling. The OUTPUT SCALE option is needed for floating point numbers, since fractional portions of stored values are truncated for display. With an output scale of zero, all digits to the right of the decimal in a floating point number are eliminated when the number is truncated.

SCALE is used only for documentation.

For example, the following item attributes work with the output scale to display floating point values:

Stored value	Picture	Output scale	Display
12.54	"^^^^.^^"	0	0.13
12.54	"^^^^.^^"	2	12.54

Use the OUTPUT SCALE option to display the decimal portion of the number:

Stored value	Picture	Output scale	Display
123.456	"^^^^^^"	0	123
123.456	"^^^^^^"	1	1235
123.456	"^^^^^^"	2	12346
123.456	"^^^^^^"	3	123456
123.456	"^^^^^^"	-1	12
123.456	"^^^^^^"	-2	1
123.456	"^^^^^^"	-3	0

Use the OUTPUT SCALE and PICTURE options for proper decimal alignment on output:

Stored value	Picture	Output scale	Display
123.456	"^^^.^"	1	123.5
123.456	"^^^.^^"	2	123.46
123.456	"^^^.^^^"	3	123.456

Limit: -16 to 16. Valid only for numeric items.

Default: 0

PICTURE string

Establishes the output picture used to format the item value for display. A picture string is made up of substitution characters (^) and nonsubstitution characters.

Character items are formatted in the following way:

1. The item is processed from left to right, substituting one character from the item for each substitution character in the picture. Nonsubstitution characters remain unchanged.
2. If there are fewer substitution characters in the picture than characters in the item value, the remaining characters in the item aren't displayed.
3. If there are more substitution characters in the picture than characters in the item value, spaces are padded to the right of the item.

As an example, the item value "FHSMITH" is formatted as follows:

Picture	Display
"^^^^^^^"	FHSMITH
"^^^^^"	FHSMI
"^.^. ^^^^^"	F.H. SMITH
"^.^. ^^^^^^^^^^"	F.H. SMITH

Numeric items are formatted in the following way:

1. The item is scaled by the output scale and rounded to the nearest whole number.
2. The result of Step 1 is processed from right to left, substituting one digit from the item for each substitution character in the picture until all significant (non-zero) digits have been processed. Nonsubstitution characters remain unchanged.

3. Until the element significance is reached, leading zeros are substituted for each substitution character. Nonsubstitution characters remain unchanged.
4. The float character is added.
5. Leading and trailing signs are added for negative values.
6. The remaining portion of the picture is filled with the fill character.
7. If there isn't enough room in the picture to hold the significant digits of the item value or the leading or trailing sign, the item is filled with the overflow character (#).

For example, the value 1578 is formatted as follows:

Picture	Display
"^^^"	####
"^^^^"	1578
"^^.^^"	15.78
"^^,^^^,^^^"	1,578

Specifying a PICTURE in QUIZ causes any SIGNIFICANCE specified in the dictionary to be ignored. A new default significance is calculated based on the specified PICTURE. To change both the PICTURE and SIGNIFICANCE, specify the PICTURE first followed by the new SIGNIFICANCE.

Limit: 60 characters per string for character items, 30 for numeric items. Not valid for date items. If the PICTURE and SIGNIFICANCE options are used together, specify PICTURE first.

SEPARATOR char

Specifies the character that overrides the separator character specified for dates, by default, the System Options separator or slash (/). The separator character separates the day, month, and year portions of a date element when it's displayed.

If the SEPARATOR option is used but the FORMAT option isn't used, then the specified separator character is used with the default date format.

Limit: Valid for date element values only.

Default: A slash (/), unless otherwise specified by the SEPARATOR option of the SYSTEM OPTIONS statement in PDL.

SIGNIFICANCE n

Specifies the minimum number of digits and characters displayed. SIGNIFICANCE forces the display of leading nonsubstitution characters and leading zeros.

If letters or special characters always appear as part of a defined item's picture, the significance must be large enough to force the display of all desired letters or characters.

For example, the value "1578" is displayed as follows, based on the indicated significance:

Picture	Significance	Display
"^^.^^^^"	6	0.1578
"^^^^.^^"	3	15.78
"^^^^.^^"	7	0015.78
"^^.^^^^ "	4	1568
" P.O.^^^^-^^"	11	P.O.0015-78
"^^.^^^^%"	5	1578%

Limit: 255. Valid for numeric values only. If the PICTURE and SIGNIFICANCE options are used together, specify PICTURE first.

Default: The number of decimal positions plus 1.

TRAILING [SIGN] string

Specifies a string of one or two characters placed to the right of the picture when the value is negative. To ensure that there is enough room for the trailing sign, add sufficient nonsubstitution characters to the rightmost portion of the picture.

SIGN is used only for documentation.

To place parentheses around negative numbers, use the LEADING SIGN and TRAILING SIGN options together. For example, a leading sign of "(" with a trailing sign of ")" displays the value -123.45 as

```
(123.45)
```

For example, the value -1578 is formatted as follows:

Picture	Leading sign	Trailing sign	Display
"^^^,^^^ "	none	CR	1,578CR
" ^^^,^^^ "	(	)	(1,578)

Limit: Valid for numeric values only.

Default: None

Discussion

The DEFINE statement calculates expressions or comparisons that are calculated for each record complex. The expression is calculated once for every record complex as soon as the data required is available.

The purpose of the DEFINE statement is to combine and manipulate data contained in files or to create a new item not contained in a record complex. The DEFINE statement is evaluated as soon as the data required for that evaluation becomes available. The DEFINE statement is re-evaluated based on default values if the record on which it depends is not retrieved.

You can include multiple DEFINE statements with the PARM option, but PowerHouse 4GL accepts only one value for each defined item.

Limit: There can be a maximum of 1023 defined items in a report.

Limit: QUIZ has a 32,767 byte buffer limit that includes all record-structures in the ACCESS statement, all defined items, and space for copies of values used in linkage.

Responding to Prompts

The user has three minutes to respond to any prompt; otherwise, the report is terminated.

A null entry (pressing [Return]) in response to a prompt from a DEFINE statement results in zeros or spaces being assigned to the defined item.

Interrupting Prompting

The user can interrupt prompting by pressing the following in response to a prompt: [Ctrl-C] (**OpenVMS, UNIX, Windows**).

When the user does this, QUIZ issues a message asking if the user wants to continue processing:

- If the user responds "yes", QUIZ restarts processing of the report from the point of interruption.
- If the user responds "no", QUIZ terminates the execution of the current report.

Editing Execution-Time Parameters

You can use the EDIT statement to specify both a pattern and acceptable values for execution-time parameters. When you use the EDIT statement with execution-time parameters, QUIZ checks values as they're entered. If a numeric or date item contains characters, QUIZ issues a message and repeats the prompt.

Default: If an editing error occurs, QUIZ reprompts users indefinitely unless the ERRORS REPROMPT option is specified. If editing errors aren't resolved within the specified number of reprompts, the QUIZ report is terminated.

Example

This report prints a list of suppliers offering a specified part. The example uses the DEFINE statement to allow execution-time entry of the part numbers. Other DEFINE statements allow the report user to enter the quantity of each part needed and customized report titles. In this example:

- PARM PROMPT prompts the report user to enter the part number and the quantity needed, and assigns these values to REQPRT and REQQTY.
- NUMERIC * 6 indicates that the type must be specified as a NUMERIC and that the maximum size of REQQTY is six digits.
- using the value of the item REQPRT, another DEFINE statement is used to produce a report heading that includes the part number in the item RPTHDR. A similar technique is used for the report footer.

```
> ACCESS PARTS &
>   LINK TO PARTSUPPLIERS &
>   LINK TO SUPPLIERS
>
> DEFINE REQPRT NUMERIC*4 = &
>   PARM PROMPT "Enter a part number: " &
>   ON ERRORS REPROMPT 3 TIMES
>
> DEFINE REQQTY NUMERIC*6 = &
>   PARM PROMPT &
>   "Enter the quantity required: "
>
> EDIT REQPRT VALUES 1000 TO 2999
>
> SELECT PARTSUPPLIERS IF PARTNUMBER = REQPRT
> SELECT PARTS IF QOH GE REQQTY
> SORT &
>  ON SUPPLIERKEY OF PARTSUPPLIERS &
>  ON PARTVARIANT OF PARTS
>
> DEFINE RPTHDR CHARACTER*40 = &
>   "SUPPLIER LIST FOR PART " + &
>   ASCII(PARTNUMBER OF PARTS)
>
> DEFINE RPTFTR CHARACTER*40 = &
>   "SUMMARY REPORT FOR PART " + &
>   ASCII(PARTNUMBER OF PARTS)
>
> PAGE HEADING &
>   TAB 22 RPTHDR &
>   KEEP COLUMN HEADINGS &
>   SKIP 2
>
> REPORT &
>   SUPPLIERNAME OF SUPPLIERS &
>   PRINT AT SUPPLIERKEY OF PARTSUPPLIERS &
>   PARTVARIANT OF PARTS &
>   QOH OF PARTS &
>   UNITCOST OF PARTS
>
> FOOTING AT &
>   TAB 22 RPTFTR &
```

```
>   SUPPLIERKEY OF PARTSUPPLIERS &
>   SKIP 2
>
> GO
```

DISPLAY

Displays a message.

Syntax

DISPLAY string

string

Specifies the message to be displayed.

Limits: 256 characters per string.

Discussion

The DISPLAY statement creates messages to be displayed at parse time. To display messages at execution-time, include them in the DISPLAY option of the BUILD statement.

To display only the string (rather than the DISPLAY statement itself), include the NOLIST option of the USE statement when you use the source statements that contain the DISPLAY statements.

EDIT

Validates entries made in response to a PARM prompt.

Syntax

EDIT item [option]...

item

Names the item with values that were entered in response to a PARM prompt from a CHOOSE or DEFINE statement. The item must pass the editing specifications declared in the data dictionary or in the EDIT statement itself.

The item can be one of the following:

- a defined item that is prompted for in a DEFINE statement
- an item that is prompted for in a CHOOSE statement

Options

The options are DOWNSHIFT, UPSHIFT, PATTERN, and VALUE.

DOWNSHIFT|UPSHIFT

Shifts the entered value of a character item to either lowercase or uppercase. This option is applied before any other editing.

Limit: Valid for character values only; non-alphabetic characters within a character item are not affected.

PATTERN string

Specifies a string of characters and metacharacters that provide a general description of values. To be valid, the entry must match the values specified in the pattern string.

For information about pattern matching, see Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Limit: 60 characters per string

VALUE value-set

Specifies one or more values and/or one or more ranges of values. The general form is:

value [TO value][[,] value [TO value]]...

A value is a single string or a single number depending on the item type.

Limit: Century-included date values must be in YYYYMMDD format. Century-excluded date values must be in YYMMDD format. Commas must separate value sets.

Discussion

The EDIT statement validates values that have been entered in response to a DEFINE or CHOOSE statement prompt.

If an execution-time parameter in a CHOOSE statement is edited with an EDIT statement, a subsequent CHOOSE statement cancels both the EDIT statement and the previous CHOOSE statement, even if the item is the same in both CHOOSE statements.

Data Dictionary Editing Specifications

Data dictionary editing specifications can be used for index values that are entered with the CHOOSE statement. An option specified in an EDIT statement overrides an identical type of edit from the data dictionary, although other editing from the data dictionary still applies. Only one EDIT statement is allowed for each item. Dictionary-based editing doesn't apply to defined items.

Validating Execution-Time Parameters

You can use the EDIT statement to specify both a pattern and acceptable values for execution-time parameters. When you use the EDIT statement with execution-time parameters, QUIZ checks values as they're entered. If a numeric or date item contains characters, QUIZ issues a message and repeats the prompt.

If an editing error occurs, QUIZ reprompts you indefinitely unless you specified the ON ERRORS REPROMPT option. If editing errors aren't resolved within the specified number of reprompts, then the QUIZ report is terminated.

Example

This example prints the address for a specified customer. The EDIT statement checks to make sure that the entered customer account number is valid (between 1000 and 3000):

```
> ACCESS CUSTOMERS
> CHOOSE ACCOUNTNUMBER &
>   PARM PROMPT  &
>   "Enter an account number in the range 1000 to 3000: "
>
> EDIT ACCOUNTNUMBER VALUES 1000 TO 3000
>
> REPORT &
>   CUSTOMERNAME &
>   STREET &
>   CITY &
>   PROVSTAT &
>   POSTALZIP &
>   ACCOUNTNUMBER
```

If you enter an account number outside of these parameters, the following error message appears:

```
Item has failed values edit.
```

EXECUTE

Executes a compiled QUIZ report.

Syntax

EXECUTE filespec [GO|NOGO]

filespec

Specifies the file that contains the compiled QUIZ report that you want to execute.

GO|NOGO

GO instructs QUIZ to load the compiled report and begin execution immediately.

NOGO instructs QUIZ to load the compiled report and prompt for additional statements that may modify the report. The actual report is then executed when the GO statement is entered.

You can modify a compiled report at execution-time by specifying the NOGO option. This lets you specify additional selection conditions or change options used when the report was compiled, as in

```
> EXECUTE LABELS NOGO
> AND SELECT IF BRANCH = "OT"
> SET REPORT COPIES 2
> GO
```

Default: GO

Discussion

The EXECUTE statement instructs QUIZ to load and execute the named compiled QUIZ report.

You can't use the EXECUTE statement to run uncompiled reports.

The statement

```
> EXECUTE LABELS
```

loads the compiled report named LABELS. When you execute a compiled report, QUIZ reloads the internal tables and produces the report. The existing information in the internal tables is replaced. Any previously entered SET JOB statement is still in effect.

The compiled report feature of QUIZ eliminates the parsing process.

For more information about how QUIZ accesses compiled reports, see the section, "Locating Files", in Chapter 1, "Running PowerHouse 4GL", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

The **procloc** parameter affects how PowerHouse 4GL uses unqualified file names that are specified in the EXECUTE statement. For more information about the **procloc** program parameter, see Chapter 2, "Program Parameters", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Examples

The first example builds the compiled report and the second example uses the EXECUTE statement to run the compiled report called SUPPLIST. SUPPLIST generates a catalog of parts, sorted by suppliers. The source statements for SUPPLIST are included as an aid to understanding the example:

```
> ACCESS PARTS &
>   LINK TO PARTSUPPLIERS &
>   LINK TO SUPPLIERS
>
> SORT &
>   ON SUPPLIERKEY OF PARTSUPPLIERS &
>   ON PARTNUMBER OF PARTS &
```

```
>   ON PARTVARIANT OF PARTS
>
> REPORT &
>   SUPPLIERNAME OF SUPPLIERS &
>   PRINT AT SUPPLIERKEY OF PARTSUPPLIERS &
>   PARTNUMBER OF PARTS &
>   PRINT AT PARTNUMBER OF PARTS &
>   PARTVARIANT OF PARTS &
>   QOH OF PARTS &
>   UNITCOST OF PARTS
>
> FOOTING AT &
>   SUPPLIERKEY OF PARTSUPPLIERS &
>   SKIP 2
>
> BUILD SUPPLIST &
>   DISPLAY "Parts Catalog being generated..."
```

Each time the compiled report SUPPLIST is executed, it generates the same report. However, you can modify the way that the report works at execution-time by executing SUPPLIST with the NOGO option. NOGO allows the report user to enter statements in order to establish additional conditions.

```
> EXECUTE SUPPLIST NOGO
>
> CHOOSE PARTNUMBER 1000 TO 1100
>
> GO
```

The following lines of code report suppliers for a specific part number selected at the execution-time prompt. QUIZ looks for a compiled report in a file named SUPPLIST.

```
> EXECUTE SUPPLIST NOGO
>
> DEFINE PARTREPORT &
>   NUMERIC*4 = &
>   PARM &
>   PROMPT "Enter a part number: "
>
> SELECT &
>   IF PARTREPORT = PARTNUMBER
>
> GO
```

The SELECT statement compares the value against the PARTNUMBER item of the PARTS file, and reports the suppliers that provide that part.

EXIT

Ends a QUIZ session.

Syntax

EXIT

Discussion

The EXIT statement ends the QUIZ session and returns control to the operating system or to the invoking program.

The EXIT statement can be abbreviated to E, EX, or EXI.

The EXIT statement and the QUIT statement perform in the exact same manner.

FINAL FOOTING

Sets the content and format of the footing at the end of the report.

Syntax

FINAL FOOTING [ALIGN|NOALIGN] [report-group]

ALIGN|NOALIGN

ALIGN aligns the items in the FINAL FOOTING statement with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the footing that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

A summary operation on the FINAL FOOTING aligns with a like summary operation of the same item in the REPORT statement. If a matching item and a summary operation doesn't exist in the REPORT statement, the alignment is attempted with a matching non-summary item in the REPORT statement. If neither is found, no alignment is done.

NOALIGN does not align items. The NOALIGN option overrides SET REPORT ALIGN.

Default: NOALIGN

report-group

Establishes the report-group to be printed at the conclusion of the report. If no report-group is included, any previous FINAL FOOTING statement is canceled.

For more information about report-groups, see (p. 101).

Discussion

The FINAL FOOTING statement prints a one-time footing that appears on the last page of the report. It is the last thing that's printed in a QUIZ report. The final footing can include summaries, item values, or anything else that can be part of a report-group. If a final footing is printed alone on the last page of a report, the page headings and footings are omitted from that page. Also, there is no page footing on a page with a final footing. If you use the NOREPORT statement, then the final footing is not displayed if no records are reported.

Example

This report uses FINAL FOOTING to show the total amount owing from all customers. SKIP 2 leaves a blank line between the final footing and the customer name footing, to make the report easier to read.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
>   LINK TO PARTS
>
> SORT &
>   ON CUSTOMERNAME &
>   ON INVOICEDATE D
>
> DEFINE INVOICEAMOUNT &
>   NUMERIC * 9 &
>   PICTURE " ^,^^^,^^^.^^" = &
>   QUANTITYSHIPPED OF INVOICEDETAIL * &
>   (UNITCOST OF PARTS * UNITMARKUP OF PARTS)
>
> REPORT &
>   CUSTOMERNAME PRINT AT CUSTOMERNAME &
```

```
>   INVOICENUMBER PRINT AT CUSTOMERNAME &
>   INVOICEDATE PRINT AT CUSTOMERNAME &
>   INVOICEAMOUNT HEADING "Amount^Owing"
>
> FOOTING AT CUSTOMERNAME &
>   SKIP 2 &
>   "Total:" &
>   TAB 46 &
>   INVOICEAMOUNT SUBTOTAL &
>   SKIP 2
>
> FINAL FOOTING &
>   SKIP 2 &
>   "Grand Total:" &
>   TAB 46 &
>   INVOICEAMOUNT SUBTOTAL
>
> GO
```

The resulting QUIZ report looks like this:

```
1991-06-17                  FUTURE INDUSTRIES Inc.                    PAGE   1

  Customer                      Invoice     Invoice       Amount
    Name                        Number       Date         Owing

  SAMSON RESORTS                 0105     1990-03-15       3,417.84
                                                            8,544.60
                                                            6,835.68
                                                            2,563.38
                                                            3,232.00
                                                            8,080.00
                                                            6,464.00
                                                            2,424.00
                                                            2,939.10
                                                            7,347.75
                                                            5,878.20
                                                            2,204.33

  Total:                                                   59,930.88

  YAKASUMA DISTRIBUTORS          0100     1991-03-19       17,089.20
                                                           12,816.90
                                                           17,089.20
?
```

The grand total is calculated by the FINAL FOOTING statement.

```
2001/07/25                  FUTURE INDUSTRIES Inc.                    PAGE   2

  Customer                      Invoice     Invoice       Amount
    Name                        Number       Date         Owing

  YAKASUMA DISTRIBUTORS          4        1993-02-03          30.00
                                                             35.00
                                                          1,002.00

  Total:                                                100,323.30

  Grand Total:                                            8,749.00
```

Without the ALIGN Option

In this example, the "TOTAL BILLINGS" label is aligned to the left-hand side of the report and the total billings follows. The default is NOALIGN.

```
> ACCESS BILLINGS
> REPORT TAB 20 PROJECT TAB 40 BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT SKIP 2 "BILLING SUBTOTAL:" &
>    BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2 &
>   "TOTAL BILLINGS:" BILLING SUBTOTAL &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   5

                  Project             Billing

                  P000010               945.00
                  P000010               105.00

  BILLING SUBTOTAL:    3,940.00

  TOTAL BILLINGS:        34,562.50

?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the billings total is aligned with the billing on the REPORT statement.

```
> ACCESS BILLINGS
> REPORT TAB 20 PROJECT TAB 40 BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT NOALIGN SKIP 2 "BILLING SUBTOTAL:" &
>    BILLING SUBTOTAL SKIP 2
> FINAL FOOTING ALIGN SKIP 2 &
>     "TOTAL BILLINGS:" BILLING SUBTOTAL &
>     PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   5

                  Project             Billing

                  P000010               945.00
                  P000010               105.00

  BILLING SUBTOTAL:    3,940.00

  TOTAL BILLINGS:                    34,562.50

?
```

Aligning Summary Operations

In this example, the BILLING SUBTOTAL on the final footing aligns with the BILLING SUBTOTAL in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> FINAL FOOTING ALIGN BILLING SUBTOTAL
```

In the following example, there is no matching summary operation and item in the REPORT statement. Therefore, the BILLING MAXIMUM on the final footing aligns with the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
```

```
> REPORT BILLING BILLING SUBTOTAL
> FINAL FOOTING ALIGN BILLING MAXIMUM
```

In the next example, the two BILLING items will not align because BILLING SUBTOTAL on the final footing is aligned with BILLING SUBTOTAL in the REPORT statement. The print position for the BILLING item of the final footing is already positioned past the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> FINAL FOOTING ALIGN BILLING SUBTOTAL BILLING
```

FOOTING AT

Sets the content and format of control break footings.

Syntax

FOOTING AT sort-item [ALIGN|NOALIGN] [report-group]

sort-item

Identifies the control break associated with this footing. A control break occurs when the value of the sort-item changes. The sort-item must be named in a previous SORT or SORTED statement.

ALIGN|NOALIGN

ALIGN aligns the items in the FOOTING AT statement with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the footing that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

NOALIGN does not align items. The NOALIGN option overrides SET REPORT ALIGN.

Default: NOALIGN

report-group

Establishes the report-group to be printed after the last detail line associated with each sort-item value. If no report-group is included, any previous FOOTING AT statement for this sort-item is canceled.

For more information about report-groups, see (p. 101).

Discussion

The FOOTING AT statement prints a control footing at the end of the specified control break, immediately before any headings or detail lines associated with the next control break.

For more information about sort-items and control breaks, see (p. 156).

Summary Operations

In addition to printing simple footings, the FOOTING AT statement is also useful for presenting summary information. You can use one or more of the following summary operations:

AVERAGE	COUNT	MAXIMUM
MINIMUM	PERCENT	RATIO
SUBTOTAL		

For more information about these summary operations, see (p. 75) and (p. 101).

Example

This report uses FOOTING AT to show the total amount invoiced to each customer. In this example:

- a footing appears each time the value for the sort-item CUSTOMERNAME changes; this allows summary information such as counts, subtotals, and labels to be reported for each customer.
- SKIP 2 leaves a blank line between the report-group and the footing to make the report easier to read.

