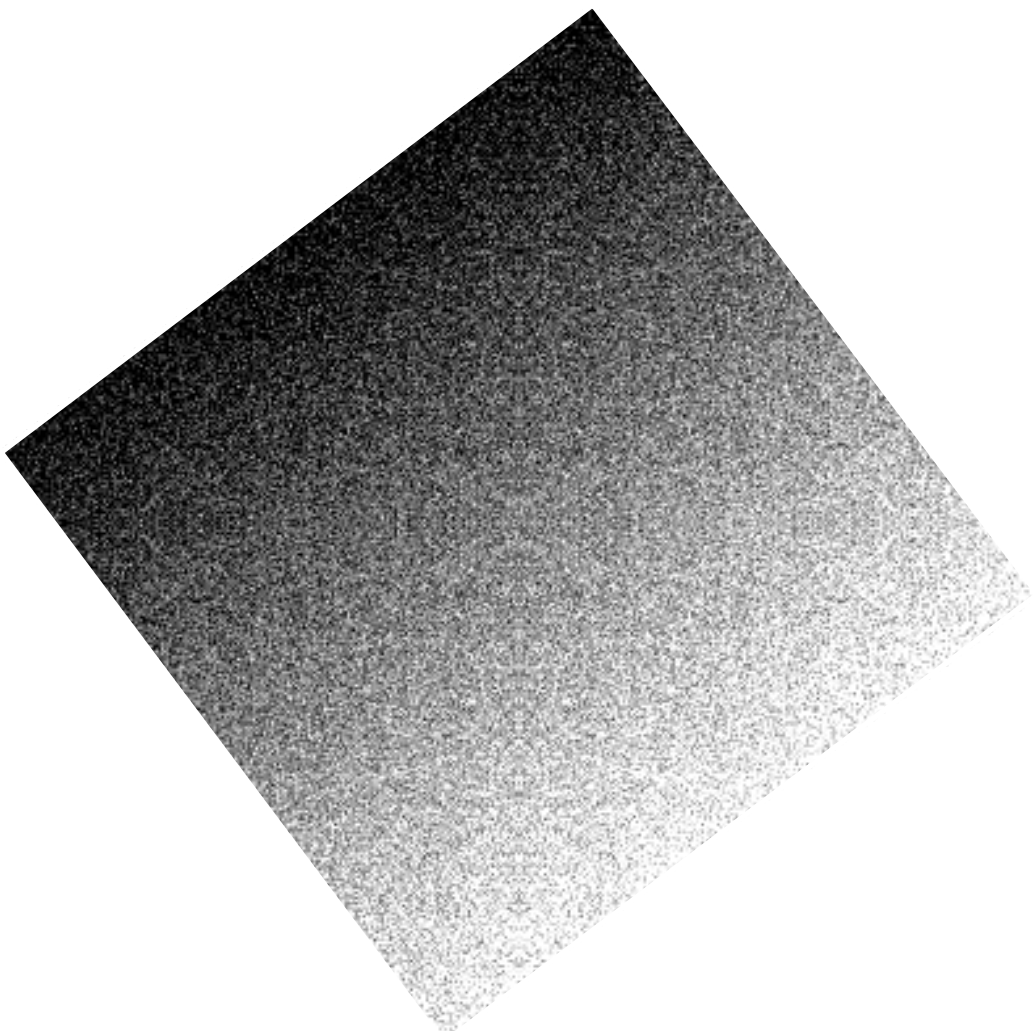
Intelligent Decision Server

IBM

SQL Entry User's Guide

Version 1



SC26-8898-00

Intelligent Decision Server

IBM

SQL Entry User's Guide

Version 1

SC26-8898-00

Note

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

First Edition (October 1996)

This edition applies to Version 1 of Intelligent Decision Server, 5639–IDS, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1996. All rights reserved.**
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices v
Trademarks v

About This Book vii
Who Should Read This Book vii
We'd Like Your Comments vii

Chapter 1. What Is SQL Entry? 1
The SQL Entry Window 1
The SQL Controls Window 1
The Source Document 2
The SQL Entry Tool in a Capsule Application 2

Chapter 2. Getting Started 3
Configuring an SQL Entry Icon 3
Securing an SQL Entry Icon 6
Creating the Source Document 7
Specifying the Source of the Program 8
Editing the SQL Program in the Source Document 9

Chapter 3. Writing the SQL Program 11
Using @-Variables 11
Using @-Keywords as @-Variables 11
Substituting String Values for @-Variables 11
Displaying @-Values 12
Using Logic Statements 12
&DISPLAY 13
&ERROR 13
&GOTO 15
&IF 15
&IMPMSG 16
&LABEL 16
&SECURECONNECT 17
&SET 18
Using Functions 19
Using Logic Statement Functions 20
Using SQL Statement Functions 22
Using SQL with the SQL Entry Tool 25
Continuing Statements 25
Including Comments 25

Chapter 4. Running the SQL Program 27
Running the SQL Entry Tool in Stand-alone Mode 27
Creating an SQL Log 28
Copying Information to Other Tools 29

© Copyright IBM Corp. 1996

Running the SQL Entry Tool in a Capsule Application 30
Creating an SQL Log 30
Transferring the Results of the SQL Program 31
Stopping the SQL Program 31

Glossary 33

IDS Publications 35

Index 36

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
555 Bailey Avenue, W92/H3
P.O. Box 49023
San Jose, CA 95161–9023

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of IBM Corporation in the United States or other countries or both:

ACF/VTAM	DXT	PROFS
AIX	Enterprise System/9000	PS/2
Application System/400	ES/9000	QMF
APPN	IBM	RACF
AS/400	MVS/ESA	RISC System/6000
Capsule	MVS/XA	RS/6000
DATABASE 2	OfficeVision	S/390

© Copyright IBM Corp. 1996

DataJoiner	OfficeVision/VM	SQL/DS
DB2	Operating System/2	System/360
DB2 for OS/2	Operating System/400	System/370
DB2/6000	OS/2	Virtual Machine/Enterprise Systems Architecture
DFSORT	OS/400	VM/ESA
Distributed Relational Database Architecture	Personal System/2	VTAM
DRDA	Presentation Manager	Win-OS/2

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About This Book

The Intelligent Decision Server (IDS) *SQL Entry User's Guide* contains reference information on the SQL Entry tool for running Structured Query Language (SQL) programs against a database.

Who Should Read This Book

This book is intended for IDS users who use the SQL Entry tool to submit written SQL statements to a database service. This book assumes you know:

The IDS environment

SQL

We'd Like Your Comments

Your feedback is important to us in providing you with the most accurate and high-quality information. If you have any comments about this book or any other IDS documentation, please visit our Website at:

http://www.dss.ibm.com

There you'll find the Feedback Connection page where you can enter comments and send them to us.

© Copyright IBM Corp. 1996

Chapter 1. What Is SQL Entry?

The SQL Entry tool processes Structured Query Language (SQL) commands, including SQL extensions supported by the relational database management systems (RDBMS) used by the IBM Intelligent Decision Server (IDS).

While working with the SQL Entry tool, you can:

Design complex applications by entering SQL statements into a source document

Use logic statements for flow control and conditional processing

Use functions and @-variables in generic applications

Supplement other tools in a capsule application by running a complex SQL program that cannot be generated from other tools, or use it in stand-alone mode

The SQL Entry Window

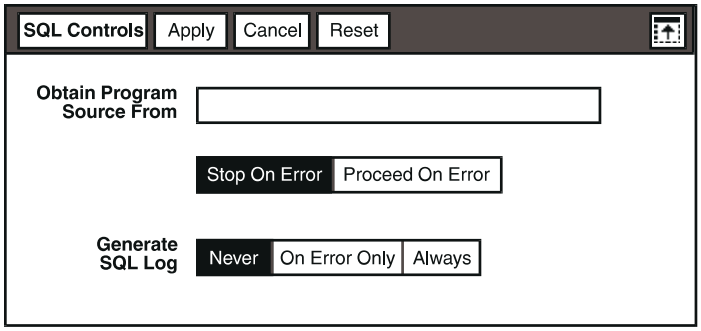
The SQL Entry window appears when you double-click on a configured SQL Entry icon. (See “Configuring an SQL Entry Icon” on page 3.) The display area of the window shows the SQL program and commands. You can also replace the SQL program in the display area by copying the program from the source document into it, or can copy the SQL program from the display area to other tools.

