Wonderware® FactorySuite™ SQL Access Manager

User's Guide

Revision A
December, 1997

Wonderware Corporation

100 Technology Drive
Irvine, CA 92618
U.S.A.
(714) 727-3200
http://www.wonderware.com

Trademarks

Contents

CHAPTER 1

SQL Access Manager

Wonderware® FactorySuite™ SQL Access Manager allows you to access, modify, create and delete tables in a database. A database stores information in tables that share a common attribute or field. Structured Query Language (SQL) is the language used to access that information.

Contents

Introduction

The InTouch SQL Access Manager add-on program is designed to easily transfer data, such as batch recipes from a SQL database to an InTouch application. It also facilitates the transfer of run-time data, alarm status or historical data from InTouch to the SQL database. For example, after a machine cycle is completed, a company may need to save several sets of data, each for a different application. SQL databases provide the ability for information to be transferred between one or more third-party applications easily. SQL Access Manager allows this data to be accessed and displayed in any InTouch application.

The InTouch SQL Access product consists of the SQL Access Manager program and the SQL Functions. The SQL Access Manager program is used to create and associate database columns with tagnames in your InTouch tagname dictionary. The process of associating database columns and InTouch database tagnames is called "binding." Binding the InTouch database tagnames to the database columns allows the SQL Access Manager to directly manipulate the data in the database. SQL Access Manager saves the database field names and their associations in a comma-separated variable (.CSV) formatted file named "SQL.DEF." (This file resides in the InTouch application directory and may be viewed or modified using SQL Access Manager or any text editor, such as Notepad.) The SQL Access Manager also creates Table Templates defining database structure and format.

For more information on Bind Lists and Table Templates, see Chapter 3 - Configuring SQL Access Manager.

SQL Functions can be used in any InTouch action script. These functions can be used to automatically execute based on operator input, a tagname value changing or when a particular set of conditions exist. For example, if an alarm condition exists, the application would execute a **SQLInsert()** or **SQLUpdate()** command to save all of the applicable data points and the state of the alarm. The SQL Functions can be used to create new tables, insert new records into tables, edit existing table records, clear tables, delete tables, select and scroll through records, etc.

InTouch SQL Access uses Intersolv/Q+E Software's Database Library QELIB (which works with a variety of different database systems) and contains all code necessary to interface with ODBC compliant database drivers. These drivers are not included with InTouch SQL Access software product and must be purchased separately (from manufacturers like Intersolv/Q+E and Microsoft Corporation). ODBC drivers are required as the actual "connection" to the database. For example, if attempting to implement SQL functions to a Microsoft® Access database, the Microsoft® ODBC Desktop Driver Set must be purchased, installed and configured separately. The next section of this chapter contains some general examples for driver configuration.

Note Database systems not discussed in this user's guide are not supported.

About this Manual

This manual is divided into a series of logical building block chapters that describe the various aspects of using SQL Access Manager. It is written in a "procedural" format that tells you in numbered steps how to perform most functions or tasks.

If you are viewing this manual online, when you see a cross reference like this one, it is actually a "hot link" to the referenced section or chapter. Click it to "jump" to that section or chapter. When you jump to another section or chapter and you want to come back to the original section, a "back" option is provided.

These types of cross references indicate that you need to look in another FactorySuite book for more information.

These are "tips" that tell you an easier or quicker way to accomplish a function or task.

The *FactorySuite Systems Administrator's Guide* provides you with complete information on the other component programs in the suite, system requirements, networking considerations, product integration, technical support, and so on.

The *InTouch User's Guide* will help you familiarize yourself with the WindowMaker development environment and its tools, read Chapter 1, "WindowMaker Program Elements." To learn about working with windows, graphic objects, wizards, ActiveX controls and so on, read Chapter 2, "Using WindowMaker."

For details on InTouch runtime environment (WindowViewer), see your online *InTouch Runtime User's Guide*.