```
> ACCESS CUSTOMERS &
```

```
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
>   LINK TO PARTS
>
> SORT &
>   ON CUSTOMERNAME &
>   ON INVOICEDATE D
>
> DEFINE INVOICEAMOUNT &
>   NUMERIC * 9 &
>   PICTURE " ^,^^^,^^^.^^" = &
>     QUANTITYSHIPPED OF INVOICEDETAIL * &
>     (UNITCOST OF PARTS * UNITMARKUP OF PARTS)
>
> REPORT &
>   CUSTOMERNAME PRINT AT CUSTOMERNAME &
>   INVOICENUMBER PRINT AT CUSTOMERNAME &
>   INVOICEDATE PRINT AT CUSTOMERNAME &
>   INVOICEAMOUNT HEADING "Amount^Owing"
>
> FOOTING AT CUSTOMERNAME &
>   SKIP 2 &
>   "Total:" &
>   TAB 46 &
>   INVOICEAMOUNT SUBTOTAL &
>   SKIP 2
>
> FINAL FOOTING &
>   "Grand Total:" &
>   TAB 46 &
>   INVOICEAMOUNT SUBTOTAL
>
> GO
```

For a sample of what the resulting QUIZ report looks like, see (p. 69).

Without the ALIGN Option

In this example, the "BILLING SUBTOTAL" labels are aligned to the left-hand side of the report and the billing subtotals follow. The default is NOALIGN.

```
> ACCESS BILLINGS
> REPORT TAB 20 PROJECT TAB 40 BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT SKIP 2 "BILLING SUBTOTAL:" &
>   BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2 &
>   "TOTAL BILLINGS:" BILLING SUBTOTAL  &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   2

                  Project             Billing

                  P000003               400.00
                  P000003               420.00
                  P000003               315.00

  BILLING SUBTOTAL:    1,135.00

                  P000004               200.00
                  P000004               400.00
                  P000004               200.00

  BILLING SUBTOTAL:      800.00

                  P000005               400.00
                  P000005               400.00
                  P000005               400.00
                  P000005               400.00

  BILLING SUBTOTAL:    1,600.00

?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the billing subtotal is aligned with the billing in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT TAB 20 PROJECT TAB 40 BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT ALIGN SKIP 2 "BILLING SUBTOTAL:" &
>   BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2 &
>  "TOTAL BILLINGS:" BILLING SUBTOTAL  &
>  PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   2

                  Project             Billing

                  P000003               400.00
                  P000003               420.00
                  P000003               315.00

  BILLING SUBTOTAL:                   1,135.00

                  P000004               200.00
                  P000004               400.00
                  P000004               200.00

  BILLING SUBTOTAL:                     800.00

                  P000005               400.00
                  P000005               400.00
                  P000005               400.00
                  P000005               400.00

  BILLING SUBTOTAL:                   1,600.00

?
```

Aligning Summary Operations

In this example, the BILLING SUBTOTAL on the footing aligns with the BILLING SUBTOTAL in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT
> REPORT BILLING BILLING SUBTOTAL
> FOOTING AT PROJECT ALIGN BILLING SUBTOTAL
```

In the following example, there is no matching summary operation and item in the REPORT statement. Therefore, the BILLING MAXIMUM on the footing aligns with the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT
> REPORT BILLING BILLING SUBTOTAL
> FOOTING AT PROJECT ALIGN BILLING MAXIMUM
```

In the next example, the two BILLING items will not align because BILLING SUBTOTAL on the footing is aligned with BILLING SUBTOTAL in the REPORT statement. The print position for the BILLING item of the footing is already positioned past the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT
> REPORT BILLING BILLING SUBTOTAL
> FOOTING AT PROJECT ALIGN BILLING SUBTOTAL BILLING
```

GO

Initiates execution of a QUIZ report.

Syntax

GO [OMNIREUSE]

Discussion

The GO statement signifies that the specifications for the current QUIZ report are complete and executes the report.

The **procloc** program parameter affects how PowerHouse 4GL searches for process files in the GO statement. For more information about the **procloc** program parameter, see Chapter 2, "Program Parameters", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

HEADING AT

Sets the content and format of control break headings.

Syntax

HEADING AT sort-item [ALIGN|NOALIGN] [report-group]

sort-item

Specifies a control break which occurs when the value of the sort-item changes. The sort-item is an item named in a SORT or SORTED statement.

ALIGN|NOALIGN

ALIGN aligns the items in the HEADING AT statement with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the heading that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

NOALIGN does not align items. The NOALIGN option overrides SET REPORT ALIGN.

Default: NOALIGN

report-group

Establishes the report-group to be printed before the first detail line associated with each sort-item value. If no report-group is included, any previous HEADING AT statement for this sort-item is canceled.

For more information about report-groups, see (p. 101).

Discussion

The HEADING AT statement prints a report-group at the beginning of the specified control break, immediately after any footings or detail lines associated with the previous control break.

For more information about sort-items and control breaks, see (p. 156).

Summary Operations

In addition to printing simple titles, the HEADING AT statement is also useful for presenting summary information. You can use one or more of the following summary operations:

AVERAGE	COUNT	MAXIMUM
MINIMUM	PERCENT	RATIO
SUBTOTAL		

For more information about these summary operations, see (p. 81) and (p. 101).

Examples

This report uses HEADING AT to show orders for all parts. In the following example

- "Part Number" is reported each time the value of the sort-item PARTNUMBER changes. The new value for PARTNUMBER is then reported.
- SKIP 2 tells QUIZ to skip to the start of the second line following the previous group of report detail lines, leaving one line blank before the heading.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL
```

```
>
> SORT &
>   ON PARTNUMBER &
>   ON CUSTOMERNAME
>
> HEADING AT &
>   PARTNUMBER &
>   SKIP 2 &
>   "Part Number: " &
>   PARTNUMBER &
>   SKIP &
>   "-----------------------------------------"
>
> FOOTING AT CUSTOMERNAME &
>   "Orders for: " &
>   CUSTOMERNAME &
>   TAB 40 QUANTITYORDERED SUBTOTAL
>
> FOOTING AT &
>   PARTNUMBER &
>
>   SKIP &
>   "-----------------------------------------" &
>   "Total Sales for Part: " &
>   PARTNUMBER &
>   TAB 40 QUANTITYORDERED SUBTOTAL
>
> FINAL FOOTING &
>   SKIP 2 &
>   "TOTAL SALES: " &
>   TAB 40 QUANTITYORDERED SUBTOTAL &
>   "UNITS" &
>   RESERVE 2 LINES
>
> GO
```

The resulting QUIZ report looks like this:

```
2001/07/25                  FUTURE INDUSTRIES Inc.

                Customer              Quantityshipped
                  Name                  (SUBTOTAL)

  Part Number:   1802
  ------------------------------------------
  Orders for:   SAMSON RESORTS                 23
  Orders for:   VAKASUMA DISTRIBUTORS           3
                                               26  UNITS

  Part Number:   1803
  ------------------------------------------
  Orders for:   VAKASUMA DISTRIBUTORS          35
                                               35  UNITS

  Part Number:   1804
  ------------------------------------------
  Orders for:   VAKASUMA DISTRIBUTORS          10
                                               10  UNITS
```

Without the ALIGN Option

In this example, the "BILLINGS FOR PROJECT" label is aligned at the left-hand side of the report and the project number follows. The default is NOALIGN.

```
> ACCESS BILLINGS
> SORT ON PROJECT ON EMPLOYEE
> REPORT TAB 30 PROJECT PRINT AT PROJECT  &
>   EMPLOYEE BILLING
> HEADING AT PROJECT  &
>   SKIP PAGE " " SKIP 3 &
```

```
>   "BILLINGS FOR PROJECT " &
>   PROJECT &
>   TAB 70 "PAGE" SYSPAGE SKIP 2
> FOOTING AT PROJECT ALIGN SKIP 2 &
>   "SUBTOTAL FOR PROJECT " &
>   PROJECT BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2  &
>   "TOTAL BILLINGS:"   &
>   BILLING SUBTOTAL    &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

```


  BILLINGS FOR PROJECT   P000001                                  PAGE     1

                         P000001  00001     100.00
                                  00001     945.00
                                  00001   1,207.50
                                  00003     105.00
                                  00003   1,470.00
                                  00004     315.00
                                  00004     105.00
                                  00004     945.00
                                  00004     420.00

  SUBTOTAL FOR PROJECT       P000001         5,612.50




?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the project is aligned with the project in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT ON EMPLOYEE
> REPORT TAB 30 PROJECT PRINT AT PROJECT &
>   EMPLOYEE BILLING
> HEADING AT PROJECT ALIGN &
>   SKIP PAGE " " SKIP 3 &
>   "BILLINGS FOR PROJECT " &
>   PROJECT &
>   TAB 70 "PAGE" SYSPAGE SKIP 2
> FOOTING AT PROJECT ALIGN SKIP 2 &
>   "SUBTOTAL FOR PROJECT " &
>   PROJECT BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2  &
>   "TOTAL BILLINGS:"  &
>   BILLING SUBTOTAL   &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
BILLINGS FOR PROJECT      P000001                              PAGE     1

                          P000001  00001     100.00
                                   00001     945.00
                                   00001   1,207.50
                                   00003     105.00
                                   00003   1,470.00
                                   00004     315.00
                                   00004     105.00
                                   00004     945.00
                                   00004     420.00

SUBTOTAL FOR PROJECT      P000001           5,612.50




?
```

Aligning Summary Operations

In this example, the BILLING SUBTOTAL on the heading aligns with the BILLING SUBTOTAL in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT
> REPORT BILLING BILLING SUBTOTAL
> HEADING AT PROJECT ALIGN BILLING SUBTOTAL
```

In the following example, there is no matching summary operation and item in the REPORT statement. Therefore, the BILLINGS MAXIMUM on the heading aligns with the BILLINGS item in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT
> REPORT BILLING BILLING SUBTOTAL
> HEADING AT PROJECT ALIGN BILLING MAXIMUM
```

In the next example, the two BILLINGS items will not align because BILLINGS SUBTOTAL on the heading is aligned with BILLINGS SUBTOTAL in the REPORT statement. The print position for the BILLINGS item of the heading is already positioned past the BILLINGS item in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT
> REPORT BILLING BILLING SUBTOTAL
> HEADING AT PROJECT ALIGN BILLING SUBTOTAL BILLING
```

INITIAL HEADING

Sets the content and format of the heading at the beginning of the report.

Syntax

INITIAL HEADING [ALIGN|NOALIGN] [report-group]

ALIGN|NOALIGN

ALIGN aligns the items in the INITIAL HEADING statement with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the heading that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

NOALIGN does not align items. The NOALIGN option overrides SET REPORT ALIGN.

Default: NOALIGN

report-group

Establishes the report-group to be printed at the beginning of the report. If no report-group is included, any previous INITIAL HEADING statement is canceled.

For more information about report-groups, see (p. 101).

Discussion

The INITIAL HEADING statement prints a one-time heading that appears on the first page of the report. When an initial heading appears on a page, any page heading is omitted for that page. If you use the NOREPORT statement, then the initial heading is not displayed if no records are reported.

To create a separate title page, simply add the SKIP PAGE option to the end of the initial heading report-group.

A skip option at the beginning of the initial heading statement's report-group is ignored. To skip lines before printing the initial heading, print a space before skipping. For example:

```
> INITIAL HEADING " " SKIP
```

Example

This report uses INITIAL HEADING to print a title for the report. In this example, INITIAL HEADING creates a title for the first page of the report. The first line of a report must be a default heading or a report-item; an initial SKIP is ignored as an initial heading. In this example, a skip is forced by inserting a blank report-item before the SKIP option.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL
>
> DEFINE TITLE CHARACTER * 45 &
>  = SPREAD("Overall Orders Report")
>
> SORT &
>  ON PARTNUMBER &
>  ON CUSTOMERNAME
>
> INITIAL HEADING  " " &
>   SKIP 1 &
>   TITLE &
>   SKIP 1
>
> HEADING AT PARTNUMBER &
>   SKIP 2 &
```

```
>   "Part Number: " &
>   PARTNUMBER SKIP &
>   "--------------------------------------"
>
> FOOTING AT CUSTOMERNAME &
>   "Orders for: " &
>   CUSTOMERNAME TAB 40 QUANTITYORDERED SUBTOTAL
>
> FOOTING AT PARTNUMBER &
>   SKIP &
>   "----------------------------------------" &
>   SKIP &
>   "Total Sales for Part: " &
>   PARTNUMBER TAB 40 QUANTITYORDERED SUBTOTAL
>
> FINAL FOOTING &
>   SKIP 2 &
>   "Total Sales: " &
>   TAB 40 QUANTITYORDERED SUBTOTAL &
>   "Units"
>
> GO
```

Without the ALIGN Option

In this example, the "These are the billings for Project" label is aligned to the left-hand side of the report and the project number follows. The default is NOALIGN.

```
> ACCESS BILLINGS
> SORT ON PROJECT ON EMPLOYEE
> CHOOSE PROJECT PARM PROMPT &
>   "ENTER PROJECT NUMBER REQUIRED " 1 TIMES
> REPORT TAB 20 BILLING EMPLOYEE PROJECT  &
>   PRINT AT PROJECT
> INITIAL HEADING &
>   SKIP PAGE " " SKIP 3 &
>   "These are the billings for Project " &
>   PROJECT &
>   TAB 60 "DATE" SYSDATE SKIP 3
> FINAL FOOTING ALIGN SKIP 2  &
>   "TOTAL BILLINGS FOR THIS PROJECT:"   &
>   SKIP 1 &
>   BILLING SUBTOTAL PICTURE "^^^,^^^,^^^.^^" &
>   PROJECT
> GO
```

```
These are the Billings for Project   P000001           DATE  93/04/14


                            100.00  00001  P000001
                          1,207.50  00001
                            945.00  00001
                            105.00  00003
                          1,470.00  00003
                            945.00  00004
                            315.00  00004
                            105.00  00004
                            420.00  00004

TOTAL BILLINGS FOR THIS PROJECT
                          5,612.50        P000001




?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the project is aligned with the project in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT ON EMPLOYEE
> CHOOSE PROJECT PARM PROMPT &
>   "ENTER PROJECT NUMBER REQUIRED " 1 TIMES
> REPORT TAB 20 BILLING EMPLOYEE PROJECT  &
>   PRINT AT PROJECT
> INITIAL HEADING ALIGN &
>   SKIP PAGE " " SKIP 3 &
>   "These are the billings for Project " &
>   PROJECT &
>   TAB 60 "DATE" SYSDATE SKIP 3
> FINAL FOOTING ALIGN SKIP 2  &
>   "TOTAL BILLINGS FOR THIS PROJECT:"   &
>   SKIP 1 &
>   BILLING SUBTOTAL PICTURE "^^^,^^^,^^^.^^" &
>   PROJECT
> GO
```

```
These are the Billings for Project          P000001     DATE  93/04/14


                             100.00  00001  P000001
                           1,207.50  00001
                             945.00  00001
                             105.00  00003
                           1,470.00  00003
                             945.00  00004
                             315.00  00004
                             105.00  00004
                             420.00  00004

TOTAL BILLINGS FOR THIS PROJECT
                           5,612.50         P000001




?
```

Aligning Summary Operations

In this example, the BILLING SUBTOTAL on the heading aligns with the BILLING SUBTOTAL in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> INITIAL HEADING ALIGN BILLING SUBTOTAL
```

In the following example, there is no matching summary operation and item in the REPORT statement. Therefore, the BILLINGS MAXIMUM on the heading aligns with the BILLINGS item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLINGS SUBTOTAL
> INITIAL HEADING ALIGN BILLING MAXIMUM
```

In the next example, the two BILLING items will not align because BILLING SUBTOTAL on the heading is aligned with BILLING SUBTOTAL in the REPORT statement. The print position for the BILLING item of the heading is already positioned past the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> INITIAL HEADING ALIGN BILLING SUBTOTAL BILLING
```

NOREPORT

Specifies what to print if no record complexes are selected.

Syntax

NOREPORT [report-group]

report-group

Establishes the report-group that is printed if no record complexes are selected. A NOREPORT statement with no report-group cancels a previous NOREPORT statement.

For more information about report-groups, see (p. 101).

Discussion

The NOREPORT statement specifies the report-group that is to be reported when no record complexes are selected by QUIZ (for example, when no data records meet selection criteria). Only the contents of the NOREPORT statement are printed; none of the page headings or footings are printed.

Using Report-Items

Any report-item can be included in the report-group, although only default values are reported. Typically, a report-item in a NOREPORT statement is a defined item, a system function, or a string.

Using Report-Groups

The NOREPORT statement uses a limited report-group. Operations performed at sort-items can't be used with the NOREPORT statement, since control breaks don't exist if no data is reported. These operations are PRINT AT, RESET AT, and any of the summary operations with the AT sort-item option that are used with sort-items. However, any of the other report-group options may be used.

Example

This example prompts the report user for a part number. QUIZ then reports all suppliers who offer that part. In this example, NOREPORT is executed when no records are found with values that match the entered part number.

```
> ACCESS PARTS &
>   LINK TO PARTSUPPLIERS &
>   LINK TO SUPPLIERS
>
> DEFINE REQPRT NUMERIC * 4 &
>   = PARM PROMPT &
>   "Enter a part number: "
>
> SELECT PARTSUPPLIER IF PARTNUMBER = REQPRT
>
> SORT &
>   ON SUPPLIERKEY OF SUPPLIERS &
>   ON PARTNUMBER OF PARTS &
>   ON PARTVARIANT OF PARTS
>
> REPORT &
>   SUPPLIERNAME OF SUPPLIERS &
>     PRINT AT SUPPLIERKEY OF PARTSUPPLIERS &
>   PARTNUMBER OF PARTS &
>     PRINT AT PARTNUMBER OF PARTS &
>   PARTVARIANT OF PARTS &
>   QOH OF PARTS
>
```

```
> NOREPORT &
>   "Summary" &
>   SKIP 1 &
>   "-------------------------------------------------" &
>   SKIP 1 &
>   "No supplier currently offers this part." &
>   SKIP
>
> GO
```

PAGE FOOTING

Sets the content and format of a footing at the end of the page.

Syntax

PAGE FOOTING [ALIGN|NOALIGN] [report-group]

ALIGN|NOALIGN

ALIGN aligns the items in the PAGE FOOTING statement with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the footing that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

NOALIGN does not align items. The NOALIGN option overrides SET REPORT ALIGN.

Default: NOALIGN

report-group

Establishes the report-group to be printed at the bottom of each page. If no report-group is included, any previous PAGE FOOTING statement is canceled.

For more information about report-groups, see (p. 101).

Discussion

A page footing can contain more than one report-item, and, if necessary, can extend over several lines.

Limit: Page footings don’t print on pages where either a final footing appears or an initial heading appears alone.

Example

This report lists customer invoice numbers by customer name. Using the PAGE FOOTING statement, the date and page number are printed at the bottom of each page.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER
>
> SORTED ON CUSTOMERNAME
>
> PAGE HEADING &
>   TAB 20 "Invoices for Customers" &
>   KEEP COLUMN HEADINGS &
>   SKIP 2
>
> REPORT CUSTOMERNAME &
>   PRINT AT CUSTOMERNAME &
>   INVOICENUMBER
>
> PAGE FOOTING &
>   SKIP 3 &
>   SYSDATE &
>   TAB 50 &
>   "Page" &
>   TAB 56 &
>   SYSPAGE
>
> GO
```

Without the ALIGN Option

In this example, the "TOTAL BILLINGS" label is aligned to the left-hand side of the report and the total billings follow. The default is NOALIGN.

```
> ACCESS BILLINGS
> REPORT TAB 25 PROJECT BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT ALIGN SKIP 2 PROJECT &
> BILLING SUBTOTAL &
> SKIP 2
> PAGE FOOTING "TOTAL BILLINGS:" &
>   BILLING SUBTOTAL &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   1

                      Project   Billing

                      P000001     100.00
                      P000001   1,470.00
                      P000001     945.00
                      P000001     105.00
                      P000001     105.00
                      P000001     315.00
                      P000001   1,207.50
                      P000001     945.00
                      P000001     420.00

                      P000001   5,612.50

                      P000002     105.00
                      P000002   1,470.00
                      P000002   1,207.50
                      P000002     157.50
                      P000002   1,470.00

  TOTAL BILLINGS :        8,552.50
?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the total billings is aligned with the billing on the REPORT statement.