The SQL Controls Window

The SQL Controls window appears when you click on the SQL Entry window header. The SQL Controls window contains fields let you specify the source document con-

© Copyright IBM Corp. 1996

taining the SQL program and determine the behavior of the SQL Entry tool when running the SQL program.



The Source Document

The source document is a text document containing the SQL program. The source document can be stored in a central file drawer for easy maintenance.

The SQL Entry Tool in a Capsule Application

In a capsule application, you can use @-variables to control the processing within an SQL Entry tool. A capsule application also allows you to automatically transfer the results of the SQL program to an icon such as Spreadsheet or Text.

Chapter 2. Getting Started

Before you can use an SQL Entry icon, you must configure it to a specific database. You might also want to secure your icon after you have written a program. The program is written in a source document. This chapter describes how to:

Configure and name an SQL Entry icon

Secure an icon

Create the source document

Configuring an SQL Entry Icon

The SQL Entry icons in the New Icons file drawer are not configured for any user or database. If you cannot find an SQL Entry icon, consult your system administrator for assistance.

SQL Entry icons must be configured for a specific database before you open them. To configure the icon, you must know the password and the name of the database or DB2 subsystem you want to access. If you do not have this information, consult your database administrator.

To configure an SQL Entry icon:

1. Select the SQL Entry icon.
2. Open the Icon Options window.
3. Select **Other Data** in the **Display** field.

   The **Other Data** settings are displayed, as shown in Figure 1 on page 4.

© Copyright IBM Corp. 1996

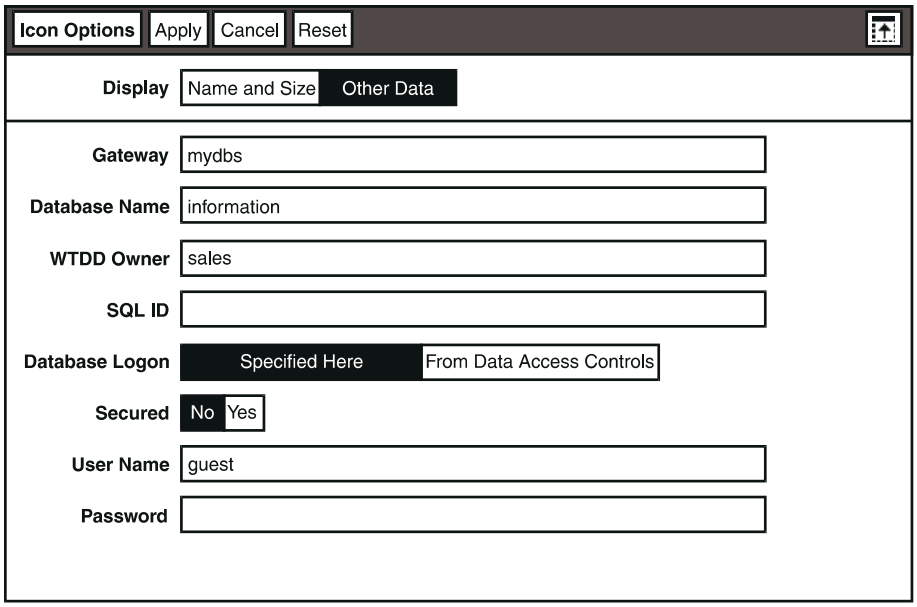


Figure 1. Other Data page of the Icon Options window

4. Using the guidelines provided in Table 1, type the required information into the text fields and select the appropriate choices in the other fields.

   If your server stores more than one physical database, you must identify the database you want to access in the **Database Name** field. You don't need to supply a database name if only one physical database exists, or if you want to access a default database.

Table 1 (Page 1 of 3). Field Guidelines

Field	Guidelines
Gateway	Type the name of the database gateway that you want the SQL Entry icon to access. If the gateway service is on a different realm, follow the gateway name with a colon and then the realm name. For example: mydbs:HQ

Table 1 (Page 2 of 3). Field Guidelines

Field	Guidelines
Database Name	Type the name of the physical database you want the SQL Entry icon to access. For DB2 databases, type the name of the DB2 subsystem you want to access. For SQL/DS databases, type the name of the virtual machine running SQL/DS. For AS/400 databases, it is not necessary to specify anything in this field, because the AS/400 is a single-database system. To connect to a supported, remote IBM database using Distributed Relational Database Architecture (DRDA), follow the local database name with a period and then the name of the DRDA server through which the connection is being made. For example: DSN.TD1 To connect to a remote SYBASE or Red Brick database, follow the SQL Server name with a vertical bar and the name of the database you want to access. For example: SYBSERVER2|SYBPROD To connect to a remote ORACLE database, specify the full database name or its alias. For the full database name, specify the standard single-character prefix used for the remote connection (such as **T** for TCP/IP), followed by a colon, the name of the target host, another colon, and finally the system ID (SID) of the target ORACLE database. For example: T:host1:DBS
Category	Type the print name or internal name of the database category for this SQL Entry icon. The name you specify appears in the window header of the category window.
WTDD Owner	If the SQL Entry tool is configured to a database that supports multiple WTDDs and qualified table names, type the name of the owner of the WTDD tables. This field makes it unnecessary to drop and recreate synonyms whenever you access a different WTDD within the same database.
SQL ID	If you are using DB2, type the table qualifier name. The ID entered in the **SQL ID** field overrides the default table qualifier name, which is the name entered in the **User Name** field. If you have questions about the table qualifier name, consult your database administrator.
Database Logon	Select **Specified Here** if you want database access to be based on information you type in the **User Name** and **Password** fields. Select **From Data Access Controls** if you want access to be based on information in the desktop's Data Access Controls window. Consult your database administrator if you have security questions.
Secured	Select **Yes** if you want the icon's **User Name** field to be set to the user ID of the desktop on which it resides. See “Securing an SQL Entry Icon” on page 6 for information on securing an icon.

Table 1 (Page 3 of 3). Field Guidelines

Field	Guidelines
user ID	Type a valid user ID. This might be the same as your desktop user ID depending on how your database security is set up. When the SQL Entry icon connects to the RDBMS, this is the user ID CPI sent to the RDBMS. If you are using the OS2-DRDA gateway to connect to a DRDA server and your DB2 for OS/2 user ID is different from your DRDA server user ID, you must specify both user IDs separated by a period. For example: DB2 for OS/2_userid.DRDAID
Password	Type the database user's password. This password might be the same as your desktop password depending on how your database security is set up. For security, the password is never displayed. If you are using the OS2-DRDA gateway to connect to a DRDA server and your DB2 for OS/2 password is different from your DRDA server password, you must specify both passwords separated by a period. For example: DB2 for OS/2_password.DRDA_password

When you finish entering data into the fields, rename the SQL Entry icon so the name reflects its data source, the name of the database being used, or the name of the application or task you are performing with the icon.

To rename the SQL Entry icon:

1. In the **Display** field, select **Name and Size**.
2. Delete the old name.
3. Type a new name into the **Name** field.
4. Close the Icon Options window.

Securing an SQL Entry Icon

When you write an SQL program you want to protect, you can secure the SQL Entry icon. The secured icon can then be opened only by those users whose desktops are authorized to use the specified database.

When you secure an SQL icon in the Icon Options window, the user ID of the desktop where the icon resides becomes the user ID. When a secured icon is mailed, the user ID changes to the user ID of the recipient. The icon cannot be opened or used by the recipient unless his or her user ID has the appropriate privileges for the database. You cannot change the **user ID** field of a secured icon.

If you receive a secured icon from another user and the database does not recognize your user ID, you will not be able to open the icon.

Important: Securing an icon is an irreversible operation. Before you secure an icon that you plan to distribute to other users, check with your database administrator to ensure that their user IDs are authorized to use the specified database.