In addition, the *InTouch Reference Guide* provides you with an in-depth reference to the InTouch script language, system tagnames, and tagname **.fields**.

Online documentation is included in your FactorySuite software package for all FactorySuite components included in your package. For example, FactorySuite System Administrator's Guide, SPC, SQL Access Manager, Recipe Manager, IndustrialSQL Sever, InControl and all Wonderware 32-bit I/O Servers. If you purchase FactorySuite+ you also get the online documentation for the InTrack and InBatch components.

Assumptions

This manual assumes you are:

- Familiar with the Windows 95 and/or Windows NT operating system working environment.
- Knowledgeable of how to use of a mouse, Windows menus, select options, and accessing online Help.
- Experienced with a programming or macro language. For best results, you should have an understanding of programming concepts such as variables, statements, functions and methods.

Technical Support

Wonderware Technical Support offers a variety of support options to answer any questions on Wonderware products and their implementation.

Prior to contacting technical support, please refer to the relevant chapter(s) in your *SQL Access ManagerUser's Guide* for a possible solution to any problem you may have with your system. If you find it necessary to contact technical support for assistance, please have the following information available:

1. Your software serial number.
2. The version of InTouch you are running.
3. The type and version of the operating system you are using. For example, Microsoft Windows NT Version 4.0 workstation.
4. The exact wording of system error messages encountered.
5. Any relevant output listing from the Wonderware Logger, the Microsoft Diagnostic utility (MSD), or any other diagnostic applications.
6. Details of the attempts you made to solve the problem(s) and your results.
7. Details of how to recreate the problem.
8. If known, the Wonderware Technical Support case number assigned to your problem (if this is an on-going problem).

📖 For more information on Technical Support, see your online *FactorySuite System Administrator's Guide*.

Viewing Your FactorySuite License

Your FactorySuite system license information can be viewed through the license viewing utility that is launched from the InTouch Help **About** dialog box.

To access **About** dialog box, select the **About** command on the InTouch **Help** menu.

📖 For more information on the licensing viewing utility, see your *FactorySuite System Administrator's Guide*.

ODBC Compliant

SQL Access Manager is an ODBC compliant application that communicates with any database system, provided the database system has an ODBC driver available for it. Before you can use an ODBC driver, it must be configured via the Microsoft ODBC Administrator program to set up the links between the ODBC compliant application and the database.

➢ **To configure an ODBC driver:**

1. Run the Microsoft ODBC Administrator program.
2. Select a driver or data source, and then click **Add New Name**, **Set Default** or **Configure**. The **ODBC Driver Setup** dialog box.

Option	Description
Data Source Name	User-defined name which identifies the data source.
Description	User-defined description of this data source.
Database Directory	Identify the directory that contains the database files. If none is specified, the current working directory is used.

   - Enter any other information required to configure the selected driver.

3. Click **OK**.
   - The driver writes the values of each field to the ODBC.INI file. These values are the default values of a connection to the data source. The default values can be changed by modifying the data source fields. Entries can be inserted manually in the appropriate data source section of the ODBC.INI file for any attribute that is not supported by the ODBC Driver Setup dialog box.

Connection Requirements

The connection string used by the **SQLConnect()** function in Version 4.x of InTouch SQL Access Manager is the convention "DRV=database_dll".

Because Version 5.6 (or later) is an ODBC compliant application, the database_dll field in the "DRV=database_dll" must be the exact name of the datasource as you configure it using the ODBC Administrator Program (described above). Alternatively, "DSN" can be used in place of "DRV."

CHAPTER 2

Configuring and Connecting Databases

The databases included in this chapter have been tested and are currently supported. Each database's requirements are unique and particular. This chapter includes separate sections for each database, describing how to configure the particular database for communication with SQL Access Manager.

Contents

Using Oracle 6

➢ **To communicate with Oracle 6:**

1. Configure your Windows database client.
2. Start the SQL*Net TSR and the NETINIT.EXE program.
3. Connect to Oracle® by executing the **SQLConnect()** function in an InTouch action script.