```
> ACCESS BILLINGS
> REPORT TAB 25 PROJECT BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT ALIGN SKIP 2 PROJECT &
> BILLING SUBTOTAL &
> SKIP 2
> PAGE FOOTING ALIGN "TOTAL BILLINGS: " &
>   BILLING SUBTOTAL &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   1

                      Project   Billing

                      P000001     100.00
                      P000001   1,470.00
                      P000001     945.00
                      P000001     105.00
                      P000001     105.00
                      P000001     315.00
                      P000001   1,207.50
                      P000001     945.00
                      P000001     420.00

                      P000001   5,612.50

                      P000002     105.00
                      P000002   1,470.00
                      P000002   1,207.50
                      P000002     157.50
                      P000002   1,470.00

  TOTAL BILLINGS :              8,552.50
?
```

Aligning Summary Operations

In this example, the BILLING SUBTOTAL on the footing aligns with the BILLING SUBTOTAL in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> PAGE FOOTING ALIGN BILLING SUBTOTAL
```

In the following example, there is no matching summary operation and item in the REPORT statement. Therefore, the BILLING MAXIMUM on the footing aligns with the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> PAGE FOOTING ALIGN BILLING MAXIMUM
```

In the next example, the two BILLING items will not align because BILLING SUBTOTAL on the footing is aligned with BILLING SUBTOTAL in the REPORT statement. The print position for the BILLING item of the footing is already positioned past the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> PAGE FOOTING ALIGN BILLING SUBTOTAL BILLING
```

PAGE HEADING

Sets the content and format of a heading at the beginning of the page.

Syntax

PAGE HEADING [ALIGN|NOALIGN] [report-group]
[KEEP [COLUMN] [HEADINGS] [SKIP [n]]]

ALIGN|NOALIGN

ALIGN aligns the items in the PAGE HEADING statement with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the heading that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

NOALIGN does not align items. The NOALIGN option overrides SET REPORT ALIGN.

Default: NOALIGN

report-group

Establishes the report-group to be printed at the beginning of each page. The report-group usually consists of a customized title and column headings. If no report-group is included, any previous PAGE HEADING statement is canceled. QUIZ provides a default page heading unless the NOHEAD option is specified in the SET statement.

For more information about report-groups, see (p. 101).

KEEP [COLUMN] [HEADINGS] [SKIP [n]]

Retains the default column headings. SKIP sets the number of lines to be skipped after the default column headings have been printed.

COLUMN and HEADINGS are used only for documentation.

Default: If the number isn't included with the SKIP option, the default is one line. The first SKIP option jumps to the start of the next line. SKIP 2 leaves one blank line.

Discussion

The PAGE HEADING statement replaces the standard page and column headings with your own headings. The page heading is positioned immediately above the first line of your report.

QUIZ provides a default page heading that consists of a system date, system title, current page number, and column headings. The default page heading isn't provided if the PAGE HEADING or SET NOHEAD statements are in effect. PAGE HEADING overrides QUIZ default column headings unless the KEEP COLUMN HEADINGS option is specified.

Example

PAGE HEADING is used to print a title on every page of this report of invoice numbers sorted by customers. In this example

- PAGE HEADING specifies that the string stored as PAGETITLE (Customer/Invoice Cross Reference) is displayed at the top of each report page.
- KEEP COLUMN HEADINGS tells QUIZ to retain the default column headings for the element.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER
>
> DEFINE PAGETITLE &
>   CHARACTER*45 = "Customer/Invoice Cross Reference"
```

```
> SORTED ON CUSTOMERNAME
>
> PAGE HEADING &
>   TAB 20 &
>   PAGETITLE &
>   KEEP COLUMN HEADINGS &
>   SKIP 2
>
> REPORT CUSTOMERNAME &
>   PRINT AT CUSTOMERNAME &
>   INVOICENUMBER
>
> FOOTING AT CUSTOMERNAME &
>   SKIP 2
>
> PAGE FOOTING &
>   SKIP 3 &
>   SYSDATE &
>   TAB 50 &
>   "Page" &
>   TAB 56 &
>   SYSPAGE
>
> GO
```

Without the ALIGN Option

In this example, the "THESE ARE THE BILLINGS FOR PROJECT" label is aligned to the left-hand side of the report and the project number follows. The default is NOALIGN.

```
> ACCESS BILLINGS
> SORT ON PROJECT ON EMPLOYEE
> REPORT TAB 20 BILLING EMPLOYEE PROJECT &
>   PRINT AT PROJECT
> HEADING AT PROJECT SKIP PAGE
> PAGE HEADING SKIP 3 &
>   "THESE ARE THE BILLINGS FOR PROJECT " &
>   PROJECT  TAB 70  "PAGE"  SYSPAGE &
>   SKIP 3 KEEP COLUMN HEADINGS
> FOOTING AT PROJECT ALIGN SKIP 2 BILLING SUBTOTAL &
>   PROJECT SKIP 2
> FINAL FOOTING SKIP 2 "TOTAL BILLINGS:"    &
> BILLING SUBTOTAL PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
THESE ARE THE BILLINGS FOR PROJECT   P000001                    PAGE     1

                Billing   Employee  Project


                  100.00   00001    P000001
                  945.00   00001
                1,207.50   00001
                  105.00   00003
                1,470.00   00003
                  315.00   00004
                  105.00   00004
                  945.00   00004
                  420.00   00004

                5,612.50            P000001




?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the project is aligned with the project in the REPORT statement.

```
> ACCESS BILLINGS
> SORT ON PROJECT ON EMPLOYEE
> REPORT TAB 20 BILLING EMPLOYEE PROJECT  &
>   PRINT AT PROJECT
> HEADING AT PROJECT SKIP PAGE
> PAGE HEADING ALIGN SKIP 3 &
>   "THESE ARE THE BILLINGS FOR PROJECT " &
>   PROJECT  TAB 70  "PAGE"  SYSPAGE &
>   SKIP 3 KEEP COLUMN HEADINGS
> FOOTING AT PROJECT ALIGN SKIP 2 BILLING SUBTOTAL &
>   PROJECT SKIP 2
> FINAL FOOTING SKIP 2 "TOTAL BILLINGS:"    &
> BILLING SUBTOTAL PICTURE "^^^,^^^,^^^.^^"
> GO
```

```
THESE ARE THE BILLINGS FOR PROJECT    P000001                    PAGE     1

                 Billing   Employee  Project


                   100.00   00001    P000001
                   945.00   00001
                 1,207.50   00001
                   105.00   00003
                 1,470.00   00003
                   315.00   00004
                   105.00   00004
                   945.00   00004
                   420.00   00004

                 5,612.50            P000001




?
```

Aligning Summary Operations

In this example, the BILLING SUBTOTAL on the heading aligns with the BILLING SUBTOTAL in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> PAGE HEADING ALIGN BILLING SUBTOTAL
```

In the following example, there is no matching summary operation and item in the REPORT statement. Therefore, the BILLING MAXIMUM on the heading aligns with the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> PAGE HEADING ALIGN BILLING MAXIMUM
```

In the next example, the two BILLING items will not align because BILLING SUBTOTAL on the footing is aligned with BILLING SUBTOTAL in the REPORT statement. The print position for the BILLING item of the heading is already positioned past the BILLING item in the REPORT statement.

```
> ACCESS BILLINGS
> REPORT BILLING BILLING SUBTOTAL
> PAGE HEADING ALIGN BILLING SUBTOTAL BILLING
```

QSHOW

Runs QSHOW from QUIZ.

Syntax

QSHOW

Discussion

The QSHOW statement initiates a QSHOW session. QSHOW enables you to make quick online inquiries about entities (such as elements, files, and record-structures) in the data dictionary.

QSHOW is ready when its prompt character (-) appears.

When you exit from QSHOW, your session resumes at the point at which it was interrupted.

For more information about QSHOW, see Chapter 4, "QSHOW Statements", in the *IBM Cognos PowerHouse 4GL PDL and Utilities Reference*.

query-specification (SELECT)

Defines a collection of rows that will be accessible when the cursor is opened.

Syntax

SELECT [ALL|DISTINCT] {*|project-list}
FROM tablespec [,tablespec]...
[WHERE sql-condition]
[GROUP BY columnspec [,columnspec]...]
[HAVING sql-condition]

The syntax for a subquery is the same as for a query-specification with two exceptions: the subquery must project a single-column table and the syntax of the subquery must include enclosing brackets.

ALL|DISTINCT

ALL selects all the columns from the specified tables. ALL indicates that duplicate rows are included. DISTINCT indicates that duplicate rows are eliminated.

Default: ALL

*

Selects all the columns from the specified tables.

project-list

Project-list selects all the columns for the specified table. A project-list is a columnspec or derived column, or a list of columnspecs and derived columns separated by commas. If names are ambiguous, they must be qualified to ensure they can be identified uniquely.

The syntax for a columnspec is:

[table-name.|correlation-name.]column-name

The syntax for a derived column is:

expression [AS name]

The AS option assigns an alias to a column. It can be used to

- save typing because it is usually shorter than the column-name
- uniquely identify multiple references to the same column
- give a name to a derived column so it can be referenced in a program

For more information about expressions, see the section, "Expressions in SQL", in Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

FROM tablespec [,tablespec]...

The FROM option identifies the tables where data in the project-list is retrieved. Rows can be retrieved from simple tables, derived tables, or joined tables. A derived table is a full query-specification including an optional ORDER BY option.

Joins are used to combine data from two or more tables based on the relationships between data in those tables. The type of join affects the rows retrieved by the query-specification.

If a correlation name is defined for the tablespec, subsequent PowerHouse 4GL references to the table must use the correlation name.

The general form of the tablespec syntax used in the following options is:

[owner.]table-name [correlation-name]

In addition to the general form of the tablespec, the following forms are also valid for this option:

(derived table) correlation-name

The correlation name must be defined for a derived table, and subsequent PowerHouse 4GL references to the derived table must use the correlation name.

tablespec CROSS JOIN tablespec

In a cross join, every possible combination of rows from the two tables being joined is created, without regard for any matching.

tablespec [INNER] JOIN tablespec
ON columnspec = columnspec
[AND columnspec = columnspec]...

In an inner join, a row is included in the result-set only if it has a matching row in the other table. The INNER keyword is for documentation only.

The ON option specifies the condition of the join.

tablespec LEFT|RIGHT|FULL [OUTER] JOIN tablespec
ON columnspec = columnspec
[AND columnspec = columnspec]...

An outer join includes all rows in the tables whether or not there are matching rows.

The left outer join returns rows from the table listed before the JOIN keyword, even if they don't have a matching row in the second table listed.

The right outer join returns rows from the table listed after the JOIN keyword, even if they don't have a matching row in the first table listed.

The full outer join returns rows from both tables listed, even if they don't have a matching row in the other table listed.

The ON option specifies the condition of the join.

[WHERE sql-condition]

The sql-condition defines linkage between tables in the query, and search criteria for rows to be retrieved. Only data that meets the criteria is available for use by PowerHouse 4GL.

The sql-condition is a condition that is limited for use within IBM Cognos SQL syntax. For more information about SQL conditions, see the section, "Conditions in SQL", in Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

[GROUP BY columnspec [,columnspec]...]

Rearranges the result-set into the minimum number of groups such that all rows within any one group have the same value for the GROUP BY columns. Rows that do not satisfy the WHERE option are eliminated before any grouping is done. The result is known as a grouped table.

To use the GROUP BY option:

- the grouping columns need not appear in the project-list
- any non-aggregate in the project-list must be used in the GROUP BY option

```
>  SQL DECLARE X CURSOR FOR &
>  SELECT SP.PNO, MAX(SP.QTY), MIN(SP.QTY) &
>      FROM SP &
>      WHERE SP.SNO <> 'S1' &
>      GROUP BY SP.PNO
```

For detailed information about the GROUP BY option, refer to an SQL reference manual.

[HAVING sql-condition]

Eliminates groups, just as the WHERE option eliminates rows.

The sql-condition is limited for use within IBM Cognos SQL syntax. For more information about SQL conditions, see the section, "Conditions in SQL", in Chapter 5, "PowerHouse 4GL Language Rules", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document, or refer to an SQL reference manual.

```
>   SQL DECLARE X CURSOR FOR &
>   SELECT SP.PNO &
>       FROM SP &
>       GROUP BY SP.PNO &
>       HAVING COUNT(*) > 1
```

Limit: The sql-condition must evaluate to a single value per group.

Discussion

Specifying Selection Criteria Using the WHERE Option and Substitutions

The WHERE option is used to define selection criteria for rows to be retrieved. Only data that meets the criteria is available for use by PowerHouse 4GL. In addition to specifying selection criteria within the query-specification, you can specify selection criteria on the following statements:

- ACCESS
- DECLARE CURSOR
- LOOKUP option on the EDIT statement

Examples

The following example demonstrates the way PowerHouse 4GL creates a single SQL query combining multiple conditions from multiple statements:

```
> SET LIST SQL
> SQL IN EMPBASE DECLARE X CURSOR FOR &
> SELECT * FROM EMPLOYEES &
>  WHERE CITY = 'BOSTON'
> SCREEN EMPLOYC
> CURSOR X WHERE(EMPLOYEE BETWEEN 1000 AND 5000) &
>  PRIMARY KEY EMPLOYEE
> ACCESS WHERE(POSITION = 'PRG') &
>  VIA EMPLOYEE ORDERED
```

The final query includes all three conditions specified in the WHERE options.

```
__ Sql after substitutions are applied:
__ SELECT *
__     FROM EMPLOYEES
__     WHERE POSITION = 'PRG' and
__           EMPLOYEE BETWEEN 1000 AND
__           5000 and
__           CITY = 'BOSTON'
```

Inner Joins

The following inner join would report all customers with matching invoices.

```
> SQL DECLARE X CURSOR FOR &
> SELECT * FROM CUSTOMER C &
>  INNER JOIN INVOICES I &
>  ON C.CUSTOMER_NUM=I.CUSTOMER_NUM
```

Outer Joins

The following example would report all customers even if they didn't have any invoices. Invoices without matching customers would not be reported.

```
> SQL DECLARE X CURSOR FOR &
> SELECT * FROM CUSTOMER C &
>  LEFT OUTER JOIN INVOICES I &
>  ON C.CUSTOMER_NUM=I.CUSTOMER_NUM
```

The next example would report all invoices even if they didn't have any matching customer information. Customers without invoices would not be reported.

```
> SQL DECLARE X CURSOR FOR &
> SELECT * FROM CUSTOMER C &
>  RIGHT OUTER JOIN INVOICES I &
>  ON I.CUSTOMER_NUM=C.CUSTOMER_NUM
```

QUIT

Ends a QUIZ session.

Syntax

QUIT

Discussion

The QUIT statement ends your QUIZ session and returns control to the operating system or invoking program.

The QUIT statement can be abbreviated to Q, QU, or QUI.

The QUIT statement and the EXIT statement perform in the exact same manner.

REPORT

Sets the content and format of report detail lines.

Syntax

REPORT [SUMMARY] [report-group|ALL [SKIP n]]

SUMMARY

States that displayed or printed output isn't required. The SUMMARY option is typically used with the SET SUBFILE statement.

report-group

Establishes the report-group to be printed for each record complex. If no report-group is included, any previous REPORT statement is canceled, and no detail lines are printed.

For more information about report-groups, see (p. 101).

ALL

Reports all values for available record items and defined items. If a record-structure in the ACCESS statement includes an array, all occurrences of the item in the array are reported.

If a record-structure in the ACCESS statement includes a blob item, a default option of WRAP 18 is used. For more information about the WRAP option, see (p. 102).

SKIP n

Sets the number of lines that should be skipped before the next record complex is reported.

Default: 1

Discussion

The REPORT statement prints a report-group for every record complex.

Default: As a convenience for testing, QUIZ, by default, reports the following:

- only the first 100 records in an interactive session
- only the first 1000 records when QUIZ is executed in batch

To increase this limit, use the SET REPORT LIMIT statement.

Example

The following report lists invoice numbers, invoice dates, and invoice amounts for customers. In this example

- PRINT AT tells QUIZ to print the value for CUSTOMERNAME only when it changes.
- HEADING replaces the default column heading as defined in the dictionary.
- ^ splits the heading into two lines: "Amount" on the first, "Owing" on the second.
- RESET AT CUSTOMERNAME is the default; you don't have to state it explicitly.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
>   AND TO PARTNUMBER OF PARTS
>
> DEFINE INVOICEAMOUNT NUMERIC*9 &
>   PICTURE "$^^^,^^^.^^" &
>   = QUANTITYSHIPPED OF INVOICEDETAIL * &
>   (UNIT-COST * UNITMARKUP)
>
```

```
> SORT &
>   ON CUSTOMERNAME &
>   ON INVOICEDATE D
>
> REPORT &
>   CUSTOMERNAME PRINT AT CUSTOMERNAME &
>   INVOICENUMBER &
>   INVOICEDATE &
>   INVOICEAMOUNT &
>   HEADING "Amount^Owing"
>
> FOOTING AT &
>   CUSTOMERNAME &
>   SKIP 2  &
>   "Total:" &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL &
>   RESET AT CUSTOMERNAME &
>   SKIP 2
>
> FINAL FOOTING &
>   "Grand Total:" &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL
>
> GO
```

The PRINT AT option eliminates redundant report information. The item specified after PRINT AT must be defined as a sort-item in a SORT or SORTED statement. The item being printed at the sort-item only appears at the first occurrence of the sort-item.

Summary operations such as SUBTOTAL can be applied to any sort-item. The amount owing is calculated for each customer in this report. The subtotal is then reset to zero in preparation to calculate the amount owing for the next customer.

report-group

Determines the content and format of detail lines, headings, and footings.

Syntax

[report-item]... [RESERVE n [LINES]]

Report-group isn't a QUIZ statement. Rather, report-group is a construct that is used frequently in several QUIZ statements.

Options

You can use report-group in the following QUIZ statements:

FINAL FOOTING	FOOTING AT	HEADING AT
INITIAL HEADING	NOREPORT	PAGE FOOTING
PAGE HEADING	REPORT	

report-item

Specifies a single component of a report-group. For more information, see (p. 102).

RESERVE n [LINES]

Starts a new page before printing the report-group if there are fewer than the specified number of lines left on the current page.

LINES is used only for documentation.

Default: The number of lines required for the entire report group, including report headings, footings, and so on.

Discussion

If a multi-line report-group can't fit on the remaining lines of a monitor or page, QUIZ skips to the next page by default. This action can be suppressed with the SET NOSPACE option.

NOREPORT uses a limited report-group. Operations performed at sort-items can't be used with NOREPORT since control breaks do not exist if no data is reported. These operations are PRINT AT, RESET AT, and any of the summary operations with the AT sort-item option. Any of the other report-group options may be used.

Summary operations that are used with the FOOTING statement are reset at the control breaks.

Summary operations that are used with the REPORT statement are reset at the lowest-level control break. The lowest-level control break is set by the last item named in a SORT or SORTED statement.

If you're using the SET NOBLANKS statement and there is a possibility that some of the lines in a single REPORT (or HEADING AT or FOOTING AT) statement will be all blank, then you may want to use the RESERVE option to reduce the number of lines that QUIZ automatically reserves.

For more information about summary operations, see (p. 102).

report-item

Specifies what's to be reported, its position and format, and whether it's to be printed on every line or only at a control break.

Syntax

[SKIP [n|PAGE]] [TAB n] [content] [format-option]
[PRINT AT sort-item]

SKIP [n|PAGE]

Skips either a specified number of lines or skips to a new page before printing the content.

TAB n

Starts printing the content in the specified column.

content

Designates what's to be printed. The content can be a record item, a defined item, a string, a system function, COUNT, or SYSPAGE. When a record item is part of an array, only the first occurrence is reported unless it's subscripted. Summary operations are valid for record items. The item syntax is:

item [WRAP n] [summary-operation]

WRAP n

Specifies how wide the column is on the report. QUIZ uses the same word-wrapping algorithm that QUICK uses for displaying multi-line fields: a long field is broken at the last space prior to the WRAP size, or at a new line.

Several report-items may have defined WRAP options. QUIZ prints enough lines in the report to accommodate the longest item. Blanks are printed for other items until the end of the longest item's text has been reached.

A default option of WRAP 18 is used for all blob items.

Limit: The WRAP option can be used for any multi-line field, such as character, text blobs (maximum size 32,767), and varying character items as documented in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document. QUIZ makes no attempt to ensure that the text will fit on a single page, or won't extend onto the next page. Also, it does not verify that the item being reported contains text. It is not recommended to report a non-text item.

summary-operation

The summary-operations are:

AVERAGE	COUNT	MAXIMUM
MINIMUM	PERCENT	RATIO
SUBTOTAL		

Indicates that a summary operation is performed on the content of the report-item. Summary operations are valid for record items and defined items.

To specify a summary operation, include the name of the summary operation in your report-item specification, as in