The preferred way to secure an SQL icon is to configure it to read the user ID and password setting from the desktop's Data Access Controls window. For more information about the Data Access Controls window, see *Working With the Intelligent Decision Server*, or consult your system administrator.

To secure an SQL Entry icon:

1. Make a copy of the SQL Entry icon you intend to secure.
2. Select the icon you want to secure and open the Icon Options window.
3. In the **Display** field, select **Other Data**.
4. In the **Secured** field, select **Yes**.

   A caution message warns you that this is a permanent change.
5. Close the icon.

When you secure an icon it is a good idea to name the icon so you can easily recognize it as a secured icon. For example, always use the same symbol at the beginning of the icon name to indicate that the icon is secured.

For more information about securing icons, see *Administering Database with IDS* and the *System Administration Guide and Reference Guide*.

Creating the Source Document

The source document is a Text document containing the SQL program that is processed in the SQL Entry tool. The SQL program is automatically read from the source document. You can change the SQL program by modifying the source document and copying the contents from the source document into the SQL Entry window.

To create the source document, copy a blank Text icon to your desktop or container in which you will run the SQL Entry tool. The source document must have a unique name.

An open source document with a sample program is shown in Figure 2 on page 8.

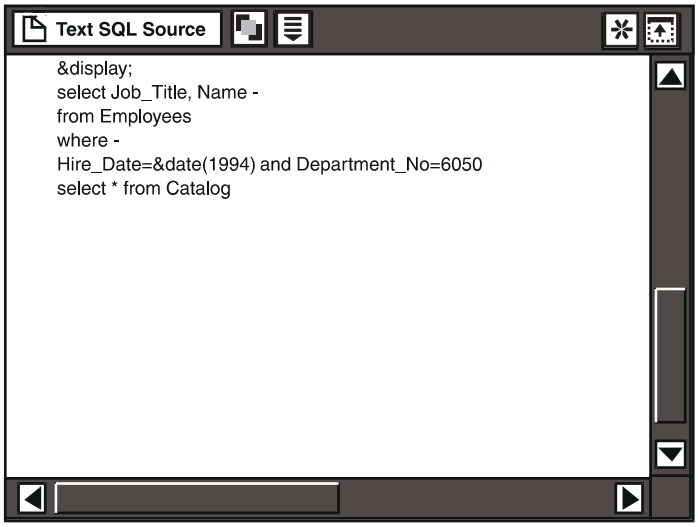


Figure 2. Sample source document

Specifying the Source of the Program

The SQL Control window contains the **Obtain Program Source From** field for specifying the name or path name of the source document. It is important to enter the exact name of the source document so the SQL Entry tool can read the SQL program. The SQL Control window and the **Obtain Program Source From** field containing the name of a source document are shown in Figure 3.

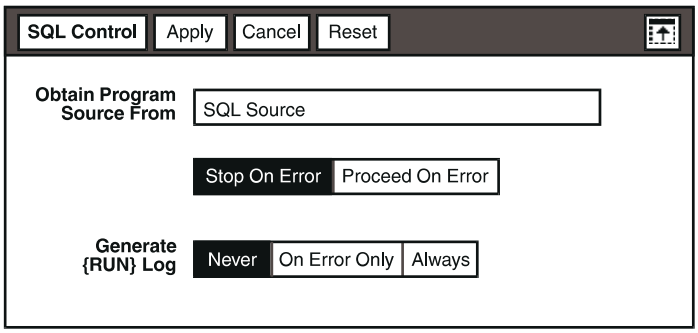


Figure 3. SQL Control window containing the source document name

To specify the name of the source document:

1. In an open SQL Entry window, click on the SQL Control window.
2. In the **Obtain Program Source From** type the name or path name of the source document.

The name must match the source document name or path name exactly, including upper and lowercase characters and spaces.

3. Close the SQL Control window.

When you specify the source document with a path name:

You must use a vertical bar as the delimiter.

The path name can specify one of the file drawers in the File Drawers container, but the specified file drawer must be on the system disk.

The path name must specify the containers that lead to the source document.

For example, if your source document, called SQL Source, is in a folder named Programs in the same container as the SQL Entry tool, specify the path name as:

Programs|SQL Source

However, if the Programs folder is in the Data Entry Icons file drawer, specify the path name as:

Data Entry Icons|Programs|SQL Source

The SQL Entry tool searches for the source document in the following order, using the path you specify:

1. The tool first looks for the path beginning at the same level as the tool. For example, if the tool is in a container, such as a folder or a capsule application, it looks for the path within that container. If the tool is on the desktop, it checks the desktop for a source document with the specified name or path name.
2. If the search within a container is unsuccessful, the SQL Entry tool next searches for the path at the desktop level.
3. If the desktop search is unsuccessful, the SQL Entry tool finally searches for the path to the source document beginning from System|File Drawers for the source document.

The first successful match is used as the source document. If a match is not found, the SQL Entry tool returns an error message.

Editing the SQL Program in the Source Document

The SQL program cannot be modified within the SQL Entry window, but must be edited in the source document. The changed SQL program can then be input into the SQL Entry tool by:

Copying the edited SQL program into the SQL Entry window

Reading the edited SQL program from the source document

When you create and test SQL programs, you can quickly update SQL statements in the SQL Entry window by copying the statements into the SQL Entry window from the source document, except when you click on the **Show Results** button. See “Creating an SQL Log” on page 28 for more information on the **Show Results** button.

To modify the SQL program:

1. Open the SQL Entry icon.
2. Open the source document and edit the SQL program as needed.
3. Click on the **Select All** button, or select as many of the SQL statements as you want to test.
4. Copy the statements into the display area of the SQL Entry window.

Important: The SQL statements you copy into the display area replace any existing statements in the SQL Entry window; they are not appended to the contents of the window.

If SQL statements are not copied into the SQL Entry window, but read from the source document, edits do not take effect until you close the source document, then close and reopen the SQL Entry icon. If you leave the source document open, then close and reopen the SQL Entry icon, the original SQL statements reappear in the SQL Entry window.

To read the modified SQL program from the source document:

1. Open the source document and edit the SQL program as needed.
2. Close the source document to save the edits to the SQL program.
3. Close the SQL Entry icon.
4. Reopen the SQL Entry icon.

Chapter 3. Writing the SQL Program

You can use unique logic statements and functions with the SQL Entry tool that provide short cuts for writing programs. When you write the SQL program, there are certain guidelines you must follow. This chapter contains information on how to:

Use @-variables in the SQL program

Use logic statements in the SQL program

Use functions in the SQL program

Use SQL with the SQL Entry tool

This chapter also includes tables at the beginning of the sections for logic statements and function. These tables provide syntax and a definition of each logic statement and functions. Following each table, each logic statement and function is listed alphabetically with a detailed description and examples.

Using @-Variables

You can include @-variables in your SQL statements when the SQL Entry tool is used within a capsule application. In the source document, enter the @-variable at the appropriate point in the SQL statement. For example:

```
select column1 from table where column2 = @C
```

Using @-Keywords as @-Variables

You cannot directly use the system @-keywords, such as **@Date**, **@User**, and **@Time**, in SQL statements. However, when you use the SQL Entry tool in a capsule application, you can define an @-keyword in the User Input Control window of the Capsule tool as the value of one of the @-variables, such as **@A**. In the following example, **@User** is defined as **@A**.

```
Select   from Sales_data where salesperson = '@A'
```

This example finds and lists all sales data for the user.

For more information on defining @-variables, see the *Capsule User's Guide*.

Substituting String Values for @-Variables

When a string value is substituted for an @-variable, the @-variable must be enclosed by single quotation marks to be correctly interpreted. In this example, **@A** is entered as a quoted string.

```
select Name, Dept_no from Employees where (Number > 65  ) and (Hire_date <= '@A')
```

For more information on using single quotation marks in SQL statements, see the database reference manual for the RDBMS with which you are working.

© Copyright IBM Corp. 1996

Displaying @-Values

To display the actual values of the @-variables used in the SQL program, select **Show @-Values** from the Special menu.