For more information on SQLConnect(), see Chapter 4 - Using SQL Functions.

Configuring the Client

Oracle databases are accessed by installing Oracle's SQL*Net product on the local computer. The Oracle SQL*Net product includes the SQL*Net TSR and the NETINIT.EXE program. Both SQL*Net TSR and the NETINIT.EXE must be running in order to communicate with the Oracle database server.

Starting the SQL*Net TSR and NETINIT.EXE

The SQL*Net TSR appropriate for the network being used must be loaded from DOS before running Windows. The NETINIT.EXE program must be started in order to enable communications to the Oracle server. To automatically execute NETINIT.EXE when Windows is started, place its icon in the Windows StartUp program group.

SQLConnect() Format

The **SQLConnect()** function is used to connect to Oracle databases. The connection string used by the **SQLConnect()** function is formatted as follows:

```
SQLConnect(ConnectionId,"<attribute>=<value>;<attribute>=<value>;
...");
```

The following describes the attributes used by Oracle. They must be specified in the following order:

Attribute	Value
DSN	The name of the data source as configured in the Microsoft ODBC Administrator, or
DRV	For compatibility with SQL Access Manager , this value is used if a data source name (DSN) is not present in the connection string. QELIB changes it to the data source name.
UID	User name.
PWD	Password.

Attribute	Value
SRVR	The SQL*Net connect string designating the server computer and database to be accessed.

Example:

```
SQLConnect(ConnectionId,"DSN=Oracle_Data;UID=SCOTT;PWD=TIGER;
SRVR=B:MKTG_SRV");
```

Where: **B:** specifies the prefix for the NetBIOS communications layer (in the **SRVR=B:MKTG_SRV** part of this string), or

T: specifies the prefix for TCP/IP, or

P: specifies the prefix for Named Pipes

MKTG_SRV is the name assigned to the Oracle Database Server.

Note A difficult attribute value to specify is the SRVR value. The information required varies depending on the SQL*Net driver. This information can be found in the SQL*Net documentation (or possibly in the CONFIG.ORA file on the "remote=" line).

Data Types Supported

SQL Access Manager supports two data types for Oracle 6 databases. The char data type contains variable length character strings. InTouch Message tagnames require a char data type. If a length is not specified, the default is one (1) character. Oracle supports a char field with a maximum length of 255 characters. However, InTouch Message tagnames are limited to 131 characters. If a message variable contains more characters than the length specified for a database field, the string will be truncated when inserted into the database.

The number data types represent InTouch Integer and Real tagnames. If the length is not specified, the value will be represented as a floating point value with 38 digits of precision. If the length is specified, the format is Width.Decimal. The Width value determines the maximum number of digits for the column. The Decimal value specifies the number of digits to the right of the decimal point. A field length for this data type is not required.

Using Oracle 7.2

➢ **To communicate with Oracle 7.2:**

1. Install Oracle Standard Client on your InTouch Node.
2. Run SQL_Net Easy Configuration to locate Database Alias for SQL connection string.
3. Create a Data Source name.
4. Connect to Oracle by executing the **SQLConnect()** function in an InTouch action script.

   ✍ For more information on the usage of **SQLConnect()**, see Chapter 4, "SQL Functions."

Configuring the Client

➢ **To run the Oracle 7 setup program:**

1. On the Oracle 7 Workgroup Server screen, select **Standard Client** as the type of installation you want to perform, and then click **OK**.
2. Select **Application User Setup** as the type of client installation you want to perform, and then click **OK**.
3. From the **Database Connection Setup** screen, enter the hostname where the Oracle7 Workgroup Server is installed. For example: WWServer.
4. Click **OK**.