```
> REPORT INVOICEAMOUNT SUBTOTAL
```

By default, when you specify a summary operation, it's calculated for each record complex. The summary operation is performed at the beginning of each control break when AT sort-item is included.

Summary operations used with the REPORT statement are reset at the lowest-level control break. The lowest-level control break is associated with the last sort-item declared in a SORT or SORTED statement.

If null value support is enabled in the PowerHouse dictionary, null values are not included in summary and statistical computations. The summary and statistical operations, AVERAGE, MAXIMUM, MINIMUM, and SUBTOTAL return the value null only when all the values used in the calculation are null.

If null value support is disabled, any character item that is null is initialized to spaces and any date or numeric item is initialized to zeros, and will be included in summary and statistical calculations.

AVERAGE [AT sort-item] [RESET AT sort-item|NORESET]

Reports the subtotal of the content of the report-item divided by the count of the number of record complexes.

COUNT [AT sort-item] [RESET AT sort-item|NORESET] [INITIAL VALUE item|system-function]

Reports a number that increments by one for each record complex. Unlike other summary operations, COUNT isn't associated with the content of the report-item and must be used alone.

MAXIMUM [AT sort-item] [RESET AT sort-item|NORESET] [INITIAL VALUE item|system-function]

Reports the maximum value of the content of the report-item.

MINIMUM [AT sort-item] [RESET AT sort-item|NORESET] [INITIAL VALUE item|system-function]

Reports the minimum value of the content of the report-item.

PERCENT item2 [AT sort-item] [RESET AT sort-item|NORESET]

Reports the ratio of the content of the report-item and item2 multiplied by 100.

RATIO item2 [AT sort-item] [RESET AT sort-item|NORESET]

Reports the subtotal of the content of the report-item divided by the subtotal of item2.

SUBTOTAL [AT sort-item] [RESET AT sort-item|NORESET] [INITIAL [VALUE] item|system-function]

Reports the sum of the values of the content of the report-item.

Options

The summary-operation options are AT, INTIAL VALUE, RESET, and NORESET.

AT sort-item

Performs the summary action at the start of the control break specified by the sort-item.

INITIAL [VALUE] item|system-function

Assigns a starting value using the current value of this item or system function. VALUE is used only for documentation.

RESET AT sort-item|NORESET

Sets the summary operation to either zero or an initial value at the start of the control break specified by the sort-item.

If the INITIAL VALUE option is used to assign a starting value, the RESET option sets the starting value at the control break.

If INITIAL VALUE isn't used, the RESET option sets the summary to zero. NORESET declares that the summary isn't to be reset.

When the RESET option isn't used, default resetting applies.

format-option

Indicates how the content is formatted when displayed.

format-option		
BWZ|NOBWZ	FILL	FLOAT
FORMAT	HEADING	LEADING
NULLSEPARATOR|NONULLSEPARATOR	OUTPUT SCALE	PICTURE
SEPARATOR	SIGNIFICANCE	TRAILING

BWZ|NOBWZ

If BWZ (blank when zero) is specified and the item value equals zero, the value is displayed as a blank. NOBWZ displays one or more zeros, depending on the significance of the item.

Limit: Valid for numeric values only.

FILL char

Specifies the character used to fill any unused space to the left of the most significant digit, float character, or leading sign in the picture.

The fill character also replaces unnecessary leading nonsubstitution characters, including commas and leading spaces.

For example, the following attributes work together to display a dollar amount that is preceded by leading asterisks:

Fill	Float	Picture	Value	Display
*	$	"^,^^^,^^^.^^"	123456	***$1,234.56

Limit: Valid for numeric values only.

FLOAT char

Specifies the float character. The float character is inserted immediately to the left of the most significant digit. To ensure that there is enough room for the float character, add either a space or an extra substitution character (^) to the left side of the picture.

For example, the following element attributes work together to display a dollar sign ($) in front of an item value:

Stored value	Picture	Float	Display
1234	"^^^^^"	$	$1234
56789	" ^^^.^^"	$	$567.89

Limit: Valid for numeric values only.

FORMAT date-format

Specifies the format for displaying date item values. A date-format can be one of the following:

Date-format	Example	Date-format	Example
YYMMDD	01/05/23	YYMMMDD	01/MAY/23
YYYYMMDD	2001/05/23	YYYYMMMDD	2001/MAY/23
YYMM	01/05	YYMMM	01/MAY

Date-format	Example	Date-format	Example
YYYYMM	2001/05	YYYYMMM	2001/MAY
YYDDD	01/125	YYYYDDD	2001/125
MMDDYY	05/23/01	MMMDDYY	MAY/23/01
MMDDYYYY	05/23/2001	MMMDDYYYY	MAY/23/2001
MMYY	05/01	MMMYY	MAY/01
MMYYYY	05/2001	MMMYYYY	MAY/2001
MMDD	05/23	MMMDD	MAY/23
DDMMYY	23/05/01	DDMMMYY	23/MAY/01
DDMMYYYY	23/05/2001	DDMMMYYYY	23/MAY/2001
DDMM	23/05	DDMMM	23/MAY
DDDYY	125/01	DDDYYYY	125/2001

YYYY - four digit year (e.g., 2001)

MM - two digit month (e.g., 05)

MMM - three character month name (e.g., MAY)

DD - two digit day for a month (e.g., 23)

DDD - three digit day for a year (e.g., 365)

Regardless of the output order of the date, the internal working format is YYMMDD (for dates without centuries), YYYYMMDD (for dates with centuries).

Limit: Valid only for date items. This option only affects the display format; the entry format isn't affected.

Default: If DATE FORMAT is not specified, the system option is used. If the system option is not specified, the default is YYYYMMDD.

HEADING string

Provides a default column heading for QUIZ reports. The HEADING option customizes the REPORT statement.

To produce multi-line headings, embed a multi-line heading character (^) in the heading string. For example, the heading "Employee^Number" displays a two-line heading in a QUIZ report:

```
Employee
 Number
```

If you do not specify a multiline heading, when the column heading is much longer than the column data width, PowerHouse 4GL wraps the heading automatically. PDC dictionaries determine the heading at dictionary compile time; PHD dictionaries determine the heading at component runtime. The algorithms used to determine where to split the text and the resultant column width are different for PDC and PHD dictionaries. If you are switching between dictionary types, you may have to reformat your reports or specify your own wrapped headings.

Limit: 60 characters per string

Default: For defined items only, if HEADING is not specified, the item name is used.

LEADING [SIGN] char

Determines the character that indicates a negative number. The leading sign is displayed to the left of the most significant digit and the float character. To ensure that there is enough room for the leading sign, add sufficient substitution characters (^) or nonsubstitution characters to the picture.

If the picture is too small, overflow occurs and QUIZ displays crosshatches (#).

SIGN is used only for documentation.

Limit: Valid for numeric values only.

Default: A negative sign (-)

NULLSEPARATOR|NONULLSEPARATOR

NULLSEPARATOR specifies that all dates are to be displayed without a separator. This allows display of century-included dates in the same space as century-excluded dates.

The DATE SEPARATOR is used for display formatting if NULLSEPARATOR is not used, or is canceled by the NONULLSEPARATOR option.

The DATE SEPARATOR may be used during input. If NULLSEPARATOR is specified, the value is redisplayed after formatting without the separator.

Default: To find out the active value of the option, you must look at the ELEMENT, the USAGE, and the SYSTEM OPTIONS statements in PDL. If the option is unspecified on the report-item, the active value is taken from the ELEMENT. If the option is unspecified on the ELEMENT statement or a related USAGE, the active value is taken from the SYSTEM OPTIONS.

OUTPUT SCALE n

Establishes the output scaling factor. The stored value of the item is multiplied by 10 raised to the power of the output scale value (that is, 10^n). The result is rounded after scaling. The OUTPUT SCALE option is needed for floating-point numbers, since fractional portions of stored values are truncated for display. With an output scale of zero, all digits to the right of the decimal in a floating-point number are eliminated when the number is truncated.

For example, the following item attributes work with the output scale to display floating point values:

Stored value	Picture	Output scale	Display
12.54	"^^^^.^^"	0	0.13
12.54	"^^^^.^^"	2	12.54

Use the OUTPUT SCALE option to display the decimal portion of the number:

Stored value	Picture	Output scale	Display
123.456	"^^^^^^"	0	123
123.456	"^^^^^^"	1	1235
123.456	"^^^^^^"	2	12346
123.456	"^^^^^^"	3	123456
123.456	"^^^^^^"	-1	12
123.456	"^^^^^^"	-2	1
123.456	"^^^^^^"	-3	0

Use the OUTPUT SCALE and PICTURE options for proper decimal alignment on output:

Stored value	Picture	Output scale	Display
123.456	"^^^.^"	1	123.5
123.456	"^^^.^^"	2	123.46

Stored value	Picture	Output scale	Display
123.456	"^^^.^^^"	3	123.456

Limit: -16 to 16. Valid for numeric items only.

Default: 0

PICTURE string

Establishes the output picture used to format the item value for display. A picture string is made up of substitution characters (^) and nonsubstitution characters.

Character items are formatted in the following way:

1. The item is processed from left to right, substituting one character from the item for each substitution character in the picture. Nonsubstitution characters remain unchanged.
2. If there are fewer substitution characters in the picture than characters in the item value, the remaining characters in the item aren't displayed.
3. If there are more substitution characters in the picture than characters in the item value, spaces are padded to the right of the item.

As an example, the item value "FHSMITH" is formatted as follows:

Picture	Display
"^^^^^^^"	FHSMITH
"^^^^^"	FHSMI
"^.^. ^^^^^"	F.H. SMITH
"^.^. ^^^^^^^^^^"	F.H. SMITH

Numeric items are formatted in the following way:

1. The item is scaled by the output scale and rounded to the nearest whole number.
2. The result of Step 1 is processed from right to left, substituting one digit from the item for each substitution character in the picture until all significant (non-zero) digits have been processed. Nonsubstitution characters remain unchanged.
3. Until the element significance is reached, leading zeros are substituted for each substitution character. Nonsubstitution characters remain unchanged.
4. The float character is added.
5. Leading sign characters and trailing sign characters, or both, are added for negative values.
6. The remaining portion of the picture is filled with the fill character.
7. If there isn't enough room in the picture to hold the significant digits of the item value or the leading or trailing sign, the item is filled with the overflow character, the crosshatch (#).

For example, the value 1578 is formatted as follows:

Picture	Display
"^^^"	####
"^^^^"	1578
"^^.^^"	15.78
"^^,^^^,^^^"	1,578

Specifying a PICTURE in QUIZ causes any SIGNIFICANCE specified in the dictionary to be ignored. A new default significance is calculated based on the specified PICTURE. To change both the PICTURE and SIGNIFICANCE, specify the PICTURE first followed by the new SIGNIFICANCE.

Limit: 60 characters per string. Valid for numeric and character values only. If the PICTURE and SIGNIFICANCE options are used together, specify PICTURE first.

SEPARATOR char

Sets the character that separates the day, month, and year portions of a date element for display.

Limit: Valid for date and numeric values only.

Default: The System Options separator or a slash (/).

SIGNIFICANCE n

Specifies the minimum number of digits and characters displayed. SIGNIFICANCE forces the printing of leading nonsubstitution characters and leading zeros.

Limit: 255. Valid for numeric values only. If the PICTURE and SIGNIFICANCE options are used together, specify PICTURE first.

TRAILING [SIGN] string

Specifies a string of one or two characters placed to the right of the picture when the value is negative. To ensure that there is enough room for the trailing sign, add sufficient nonsubstitution characters to the rightmost portion of the picture.

SIGN is used only for documentation.

To place parentheses around negative numbers, use the LEADING SIGN and TRAILING SIGN options together. For example, a leading sign of "(" with a trailing sign of ")" displays the value -123.45 as

```
(123.45)
```

For example, the value -1578 is formatted as follows:

Picture	Leading sign	Trailing sign	Display
"^^^,^^^ "	none	CR	1,578CR
" ^^^,^^^ "	(	)	(1,578)

Limit: Valid for numeric values only.

Default: None

PRINT AT sort-item

Specifies that the content is reported once at the start of each control break at the sort-item's level. The PRINT AT option suppresses the display of repetitive values in report-groups reported by the REPORT statement. If QUIZ skips to a new page, the content is reprinted.

Discussion

A report-item is an item in a statement combined with any optional specifications you add. In QUIZ, a report-item specifies the following:

- what is to be reported
- its position and format
- whether it is to be printed on every line or only at control breaks

You can use system functions with the COUNT, MAXIMUM, MINIMUM, and SUBTOTAL summary operations.

For more information about these functions, see Chapter 6, "Functions in PowerHouse 4GL", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

REVISE

Edits the current temporary save file or a specified file.

Syntax

REVISE [*|filespec [option]...]

*****

Signifies that the temporary source statement save file is to be edited. The save file is quizsave.qks (**OpenVMS, UNIX, Windows**).

The source statement save file is a temporary file that PowerHouse 4GL opens at the beginning of a session.

All QUIZ statements you've entered since the last CANCEL CLEAR, SAVE CLEAR, or SET SAVE CLEAR statement are recorded in QUIZSAVE as you enter them. However, CANCEL CLEAR, SAVE CLEAR, SET SAVE CLEAR, SAVE, and EXIT are not recorded in QUIZSAVE.

The asterisk isn't required if you are editing the QUIZSAVE file without changing any default options. However, the asterisk is required if you're overriding any of the default options, as in

```
> REVISE * NOUSE
```

filespec

The name of an existing permanent file. If this file does not contain QUIZ statements, use the NOUSE option so that QUIZ doesn't try to process the file when you exit from the system editor.

Options

The options are DETAIL, NODETAIL, LIST, NOLIST, USE, and NOUSE.

DETAIL|NODETAIL

DETAIL copies the contents of the revised file into the QUIZSAVE file when you exit from the system editor; NODETAIL does not.

If you're revising a permanent file and the USE and NODETAIL options are in effect, then a USE statement is written to the current QUIZSAVE file and invoked when you exit from the system editor, as in

```
> USE ORDERS NODETAIL
```

Limit: NODETAIL is not valid with QUIZSAVE.

Default: DETAIL

LIST|NOLIST

LIST displays the statements in the revised file as QUIZ processes them; NOLIST does not.

Default: LIST

USE|NOUSE

USE processes the revised statements when you exit from the system editor. NOUSE returns you to QUIZ at the point from which you left it, without processing the revised statements.

Default: USE

Discussion

The REVISE statement indicates which file is to be edited, and, optionally, how the revised file is to act upon re-entering QUIZ. You use the system editor to edit either the QUIZSAVE file or a permanent file from within QUIZ. The QUIZSAVE file is edited by default.

When you enter the REVISE statement without a file name, QUIZ automatically performs a CANCEL CLEAR statement prior to processing the statements. When you enter the REVISE statement with a file name, the automatic CANCEL CLEAR statement isn't performed.

The **procloc** parameter affects how PowerHouse 4GL uses unqualified file names that are specified in the REVISE statement. For more information about the **procloc** program parameter, see Chapter 2, "Program Parameters", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Choosing a Different Editor

OpenVMS

The REVISE statement invokes the DCL command assigned to the global symbol PHEDIT (usually used to designate an editor). By default, the SET POWERHOUSE command sets PHEDIT to

```
$PHEDIT :==EDIT/EDT
```

causing the REVISE statement to invoke the EDT editor.

You can change the default editor by changing the setting of the PHEDIT symbol. For example, to use the special interface to EDT called UTILITIES:EDT.COM, change the setting to

```
$PHEDIT :==@UTILITIES:EDT.COM
```

It is recommended that you use either EDIT/EDT or EDIT/TPU as the setting for PHEDIT. In either of these cases, the editor can be called directly; otherwise, a subprocess is spawned.

If you intend to use the **nodcl** program parameter to restrict user access to the operating system, it is further recommended that you do not select editors (such as TPU) that provide operating system access. When **nodcl** is in effect, users will continue to be able to access the system editor through the REVISE statement.

UNIX

By default, the REVISE statement uses the editor defined by the environment variable PHEDIT. If PHEDIT is not defined, the system checks the environment variable EDITOR. If you have not defined either of these variables, the REVISE statement fails.

Windows

By default, the REVISE statement uses the editor defined by the environment variable PHEDIT. If PHEDIT is not defined, the system checks the environment variable, EDITOR. The PowerHouse 4GL installation procedure sets PHEDIT to specify the Windows Notepad as the editor unless the PHEDIT environment variable is already set, in which case the setting is left as is.

SAVE

Saves the current QUIZ source statements in a file.

Syntax

SAVE filespec [CLEAR]

filespec

Names a file that will contain the QUIZ statements.

If QUIZ finds an existing file with the same name, it prompts for confirmation before creating a new version or overwriting the existing file. If SET NOVERIFY DELETE is in effect, no prompting takes place.

CLEAR

Removes any source statements in the temporary save file, QUIZSAVE, once the contents are copied to a permanent file.

Discussion

The SAVE statement writes statements to the temporary source statement file, QUIZSAVE, as you enter them. The SAVE statement itself is not included in the file.

The SAVE statement creates a permanent file and copies the contents of QUIZSAVE into it. You can use the saved contents as a source file for documentation and future changes, or as a working file for modification using the system editor. The saved statements can also be processed by QUIZ with the USE statement.

The CLEAR option clears the temporary save file after its contents have been saved so that you can enter and then save a new set of QUIZ statements in the same session. To clear the temporary save file without saving its contents, use the SET SAVE CLEAR statement.

For more information about where QUIZ saves source statement files, see the section, "Locating Files", in Chapter 1, "Running PowerHouse 4GL", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Example

This report generates mailing labels for customers. The statements that make up the report are saved in a file named LABELS. In the following example

- SAVE creates a permanent file, LABELS, that contains the report source statements.
- CLEAR removes these report statements from the temporary save file, clearing it for the next group of source statements.

```
> ACCESS CUSTOMERS
>
> SORTED ON CUSTOMERNAME
>
> DEFINE CITYPROV CHARACTER*38 &
>   = TRUNCATE(CITY) + ", " + TRUNCATE(PROVSTATE)
>
> REPORT &
>   CUSTOMERNAME SKIP &
>   STREET SKIP &
>   CITYPROV SKIP &
>   COUNTRY SKIP &
>   POSTALZIP SKIP PAGE
>
> SET REPORT DEVICE PRINTER
> SET PAGE WIDTH 40
> SET PAGE LENGTH 12
> SET PAGE IMAGES 2
```

```
> SET NOHEAD
>
> SAVE LABELS CLEAR
```

SELECT

Applies selection conditions to record-structures in the ACCESS statement when building a record complex.

Syntax

SELECT [record-structure] [IF condition]
SELECT [IF condition]
AND SELECT [IF condition]

SELECT record-structure [IF condition]

Applies a condition to a data record as it is read. Specifies that if the selection condition is satisfied, the data record is included in the record complex. If the condition isn't satisfied, the data record is bypassed and the next data record is read before the record complex is constructed.

Only one SELECT record-structure statement can be in effect for each record-structure named in the ACCESS statement. The conditions in the statement cannot be based on items from files later in the ACCESS statement. It is recommended that items occurring in more than one file be qualified with OF. This is because the record structure in the SELECT file IF is searched first, followed by the record structures in the ACCESS statement in first to last order.

A second SELECT record-structure IF overrides the previous SELECT record-structure IF for the same record-structure; a SELECT record-structure (with no conditions) cancels any previous SELECT record-structure statement for that record-structure.

The SELECT record-structure IF statement applies the conditions only to items in the specified record-structure. If the condition fails, QUIZ skips the current record (of the specified record-structure) and tries to build the record complex with the next data record from the file.

Limit: This form of the SELECT statement cannot be applied to a cursor. To apply selection criteria to a cursor, use the WHERE clause of the cursor declaration.

SELECT [IF condition]

Applies a condition to the record complex as a whole. The condition is evaluated as the record complex is being created. If the condition is not satisfied, the creation of the record complex is halted immediately.

Only one SELECT statement can be in effect at one time. A second SELECT IF overrides the previous SELECT IF statement; a SELECT (with no conditions) cancels any previous SELECT IF statement.

Limit: This form of the SELECT statement cannot be applied to a cursor.

AND SELECT [IF condition]

Allows specification of a selection condition in addition to those in the SELECT IF without canceling the previous SELECT IF statement. A second AND SELECT IF overrides any previous AND SELECT IF statement; an AND SELECT (with no conditions) cancels any previous AND SELECT IF statement. AND SELECT is another way of adding conditions; this is useful for adding additional selection criteria to a previously compiled report.

Limit: This form of the SELECT statement cannot be applied to a cursor.

condition

The condition may reference any items that are currently accessible, which includes items accessed through a cursor. The condition is applied against the record complexes that QUIZ has built. By this time QUIZ has read the data.

The following is valid:

```
>   SQL DECLARE emp CURSOR FOR &
>   SELECT * FROM employees
>   ACCESS emp
>   SELECT IF firstname > 'H'
```

but it may be more efficient to let the database do the work, as in

```
>  SQL DECLARE emp CURSOR FOR &
>  SELECT * FROM employees &
>     WHERE firstname > 'H'
```

Discussion

SELECT Statement Conditions

Note that both SELECT statements ignore trailing blanks in any selection values. This means that SELECT statement values "1000 " and "1000" are treated as the same value for retrieval. This is consistent with the way in which relational systems display data.

The record or record complex is bypassed if evaluating the condition results in a data expression error.

Parallel Relationships

Use caution when using the SELECT statement with a condition involving items from record-structures in a parallel relationship. Since the record complex is abandoned if data records can't be retrieved, potentially significant data can be lost when the record complex is abandoned. You can avoid this situation by using the SELECT record-structure statement.

Given the statement defining a parallel detail relationship,

```
>  ACCESS PARTS LINK TO PARTSDETAIL AND TO PARTSINFO
```

with representative values,

PARTS	PARTSDETAIL	PARTSINFO
P1	D1 D2	I1 I2 I3

with no SELECT, the record complex is:

```
P1      D1         I1
P1      D2         I2
P1      -          I3
```

with the following SELECT IF,

```
>  SELECT IF ITEM OF PARTSINFO = I2
```

the record complex is:

```
P1      D2         I2
```

with the following SELECT record-structure IF,

```
> SELECT PARTSINFO IF ITEM OF PARTSINFO = I2
```

the record complex is:

```
P1      D1         I2
P1      D2         -
```

Speed and Efficiency of Execution

The order of the record-structures in the ACCESS statement affects the speed of execution. Conditions applied to record-structures that appear earlier in the ACCESS statement result in lower execution-times than those applied to record-structures that appear later in the ACCESS statement.

Differences Between the CHOOSE and SELECT Statements

The CHOOSE statement can also be used to retrieve data records. The SELECT statement always reads records sequentially; the CHOOSE statement retrieves records by indexes.

For more efficient performance, use the CHOOSE statement instead of the SELECT statement when possible. However, the CHOOSE and SELECT statements aren't mutually exclusive. If CHOOSE and SELECT are used in the same report, the CHOOSE statement is performed before the SELECT statement.

When a new collating sequence is defined in the data dictionary, the CHOOSE and SELECT statements perform differently; though they may seem to be asking for the same thing, the results may actually be different. The changed collating sequence has no impact on the records retrieved with the CHOOSE statement, but does affect the data records you can retrieve with SELECT.

For example, if the order of the standard English alphabet is reversed in the data dictionary, and you enter the following syntax

```
> SELECT IF LASTNAME > "A"
```

no data records are selected.

However, if you enter this syntax

```
> CHOOSE LASTNAME "a@@"
```

all data records are selected.

Adding Additional Selection Criteria

You can use AND SELECT IF to add selection criteria to a previously compiled report. In the following example, PROJREP is a compiled report that includes a SELECT statement that selects all current projects. In order to make the report execute for only one project, an AND SELECT IF statement is added:

```
> EXECUTE PROJREP NOGO
> AND SELECT IF PROJECTCODE = 1248
> G0
```

The report can't already include an AND SELECT IF statement; it would be overridden.

Example

The following example reports all orders where the quantity shipped is less than the quantity ordered, and the order is more than two months old. For this example, assume the current date is June 15, 2001:

```
> ACCESS &
>   ORDERMASTER &
>   LINK TO ORDERDETAIL
>
> DEFINE ORDERAGE = DAYS(SYSDATE) - DAYS(ORDERDATE)
>
> SELECT ORDERMASTER IF ORDERAGE > 60
> SELECT ORDERDETAIL IF &
>   QUANTITYSHIPPED < QUANTITYORDERED
>
> REPORT &
>   ORDERNUMBER &
>   QUANTITYSHIPPED &
>   QUANTITYORDERED &
>   ORDERDATE
>
> GO
```

The first SELECT statement retrieves a record from the ORDERMASTER file only if the value of ORDERDATE is before April 15, 2001. The second SELECT statement retrieves a record from the ORDERDETAIL record-structure only if items have been back-ordered.

The resulting QUIZ report looks like this:

```
1991-06-17                  FUTURE INDUSTRIES Inc.                  PAGE   1

   Order    Quantity   Quantity     Order
   Number   Shipped    Ordered      Date

   1003         0          23    1991-03-15
   1004         0           3    1991-03-18
   1006         0          35    1991-03-18
   1007         0          10    1991-03-18

?
```

SET

Changes the default report-control settings.

Syntax

SET DEFAULT|option...

DEFAULT

Resets all the SET options, except SET DICTIONARY, to default values. The DATABASE option is reset to what is specified in your current dictionary.

Interactive	Batch
BLANKS	BLANKS
DUPLICATES	DUPLICATES
NOFORMFEED	FORMFEED
HEAD	HEAD
LIST	LIST
NOCLOSE	CLOSE
NOJOB	NOJOB
NOLIST SQL	NOLIST SQL
NOPRINT	NOPRINT
NOSTOPONERROR	NOSTOPONERROR
NOSUBFILE	NOSUBFILE
NOVERIFY ERRORS	NOVERIFY ERRORS
PAGE	PAGE
IMAGES 1	IMAGES 1
LENGTH 24	LENGTH 60
NUMBER 1	NUMBER 1
TITLE	TITLE
WIDTH 79	WIDTH 132
REPORT	REPORT
COPIES 1	COPIES 1
DEVICE TERMINAL	DEVICE PRINTER
LIMIT 100	LIMIT 1000
NAME QUIZLIST	NAME QUIZLIST
SPACING 2	SPACING 2

Interactive	Batch
SPACE	SPACE
STACKSIZE 400	STACKSIZE 400
STATISTICS	STATISTICS
VERIFY DELETE	VERIFY DELETE[1]
WAIT	NOWAIT

[1] *In batch, VERIFY DELETE is ignored.*

Options

Set Options		
BLANKS|NOBLANKS	CLOSE|NOCLOSE	DATABASE
DBMODE	DICTIONARY	DOWNSHIFT|UPSHIFT|NOSHIFT
DUPLICATES|NODUPLICATES	FILE	FORMFEED|NOFORMFEED
HEAD|NOHEAD	JOB|NOJOB	LIST|NOLIST
NOVERIFY	PAGE	PRINT|NOPRINT
REPORT	SAVE CLEAR	SPACE|NOSPACE
STACKSIZE	STATISTICS|NOSTATISTICS	STOPONERROR|NOSTOPONERROR
SUBFILE|NOSUBFILE	VERIFY	WAIT|NOWAIT
WARNINGS|NOWARNINGS		

BLANKS|NOBLANKS

BLANKS prints report lines that are blank; NOBLANKS does not. This option applies to the report-group, not the lines skipped due to carriage control.

Default: BLANKS

CLOSE|NOCLOSE

CLOSE closes the output file after each report is produced; NOCLOSE does not. A subsequent SET REPORT DEVICE statement resets the defaults.

Default: CLOSE for printer and disk output; NOCLOSE for terminal and tape output.

DATABASE database-name

For SQL support, each SQL statement requires a name to attach to the database. The database name must exist as a logical name in the current dictionary.

The database name can also be set when loading the dictionary, or by using the IN database option of the DECLARE CURSOR statement, or in the PowerHouse resource file.

For more information about setting the database, see the section, "SQL Overview", in Chapter 1, "About PowerHouse 4GL and Relational Databases", in the *IBM Cognos PowerHouse 4GL PowerHouse and Relational Databases* document.

DBMODE (UNIX, Windows)

Used to specify the open mode for Eloquence databases. Eloquence accepts open modes from 1 to 9. DBMODE 9 only allows PowerHouse 4GL to read the database, but allows other concurrent users of the database to read and update data. Eloquence only fully supports modes 1, 3, 8 and 9. All other modes are mapped to one of these supported modes. The other modes are retained for backward compatibility. For more information about open modes and mapping, refer to your Eloquence documentation.

Default: 9

DICTIONARY filespec [TYPE PHD|PDC]

Changes the data dictionary used for the current QUIZ session. The SET DICTIONARY statement can be used any number of times in a single session, and is helpful when more than one dictionary is being referenced. This option overrides any other dictionary setting method.

filespec

Names the dictionary that you want to use.

Default: PHD (**OpenVMS**) or phd.pdc (**UNIX, Windows**)

[TYPE PHD|PDC] (OpenVMS)

Specifies the default dictionary type. When the TYPE option is specified in a PowerHouse 4GL component, it applies to all SET DICTIONARY statements in the component.

When searching for a dictionary, PowerHouse 4GL limits searches to the dictionary type specified by the TYPE option. If the TYPE option is not specified, PowerHouse 4GL searches first for a PHD dictionary, then a PDC dictionary.

Default: PHD

DOWNSHIFT|UPSHIFT|NOSHIFT

Specifies that the names of entered identifiers be shifted to lowercase, uppercase, or left as entered. This option overrides the dictionary or the program parameter setting.

DUPLICATES|NODUPLICATES

DUPLICATES reports the record complex if the report detail line is identical to the preceding report detail line; NODUPLICATES does not.

Default: DUPLICATES

FILE record-structure [WAIT|NOWAIT [ON RECEIVE]] (OpenVMS)

WAIT ON RECEIVE indicates that when reading messages from a mailbox, the process will wait for a message to be posted if one is not already there. When using this option, an empty mailbox is not treated as an End-of-Data condition; instead, an End-of-File condition is always processed as an End-of-Data condition and could be used to control processing. NOWAIT ON RECEIVE specifies that a message should be returned if one exists, but if the mailbox is empty, control should be returned to the process immediately.

FORMFEED|NOFORMFEED

FORMFEED uses carriage control to skip the report to a new page. With FORMFEED in effect, QUIZ prespaces for carriage control (that is, it performs the carriage control before it prints). NOFORMFEED generates blank lines to fill out the current page. A subsequent SET REPORT DEVICE statement resets the defaults.

Default: FORMFEED for printers; NOFORMFEED for terminal, disk, and tape output.

HEAD|NOHEAD

HEAD prints default page headings; NOHEAD does not. NOHEAD sets NOFORMFEED so that QUIZ can be used for special forms handling. SET HEAD will reset to FORMFEED. If automatic page breaks are required, the PAGE LENGTH or FORMFEED options must be explicitly set because the NOHEAD option turns off automatic page ejection.

Default: HEAD

JOB|NOJOB (UNIX, Windows)

JOB streams the report as a batch job; NOJOB processes the report interactively. For information on submitting reports as a batch job, see (p. 126).

Limit: JOB and NOJOB are not saved in compiled reports.

Default: NOJOB

JOB [option] (OpenVMS)

For information on submitting reports as a batch job, see (p. 126).

JOB options		
AFTER	CHARACTERISTICS|NOCHARACTERISTICS	CPUTIME
HOLD|NOHOLD	IDENTIFY|NOIDENTIFY	KEEP|NOKEEP
LOGFILE|NOLOGFILE	NAME	NOTIFY|NONOTIFY
PARAMETERS	PRINTER|NOPRINTER	PRIORITY
QUEUE	RESTART|NORESTART	USER
WSDEFAULT	WSEXTENT	WSQUOTA

AFTER [absolute-time][+|- delta-time] (OpenVMS)

Specifies the time at which the job is to start executing. The default is the time that the job reaches the top of the queue. An absolute-time is a specific date and/or time of day. The general form is:

{dd-mmm[-yyyy][:[hours][:[minutes][:[seconds][.[hundredths]]]]]}|{TODAY|TOMORROW|YESTERDAY}

A delta-time is an offset from the current time to a time in the future. The general form is:

[days][-[hours]][:[minutes][:[seconds][.[hundredths]]]]

You must indicate an absolute time, a delta time, a combination of both absolute and delta time, or the time indicators TODAY, TOMORROW, and YESTERDAY.

For example, the following job would print at noon on May 15, 2001:

```
> SET JOB AFTER 15-MAY-2001:12
```

The next job would print three hours from the current time:

```
> SET JOB AFTER +3
```

This last job would print at 11:00 p.m. on the current date:

```
> SET JOB AFTER TOMORROW -1
```

CHARACTERISTICS number|string[[,]number|string]... NOCHARACTERISTICS (OpenVMS)

CHARACTERISTICS specifies one or more of the characteristics, to a maximum of 127, that you can use in defining the job queue for executing a job.

Limit: Specified numbers can range from 0 to 127. If you specify a characteristic name, there is a maximum of 31 characters allowed for a physical characteristic and a maximum of 255 characters allowed for a logical characteristic. A string is a series of displayable characters (letters, numbers, or special characters) enclosed in double or single quotation marks. The DCL command $SHOW QUEUE/CHAR shows you the characteristics that are in effect for your system.

NOCHARACTERISTICS specifies that any previously set characteristics are to be canceled.

CPUTIME n (OpenVMS)

Specifies the maximum CPU time for the job, as in

```
> SET JOB CPUTIME 3
```

In the example above, the job CPUTIME is set to 3 seconds.

For more information, see the *OpenVMS DCL Dictionary*.

Limit: You cannot request more time than permitted by either the base queue limit or your own UAF record-structure.

HOLD|NOHOLD (OpenVMS)

HOLD stipulates that the job is to be held in a queue until specifically released; NOHOLD does not.

Default: NOHOLD

IDENTIFY|NOIDENTIFY (OpenVMS)

IDENTIFY indicates that a message containing the job number and queue message is to be displayed when the job is sent to the batch queue; NOIDENTIFY does not.

Default: IDENTIFY

KEEP|NOKEEP (OpenVMS)

KEEP indicates that the job log file is to be kept after printing; NOKEEP does not, as in

```
> SET JOB NOKEEP
```

Default: NOKEEP

LOGFILE [name]|NOLOGFILE (OpenVMS)

LOGFILE specifies the name to be given to the log file. The name is a unique name identifying a PowerHouse 4GL entity. It is a valid PowerHouse 4GL name used in conjunction with the keyword that appears immediately prior to it. All PowerHouse 4GL names must start with a letter, and can contain letters, digits, and any of the special characters specified in the PDL SYSTEM OPTIONS statement, or in the PHD SYSTEM screen.

Limit: You can specify a name up to 31 characters long except for element usage names, which can be up to 10 characters long. All names should be meaningful to system users, designers, and programmers.

NOLOGFILE specifies that no log file is to be kept for the job. This also implies the NOKEEP option.

In the following example, the job log information is written to the LOGTEMP file:

```
> SET JOB LOGFILE LOGTEMP
```

Default: The job name with an extension of .LOG.

NAME filespec (OpenVMS)

Used to give the job a name. A filespec is a name of an OpenVMS file (which may consist of the node, device, directory, filename, type, and version) or a logical name.

Limit: The maximum length for a filespec in PowerHouse 4GL for OpenVMS is 255 characters. Filespecs are restricted to alphanumeric and punctuation characters. The characters semi-colon (;), dollar sign ($), and leading question mark (?) have special meanings in PowerHouse 4GL and are prohibited.

A file specification takes the general form:

[NODE::][DEVICE:][[DIRECTORY]]FILENAME.EXT;1

The square brackets are required when you enter a directory name.

In the following example, the name EMPLIST is assigned to the QUIZ report:

```
> SET JOB NAME EMPLIST
```

Default: The name of the temporary job file created by QUIZ, which has the file extension .JOB.

NOTIFY|NONOTIFY (OpenVMS)

NOTIFY indicates that the user should be notified when the job is completed; NONOTIFY does not.

Default: NONOTIFY

PARAMETERS string [,string]... (OpenVMS)

Specifies a list of parameters, to a maximum of eight, to be passed to the job. Parameters are typically used when you are running a command file that requires them. A string is a series of displayable characters (letters, numbers, or special characters) enclosed in double or single quotation marks.

Limit: 255 characters per string

PRINTER name|NOPRINTER (OpenVMS)

PRINTER name specifies the name of the print queue to which the log file is to be directed when the job is completed; NOPRINTER does not. The name is a unique name identifying a PowerHouse 4GL entity. It is a valid PowerHouse 4GL name used in conjunction with the keyword that appears immediately prior to it. All PowerHouse 4GL names must start with a letter, and can contain letters, digits, and any of the special characters specified in the PDL SYSTEM OPTIONS statement, or in the PHD SYSTEM screen. You can specify a name up to 31 characters long except for element usage names, which can be up to 10 characters long. All names should be meaningful to system users, designers, and programmers.

Default: SYS$PRINT

PRIORITY n (OpenVMS)

Specifies the job's scheduling priority. The range is from 0 through 255.

QUEUE queuename (OpenVMS)

Specifies the name of the queue where the job is waiting for execution.

Limit: The maximum size for an actual queuename is 31 characters; for a logical queuename, the maximum size is 255 characters.

Default: SYS$BATCH

RESTART|NORESTART (OpenVMS)

RESTART indicates that the job is to restart after a system crash or after a $ STOP /QUEUE /REQUEUE command has been issued; NORESTART does not.

Default: RESTART

USER username (OpenVMS)

Specifies the name of the user for whom you're submitting the job. Privileges are necessary to use this option.

WSDEFAULT n (OpenVMS)

Defines a working set default for the batch job. For more information, see the *OpenVMS DCL Dictionary*.

WSEXTENT n (OpenVMS)

Defines a working set extent for the batch job. For more information, see the *OpenVMS DCL Dictionary*.

WSQUOTA n (OpenVMS)

Defines the maximum working set size for the batch job. For more information, see the *OpenVMS DCL Dictionary*.

LIST|NOLIST

LIST displays the contents of a QUIZ source statement file referenced by a USE statement; NOLIST does not.

Limit: The LIST and NOLIST options are not saved in compiled reports.

Default: LIST

LIST|NOLIST SQL

The SQL option controls the listing of SQL statements. It shows the SQL requests sent from PowerHouse 4GL to the database, including the effects of any substitutions.

Default: NOLIST SQL

Limit: Only applies to cursors declared using a DECLARE CURSOR statement.

NOVERIFY [DELETE] [ERRORS]

Disables requests for authorization to proceed with processing. An entry of SET NOVERIFY without another specified option included in the statement sets the specification to NOVERIFY DELETE ERRORS. NOVERIFY isn't saved in compiled reports.

NOVERIFY and VERIFY are mutually exclusive if they include identical options.

Default: NOVERIFY ERRORS

DELETE

Replaces an existing file without requesting authorization to proceed with processing.

ERRORS

Doesn't pause when errors are encountered in a file processed by a USE statement.

PAGE

For information about how to change the default page-control settings, see the SET PAGE statement on (p. 130).

PRINT|NOPRINT

The PRINT option sends the source listing to:

OpenVMS:	The file defined with the logical name SYSPRINT, or to SYS$PRINT if SYSPRINT is not defined.
UNIX, Windows:	The default printer. The default printer is obtained from the value of the environment variable PH_PRINTER. If this variable is not set, the system default printer is used.

Limit: PRINT and NOPRINT are not saved in compiled reports.

Default: NOPRINT

REPORT

For information about how to change the default report-control settings, see the SET REPORT statement on (p. 132).

SAVE CLEAR

Removes any source statements from the temporary save file that have been entered at the point that SET SAVE CLEAR is entered.

Limit: The SAVE CLEAR option isn’t saved in compiled reports.

SPACE|NOSPACE

SPACE skips to a new page if the current report-group can't fit on the current page. NOSPACE prints as much of the current report-group as possible before skipping to a new page. SPACE and NOSPACE are saved in compiled reports.

Default: SPACE

STACKSIZE n

Changes the area used for expression processing. The stacksize is an internal buffer where expressions are stored. This option is needed only when QUIZ issues a message that the stacksize is too small.

This option is saved in compiled reports.

Default: 400

STOPONERROR|NOSTOPONERROR

By default in QUIZ, a GO statement starts report processing even if there has been a parser error. STOPONERROR instructs QUIZ not to start processing if there has been a parsing error in the current report. NOSTOPONERROR, the default, instructs QUIZ to start processing even if there has been a parsing error in the current report. The STOPONERROR|NOSTOPONERROR option of the SET statement overrides the **stoponerror|nostoponerror** program parameter or STOP ON ERROR Resource file statement.

If STOPONERROR is in effect and a parsing error has occurred, the error must be cleared using an ACCESS, CANCEL, EXECUTE, or USE statement, or the current report must be corrected using the REVISE statement.

Default: NOSTOPONERROR

STATISTICS|NOSTATISTICS

STATISTICS issues statistics at the end of a report; NOSTATISTICS does not. The statistics consist of the numbers of records selected, lines printed, and pages printed. Both options are saved in compiled reports.

Default: STATISTICS

SUBFILE|NOSUBFILE

For information about directing output to a subfile and changing the default subfile settings, see the SET SUBFILE statement on (p. 142).

VERIFY [DELETE] [ERRORS]

Requests authorization to proceed with processing. An entry of SET VERIFY without another specified option in the statement sets the specification to VERIFY DELETE ERRORS.

VERIFY and NOVERIFY are mutually exclusive if they include identical options.

Default: VERIFY DELETE

DELETE

Requests authorization to proceed with processing when it's necessary to replace an existing file.

ERRORS

Pauses when errors are encountered in a file processed by a USE statement.

WAIT|NOWAIT

WAIT pauses after displaying each page of the report; NOWAIT does not. A subsequent SET REPORT DEVICE TERMINAL or SET REPORT DEVICE PRINTER statement resets the defaults.

Defaults: WAIT for terminals; NOWAIT for output to other devices.

WARNINGS|NOWARNINGS

WARNINGS issues warning messages as required; NOWARNINGS does not.

The WARNINGS and NOWARNINGS options are saved in compiled reports.

Default: WARNINGS

Discussion

The SET statement specifies the settings that control the functioning of a QUIZ report. The SET statement entered without any options does nothing.

Individual SET statements can be listed in any sequence. To see which SET statements are in effect, enter

```
> SHOW STATUS
```

Combining SET Statements

Although the statements are easier to read if each statement is separated, the different SET statements (SET PAGE, SET REPORT, SET SUBFILE) can be combined into one statement. If there is more than one option containing the leading keyword PAGE, REPORT, or SUBFILE in a combined statement, you only have to specify the keyword once.

Each statement is effective unless mutually exclusive options are entered. If mutually exclusive options are entered, only the last option that is entered is effective.

SET Statement Options Saved in Compiled Reports

The following lists the SET statement options that are saved in compiled reports, and those that are not saved:

Saved	
BLANKS|NOBLANKS	CLOSE|NOCLOSE
DBMODE (**UNIX, Windows**)	DUPLICATES|NODUPLICATES
FORMFEED|NOFORMFEED	HEAD|NOHEAD
PAGE	REPORT
SPACE|NOSPACE	STACKSIZE
STATISTICS|NOSTATISTICS	SUBFILE|NOSUBFILE
WARNINGS|NOWARNINGS	WAIT|NOWAIT

Not Saved	
DICTIONARY	JOB|NOJOB
LIST|NOLIST	NOVERIFY [DELETE] [ERRORS]
PRINT|NOPRINT	SAVE CLEAR
VERIFY [DELETE] [ERRORS]	

Ways to Run a Batch Job

There are two ways to run a QUIZ report as a batch job:

- use the SET JOB statement to build a job command file

- submit a job command file containing operating system commands, QUIZ statements, and execution-time parameters

Generating a Batch Job Using SET JOB

Using the set job statement allows online prompting for execution-time parameters while the QUIZ report is executed in batch. It works as follows:

1. The SET JOB statement suppresses the interactive execution of any subsequent QUIZ report.
2. When the SET JOB statement is entered in QUIZ, the QUIZSAVE file is cleared.
3. If QUIZ encounters the designated file QUIZJOB, the contents are written to the QUIZSAVE file.

OpenVMS:	Operating system commands prefixed by two operating system prompts ($$) write to the currently active save file (prefixed with a single prompt). Operating system commands prefixed with a single prompt ($) execute immediately.
UNIX:	Operating system commands prefixed by two exclamation marks (!!) write the Shell command to the currently active save file with one exclamation mark. Operating system commands prefixed with a single exclamation mark (!) suspend the current process, and execute the Shell command immediately.
Windows:	Operating system commands prefixed by two exclamation marks (!!) write the command to the currently active save file with one exclamation mark. Operating system commands prefixed with a single exclamation mark (!) execute immediately.

4. All other statements are parsed and written to QUIZSAVE.
5. When an EXECUTE or GO statement (or a USE statement that refers to a compiled QUIZ report) is encountered, QUIZ prompts for any execution-time parameters. When the parameters are entered and accepted, they're written to QUIZSAVE.
6. If the SET JOB statement is in effect, and a report specifies the creation of a subfile, QUIZ creates an interim minidictionary. Succeeding reports that reference the subfile in the same job can determine formats from the minidictionary. Once the job has been submitted and the PowerHouse 4GL session has been terminated, the interim minidictionary is deleted.
7. When an EXIT or QUIT statement is encountered, the QUIZSAVE file is saved as a job file that is submitted immediately for execution.

   OpenVMS: Job files are created in the directory SYS$SCRATCH (and given the extension .JOB) where they are held until the job is complete, and then deleted. Job files may not be deleted after a system crash.

   Windows: The job is submitted as a separate spawned process just before the product exits. By default, the spawned process window is hidden. To show the window, use the **setjobshow** program parameter. The default is **nosetjobshow**. There is also a resource file statement SETJOBSHOW.

The REVISE statement cannot be used when the SET JOB statement is in effect.

The **noosaccess** and **nodcl** (**OpenVMS**) program parameters cause QUIZ to reject any command that's preceded by an operating system prompt. If QUIZ is invoked with the **noosaccess** and **nodcl** (**OpenVMS**) program parameters, then the SET JOB feature is also rejected.

Any operating system commands necessary to execute QUIZ as a batch job must either be entered after the SET JOB statement or be included in the QUIZJOB file. Consider the following examples.

Examples

These examples show you how to generate a batch job using the SET JOB statement.

UNIX, Windows: It is not necessary to include a command to run QUIZ. QUIZ automatically generates the appropriate command after the SET JOB statement. Since the QUIZ command is generated automatically, use a SET DICTIONARY statement to specify a non-default dictionary.

OpenVMS