The SQL Entry tool reads the values of the @-variables from the User Input Control window of the Capsule tool.

To hide the @-variables in the SQL program again, select **Hide @-Values** from the **Special** menu.

Using Logic Statements

Logic statements are used in the SQL program for flow control and conditional processing. When you use logic statements with the SQL Entry tool, each logic statement must:

Begin on a new line

Have an ampersand (&) as the first nonblank character of the line

End with a semicolon (;)

Table 2 lists each logic statement and provides syntax and a brief description.

Table 2. Logic statements

Logic Statement	Syntax	Description
&DISPLAY	&DISPLAY;	Specifies the SELECT statement that displays when used in stand-alone mode or that transfers data when used in a capsule application.
&ERROR	&ERROR *action*;	Specifies the action taken by the SQL Entry tool when it encounters errors from SQL statement execution.
&GOTO	&GOTO *label*;	Transfers program control to a line tagged with the &LABEL statement.
&IF	&IF *condition* THEN GOTO *label*;	Conditionally transfers program control to a line tagged with the &LABEL statement.
&IMPMSG	&IMPMSG(*string*');	Posts a message in the Important Message window.
&LABEL	&*label*:;	Specifies a word as a tagged line.
&SECURECONNECT	SECURECONNECT(['*database*'], ['*service*']);	Establishes a connection to the specified database and service.
&SET	&SET *@-variable = expression*;	Sets @-variables during program execution.

The following sections about logic statements contain an explanation, a syntax illustration, remarks when needed, and examples. The logic statements are listed in alphabetical order.

&DISPLAY

The **&DISPLAY** statement allows you to specify the **SELECT** statement that transfers the data from the SQL Entry tool when the tool is used in a capsule application. It also allows you to specify which **SELECT** statement is displayed during stand-alone use.

Syntax

&DISPLAY;

Remarks

Since the **&GOTO** and **&IF** statements can cause nonsequential execution of SQL statements, you can use the **&DISPLAY** statement to identify which **SELECT** statement was used for the output data.

The **&DISPLAY** statement must be on its own physical line and immediately precede the **SELECT** statement that provides the output data.

To output data from the SQL Entry tool, the execution flow must encounter an **&DISPLAY** statement followed by a **SELECT** statement. If the **&DISPLAY** statement is omitted, the output is produced by the last **SELECT** statement in the physical order of the program.

Only one **&DISPLAY** statement should be executed during a single run. If more than one **&DISPLAY** statement is executed in an SQL program, an error results.

Example

In this example, the **&DISPLAY** statement immediately precedes the first statement, so the result of the first **SELECT** statement is output.

```
&DISPLAY;

select Number, Description from Problems where Number > 1

select Number from Problems where Assigned_To = 'John'
```

&ERROR

The **&ERROR** statement allows you to control the action taken by the SQL Entry tool when it encounters errors resulting from SQL statement execution. The effect the **&ERROR** statement has on an SQL program depends upon the options set in the SQL Controls window. See “Stopping the SQL Program” on page 31 for more information.

Syntax

&ERROR action

where:

action Is one of the following:

STOP

MESSAGE

IGNORE

Remarks

The **&ERROR** statement changes the action taken by the SQL Entry tool for all SQL statements executed after this statement and until the next **&ERROR** statement is executed.

The action taken is dependent on the setting of the options in the SQL Controls window. The choices are **Stop On Error** and **Proceed On Error**.

If you select the **Stop On Error** choice in the SQL Controls window, the SQL Entry tool ignores any **&ERROR** statements within the SQL program. Processing stops when an error occurs. An error message is posted in the message bar, Important Message window, or in the SQL Log, if one exists.

If you select the SQL Controls window, the **&ERROR** statement can be used to override this setting. The possible **&ERROR** statements and their results are:

STOP

Processing stops when an error is encountered. An error message is posted in the message bar or Important Message window, or in the SQL Log, if one exists.

MESSAGE

Processing proceeds when an error is encountered. An error message is posted in the message bar, Important Message window, or in the SQL Log, if one exists.

IGNORE

Processing proceeds when an error is encountered. No error message is posted.

Example

Each **&ERROR** statement changes the action taken by the SQL Entry tool. Numerous **&ERROR** statements can be used in an SQL program. For example:

```
/ if update fails then stop /
&ERROR stop;
update table set sales = sales 1
/ if no rows deleted then put out a message /
&ERROR message;
delete from table where sales < 1
/ if select fails for any reason, keep on running /
&ERROR ignore;
select from WT_CONTENTS
```

&GOTO

Use the **&GOTO** statement to transfer control of the program to a line you tag with the **&label** statement. Program control is transferred to the SQL statement that appears immediately after the label.

Syntax

&GOTO label

where:

label Is the label variable as assigned using the **&label** statement

Example

```
&SET @A= ;
&loop:;
&IF @A>1  then goto endit;
&SET @A=@A+1;
insert into Problems_list (number) values (@A)
&GOTO loop;
&endit:;
```

&IF

Use the **&IF** statement to conditionally transfer control to a line you tag using the **&label** statement.

Syntax

&IF condition THEN GOTO label;

where:

condition Is a conditional expression.

label Is the label variable as assigned using the **&label** statement.

Remarks

If the conditional expression evaluates to true, then program control is transferred to the SQL statement immediately following the label; otherwise, program control is transferred to the SQL statement following the **&IF** statement.

Conditional expressions can include @-variables, logical operators (**AND, NOT,** and **OR**), relational operators (=, !=, <, <=, >, and >=), constants (integer, floating point, and strings), and mathematical operators (+, -, *, and /). The + operator can be used to concatenate character strings. Conditional expressions can also contain some of the special functions such as **COLVAL** or special variables such as **NROWS** or **ERRNUM**, and can use parentheses.

The SQL Entry tool converts integers over 100 million into real numbers.

The tagged line can appear before or after the **&IF** statement.

Example

```
select   from WT_CONTENTS
&IF errnum = -1 then goto end;
&IF nrows =   then goto end;
&SET @A = colval(1);
&end:;
```

&IMPMSG

The **&IMPMSG** statement allows you to post an important message in the message bar.

Syntax

```
&IMPMSG('string');
```

where:

string Is a constant text string or an @-variable containing the text of the specified message.

Remarks

The string must be enclosed in single quotes.

Text and @-variables can be used in the message.

Before a message is posted, @-variables are evaluated.

The identifier, **Application Message:**, prefixes each message displayed by an application.

Example

In this example, the **&IMPMSG** statement and the result that is posted to the message bar are shown.

```
&IMPMSG('Error inserting row into database');
Application Message: Error inserting row into database
```

&LABEL

By prefixing a word of your choice with an ampersand (&), you can place a tag or identifier before a line in the source document. With the **&GOTO** statement you can transfer control to the line immediately following the label definition.

The **&LABEL** statement must:

Be on a line by itself

Contain a colon (:) followed by a semicolon (;), as shown in the syntax example

The SQL statement immediately following the label is referred to as a tagged line. The tagged line receives control in the SQL program when a reference is made to the **&LABEL:;** statement immediately preceding it. Refer to "&GOTO" on page 15 for an illustration of the combined use of the **&LABEL** statement and the **&GOTO** statement.

Syntax

&label :;

where:

label Is a word of your choice, such as loop.

Remarks

The comparison of labels is not case sensitive.

You cannot assign a keyword or other reserved word as a label.

Example

This example shows a label assigned to a routine. The routine can then be referenced by other statements. When referenced in an **&GOTO** statement, program control is passed to the SQL statement immediately following the **&LABEL:;** statement.

```
&loopA:;
&set @B=@B+@G;
&if @B>@C then goto &loopD;
insert into Problems_list (number) values (@B)
&loopD:;
select  from Exceptions
```

&SECURECONNECT

The **&SECURECONNECT** statement allows an SQL Entry tool to be connected to different databases and services during program execution. The **&SECURECONNECT** statement establishes connections to the database and service specified in the logic statement.

Syntax

&SECURECONNECT([database][,service]);

where:

database Is the optional name of a database.

service Is the optional name of a service.