Configuring the SQL_Net

1. On the Windows Taskbar, click **Start**. Point to **Programs**, then point to **Oracle**, and then click **SQL_Net Easy Configuration**.
2. By default, the alias created for your server will begin with **wgs_ServerName_orcl**. However, you can change this name.

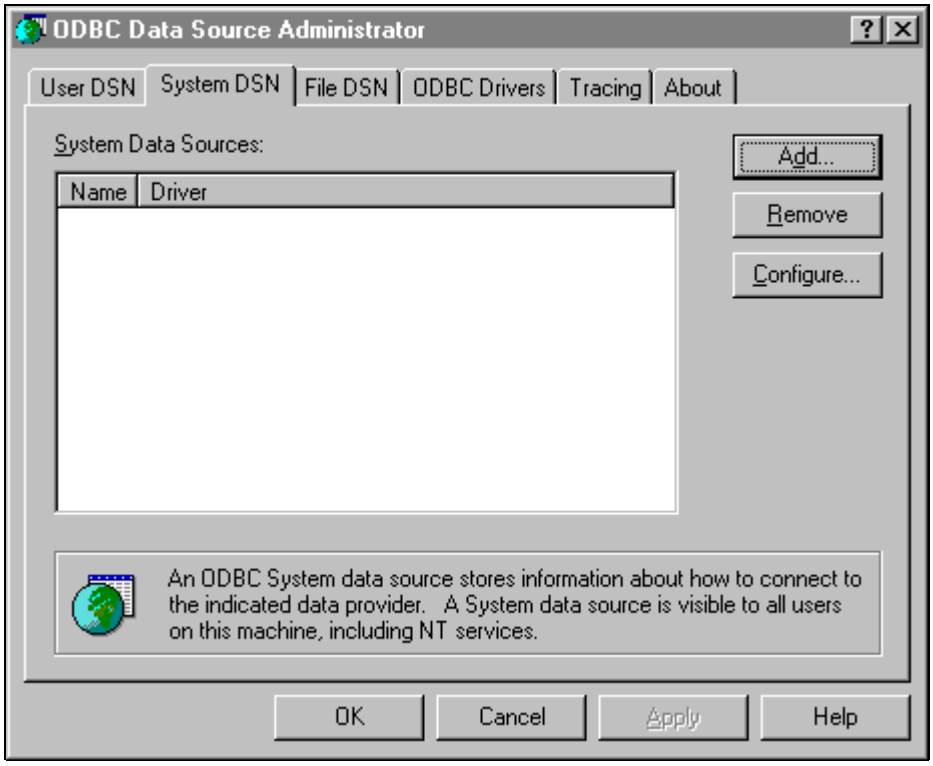
   🖱 The database alias is used in the **SQLConnect()** function in InTouch.
3. To modify the Database Alias select your server name, and then click **OK**.
4. Click **Modify Database Alias Select Network protocol**. The Named Pipe Server is the computer name for the Oracle server. The SQL_Net is now complete.

   🖱 Make sure you write down the Database alias for use later.

Configuring the Data Source Name

➢ **To Install ODBC drivers on the client node:**

1. On the Windows **Taskbar**, click **Start**. Point to **Settings**, then click on **Control Panel**. The Windows Control Panel opens.
2. Double-click **ODBC**. The **ODBC Data Source Administrator** dialog box will appear.
3. Click the **System DSN** tab:



4. Click **Add**. The **Create New Data Source** dialog box will appear.
5. Select the Oracle7 ODBC driver, and then click **Finish**. The **ODBC Oracle Driver Setup** dialog box will appear.
6. In the **Data Source Name** box, type the Server Name for your Oracle Server.
7. Click **Advanced**. Use the default settings on the **ODBC Oracle Advanced Driver Setup** dialog box. Click **Close**. The **ODBC Data Source Administrator** dialog box will reappear.
8. Click **OK**.