```
> SET JOB
> $$QUIZ DICT=personnel
> ACCESS EMLOYEES
> CHOOSE EMPLOYEE PARM PROMPT "Enter Employee Number: "
> DEFINE STARTDATE DATE=PARM PROMPT &
>   "Enter an employment starting date:  "
> SELECT IF DATEJOINED GE STARTDATE
> REPORT LASTNAME BRANCH POSITION DATEJOINED
> GO
> EXIT
```

UNIX, Windows

```
> SET JOB
> SET DICTIONARY personnel
> ACCESS EMPLOYEES in QCORA
> CHOOSE EMPLOYEE PARM PROMPT "Enter Employee Number: "
> DEFINE DATETIME DATE = PARM PROMPT &
>   "Enter an employment starting date:  "
> SELECT IF DATEJOINED GE DATETIME
> REPORT LASTNAME BRANCH POSITION DATEJOINED
> GO
> EXIT
```

Submitting Jobs as Text Files

The job command file that you create for your batch job must include execution-time values (responses) for all CHOOSE and DEFINE statement prompts. These values are placed in the job file starting on the line below the GO statement (or the EXECUTE statement, if compiled reports are used).

You must place each response to the CHOOSE statement prompt on its own line; multiple responses on a single line are not allowed. A blank line following the last response to the CHOOSE statement indicates that no more index values will be entered.

For each DEFINE statement, you must include a single response in the job file. For example, if your QUIZ report contains ten DEFINE statements with the PARM option, you must include ten values (each on its own line) in your job file.

In QUIZ, responses to CHOOSE statement prompts must appear first, followed by the required blank line and a value for each DEFINE statement. It's good coding practice to always enter CHOOSE statements before DEFINE statements.

OpenVMS

The sequence of these responses must also correspond to the appearance of the DEFINE statements in the statement list. Consider the following text file designed to run QUIZ in batch mode:

```
> $QUIZ
> ACCESS EMPLOYEES
> DEFINE STARTDATE DATE = PARM
> CHOOSE EMLOYEE PARM
> SELECT IF DATEJOINED GE STARTDATE
> REPORT LASTNAME BRANCH POSITION DATEJOINED
> GO
1
4
9
11
22

900101
> EXIT
```

The values 1, 4, 9, 11, and 22 print in response to the PARM option of the CHOOSE statement. The blank line after the value 22 indicates the end of the execution-time parameters for the CHOOSE statement. The value 900101 prints in response to the PARM option of the DEFINE statement.

UNIX

The sequence of these responses must also correspond to the appearance of the DEFINE statements in the statement list. Consider the following text file designed to run QUIZ in batch mode:

```
quiz << E_O_F
ACCESS EMPLOYEES
DEFINE STARTDATE DATE = PARM
CHOOSE EMPLOYEE PARM
SELECT IF DATEJOINED GE STARTDATE
REPORT LASTNAME BRANCH POSITION DATEJOINED
GO
1
4
9
11
22
90/01/01
EXIT
E_O_F
```

The values 1, 4, 9, 11, and 22 are responses to the PARM option of the CHOOSE statement. The blank line after the value 22 indicates the end of execution-time parameters for the CHOOSE statement. The value 90/01/01 is the response to the PARM option in the DEFINE statement.

Windows

The sequence of these responses must correspond to the appearance of the DEFINE statements in the statement list. Consider the following text file designed to run QUIZ in batch mode:

```
quiz
ACCESS EMPLOYEES
DEFINE STARTDATE DATE = PARM
CHOOSE EMPLOYEE PARM
SELECT IF DATEJOINED GE STARTDATE
REPORT LASTNAME BRANCH POSITION DATEJOINED
GO
1
4
9
11
22
90/01/01
EXIT
```

The values 1, 4, 9, 11, and 22 are responses to the PARM option of the CHOOSE statement. The blank line after the value 22 indicates the end of execution-time parameters for the CHOOSE statement. The value 90/01/01 is the response to the PARM option in the DEFINE statement.

Example

This example uses the SET statement to establish report features. In this example

- NOBLANKS compresses large reports by removing blank lines when the report is displayed.
- NOHEAD stops QUIZ from producing the default column headings.
- DEVICE PRINTER produces the report on the printer; COPIES prints two copies of the report.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
```

```
>   AND TO PARTNUMBER OF PARTS
>
>  SORT &
>   ON CUSTOMERNAME &
>   ON INVOICEDATE D
>
> DEFINE INVOICEAMOUNT NUMERIC * 9 &
>   PICTURE "$^^^,^^^.^^" &
>   = QUANTITYSHIPPED OF INVOICEDETAIL * &
>   (UNITCOST OF PARTS * UNITMARKUP OF PARTS)
>
> REPORT CUSTOMERNAME &
>   PRINT AT CUSTOMERNAME &
>   INVOICENUMBER &
>   INVOICEDATE &
>   INVOICEAMOUNT &
>   HEADING "Amount^Owing"
>
> FOOTING AT CUSTOMERNAME &
>   "Total:" &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL &
>   SKIP 2
>
> FINAL FOOTING &
>   SKIP 2 &
>   "Grand Total:" &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL
>
> SET NOBLANKS
>
> SET NOHEAD
>
> SET REPORT DEVICE PRINTER
>
> SET REPORT COPIES 2
>
> SET REPORT LIMIT 3000
>
> GO
```

SET PAGE

Changes the default page-control settings.

Syntax

SET PAGE [option]...

Options

All SET PAGE options are saved in compiled reports.

The options are IMAGES, LENGTH, NUMBER, TITLE, and WIDTH.

IMAGES n

States the number of times that a page image is repeated horizontally in a report.

IMAGES applies only to report-groups that are reported with the REPORT statement. Report-groups in other reporting statements are not repeated. When you specify a number for PAGE IMAGES, the resulting report width is equal to the number of images multiplied by the current page width setting.

For example, the statements

```
> SET PAGE IMAGES 3
> SET PAGE WIDTH 40
```

result in a page width of 120 characters.

Limit: 10

Default: 1

LENGTH n

Specifies the number of lines in each page of the report. A SET REPORT DEVICE TERMINAL or SET REPORT DEVICE PRINTER statement resets the length to the default.

Limit: 32,767 lines

Default: If the terminal is configured (in the system, not by page width) to be 80 characters wide or less, QUIZ assumes a page length of 24; otherwise, the page length is 60. If LENGTH 0 is specified, indicating no predetermined page length, LENGTH is set to 10,000 and headings are printed only on the first page.

NUMBER n

Sets the initial page number of a report. This option is useful if the report is part of a preceding report.

Limit: -2147483647 to 2147483647

Default: 1

TITLE string

Centers a page title under the standard title. TITLE is canceled by a new ACCESS statement. Multi-line titles can be created by including a multi-line symbol (^) in the string. Each multi-line symbol begins a new line. Each new line is centered.

WIDTH n

Specifies the maximum number of characters, including spaces, to be accepted on a report line or page image. A SET REPORT DEVICE TERMINAL or SET REPORT DEVICE PRINTER statement resets the width to the default.

Limit: 1 to 32,767

Default: 132 for printers; terminal width - 1 for terminals (normally 79).

Discussion

The SET PAGE statement options change the physical dimensions (length and width) and basic reporting features (report title, starting page number) of a QUIZ report.

Example

The SET PAGE statement is used in this example to generate mailing labels. In this example

- WIDTH sets the page width to 40 columns to allow for two columns of labels.
- LENGTH determines that each page will be 12 lines long.
- IMAGES prints two horizontal page images.

```
> ACCESS CUSTOMERS
>
> SORTED ON CUSTOMERNAME
>
> DEFINE CITYPROV CHARACTER * 38 &
>   = TRUNCATE(CITY) + ", " + TRUNCATE(PROVSTATE)
>
> REPORT &
>   CUSTOMERNAME SKIP &
>   STREET SKIP &
>   CITYPROV SKIP &
>   COUNTRY SKIP &
>   POSTALZIP SKIP PAGE
>
> SET REPORT DEVICE PRINTER
>
> SET PAGE WIDTH 40
>
> SET PAGE LENGTH 12
>
> SET PAGE IMAGES 2
>
> SET NOHEAD
>
> GO
```

SET REPORT

Changes the default report-control settings.

Syntax

SET REPORT [option]...

Options

All SET REPORT options are saved in compiled reports.

SET REPORT options		
ALIGN|NOALIGN	COPIES	DESTINATION
DEVICE	NAME	SPACING

ALIGN|NOALIGN

ALIGN aligns the items in all heading and footing statements with the same items in the REPORT statement. If there is an obstacle in the way, such as a label or an item positioned with the TAB option, then the ALIGN option puts the item to the right of the obstacle. Similarly, items in the heading or footing that aren't in the REPORT statement are put to the right of previous items, overflowing onto a new line if necessary.

ALIGN remains in effect until you enter either SET DEFAULT or SET REPORT NOALIGN, or until you finish your QUIZ session.

The ALIGN options on individual HEADING and FOOTING statements override the ALIGN option on SET REPORT statement.

NOALIGN does not align items.

Default: NOALIGN

COPIES n (UNIX)

Specifies the number of copies to be printed when routing output to a printer.

UNIX: The COPIES option is ignored if a string is specified on the SET REPORT DEVICE PRINTER statement.

For information about the COPIES option on OpenVMS, see (p. 136).

Limits: Valid for SET REPORT DEVICE PRINTER only. The limit is 128.

Default: 1

DESTINATION printername (UNIX)

Specifies the printer where the report is to be sent.

Use the DESTINATION option instead of the PRINTER string option to identify a specific printer if operating system access if not allowed.

The DESTINATION option is ignored if a string is specified on the SET REPORT DEVICE PRINTER statement.

Limit: Valid for SET REPORT DEVICE PRINTER only.

Default: Determined by the system used.

DEVICE device

Specifies the output device for the report. Changing the device specification changes some of the basic defaults.

For more information about QUIZ defaults, see (p. 137).

If you don't specify a device, and you compile your QUIZ report interactively, the output is directed to the terminal. The same report compiled in batch defaults to the printer.

Limit: Only one device may be used.

Default: TERMINAL for interactive reports; PRINTER for batch jobs. SET REPORT DEVICE statements reset (to defaults) any prior settings of CLOSE|NOCLOSE and FORMFEED|NOFORMFEED.

device

The device is one of DISC, DISK, NONSPOOLED PRINTER, TAPE, or TERMINAL.

DISC|DISK

Writes your reports to a disc file. This is particularly useful if you want to save the output for later use. DISK may be used as an alternative spelling.

Default: DISC for non-interactive QUIZ sessions. Unless the NAME option is used, QUIZ creates a file named QUIZLIST. When the NAME option is specified, QUIZ creates a file with the specified name.

PRINTER [EJECTPAGE|NOEJECTPAGE] (OpenVMS)
PRINTER [string] [EJECTPAGE|NOEJECTPAGE] (UNIX)
PRINTER [string|DIALOG|NODIALOG] [EJECTPAGE|NOEJECTPAGE] (Windows)

Sends your reports to a printer.

PRINTER resets the PAGE WIDTH, PAGE LENGTH, and WAIT|NOWAIT option to the defaults. QUIZ resets the page length to 66 lines and the page width to 132 columns.

OpenVMS: You can specify several options for controlling the printer and /or printer queue to which your report is sent. These options aren't lost when you switch devices. For information on the OpenVMS printer options, see (p. 135)

string (UNIX)

The optional string allows you to specify the Shell command or script file to be used to print the report. If a string is specified, the COPIES, DESTINATION, and NAME options are ignored. QUIZ assumes these arguments are placed in the lp(1) Shell command or script file, as needed.

Limit: The string must be 60 characters or less in length.

The default printer strings are:

UNIX SVR4: "lp -t%s -n%d -d%s

UNIX BSD: lpr -T%s -#%d -P%s

If the string is omitted, PowerHouse 4GL follows these steps to determine the appropriate values:

1. If a string was used in the last SET REPORT DEVICE PRINTER [string] statement, PowerHouse 4GL uses that value. The DESTINATION, COPIES, and NAME options are ignored.
2. If the value of the environment variable PH_PRINTER is defined, PowerHouse 4GL uses this value. The DESTINATION, COPIES, and NAME options are ignored. QUIZ assumes these arguments are available in the PH_PRINTER environment variable.
3. The system default string is used. If specified, PowerHouse 4GL uses the DESTINATION, COPIES, and NAME options; otherwise, the defaults for these options are used.

   If the string is null, options 2 and 3 are used to obtain a value.

   If the printer string is "slave.txt", your reports are sent to a printer attached to a terminal or to a PC emulating a terminal. For more information about slave printer support for HP-UX, see (p. 138).

   A string is not valid if the **noosaccess** program parameter is specified or the OSACCESS resource file option equals OFF. A syntax error will result.

   When a string is not allowed, the DESTINATION, COPIES, and NAME options can be used to set the more common printer settings.

   string|DIALOG|NODIALOG (Windows)

The optional string allows you to specify the name of a printer. The specified printer's configuration for the current Windows session is used. Using an empty string negates a previous string option.

Limit: The string must be 60 characters or less in length.

DIALOG causes QUIZ to open the standard Windows print dialog box when the report output file is opened. The print dialog allows the printer and number of copies to be selected. The DIALOG|NODIALOG option can be used to override the **printdialog|noprintdialog** program parameter or PRINT DIALOG Resource file statement.

Default: NODIALOG. Unless string or DIALOG is specified, the printer configuration for the current Windows session is used as the destination printer. To direct the report to another printer, change your printer configuration using the Control Panel.

EJECTPAGE|NOEJECTPAGE

When QUIZ sends a report to the printer, it will do a page eject before it starts printing the report. This will cause a blank page to appear between the banner and the start of the report. To prevent this, use NOEJECTPAGE.

Default: EJECTPAGE

TAPE (OpenVMS)

Sends your reports to tape. QUIZ creates a tape file named QUIZLIST unless the NAME option is used.

TAPE [string] (UNIX)

Sends your reports to tape. The optional string allows you to specify the destination of the report.

If the string is omitted, QUIZ takes the following steps to obtain a default value:

1. If a previous string was used in the last SET REPORT DEVICE TAPE [string] statement, PowerHouse 4GL uses that value.

   Limit: The string must be 60 characters or less in length.
2. If the value of the environment variable PH_TAPE is defined, PowerHouse 4GL uses that value.
3. The system default is used.

If the string is null, options 2 and 3 are used to attempt to obtain a value.

TERMINAL (OpenVMS, Windows)

Prints reports on your monitor.

Default: TERMINAL resets to default settings any prior settings of PAGE WIDTH, PAGE LENGTH, and WAIT|NOWAIT. These batch defaults are left on for the remainder of the QUIZ session.

TERMINAL (UNIX)

Prints reports on the terminal identified by the open name set by a SET REPORT name statement (for example, SET REPORT NAME/dev/tty4). The exception to this is when the report name is QUIZLIST. In this instance, if a SET DEVICE REPORT TERMINAL statement is issued, your default terminal is used. This allows you to easily identify the default terminal.

Default: TERMINAL resets to default settings any prior settings of PAGE WIDTH, PAGE LENGTH, and WAIT|NOWAIT. These batch defaults are left on for the remainder of the QUIZ session.

Default: 40 characters

LIMIT n|NOLIMIT (OpenVMS, UNIX, Windows)

Specifies the maximum number of record complexes that can be reported. This affects all output devices. The number of records that can be reported can't exceed the limit (n) even if SELECT is used. NOLIMIT removes any limit to the number of record complexes that can be reported. Using this option overrides the defaults.

Limit: 2,147,483,647 if the LIMIT option is used.

Default: 100 for interactive sessions; 1000 for batch sessions

NAME filespec

Sets the name of the filespec for output. How this is used depends upon which device has been set. The NAME option is saved in compiled reports.

UNIX, Windows:	The NAME option is ignored if a string is specified on the SET REPORT DEVICE PRINTER statement.

Default: QUIZLIST

SPACING n

Sets the number of blank spaces inserted between each report-item if no TAB settings are included.

Limit: 0 to 32

Default: 2

DEVICE PRINTER Options (OpenVMS)

SET REPORT DEVICE PRINTER options		
AFTER	BURST|NOBURST	COPIES
FLAG|NOFLAG	FORMS	HOLD|NOHOLD
IDENTIFY|NOIDENTIFY	LOWERCASE|NOLOWERCASE	NOCHARACTERISTIC|CHARACTERISTIC
NOTE	NOTIFY|NONOTIFY	OPERATOR|NOOPERATOR
PRIORITY	QUEUE	RESTART|NORESTART
TRAILER|NOTRAILER	USER	

AFTER [absolute-time][+|- delta-time] (OpenVMS)

Specifies the time at which the job is to start printing. The default is the time that the job reaches the top of the queue. An absolute-time is a specific date and/or time of day. The general form is:

{dd-mmm[-yyyy][:[hours][:[minutes][:[seconds]
[.[hundredths]]]]]}|{TODAY|TOMORROW|YESTERDAY}

A delta-time is an offset from the current time to a time in the future. The general form is:

[days][-[hours]]
[:[minutes][:[seconds][.[hundredths]]]]

You must indicate an absolute time, a delta time, a combination of both absolute and delta time, or the time indicators TODAY, TOMORROW, and YESTERDAY. For example,

```
> SET REPORT DEVICE PRINTER AFTER 15-MAY-2001:12
```

executes at noon on May 15, 2001;

```
> SET REPORT DEVICE PRINTER AFTER +3
```

executes three hours from the current time

```
> SET REPORT DEVICE PRINTER AFTER TOMORROW -1
```

executes at 11:00 p.m. on the current date.

For more information about time representation, see the *OpenVMS User Manual*.

BURST|NOBURST (OpenVMS)

BURST specifies that a burst page be printed prior to the flag page at the front of the report. This enables you to easily identify where one report ends and another begins. The burst page contains the same information as the flag page except that it prints over the perforation. When BURST is specified, FLAG need not be specified. NOBURST does not print the burst page.

CHARACTERISTICS number|string[[,]number|string]...
NOCHARACTERISTICS (OpenVMS)

CHARACTERISTICS specifies one or more of the characteristics, to a maximum of 127, that you can use in defining the printing format of the report. NOCHARACTERISTICS specifies that any previously set characteristics are to be canceled.

Limit: Specified numbers can range from 0 to 127. If you specify a string, there is a maximum of 31 characters for a physical characteristic and a maximum of 255 characters for a logical characteristic. These characteristics are installation specific. The DCL command $ SHOW QUEUE /CHAR shows you the characteristics that are in effect for your system.

COPIES n (OpenVMS)

Specifies the number of copies to be printed when routing output to a printer.

Limits: 255

Default: 1

FLAG|NOFLAG (OpenVMS)

FLAG indicates that a flag page is to be printed at the front of the report. The flag page contains information about the file being printed. This option need not be specified if BURST has been used as a flag page is automatically produced when BURST is specified. NOFLAG suppresses the flag page.

FORMS n|string (OpenVMS)

Specifies which form is to be used for the report. The value entered may be a number or a string of characters. This string may be an actual form name or a logical name for a form. To see the forms that are available for your system, use the DCL command $SHOW QUEUE/FORM.

Limit: If an actual form name is used, the string can be up to 31 characters in length. If a logical name is used, the string can be up to 255 characters in length.

HOLD|NOHOLD (OpenVMS)

HOLD stipulates that the job is to be held in a queue until specifically released; NOHOLD stipulates that the job is not to be held in a queue until it is specifically released.

Default: NOHOLD

IDENTIFY|NOIDENTIFY (OpenVMS)

IDENTIFY indicates that a message containing the job number and queue message is to be displayed when the job is sent to the print queue; NOIDENTIFY does not.

Default: IDENTIFY

LOWERCASE|NOLOWERCASE (OpenVMS)

LOWERCASE indicates that the report is to be printed on a printer that supports both uppercase and lowercase letters. NOLOWERCASE indicates that the report is to be printed on a printer that supports either uppercase and lowercase letters.

NOTE string (OpenVMS)

Specifies that a note is to be printed on the flag page of the report.

Limit: The note can be up to 255 characters long.

NOTIFY|NONOTIFY (OpenVMS)

NOTIFY indicates that a message is to be sent to the user's terminal when the report has been printed. NONOTIFY does not.

Default: NONOTIFY

OPERATOR|NOOPERATOR (OpenVMS)

OPERATOR specifies that a message of up to 255 characters is to be sent to the operator prior to printing. NOOPERATOR does not.

Limit: The report is not printed until the operator responds to the message.

Default: NOOPERATOR

PRIORITY n (OpenVMS)

Sets a priority for the report. A new SET REPORT DEVICE PRINTER or SET REPORT DEVICE TERMINAL statement resets the priority to the default.

Limits: 0 to 255

QUEUE queuename (OpenVMS)

Specifies the name of the queue where the report is waiting for printing, and also determines the device where the report is to be printed.

Default: SYS$PRINT

RESTART|NORESTART (OpenVMS)

RESTART indicates that the report is to be restarted after a system crash or after a $ STOP /QUEUE /REQUEUE command is issued; NORESTART does not.

Default: RESTART

TRAILER|NOTRAILER (OpenVMS)

TRAILER indicates that a trailer page is to be printed at the end of the report. NOTRAILER does not.

Default: NOTRAILER

USER username (OpenVMS)

Specifies the name of the user for whom you're submitting the report. Privileges are necessary to use this option.