Remarks

The names of the specified database and service must be enclosed by single quotes.

The user IDF and password used to establish a connection to a specified database and service are read from the desktop's Icon Options window.

You can specify the database by itself in the **&SECURECONNECT** statement. The SQL Entry tool connects to the database specified in the statement and the service specified in the Icon Options window. You cannot specify the service by itself.

If a database and service are not specified in the **&SECURECONNECT** statement, the SQL Entry tool connects to the database and service specified in the Icon Options window.

There is only one connection at any time during program execution. When the SQL Entry tool makes a new connection, it automatically closes the previous connection.

Example

```
&SECURECONNECT(dbs, HQ);
select   from WT_CONTENTS
&IF nrows =   then goto end;
&SECURECONNECT(mydbs, HQ);
&IF nrows =   then goto end;
&end:;
```

&SET

The **&SET** statement allows you to set @-variables during SQL Entry tool execution.

Syntax

&SET @-variable =expression ;

where:

@-variable	Is the name of an @-variable.
expression	Is a value, logical expression, string expression, or mathematical expression whose result is assigned to the @-variable.

Remarks

A string expression cannot be longer than 4096 characters.

When the program is executed, the @-variable is assigned the value of the expression.

If the value of the expression is over 100 million, the SQL Entry tool converts it into a real number.

The value of the @-variable is changed in the User Input Control window of the Capsule tool containing the SQL Entry icon, which affects all subsequent references to the @-variable, both within the SQL Entry tool and by other tools.

Examples

The following examples illustrate some of the values that can be used as the expression in the **&SET** statement.

```
&SET @G = 'Hello';
&SET @C = 2. 1;
&SET @D = '@C';
&SET @J = datetype(12-25-94)
&SET @F = numtype(12.1, 13.5)
&SET @A = nrows;
&SET @R = colval(2);
&SET @H = @D + 4. ;
```

Using Functions

Functions are used within logic statements to access the state of the SQL program and within SQL statements to generate parts of the SQL statement that would be difficult to specify manually.

Table 3 lists each function and provides syntax and a brief description.

Table 3. Functions

Function	Syntax	Description
COLVAL	COLVAL(*n*)	Returns the value of the specified column of the first row of data of the last executed **SELECT** statement.
&CONSTRAINT	&CONSTRAINT(*column name, data type, operator type, data value*)	Specifies an item in a field value that consists of multiple data items when an unknown amount of items have been assigned to an @-variable.
&DATE	&DATE(*date*)	Converts date strings to the internal representation of dates in the database.
DATETYPE	DATETYPE(*date*)	Identifies the value of an @-variable as a date.
ERRNUM	ERRNUM	Returns a -1 if an error code was returned from the database service by the last SQL statement executed.
NROWS	NROWS	Returns an integer indicating the number of rows affected by the last SQL statement executed.
NUMTYPE	NUMTYPE(*number*)	Identifies the value of an @-variable as a number.

The following sections about each function contain an explanation, a syntax illustration, remarks when needed, and an example. The functions are separated into two categories and listed in alphabetical order.

Using Logic Statement Functions

Logic statement functions are used in the SQL program to access the state of the program. The logic statement functions are primarily used with the **&IF** and **&SET** logic statements.

There are two special variables that are also used in logic statements: **ERRNUM** and **NROWS**.

COLVAL

The **COLVAL** function accesses the results of the last executed **SELECT** statement. It returns the value of the specified column from the first row of data of the last executed **SELECT** statement.

Syntax

COLVAL(*n*)

where:

n Is a specified column of the first row returned by the most recent **SELECT** statement.

Remarks

The **COLVAL** function returns an error message if any of these conditions are true:

- There is no **SELECT** statement
- The previous **SELECT** statement returns no rows
- The **COLVAL** argument is not a valid column number

If the **COLVAL** function is used in a parenthetical expression, **COLVAL** must be at the end of the parenthetical expression for the logic statement to be executed correctly.

Example

```
select   from WT_CONTENTS
&IF errnum = -1 then goto end;
&IF nrows =   then goto end;
&SET @A = colval(1);
&end:;
```

DATETYPE

The **DATETYPE** function identifies the value of the @-variable in the **@SET** statement as a date. This function is useful when a date is entered as a series of numbers separated by nonalphabetic characters, such as a hyphen (–) or slash (/).

Syntax

DATETYPE(*date*)

where:

date Is the variable value of a date.

Remarks

The **DATETYPE** function is required to identify a value set in an @-variable as a date. If **DATETYPE** is not used in an **&SET** statement which consists of numbers and nonalphabetic symbols, the value is treated as a number or expression and assigned a numeric type.

The list of values in a **&SET** statement can contain other @-variables if the referenced @-variables also contain date values identified by the **DATETYPE** function.

Example: In the following example, **@H** in the capsule's User Input Control window would contain the number -107 if the **DATETYPE** function was not used in the **@SET** statement, because (12-25-94) would be interpreted as a mathematical equation. The second **@SET** statement would then return an error.

```
&SET @H = datetype(12-25-94);
&SET @J = datetype(@H);
```

ERRNUM

The **ERRNUM** variable returns the integer values –1 if an error was returned from the database service for the last SQL statement executed.

Syntax

```
ERRNUM
```

Remarks: If an error does not occur in the SQL program, or if there is no SQL statement executed by the SQL Entry tool, the value is 0.

Example

```
select   from WT_CONTENTS
&IF errnum = -1 then goto end;
&SET @A = colval(1);
&end:;
```

NROWS

The **NROWS** variable returns an integer indicating the number of rows affected by the last SQL statement executed. This function is useful for checking that a **SELECT**, **INSERT**, **UPDATE**, or **DELETE** statement has affected the correct number of rows.

Syntax

```
NROWS
```

Example

```
select   from WT_CONTENTS
&IF nrows =   then goto end;
&SET @A = colval(1);
&end:;
```

NUMTYPE

The **NUMTYPE** function identifies the value of the @-variable in the **@SET** statement as a number.

Syntax

NUMTYPE(number)

where:

number Is a number or list of numbers.

Remarks

If the number argument in the **@SET** statement is a single value, the **NUMTYPE** function is not required. A single value can be an expression.

The **NUMTYPE** function must be used in an **&SET** statement with more than one value. A list of numeric values must be constants and be separated by commas.

When the **NUMTYPE** function is used, the values cannot be mathematical expressions that must be evaluated before being assigned to an @-variable.

The list of values in a **&SET** statement can contain other @-variables if the referenced @-variables also contain numeric values.

Example: In the following example, **@B** in the capsule's User Input Control window will contain two numbers, **12.1** and **13.5**.

```
&SET @B = numtype(12.1, 13.5);
```

If the **NUMTYPE** function were not used in the **&SET** statement, **@B** would contain the string **12.1** 0.

Using SQL Statement Functions

SQL statement functions are used in the SQL program to generate parts of the SQL statement that would be difficult to specify manually. SQL statement functions begin with an ampersand (&).

&CONSTRAINT

The **&CONSTRAINT** function allows you to specify an item in a field value that consists of multiple data items when you do not know how many items have been assigned to an @-variable.

Syntax

&CONSTRAINT(column name, data type, operator type,
data values)

where:

column name	Specifies the table column that is to be constrained.
data type	Specifies the type of data used as the constraint. Valid types are one of the following: **CHAR** (including **VARCHAR**) **NUMBER** (including integer and real) **DATE**
operator type	Is either **EQUAL** or **NOTEQUAL**. If **NOTEQUAL** is specified, the constraint is negated by means of the **NOT** operator.
data values	Are zero or more items used as the constraint value.

Remarks

If the data type specified as the parameter is **CHAR**, the constraint value is quoted. If the constraint for **CHAR** consists of multiple values, quotation marks are added to each value delimited by a comma.

If the data type specified as the parameter is **DATE**, the constraint value is converted to the appropriate service date.

For single data items, the **=** comparison operator is used. For multiple data items, the **IN** operator is used.