Discussion

Changing the REPORT DEVICE Specification

Changing the REPORT DEVICE specification changes several settings. The following table lists the option values that change when the device type changes. Other settings don't change when the device is changed:

TERMINAL	PRINTER	DISC	TAPE
NOCLOSE	CLOSE	CLOSE	NOCLOSE
NOFORMFEED	FORMFEED	NOFORMFEED	NOFORMFEED
WAIT	NOWAIT	NOWAIT	NOWAIT
PAGE WIDTH 79[1]	PAGE WIDTH 132	n/c	n/c
PAGE LENGTH 24	PAGE LENGTH 60	n/c	n/c

[1]*The page width may vary depending on how the terminal was set up prior to invoking QUIZ.*

n/c=no change

SET REPORT statements specify the values of parameters that control the functioning of a QUIZ report. Any number of options can be specified in a single SET statement. If there is more than one option containing the leading keywords PAGE, REPORT, or SUBFILE, the keyword need be specified only once. Individual SET statements can be listed in any sequence. SET statements are not reset by a new ACCESS statement unless indicated in the text.

Routing Reports to a Disc

The DEVICE DISC option of the SET REPORT statement instructs QUIZ to write your reports to a disc file. This feature is particularly useful if you want to save the output for later use.

The DEVICE DISC option opens the disc file specified by the last report name prior to the opening of the report. If there is no name defined, QUIZ creates a disc file with the name that you specified in the NAME option of the SET REPORT statement. If you haven't specified the NAME option, QUIZ creates a disc file name: QUIZLIST (**OpenVMS**) or quizlist.txt (**UNIX, Windows**).

OpenVMS: Note that NAME can only be a file name or a logical name, not a complete file specification with a device and directory. To use a complete file specification, set up a logical name.

The DEVICE DISC option doesn't change the page width or page length. However, the DEVICE DISC option sets the options NOFORMFEED and CLOSE.

If you specify the FORMFEED option after the SET REPORT DEVICE DISC statement, QUIZ writes a carriage control to the disc file. If you don't want a carriage control written to the disc file, specify SET NOFORMFEED. Page skipping is then handled by generating an appropriate number of blank lines.

If CLOSE is in effect, the disc file is closed after each report. Each new report that is written to the disc file overwrites the former report. To produce multiple reports on the same disc file, you must specify SET NOCLOSE. This allows QUIZ to append several reports to a single file.

Routing Reports to a Printer

The SET REPORT DEVICE PRINTER statement instructs QUIZ to spool your reports for printing on the system printer.

When you enter this statement, QUIZ automatically sets the page width to 132 columns and page length to 66 lines to reflect the dimensions of standard printed output. QUIZ also automatically sets the FORMFEED, CLOSE, and NOWAIT options.

OpenVMS: The default print queue is SYS$PRINT. If you have more than one system printer, output can be rerouted using the QUEUE option of the SET REPORT DEVICE PRINTER statement.

The default report name is QUIZLIST. This can be changed using the SET REPORT NAME statement.

Routing Reports to a Terminal

The SET REPORT DEVICE TERMINAL statement instructs QUIZ to display reports on your terminal.

QUIZ resets the page to the width and length that the terminal had been set to prior to running QUIZ.

The NOFORMFEED, NOCLOSE, and WAIT options are set.

Slave Printer Support for HP-UX (UNIX)

```
$ quiz
> SET REPORT DEVICE PRINTER "slave.txt"
```

The slave.txt script is included in the PowerHouse 4GL bin directory. The executable 'awk' must also exist on your system because it is called by this script.

This script understands two basic terminal types: HP and VT. Other terminal types can be added if they support pass-through printing operations.

Straps G and H are affected when slave.txt is executed with an HP terminal. These straps have the values 0 and 1 respectively when this script is completed, except when the environment variables PHSTRAP_G and PHSTRAP_H already exist. In this case, the straps are reset to the values specified by the environment variables. A value of 0 indicates that the strap is to be set to NO and a value of 1 indicates that the strap is to be set to YES.

When you print from an HP terminal, a completion code (S, F, or U) is sent out. The script reads this completion code but the character is also echoed to the terminal. The script does not control the echoing. For systems where the **stty** command is supported with pipes, the following command can be specified as the printer string to QUIZ to disable the echo of the completion character:

```
> quiz
> SET REPORT DEVICE PRINTER "stty -echo;slave.txt;stty echo"
```

Examples

The SET REPORT statement in the following example determines the output device and maximum size of this report, as well as the number of copies that will be printed. In this example

- the report output is spooled to the system printer.
- COPIES prints two copies.
- LIMIT specifies that a maximum of 3000 record complexes are printed in the report.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL
>
> SORT &
>   ON CUSTOMERNAME &
>   ON INVOICEDATE D
>
> REPORT &
>   CUSTOMERNAME &
>   PRINT AT CUSTOMERNAME &
>   INVOICENUMBER &
>   INVOICEDATE &
>   INVOICEAMOUNT &
>   HEADING "Amount^Owing"
>
> FOOTING AT CUSTOMERNAME &
>   SKIP 2 &
>   "Total:" &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL &
>   SKIP 2

> FINAL FOOTING &
>   "Grand Total:" &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL
>
> SET NOBLANKS
>
> SET NOHEAD
>
> SET REPORT DEVICE PRINTER
>
> SET REPORT COPIES 2
>
> SET REPORT LIMIT 3000
>
> GO
```

If you expect a large amount of output from the report, you might want to check the results of the report and its format on the screen. The EXECUTE statement with the NOGO option allows you to change the report conditions at execution-time. For example, the following statements activate the compiled report LABELS, but allow you to override the SET REPORT DEVICE PRINTER statement and direct output to the screen rather than the printer:

```
> EXECUTE LABELS NOGO
>
> SET REPORT DEVICE TERMINAL LIMIT 20
>
> GO
```

The following examples show the effect of the ALIGN and NOALIGN options on the SET REPORT statement.

With the NOALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the billing subtotal and total billings follow.

```
> SET REPORT NOALIGN
> ACCESS BILLINGS
> REPORT TAB 20 PROJECT TAB 40 BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT SKIP 2 "BILLING SUBTOTAL " &
>   BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2 "TOTAL BILLINGS " &
>   BILLING SUBTOTAL  &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

The resulting QUIZ report looks like this:

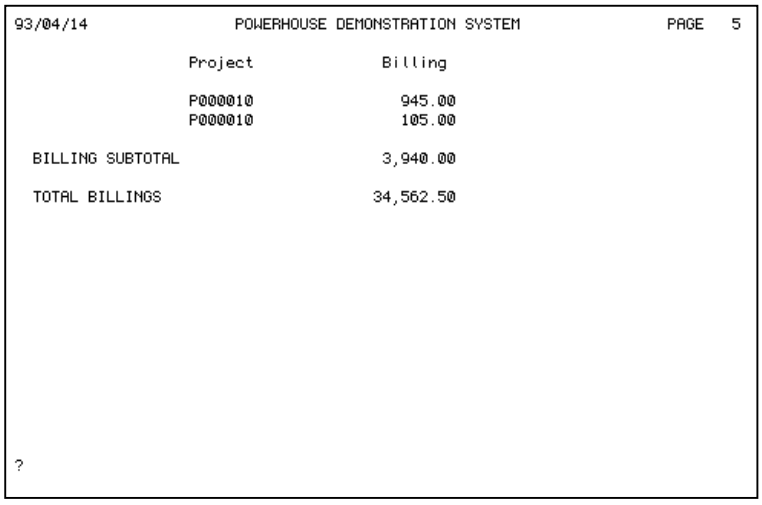
```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   5

                 Project            Billing

                 P000010              945.00
                 P000010              105.00

  BILLING SUBTOTAL    3,940.00

  TOTAL BILLINGS       34,562.50

?
```

With the ALIGN Option

In this example, the labels are aligned to the left-hand side of the report and the billing subtotal and total billings are aligned with the billing in the REPORT statement.

```
> SET REPORT ALIGN
> ACCESS BILLINGS
> REPORT TAB 20 PROJECT TAB 40 BILLING
> SORT ON PROJECT
> FOOTING AT PROJECT SKIP 2 "BILLING SUBTOTAL " &
>   BILLING SUBTOTAL SKIP 2
> FINAL FOOTING SKIP 2 "TOTAL BILLINGS" &
>   BILLING SUBTOTAL  &
>   PICTURE "^^^,^^^,^^^.^^"
> GO
```

The resulting QUIZ report looks like this:



```
93/04/14              POWERHOUSE DEMONSTRATION SYSTEM              PAGE   5

                 Project             Billing

                 P000010               945.00
                 P000010               105.00

  BILLING SUBTOTAL                   3,940.00

  TOTAL BILLINGS                    34,562.50

?
```

Example (UNIX)

The following example sets the printer destination and then directs the output to the printer.

```
> SET REPORT DESTINATION printfile
> SET REPORT DEVICE PRINTER
```

In the next example, the DESTINATION is set but is not used, because the REPORT device is the terminal.

```
> SET REPORT DESTINATION printfile
> SET REPORT DEVICE TERMINAL
>
> SORTED ON CUSTOMERNAME
>
> PAGE HEADING &
>   TAB 20 "Invoices for Customers" &
>   KEEP COLUMN HEADINGS &
>   SKIP 2
>
> REPORT &
>   CUSTOMERNAME &
>   PRINT AT CUSTOMERNAME &
>   INVOICENUMBER
>
> PAGE FOOTING &
>   SKIP 3 &
>   SYSDATE &
>   TAB 50 &
>   "Page" &
>   TAB 56 &
>   SYSPAGE
>
> GO
```

SET SUBFILE

Directs output to a subfile and changes the default subfile settings.

Syntax

```
SET { SUBFILE [options...
 [INDEX name [index-options...]
  [SEGMENT item [ASCENDING|DESCENDING]
   [, item [ASCENDING|DESCENDING]]...]
    [, INDEX name [index-options...]
     [SEGMENT item [ASCENDING|DESCENDING]
      [, item [ASCENDING|DESCENDING]]...]]...]
 |NOSUBFILE}
```

Specifies that a subfile will be created based on the contents of the REPORT statement. Because the subfile is not identified in the data dictionary, reference it as identified to the operating system.

Default: SET NOSUBFILE

Limit: You can access a maximum of 31 record-structures (including subfiles) per report.

OpenVMS, UNIX, Windows: For a regular subfile, QUIZ appends the file extension .SFD to the minidictionary portion of the subfile while the data portion is assigned .SF. Both filenames are identical. A subfile without a minidictionary is assigned the extension .DAT. For a portable subfile, QUIZ appends .PSD to the minidictionary portions and .PS to the data portion. The default name for a subfile is QUIZWORK. For more information about portable subfiles, see the PORTABLE option on (p. 144).

All SET SUBFILE options are saved in compiled reports.

INDEX name [index-option]...

Names the index to be referenced. If the name is an item from the REPORT statement, the item is considered an index segment. If the name is not an item, a segment must be specified using the SEGMENT option.

Limit: You can't use the INDEX option with the PORTABLE option. A maximum of 16 indexes with 8 segments per index is supported.

index-options		
ALTERNATE|PRIMARY	ASCENDING|DESCENDING	REPEATING|UNIQUE
SEGMENT		

ALTERNATE|PRIMARY

Specifies whether the index is an alternate index or a primary index. An indexed subfile always contains one primary index, and can also contain alternate indexes.

Default: The first index defined is the primary index. All subsequent indexes are alternate indexes.

ASCENDING|DESCENDING (OpenVMS, UNIX, Windows)

Specifies the default sort order of the index.

Default: ASCENDING if no sort order is specified on segments.

REPEATING|UNIQUE

REPEATING specifies that records may have the same index values. UNIQUE specifies that every record in the subfile must have a unique index value.

Default: REPEATING

SEGMENT item [ASCENDING|DESCENDING] [,item [ASCENDING|DESCENDING]]...

Specifies the name of an item in the record structure. The item must be listed in the REPORT statement.

ASCENDING|DESCENDING (UNIX, Windows)

Specifies the default sort order of the segment.

Default: ASCENDING

Options

SET SUBFILE Options		
APPEND|NOAPPEND	AT	DICTIONARY|NODICTIONARY
FORMAT	KEEP|TEMPORARY	NAME
PORTABLE		

APPEND|NOAPPEND

APPEND adds data records to the end of the existing subfile. The minidictionary describing the contents of the original subfile is not altered.

If the subfile has a different format from the information to be appended and the record sizes are the same, QUIZ appends to the subfile. However, problems arise when you try to access the subfile because the minidictionary contains only the original record layout.

If the subfile has a different record size from the information to be appended, QUIZ issues an error.

NOAPPEND creates a new subfile if the subfile does not exist. If the subfile already exists, QUIZ erases existing subfile data records before adding the new data records.

Default: NOAPPEND

AT sort-item

Writes records to the subfile at the last occurrence of the sort-item. The sort-item must be named in a previous SORT or SORTED statement.

Default: Writes a record to the subfile for each record complex.

DICTIONARY|NODICTIONARY

DICTIONARY writes subfile format information to a subfile minidictionary. A subfile created with NODICTIONARY does not have a related subfile dictionary and cannot be accessed by PowerHouse 4GL components. The NODICTIONARY option is ignored when overwriting an existing permanent subfile.

OpenVMS, UNIX, Windows:	If the NODICTIONARY option is used, a .DAT file suffix is added automatically.

Both options are saved in compiled reports.

Default: DICTIONARY

FORMAT 4|7|8 (OpenVMS)
FORMAT 3|5|6|7|8 (UNIX)
FORMAT 8 (Windows)

The format dictates how much information is kept about items in the subfile.

Attention to format must be given when creating subfiles that must be compatible with an earlier version of PowerHouse 4GL. However, compatibility is not an issue when using later versions of PowerHouse 4GL. A format introduced with one version of PowerHouse 4GL is supported in later versions of PowerHouse 4GL.

FORMAT 5 and higher support subfiles with PowerHouse 4GL names of up to 64 characters.

For each platform, the following tables list compatible PowerHouse 4GL versions for the various formats.

Platform	Format Option	Compatible PowerHouse 4GL Version
OpenVMS:	4	5.04 and higher
	7	7.10 and higher
	8	8.00 and higher
UNIX:	3	5.10 and higher
	5	5.23 and higher
	6	7.23 and higher
	7	7.23 and higher
	8	8.03 and higher
Windows:	8	8.01 and higher

Limits: FORMAT 0 subfiles do not accept dates with four-digit years. FORMAT [0|3] cannot be used with portable subfiles.

OpenVMS, UNIX, Windows: The maximum number of items allowed is 1023.

Default: 8

KEEP|TEMPORARY

KEEP creates a permanent subfile that may be used in future sessions. TEMPORARY creates a temporary subfile that is deleted at the end of a QUIZ session. Use temporary subfiles when the information is needed only for the current session or a batch job.

Default: Portable subfiles default to KEEP. Nonportable subfiles default to TEMPORARY.

NAME filespec (OpenVMS, UNIX, Windows)

Names the subfile. If the PORTABLE option is specified, it names the data dictionary portion of the subfile.

Default: QUIZWORK

PORTABLE [DATAFILE filespec]

Creates a portable subfile in ASCII format to simplify intercomputer transfer.

The PORTABLE option is saved in compiled reports.

OpenVMS, UNIX, Windows: The file suffix for portable subfiles with an ASCII subfile is .ps. The file suffix for portable subfiles with a minidictionary portion of the subfile is .psd.

DATAFILE filespec

The file specification for the data file of a PowerHouse 4GL portable subfile. The DATAFILE option lets you specify a file specification for a data file that is different from the dictionary portion of the portable subfile. For example:

```
SET SUBFILE abc PORTABLE DATAFILE xyz
```

When referencing the subfile, be certain to use the open name for the dictionary file. The open name for the data file is contained in the minidictionary and is used by the operating system when creating the physical file.

Although you can qualify both the dictionary file name and the data file name with a location, the data file qualification is not saved in the dictionary. So although the files are written to the locations specified, the data file cannot be located because the qualification is lost. Portable subfiles are intended to be used to transfer data from one platform to another. Including the qualification in the data file name may render the data file unusable on a platform with a different filespec format.

Limit (**OpenVMS**): If you specify a logical name for the datafile name, then you must specify a logical name for the dictionary name. The logical paths you specify for the datafile name and the dictionary must be identical.

Discussion

The SET SUBFILE statement either creates a self-describing direct file and adds new data records, or adds new data records to an existing subfile. If the SET SUBFILE statement is active, the REPORT statement lists only the items and item summaries to be written to the subfile. Control headings and footings are not written to the subfile.

When you include the SUMMARY option in a REPORT DEVICE PRINTER or REPORT DEVICE TERMINAL statement, QUIZ won't display the report information on the screen, but will write it to the subfile.

Include all options for each subfile in one SET SUBFILE statement:

```
> SET SUBFILE NAME CUSTOMER KEEP
```

OpenVMS, UNIX, Windows: PowerHouse 4GL stores subfiles as a pair of files. One file contains the information that describes the data layout or data record of the subfile, and the other contains the actual data.

Using Subfiles

You can use subfiles to

- store and access data produced by QUIZ reports
- extract and manipulate data from existing files for use in other reports or programs
- temporarily hold data
- pass data between QUIZ and QTP
- transfer data between reports
- process reports in two or three passes (one pass for data manipulation and another for report results)
- extract and build data files from large databases
- process data from more than one file with no common index
- create files for use by other systems and languages

Types of Subfiles

There are three types of subfiles:

- interim
- temporary
- permanent

QUIZ creates subfiles at the following stages:

BUILD	Creates an interim subfile.
EXECUTE	Creates permanent and/or temporary subfiles; overwrites any existing interim subfiles.

GO	Creates permanent and/or temporary subfiles; overwrites any existing interim subfiles.

Interim Subfiles

Interim subfiles are self-described files that do not contain data.

QUIZ uses interim subfiles when it requires subfile metadata at parse-time, which is before QUIZ has actually created either a temporary or a permanent subfile. QUIZ does not use interim subfiles at execution-time.

QUIZ deletes interim subfiles without warning when it creates the corresponding temporary or permanent subfile. This is true even if you specify the APPEND option, or if record-structure differences are present.

QUIZ overwrites an interim subfile when it creates another interim subfile with the same name.

If you try to create an interim subfile when a temporary subfile with the same name already exists, QUIZ issues a warning asking if you want to delete the existing temporary subfile.

Temporary Subfiles

The TEMPORARY option creates a temporary subfile that the system automatically deletes when you exit QUIZ (**OpenVMS, UNIX, Windows**) unless you use the **phtempkeep** program parameter (**UNIX, Windows**). When you create a temporary subfile in a batch job, the system deletes it when the batch job ends.

Permanent Subfiles

The KEEP and PORTABLE options create permanent subfiles that can be used in other sessions or batch jobs.

Indexed Subfiles

PowerHouse 4GL applications can use subfiles to pass information between QUICK, QUIZ, QTP, and other applications. Subfiles may be either direct or indexed. Indexed subfiles have the following advantages:

- They may reduce the number of passes required by QUIZ and QTP for data manipulation.
- They may allow you to expand dictionary functionality without recompiling QUICK screens.
- They don't need to be converted for use by other applications that require indexed files.

Indexed subfiles can have single or multiple indexes with single or multiple segments.

Accessing Subfiles

Use the ACCESS statement to access subfiles by

- using the PowerHouse 4GL logical name without an alias
- prefixing the name with an asterisk (*) when you first name it and then specifying an alias and then using the alias name

The asterisk indicates that you haven’t declared the subfile in the data dictionary.

Subfiles created by QTP can also be used as input files to QUIZ. Likewise, QUIZ subfiles can be used as input to QTP. The only limitation is that the subfile must be the primary file unless you link to the subfile by record number. The first record is assigned 0 (**UNIX, Windows**) or 1 (**OpenVMS**).

You can see a subfile record layout by using the SHOW SUBFILE statement of QSHOW.

Actions QUIZ Performs on Subfiles

If a SET SUBFILE statement specifies the name of an existing subfile, QUIZ does one of the following at parse time:

- purges the existing subfile and creates an interim minidictionary
- uses the existing subfile’s minidictionary
- reports an error

Once the report is executed, if a subfile other than the minidictionary created at parse time exists, QUIZ does one of the following:

- retains the existing file. The APPEND option affects the processing in the following ways:
  - if the APPEND option is specified, new records are added to the existing subfile
  - if the APPEND option is not specified, the data portion of the subfile is overwritten by new records
- purges the existing subfile and creates a new subfile
- reports an error

For more information about how QUIZ locates subfiles, see the section, "Locating Files", in Chapter 1, "Running PowerHouse 4GL", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

How QUIZ Writes Items to Subfiles

QUIZ writes data record items to the subfile with the same format as the original object.

QUIZ writes defined items to the subfile with the same format as the corresponding DEFINE statement. DEFINE statements must be written for each occurrence of an array and explicitly written to the subfile.

QUIZ writes items with summary operations to the subfile as follows:

Summary operation	What QUIZ writes to the subfile
AVERAGE	a 16-byte compound item, consisting of: • FLOAT SIZE 8 SUBTOTAL • INTEGER SIZE 4 COUNT • INTEGER SIZE 4 filler of binary zeros
COUNT	not written to the subfile
MAXIMUM	FLOAT SIZE 8 for numeric and date items CHARACTER SIZE m for character items (where m is the length of the item)
MINIMUM	FLOAT SIZE 8 for numeric and date items CHARACTER SIZE m for character items (where m is the length of the item)
PERCENT	two FLOAT SIZE 8 subtotals
RATIO	two FLOAT SIZE 8 subtotals
SUBTOTAL	FLOAT SIZE 8

QUIZ writes items with the AVERAGE, PERCENT, or RATIO summary operations to the subfile as two items: one item for the summary operation and one item for the count. You cannot reference the item for the count.

Defined items must sometimes be used when writing summary operations to a subfile. Because summary items take the name of the item being summarized, confusion occurs if an item is the object of more than one summary. Both summary items will have the same name on the subfile and only the first of these items can be referenced.

QUIZ writes INITIAL VALUE as FLOAT SIZE 8 to the subfile.

Also, QUIZ writes the following to the subfile:

System function	What QUIZ writes to the subfile
SYSDATE	PHDATE for dates that don't include a century prefix in the data dictionary
	INTEGER SIZE 4 for dates that include a century prefix in the data dictionary
SYSDATETIME	INTEGER SIZE 8
SYSTIME	CHARACTER SIZE 40
SYSNAME	INTEGER SIZE 2 in hhmm format

How QUIZ Writes Substructured Items to Subfiles

QUIZ only writes the specified items of the substructure. If REPORT ALL is used, QUIZ writes the items from the lowest level of structure.