If the SQL statement does not have existing constraints, the **WHERE** clause is added. If a constraint already exists in the current SQL statement, then the **AND** operator is used to relate the existing constraint with the one being added by the **&CONSTRAINT** function.

The **&CONSTRAINT** function must be the last constraint in a call for the correct keyword to be added to the SQL statement.

The **&CONSTRAINT** function must follow the **SELECT** statement it modifies and cannot be preceded by an embedded **SELECT** statement.

Examples: In the following examples, statements using the **&CONSTRAINT** functions and the resulting SQL statements are shown.

If @A = NY :

```
select sales from salestable &CONSTRAINT(state, char, equal, @A)

select sales from salestable where state = 'NY'
```

If @D = A1, Smith, and Star :

```
select sales from salestable &CONSTRAINT(brand, char, equal, @D)
select sales from salestable where brand in ('A1', 'Smith', 'Star')
```

If @D = A1, Smith, and Star, and NOTEQUAL is the third parameter:

```
select sales from salestable where city = 'Atlanta'
&CONSTRAINT(brand, char, notequal, @D)
select sales from salestable where city = 'Atlanta'
and not (brand in ('A1', 'Smith', 'Star'))
```

&DATE

The **&DATE** function allows you to convert dates to the internal representation of dates in the database to which the SQL Entry tool is connected. The appropriate representation is necessary in an **INSERT** statement and in the **WHERE** clause of a **SELECT** statement.

Syntax

&DATE(*date*)

where:

date Is the variable value of a date, specified in the same format as set in your desktop, and can be any one of the following formats shown in Table 4.

Table 4. Acceptable date formats

Date type	Formats
Month	11 nov November
Odd-Bimonth	DJ, 95
Even-Bimonth	ND, 1995
Quarter	1q95 1Q1995
Year	95 1995

Remarks

The **&DATE** function has one date argument. You can use either upper or lowercase letters when specifying the date.

The **&DATE** function cannot be used to set @-variables.

When used in an **INSERT** statement, the **&DATE** function returns a single value representing the starting date in the range.

When used in a **WHERE** clause, if the comparison operator is **EQUAL** or **NOTEQUAL** and **&DATE** represents a range of date values in a period, the **&DATE** function returns a **BETWEEN** operator.

When you mail an SQL Entry icon to a different desktop, the icon uses the formats from the recipient's desktop. The desktop date and number formats of the recipient must be the same as your desktop date formats or data in the icon might be incorrect.

Examples: The following example produces an SQL statement returning the names of all employees born between January 1, 1959 and December 31, 1959.

```
/ find employees born in 1959 /
```

```
select Name from Employees where (birthday=&DATE(1959))
```

The following example produces an SQL statement returning the names of all employees born after December 31, 1959.

```
/ find employees born after 1959 /
select Name from Employees where (birthday>&DATE(1959))
```

The following example produces an SQL statement returning the names of all employees not born between January 1, 1959 and December 31, 1959.

```
/ find employees not born in 1959 /
select Name from Employees where not(birthday=&DATE(1959))
```

Using SQL with the SQL Entry Tool

The following sections contain guidelines for using standard SQL with the SQL Entry tool.

Continuing Statements

You can use a space followed by a hyphen (-) as a continuation character to split a single SQL statement into multiple lines in the source document. Spaces are allowed after the continuation hyphen, but the last printable character in the line must be a hyphen (-). This example shows a typical continuation:

```
select Dept_no from Employee where -
(Number > 65  ) and (Hire_date <= &DATE(jan 1,94))
```

Although this SQL statement appears as two separate lines, it is executed as one SQL statement by the SQL Entry tool.

Continuation characters cannot be used with logic statements.

If you type a statement that wraps at the margin of your source document, it is still considered a single line in the SQL Entry tool. The Enter key and Shift+Enter keys are considered line separators.

Including Comments

Comments are used to clarify an SQL statement or program and are not sent to the database service.

To include a comment in your SQL program, type / before the comment and / at the end of the comment. For example:

```
/ These file numbers indicate departments selected for relocation /
```

The following rules apply to including comments in an SQL program:

- Each comment must be contained on a single line.

A comment in the last **SELECT** statement of the SQL program must appear after the first word of the **SELECT** statement. If the comment is placed before the first word, then the statement is not recognized by the SQL Entry tool as the last **SELECT** statement in the program. It is executed, but the results are not displayed.

A comment cannot appear directly after the **&DISPLAY** statement. If the comment is placed after the **&DISPLAY** statement, the **SELECT** statement is not recognized by the SQL Entry tool as the specified statement to display, and the SQL Entry tool returns an error.

Chapter 4. Running the SQL Program

The SQL program processes SQL statements differently depending on whether the SQL Entry tool is run in a container, on the desktop, or run closed within a capsule application. While the program is running, it can be captured in an SQL Log, so that errors are easily detected. Data can be transferred from the SQL Entry tool to other tools after the program is finished running. This chapter contains information on how to:

Run the SQL Entry tool in stand-alone mode

Run the SQL Entry tool in a capsule application

Stop the SQL program

If you create any temporary tables when the SQL program is run, you need to remove them by including the DROP command before the end of the SQL program.

Running the SQL Entry Tool in Stand-alone Mode

The following conditions apply to running the SQL Entry tool in stand-alone mode:

The SQL Entry tool is open in a container or on your desktop while the SQL statements are being processed.

It is not possible to automatically send output to another tool.

To run the SQL Entry tool in stand-alone mode:

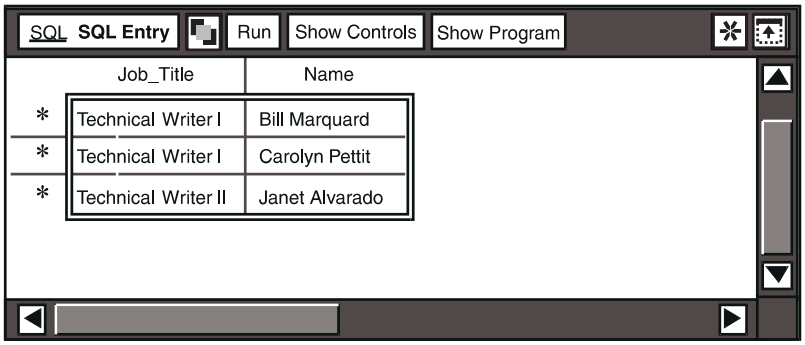
1. Open the SQL Entry icon.
2. Click on the SQL Entry window header.

An arrow points to each statement as the statement is executed.

When the program is complete, the results of the last executed **SELECT** statement appear in the SQL Entry window in the form of a spreadsheet. If you used the &DISPLAY function to display the results of a different **SELECT** statement, those results appear in the SQL Entry window.

The SQL Entry tool truncates lengthy lines in output. The length of the line that can be displayed in its entirety varies according to the font and size you have chosen for your desktop. As a rule of thumb, the window can hold about twice the width of the screen in characters without truncating data.

© Copyright IBM Corp. 1996



The **Show Program** button appears in the SQL Entry window header after the results of the SQL program are returned.

To display the program again, click on the **Show Program** button. The label in the button changes to **Show Results**.

You cannot copy SQL programs into the SQL Entry window while results are being displayed.

Creating an SQL Log

When you run the SQL Entry tool in stand-alone mode, you have the option of capturing the SQL program as it runs in a specially created text document, the SQL Log. The SQL Log does not capture the results of the program, but it does capture:

The actual execution flow of the program

The logic statements

The SQL statements as they appear after being sent to the database

The actual time the SQL statements were sent to the database

The number of rows (tuples) that were processed

Any error messages

To create the SQL Log:

1. Open the SQL Entry icon.
2. Select **Capture SQL** in the Special menu.

   The SQL Log window appears on the desktop. The created Text document is assigned the name SQL Log, with the current date and time.
3. Click on the SQL Entry window header.

   The SQL Log is shown in Figure 4 on page 29.

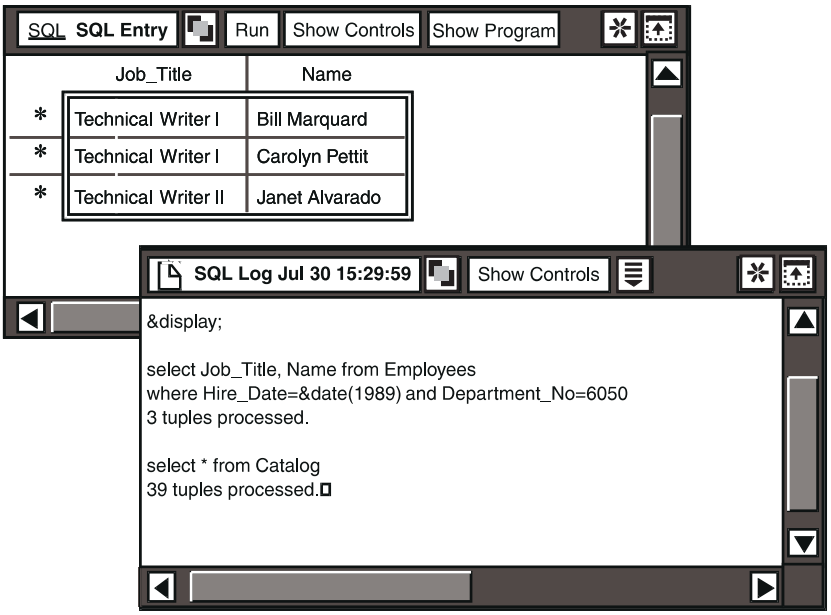


Figure 4. SQL Log window

4. Click on the title block of the SQL Log window to close it after the SQL program has finished running.

   If you close the SQL Log window while the program is running, the SQL Log stops capturing information.

If an error is encountered in the SQL program, an error message is sent to the SQL Log. A message is also posted to the message bar or Important Message window, indicating that an error occurred. If the SQL Log is not generated, the error messages are posted as important messages in the message bar or Important Message window.

Copying Information to Other Tools

The SQL program or its results can be transferred from the SQL Entry window by copying the information to other tools.

To transfer information from the SQL Entry window:

1. Click on the **Select All** button in the window header of the SQL Entry window, or select the specific rows or columns you want to copy.
2. Copy the information to the destination icon, such as a Spreadsheet or Text icon.

Information cannot be moved or deleted out of the SQL Entry window.

Running the SQL Entry Tool in a Capsule Application

The following conditions apply to running the SQL Entry tool in a capsule application:

The SQL Entry icon must be closed.

The SQL Entry icon cannot have incoming arrows.

The SQL Entry icon must have only one outgoing arrow to transfer results of a **SELECT** statement to another icon.

You can use an @-variable, such as **@A**, and define the @-value as an @-keyword, such as **@DATE**, in the User Input Control window of the capsule tool.

To run the SQL Entry tool in a capsule application, click on the capsule window header.

The SQL statements are processed in three phases:

1. The SQL statements up to, but not including, the **SELECT** statement which transfers data to another tool, are processed.
2. Either the last **SELECT** statement in the SQL program or the **SELECT** statement prefaced by **&DISPLAY** is executed.
3. The outgoing arrow and everything following the **SELECT** statement which transfers data are executed by the capsule application.

If you run the SQL Entry tool in an open capsule window, the SQL Entry tool runs as if it is in stand-alone mode, although defined @-variables are still processed.

Creating an SQL Log

When you run the SQL Entry tool in a capsule application, you have the option of capturing the program as it runs to a specially created System Logging icon, the SQL Log. The SQL Log does not capture the results of the program, but it does capture:

The actual execution flow of the program

The logic statements

The SQL statements as they look after being sent to the database

The actual time the SQL statements were sent to the database

The number of rows (tuples) that were processed

Any error messages

When SQL Logs are generated, they reside inside a Folder icon named LOGS that is automatically created and placed on your desktop. The SQL Log is a closed System Logging icon, and is assigned the name of the SQL Entry icon from which it was generated, the word Log, and a date and time stamp.

Do not confuse the SQL Log with the capsule window.

To create the SQL Log:

1. Open the SQL Entry icon.
2. Click on the **Show Controls** button in the SQL Entry window header to display the **Generate SQL Log** field in the SQL Controls window.
3. In the **Generate SQL Log** field, select one of the settings described in the following table.

Setting	Function
Never	Never generates an SQL Log.
On Error Only	Generates an SQL Log only if an error occurs. The SQL Log contains all of the statements executed by the program, regardless of where the error occurred in the program.
Always	Generates an SQL Log each time the program is run.

   Do not generate an SQL Log if the program is very lengthy or has long iterations. When an SQL Log becomes full, another is created to continue capturing information. Multiple SQL Logs could use all available disk space.
4. Close the SQL Controls window to retain the setting for the **Generate SQL Log** field.
5. Close the SQL Entry window.
6. Click on the capsule window header to begin running the SQL program.

You can sort the information in the SQL Logs or sort the SQL Logs themselves within the LOGS folder. See *Working With the Intelligent Decision Server* for more information on sorting with the System Logging and Folder icons.

Transferring the Results of the SQL Program

The results of the SQL program are automatically transferred from the SQL Entry icon to the connected icon when the SQL program is finished running. The **&DISPLAY** statement specifies which information is transferred from the SQL Entry icon. Icons that can accept data from the SQL Entry icon are Text, Spreadsheet, Plot, Layout, and Transformer icons. For more information about connecting icons and the capsule tool, see the *Capsule User's Guide*.

Stopping the SQL Program

It is possible to stop the SQL program either automatically when an error is encountered in SQL statement execution or manually.

You can set the SQL program to automatically stop when an error is generated. To set this option, select **Stop On Error** in the SQL Controls window before you run the program.

To continue running the SQL program even if an error is encountered, select **On Error** in the SQL Controls window.

The **&ERROR** statement can be used to override the **Proceed On Error** option. See "&ERROR" on page 13 for more information.

To stop the SQL program manually at any time:

1. Press the Stop key.

To begin running the SQL program again:

1. Click on the SQL Entry window header.

Glossary

@-keyword. A system-defined variable that returns current values within a capsule application or within the headers and footers of a printable icon. System-supplied @-keywords cannot be used directly in SQL statements, but they can be defined as the value of an @-variable.

@-value. The value assigned to an @-variable in the capsule's User Input Control window. In the SQL Entry tool, the &SET statement allows you to define the value of an @-variable as an expression which is evaluated to an actual value when the statement containing the expression is processed.

@-variable. A control variable used to represent a value in any tool that can be run inside a capsule application. These values are frequently used as constraints and variable titles in the Query, Plot, Spreadsheet, Layout, and other IDS tools.

capsule. A tool for building an automated processing sequence using other icons

conditional processing. Processing of logic statements such as &IF.

container. An icon, such as the Folder and File Drawer icons, that can hold other icons.

database administrator (DBA). The person who issues administrative instructions to a database service. The DBA defines the data tables used by data access tools, monitors database performance, and assigns user access.

DB2. DATABASE 2, an IBM relational database management system that runs on an MVS system.

display area. An area of the SQL Entry window where the SQL statements are shown as they are executed, and the results of the SQL program are shown.

internal name. In the IDS environment, the name by which a database item is stored in the Workstation Tools Data Dictionary (WTDD). Each database category, table, and column has an internal name based on naming conventions required by the database system. To make these names more useful to users, each internal name has a corresponding external name, called a print name, that the IDS tool user sees on the screen. For example, a table with the internal name DBA_EMP might be known to the user as the Employee table. Contrast with print name.

iteration. The process of repeatedly running a set of instructions until some condition is satisfied. The SQL Entry tool performs iterations using the &IF and &GOTO statements.

logical database. A database representation that includes print names for tables and columns, and that groups the tables into useful categories. The logical database also contains definitions for default joins between tables. The Workstation Tools Data Dictionary (WTDD) stores the description of the logical database, the mapping of print names to internal names in the physical database, and other logical attributes, such as the type and length of data.

macro. An IDS text document in the DBA file drawer that contains SQL statements and that can contain substitutable values that are passed when executed. You can use macros through either the host interface or the SA tool on a desktop, but not with the SQL Entry tool.

password. A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a desktop, gateway, database, system, or service within a system.

path name. A designation for the location of a file or icon.

physical database. The database as seen by the database management system. It is a set of physically stored data tables, grouped by user, that is represented to IDS data access tool users by the logical database.

print name. The name of a database item (such as a column or table) that is used in IDS windows instead of the more cryptic name by which the data item is identified in the database. Print names are identified in the Workstation Tools Data Dictionary (WTDD) and are converted during data retrieval to internal names that can be understood by the database. Contrast with internal name.