The following example shows a substructure as defined in the dictionary:

```
> RECORD INVOICEMASTER
> ITEM DATE_ALL DATATYPE ZONED SIZE 8
> BEGIN STRUCTURE
>   ITEM DATE_YEAR DATATYPE ZONED SIZE 4
>   ITEM DATE_MONTHDAY DATATYPE ZONED SIZE 4
>  BEGIN STRUCTURE
>   ITEM DATE_MONTH DATATYPE ZONED SIZE 2
>   ITEM DATE_DAY DATATYPE ZONED SIZE 2
>   END STRUCTURE
> END STRUCTURE
```

For either of the following two statements

```
REPORT DATE_YEAR DATE_MONTH DATE_DAY
REPORT ALL
```

QUIZ creates the following subfile:

```
91/08/14                  SUBFILE DICTIONARY                  Page    1
                        SUBFILE RECORD REPORT


   Subfile:          S2
   Type:             Special
   Format:           5
   Record Size:      8 Bytes

 Item                              Type            Offset Size  Occurs

 DATE_YEAR                         ZONED UNSIGNED  0      4
 DATE_MONTH                        ZONED UNSIGNED  4      2
 DATE_DAY                          ZONED UNSIGNED  6      2









 ?
```

Portable Subfiles

Portable subfiles are intended to be used to transfer data between different types of computers. Since data structures are often different, both the portable subfile dictionary and the datafile are character data. For example, instead of floating point data being in computer-specific format, portable subfiles use a portable float format that can be interpreted by PowerHouse 4GL on any computer that supports PowerHouse 4GL.

The Portable Subfile Dictionary

The portable subfile dictionary starts with a comment that reads

```
;Subfile created by <QUIZ|QTP> <version> on <date>
```

Each item in the subfile is listed as an element in the order they are declared in the subfile. Item format options are included if specified. If not specified, format options take their default values in the same manner as they would in a QUIZ REPORT statement. The options that are always included are shown below. The general format for each element is:

```
Element <element_name> <Type> Size <n> &
  [<format_option> &]…
  Heading  "<heading>"
```

Each format option is on a line by itself except the separator and date format which look like this:

```
Element TESTDATE Date Size 8 &
  Separator "-"    Format YYYYMMDD &
  Heading "TESTDATE"
```

All elements have a HEADING.

Character elements have a size equal to the item size.

Date elements have a size based on the type of date: 6 for century excluded (DATE CENTURY EXCLUDED, JDATE, PHDATE), 8 for century included (DATE CENTURY INCLUDED, ZDATE), and 18 for DATETIME. The SEPARATOR and date FORMAT are always included.

Numeric elements have a size based on the item datatype as shown below. The LEADING SIGN (by default "-"), SIGNIFICANCE (by default 1), and PICTURE are always included. This is regardless of whether the datatype is SIGNED or UNSIGNED.

The sizes and the number of substitution characters (^) in the PICTURE are listed below. Signed or unsigned has no affect on the number of substitution characters.

Datatype	Element size	Substitution Characters (^) in PICTURE	
FLOAT SIZE 4	9	7	
FLOAT SIZE 8	19	10	
FREEFORM SIZE n	n	n+1	
INTEGER SIZE 1	3	3	Applies to SIGNED, UNSIGNED
INTEGER SIZE 2	5	5	Applies to SIGNED, UNSIGNED
INTEGER SIZE 3	10	7	Applies to SIGNED, UNSIGNED
INTEGER SIZE 4	10	10	Applies to SIGNED, UNSIGNED
INTEGER SIZE 5	15	12	Applies to SIGNED, UNSIGNED
INTEGER SIZE 6	15	15	Applies to SIGNED, UNSIGNED
INTEGER SIZE 7	20	19	Applies to SIGNED, UNSIGNED
INTEGER SIZE 8	20	19	Applies to SIGNED, UNSIGNED

Datatype	Element size	Substitution Characters (^) in PICTURE	
PACKED SIZE n	2n-1	2n+1	Applies to SIGNED, UNSIGNED
ZONED SIZE n	n	n+1	Applies to SIGNED, UNSIGNED, NUMERIC
FLOAT SIZE 4	9	7	Applies to SIGNED, UNSIGNED

The SUBFILE statement appears after the elements. This is a special form of the FILE statement. Since you can't have an indexed portable subfile, the organization is always direct:

```
Subfile <data_portion_name> Organization Direct
```

When you declare a subfile name in QUIZ or QTP, you actually declare the dictionary name. The datafile portion name is derived from the dictionary name unless the DATAFILE option is specified to override. While you can specify a different path for the datafile portion, that path is not added to the dictionary. Only the datafile name is used. This is because paths are different on each computer type. When using portable subfiles to move data from one computer to another, ensure that your subfile dictionary and datafile names are compatible with both computers.

Following the SUBFILE statement is the record statement:

```
Record <data_portion_name>
```

Each item is then listed in an ITEM statement in the order they are declared in the subfile. The general form is

```
Item <element_name> <portable_datatype> Size <n>
```

Many PowerHouse 4GL datatypes are represented using a portable datatype that represents the data in character format. The following table lists the PowerHouse 4GL datatypes, the equivalent portable datatypes, and the size used, where n is the original item size in bytes.

PowerHouse 4GL Datatype	Portable Datatype	Size
CHARACTER	CHARACTER	n
DATE CENTURY EXCLUDED	PORTABLE DATE	6
DATE CENTURY INCLUDED	PORTABLE DATE	8
DATETIME	PORTABLE DATE	18
JDATE	PORTABLE DATE	6
PHDATE	PORTABLE DATE	6
ZDATE	PORTABLE DATE	8
FLOAT SIZE 4	PORTABLE FLOAT	15
FLOAT SIZE 8	PORTABLE FLOAT	25
FREEFORM SIZE n	FREEFORM	n
INTEGER SIZE 1	PORTABLE ZONED	4
INTEGER SIZE 2	PORTABLE ZONED	6
INTEGER SIZE 3	PORTABLE ZONED	11
INTEGER SIZE 4	PORTABLE ZONED	11
INTEGER SIZE 5	PORTABLE ZONED	16
INTEGER SIZE 6	PORTABLE ZONED	16
INTEGER SIZE 7	PORTABLE ZONED	21
INTEGER SIZE 8	PORTABLE ZONED	21

PowerHouse 4GL Datatype	Portable Datatype	Size
PACKED SIZE n	PORTABLE ZONED	2n
ZONED SIZE n	PORTABLE ZONED	n+1

The sizes for PACKED is based on the number of bytes. For example, if the original definition was PACKED*12, then the subfile will get 7 bytes ((n+2)/2 floored) which gives an element size of 13.

The last line of the subfile dictionary is

```
; end source
```

Portable Subfile Datatype Specifics

The datafile is a direct file with the data in the same sequence as the dictionary item list.

CHARACTER and FREEFORM data is copied as is.

DATE data is converted to PORTABLE DATE so that there is no confusion about the format. PORTABLE DATE SIZE 6 is in YYMMDD format. PORTABLE DATE SIZE 8 is in YYYYMMDD format. PORTABLE DATE SIZE 18, used for DATETIME, is in YYYYMMDD.HHMMSSTTT format with a decimal point separating the date and time portions.

The decimal point is always used even if the decimal character in the dictionary has been changed.

PORTABLE FLOAT SIZE 15 starts with a sign, plus or minus, followed by a decimal point and 8 digits representing the mantissa of the number. This is followed by the letter E, another sign, plus or minus, and a 3 digit exponent. For example, the number 1000 is represented by +.10000000E+004 and the number -0.05 is represented as -.49999999E-002. Remember that floating point is not an exact representation.

PORTABLE FLOAT SIZE 25 starts with a sign, plus or minus, followed by a decimal point and 18 digits representing the mantissa of the number. This is followed by the letter E, another sign, plus or minus, and a 3 digit exponent. For example, the number 1000 is represented by +.10000000000000000E+004.

PORTABLE ZONED SIZE n starts with a sign, plus or minus, followed by n-1 digits representing the actual value.

Examples

Subfiles are useful when complex calculations or summary values are required, or when a report can be simplified by breaking it down into two or three passes.

Breaking Reports into Two Passes: Example 1

In the following example, the first pass writes one record to the subfile for each customer. Each record contains the customer's account number and the total value of all their invoices. In the second pass, the information from the first pass is used to sort the customers into descending (D) order of total invoices.

The report produces a ranking by total invoices. In the following example:

- The first ACCESS statement begins the first pass.
- AT CUSTOMERKEY tells QUIZ that you want to write records to the subfile at the last occurrence of the CUSTOMERKEY sort-item.
- The second ACCESS statement begins the second pass.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
>   AND TO PARTNUMBER OF PARTS
>
> DEFINE INVOICEAMOUNT NUMERIC * 9 &
>   PICTURE "$^^^,^^^.^^" &
>   = QUANTITYSHIPPED OF INVOICEDETAIL * &
>   (UNITCOST * UNITMARKUP)
```

```
>
> SORT ON CUSTOMERKEY
>
> REPORT SUMMARY &
>   CUSTOMERKEY &
>   INVOICEAMOUNT SUBTOTAL
>
> SET SUBFILE AT CUSTOMERKEY
>
> GO
> ACCESS *QUIZWORK LINK TO CUSTOMERS
>
> SORT ON INVOICEAMOUNT D
>
> REPORT &
>   COUNT NORESET &
>   CUSTOMERNAME &
>   INVOICEAMOUNT &
>   HEADING "Total^Invoices"
>
> GO
```

Breaking Reports into Two Passes: Example 2

The following is another example of a two-pass report. In the first pass, you write a subtotal for the item you want to average, along with a count of the number of values in the subtotal. In the second pass, you calculate the average.

In the following example:

- The first ACCESS statement begins the first pass.
- The second ACCESS statement begins the second pass.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
>   AND TO PARTNUMBER OF PARTS
>
> SORTED ON CUSTOMERNAME
>
> DEFINE ORDERCOUNT NUMERIC * 4 = 1
>
> DEFINE ORDERTOTAL NUMERIC * 9 &
>   PICTURE "$^^^,^^^.^^" &
>   = QUANTITYSHIPPED OF INVOICEDETAIL * &
>   (UNITCOST OF PARTS * UNITMARKUP OF PARTS)
>
> SET SUBFILE &
>   AT CUSTOMERNAME &
>
> REPORT SUMMARY &
>   CUSTOMERNAME &
>   ORDERTOTAL SUBTOTAL &
>   ORDERCOUNT SUBTOTAL
>
> GO
>
> SET NOSUBFILE
> ACCESS *PASS1>
> DEFINE AVERAGEORDER NUMERIC * 7 &
>   HEADING "Average^Order" &
>   PICTURE " ^^,^^^.^^" &
>   = (ORDERTOTAL / ORDERCOUNT)
> REPORT &
>   CUSTOMERNAME &
>   AVERAGEORDER
>
```

```
> GO
```

QUIZ and Interim Subfiles

The following example creates a compiled report named SUPPREPT and an interim subfile named TOTAL:

```
> ACCESS SUPPLIERS
>
> REPORT ALL
>
> SET SUBFILE NAME TOTAL
>
> BUILD SUPPREPT
```

The interim subfile, named TOTAL, can now be used by QUIZ to provide metadata information for statements that reference the subfile TOTAL before it's actually created. When you are parsing a file that compiles or creates several production reports without executing them, QUIZ accesses the metadata from the interim subfile TOTAL and then goes on to compile the next production report. When QUIZ encounters a GO statement, it will execute the report and create a temporary subfile named TOTAL.

Using Defined Items

A DEFINE statement is sometimes needed to calculate subtotals or to write the result of a summary operation to the subfile. For example:

```
> REPORT SUMMARY &
>   INVOICEAMOUNT SUBTOTAL &
>   INVOICEAMOUNT MAXIMUM &
>   INVOICEAMOUNT MINIMUM
```

In this instance, QUIZ writes three items to the subfile, but they are all identified by the name INVOICEAMOUNT and only the first item can be referenced by name.

To solve this problem, create defined items. For example:

```
> DEFINE IAMTSUBTOT = INVOICEAMOUNT
>
> DEFINE IAMTMAX = INVOICEAMOUNT
>
> DEFINE IAMTMIN = INVOICEAMOUNT
>
> REPORT SUMMARY &
>   IAMTSUBTOT SUBTOTAL &
>   IAMTMAX MAXIMUM &
>   IAMTMIN MINIMUM
```

Also avoid writing the COUNT summary operation to the subfile. COUNT is an action, not an item; therefore, it can't be referenced by name when QUIZ reads the subfile. To solve this problem, create a defined item. For example:

```
> DEFINE COUNTER = 1
>
> REPORT &
>   SUMMARY &
>   CUSTOMERNAME &
>   COUNTER SUBTOTAL
```

Using Indexed Subfiles

In the following example, the subfile TOTALS is created. The subfile has a unique index, SALESIDX, consisting of two segments, EMPLOYNO and MTHYEAR:

```
> SET SUBFILE NAME TOTALS &
>   INDEX SALESIDX UNIQUE &
>   SEGMENT EMPLOYNO, MTHYEAR
```

If the subfile were not indexed, a sort would be required to get the records in the correct sequence:

```
> ACCESS *TOTALS
> SORT ON EMPLOYNO ON MTHYEAR
> REP ALL
```

```
> GO
```

With indexed subfiles, you can use the CHOOSE statement, and avoid the extra overhead of a SORT statement, provided that the subfile is the primary file, and that the desired order is that of the index.

```
> ACCESS *TOTALS
> CHOOSE VIAINDEX SALESIDX
> REP ALL
> GO
```

SHOW

Displays all record-structures or items for the report.

Syntax

SHOW [option]

Options

SHOW Options		
ACTIVITY	COMMANDS	DATABASES
FILES	ITEMS	STATUS

ACTIVITY

Displays statistics about activities performed, such as the number of data records read and selected, the number of files opened, lines and pages printed, and the amount of CPU time used.

COMMANDS

Displays all statements copied to QUIZ's temporary save file since the last ACCESS statement or SET SAVE CLEAR statement. The SHOW COMMANDS statement is not saved in the save file.

DATABASES

Lists the tables or views from databases that are declared with either FILE or DATABASE statements in PDL.

FILES

Lists the names of record structures, tables, or views from files and databases that are declared with FILE or DATABASE statements in PDL.

ITEMS

Displays the names of all items that are part of the record-structures named in the current ACCESS statement. Index items are identified by asterisks (*). If there are multiple-segment indexes, only the first segment is identified by an asterisk. Substructured items are identified by periods up to the fourth substructured level; for items substructured from levels 5 to 15 (the maximum), the display format is .5 to .15 respectively. In this way, items substructured from the fifth to the fifteenth level are identified explicitly by number rather than by nested format.

Picture overflow is identified by an ellipsis (...). The maximum picture size allowed in the SHOW ITEMS display is fifteen characters long. All characters for a picture that is exactly fifteen characters long are displayed. Redefinitions are identified by underscores (_).

The ITEMS option of the SHOW statement displays the TYPE, OUTPUT SCALE, DECIMAL, and PICTURE of the element associated with each item.

STATUS

Displays all current status-control (SET statement) settings.

SORT

Sorts record complexes in a desired sequence and defines control breaks.

Syntax

UNIX, Windows:	**SORT [ON sort-item [A|D] [RESET PAGE[n]]]** **[ON|, sort-item [A|D] [RESET PAGE[n]]]...**
OpenVMS:	**SORT [ON sort-item [A|D] [RESET PAGE[n]]]** **[ON|, sort-item [A|D] [RESET PAGE[n]]]** **[RETAIN ORDER]...**

ON sort-item

Specifies a record item or defined item on which to sequence the record complexes. The item then becomes a sort-item.

If more than one item is listed in the SORT statement, a comma or the keyword ON must precede each additional item.

Limit: Items can't be subscripted.

A|D

Specifies that sorting is done in ascending (A) or descending (D) order.

Default: Ascending (A)

RESET PAGE [n]

Resets the page number to the value specified and starts a new page whenever a control break for the sort-item occurs.

Limit: -2147483647 to 2147483647

Default: 1

RETAIN ORDER (OpenVMS)

Stipulates that records with duplicate sort-item values must be returned in the order in which they were read from the record-structure.

Discussion

The SORT statement states which record complexes are to be sorted and the sequence in which they are to be sorted.

The last item named in the SORT statements becomes the item associated with the lowest-level control break. A new SORT statement cancels any previous SORT or SORTED statement. The SORT statement used alone declares that no sort is to occur.

Control Breaks

Each sort-item establishes a control break (the point in a request where the value of a sort-item changes or "breaks") where various actions can be performed. The record complexes within a control break are collectively called a record complex group.

Example

The report in the following example displays the invoices for each customer, organized by customer name and invoice date. In the following example:

- The report output is sorted first by CUSTOMERNAME, then by INVOICEDATE.
- D tells QUIZ to report invoice dates from most recent to least recent. If D isn't specified, report data is sorted in ascending order.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER &
>   LINK TO INVOICEDETAIL &
>   AND TO PARTNUMBER OF PARTS
>
> DEFINE INVOICEAMOUNT NUMERIC * 9 &
>   PICTURE "$^^^,^^^.^^" &
>   = QUANTITYSHIPPED OF INVOICEDETAIL * &
>   (UNITCOST OF PARTS * UNITMARKUP OF PARTS)
>
> SORT &
>   ON CUSTOMERNAME &
>   ON INVOICEDATE D
>
> REPORT &
>   CUSTOMERNAME PRINT AT CUSTOMERNAME &
>   INVOICENUMBER &
>   INVOICEDATE &
>   INVOICEAMOUNT &
>     HEADING "Amount^Owing"
>
> FOOTING AT &
>   CUSTOMERNAME &
>   SKIP 2 &
>   "Total: " &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL
>
> FINAL FOOTING &
>   "Grand Total: " &
>   TAB 40 &
>   INVOICEAMOUNT SUBTOTAL
>
> GO
```

SORTED

Defines control breaks for records known to be in sorted order.

Syntax

SORTED [ON sort-item [A|D] [RESET PAGE[n]]]
[ON|, sort-item [A|D] [RESET PAGE[n]]]...

sort-item

Specifies a record item or defined item that establishes a control break.

Limit: Items can't be subscripted.

A|D

Specifies that sorting is done in ascending (A) or descending (D) order.

This option is used only for documentation; the actual sequence is neither checked nor altered.

Default: Ascending (A)

RESET PAGE [n]

Resets the page number to the value specified and starts a new page when a control break for the sort-item occurs.

Limit: -2147483647 to 2147483647

Default: 1

Discussion

The SORTED statement declares items that will be used as control breaks. It indicates that the record complexes are already in the desired sequence, so QUIZ won't sort them.

A new SORTED statement cancels all previous SORT, SORTED, FOOTING AT, and HEADING AT statements.

Use the SORTED statement to eliminate unnecessary sorting whenever records are already in the desired sequence.

Example

This report lists customer invoice numbers by customer name. It includes a SORTED statement to establish a control break for report formatting. This report uses the SORTED statement instead of the SORT statement because CUSTOMERNAME is a segment and therefore records are retrieved in sorted order.

```
> ACCESS CUSTOMERS &
>   LINK TO ORDERMASTER &
>   LINK TO ORDERDETAIL &
>   LINK TO INVOICEMASTER
>
> SORTED ON CUSTOMERNAME
>
> PAGE HEADING &
>   TAB 20 "Invoices for Customers" &
>   KEEP COLUMN HEADINGS &
>   SKIP 2
>
> REPORT &
>   CUSTOMERNAME &
>   PRINT AT CUSTOMERNAME &
>   INVOICENUMBER
>
> PAGE FOOTING &
```

```
>   SKIP 3 &
>   SYSDATE &
>   TAB 50 &
>   "Page" &
>   TAB 56 &
>   SYSPAGE
>
> GO
```

USE

Processes QUIZ source statements that are contained in a file.

Syntax

USE filespec [option]

filespec

Names the file that contains the QUIZ source statements you want to use.

If the file doesn't exist, QUIZ issues an error message.

Options

The options are DETAIL, NODETAIL, LIST, and NOLIST.

DETAIL|NODETAIL

Writes the contents of the file to QUIZ's source statement save file, QUIZSAVE, rather than just the USE statement itself. NODETAIL writes just the USE statement to the temporary save file rather than the contents of the file being used.

Default: DETAIL

LIST|NOLIST

Displays the statements as they are processed; NOLIST does not.

Default: LIST

Discussion

The USE statement instructs QUIZ to process the named file for statement input.

QUIZ reads and interprets each statement as if it had been entered from the terminal.

The **procloc** parameter affects how PowerHouse 4GL uses unqualified file names that are specified in the USE statement. For more information about the **procloc** program parameter, see Chapter 2, "Program Parameters", in the *IBM Cognos PowerHouse 4GL PowerHouse Rules* document.

Nesting USE Statements

A file referenced in a USE statement can itself contain other USE statements. USE files can be nested to a maximum of 20 levels. Permanent files containing valid source code can be included at any time providing they are consistent with QUIZ syntax and structure.

Example

The source statements in this example are stored in a file called PRICELST. To print the report, the user enters

```
> USE PRICELST
```

A second USE statement, nested within the first, reads the file, STDREP, that contains a standard report format. STDREP can be used by many reports:

```
> ACCESS PARTS
>
> USE STDREP
```

QUIZ reads the statements from the file STDREP into the current report:

```
> SET NOBLANKS
> SET NOHEAD
> SET REPORT DEVICE PRINTER &
>   LIMIT 3000 &
```

```
>   COPIES 2
```

As each statement is read, it's interpreted as if it were entered from the keyboard. (If the NOLIST option had been included, the contents of STDREP would not have been displayed.) When all the statements from the USE file have been processed, QUIZ continues to process the PRICELST file:

```
> DEFINE RETAILPRICE &
>   NUMERIC * 9 &
>   PICTURE "$^^^,^^^.^^" &
>   = (UNITCOST * UNITMARKUP)
> REPORT &
>   PARTNUMBER &
>   PARTVARIANT &
>   PARTNAME &
>   RETAILPRICE
> GO
```

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z