RDBMS. See relational database management system.

relational database management system (RDBMS). The software that makes it possible to create, use, and maintain relational data in the form of tables consisting of rows and columns of data.

© Copyright IBM Corp. 1996

source document. A Text icon that contains the SQL program that is read into the SQL Entry tool.

SQL. Structured Query Language, a command language for use with relational databases. The language consists of statements to insert, update, delete, query, and protect data. IDS tools use SQL to communicate with database servers.

SQL Entry tool. A tool used to run Structured Query Language (SQL) programs.

SQL log. A Text or System Logging icon used to record SQL statements as they are processed. The SQL log also includes a beginning and ending time stamp, and processing statistics.

stand-alone mode. The mode in which the user runs the SQL Entry icon interactively and open in a container.

Structured Query Language. **See** SQL.

table. In a database, a single file, consisting of data organized in columns and rows.

tagged line. In an SQL program, the line immediately following the line containing the &label statement. The tagged line takes control of the program when an &GOTO statement passes control to the tagged line.

user ID. A name used to access a desktop, system, gateway, or database service.

Workstation Tools Data Dictionary (WTDD). A logical data dictionary that standardizes the IDS desktop view of data across many relational database management systems. The WTDD stores the print names of database items, defines how the items are to be displayed, and includes other logical attributes of data. The WTDD is implemented as a set of standard relational tables that can be managed through the DBA tool.

WTDD. See Workstation Tools Data Dictionary.

IDS Publications

This section lists IDS publications, including order numbers. To order copies of the books listed here, or to get more information about a book, see your IBM representative.

General Publications

Getting Started with the Intelligent Decision Server (SC26-8892)

Working with the Intelligent Decision Server (SC26-8893)

Installation, Configuration, and Administration Publications

Installing IDS LAN Components (GC26-8877)

System Administration Guide and Reference (SC26-8890)

Messages and Codes (GC26-8878)

Database Gateway Services Publications

Installing Database Gateway Services for UNIX Environments (GC26-8884)

Configuring and Administering Database Gateway Services (GC26-8885)

Administering Databases with IDS (SC26-8891)

Host Services Publications

Planning MVS Host-to-LAN Connectivity (GC26-8880)

Installing Host Services and Configuring Host-to-LAN Communications for MVS (GC26-8881)

Administering Host Services for MVS: DB2 and Cooperative Application Services (GC26-8882)

Developing Applications with Cooperative Application Services (GC26-8883)

Configuring and Administering Mail Gateway Services (GC26-8923)

Open Data Access Service Publications

Developing Applications with Open Data Access Service (SC26-8888)

Data Access Tools Publications

Browser User's Guide (SC26-8900)

Data Entry User's Guide (SC26-8897)

DBA Tool User's Guide (SC26-8894)

SQL Entry User's Guide (SC26-8898)

Data Analysis Tools Publications

BASIC Tool User's Guide (SC26-8908)

Query User's Guide (SC26-8901)

Spreadsheet User's Guide (SC26-8904)

Presentation Tools Publications

Layout User's Guide (SC26-8902)

Plot User's Guide (SC26-8905)

Reporter User's Guide (SC26-8903)

Text User's Guide (SC26-8896)

Application Development Tools Publications

Transformers Guide (SC26-8907)

Capsule User's Guide (SC26-8906)

Forms Design User's Guide (SC26-8899)

Interacting with PC Applications (SC26-8895)

Installing and Configuring IDS Open Clients (GC26-8889)

© Copyright IBM Corp. 1996

Index

Special Characters

@-keywords 11
@-values 12
@-variables
  defining @-keywords as 11
  displaying @-values 12
  using in a capsule tool 11
&CONSTRAINT function 22, 23
&DATE function 24
&DISPLAY statement 13
&ERROR statement 13
&GOTO statement 15
&IF statement 15
&IMPMSG statement 16
&LABEL statement 16
&SECURECONNECT statement 17
&SET statement 18

A

Always setting 30

B

buttons
  Run 27
  Show Controls 30
  Show Program 28
  Show Results 28

C

capsule application, using SQL Entry icon in 30, 31
Capsule tool, using @-variables in 11
Capture SQL choice 28
Category 4
COLVAL function 20
comments, including 25
continuation character 25
continuing SQL statements 25

D

Database Name field 4
date formats, acceptable 24
DATETYPE function 20

E

ERRNUM function 21
errors 13, 14, 31

F

fields
  Database Name 4
  Generate SQL Log 30
  Obtain Program Source From 8
  Server 4
  SQL ID 4
functions
  &CONSTRAINT 22, 23
  &DATE 24
  COLVAL 20
  DATETYPE 20
  descriptions 19
  ERRNUM 21
  logic statement 20, 22
  NROWS 21
  NUMTYPE 22
  SQL statement 22, 25
  using 19, 25

G

Generate SQL Log field 30

H

Hide @-Values choice 12

I

Icon Options window
  fields 4
  Name and Size page 6
  Other Data page 3, 7
information, transferring 31

© Copyright IBM Corp. 1996

L

logic statements
- &DISPLAY 13
- &ERROR 13
- &GOTO 15
- &IF 15
- &IMPMSG 16
- &LABEL 16
- &SECURECONNECT 17
- &SET 18
- descriptions 12, 13
- functions 20, 22
- using 12, 19

N

Name and Size page 6
Never setting 30
NROWS function 21
NUMTYPE function 22

O

Obtain Program Source From field 8
On Error Only setting 30
Other Data page 3, 7

P

Password field 4
path name 9
Proceed On Error setting 31

Q

quotation marks, using 11

R

Reporter icon
- securing 6

Run button 27
running the SQL program 13, 14, 31

S

securing an SQL Entry icon 6, 7
securing icons 6
Server field 4
Show @-Values choice 12
Show Controls button 30
Show Program button 28
Show Results button 28
source document
- creating 7
- editing 9
- illustration 2, 7
- naming 7
- path name 9
- specifying 8, 9

Special menu
- Capture SQL choice 28
- Hide @-Values choice 12
- Show @-Values choice 12

SQL Control window
- illustration 8
- Obtain Program Source From field 8

SQL Controls window
- Generate SQL Log field 30
- illustration 1, 30
- Proceed On Error setting 31
- SQL log settings
  - Always 30
  - Never 30
  - On Error Only 30
- Stop On Error setting 31

SQL Entry icon
- configuring 3, 6
- renaming 6
- securing 6, 7
- using
  - in a capsule application 30, 31
  - in stand-alone mode 27, 29

SQL Entry tool, using SQL with 25
SQL Entry window, display area 1
SQL extensions 1
SQL ID field 4
SQL Log
- creating 28, 29
- window 28

SQL log, creating 30, 31
SQL program
- errors 13, 14, 31
- modifying 9
- running 13, 14, 31
- stopping
  - automatically 13, 14, 31
  - manually 32

SQL statements
  continuing 25
  functions 22, 25
stand-alone mode 27, 29
statements
  logic 12, 19
  SQL 25
Stop On Error setting 31
stopping the SQL program 13, 14, 31
strings, entered as values 11

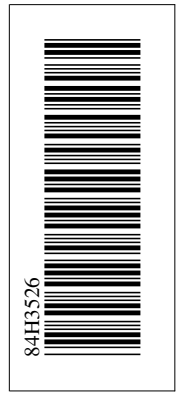
T

temporary tables, removing 27
text, entered as values 11
transferring information 31

IBM®

Part Number: 84H3526

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.



SC26-8898-