SQLConnect() Format

The **SQLConnect()** function is used to connect to Oracle databases. The connection string used by the **SQLConnect()** function is formatted as follows:

```
SQLConnect(ConnectionId,"<attribute>=<value>;<attribute>=<value>;
...");
```

The following describes the attributes used by Oracle. They must be specified in the following order:

Attribute	Value
DSN	The name of the data source as configured in the Microsoft ODBC Administrator.
UID	User name.
PWD	Password.
SRVR	SQL_NET Database Alias.

Example:

```
SQLConnect(ConnectionId,"DSN=Oracle;UID=SCOTT;PWD=TIGER;
SRVR=wsg_wwServer_orcl");
```

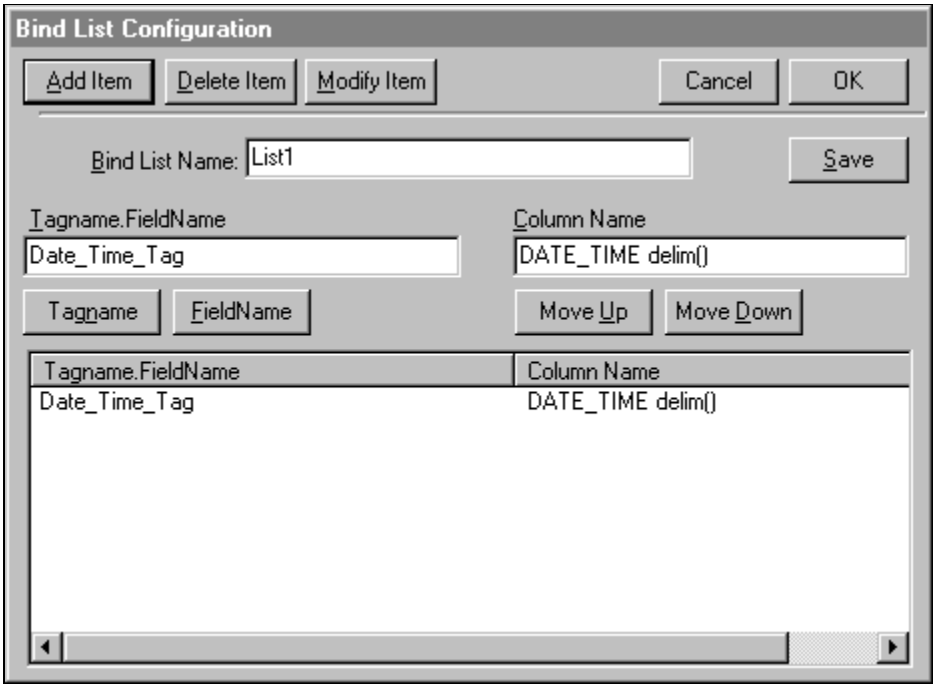
Note If the Database Alias was changed from the default format wsg_ServerName_orcl use the new name that was created using the SQL_NET Easy Configuration, SRVR=WWServer.

Logging Date and Time to an Oracle Date Field

To log the date and time to an Oracle date field, you must Configure the Bind List using the delim function.

➢ **To log both date and time to an Oracle date field:**

1. In the Application Explorer under **SQL Access Manager**, double-click **Bind List**. The **Bind List Configuration** dialog box appears:



2. In the **Tagname.FieldName** box, type the tagname that you want to use.
3. In the **Column Name** box, type the **DATE_TIME delim()** function.

Note In the **Column Name** input box, a space must be used with the **delim()** function. For example, DATE_TIME. The space between DATE and TIME must be entered.

4. In your InTouch application, create a QuickScript to prepare input data from present date and time. For example:

```
DATE_TIME_TAG = "TO_DATE('" + $DateString + "" +
StringMid($TimeString,1,8) + "','mm/dd/yy hh24:mi:ss')";
```

   The Date_Time_Tag will display as the following in runtime:

```
TO_DATE('08/22/97 23:32:18' ,'mm/dd/yy hh24:mi:ss')
```

Using Sybase or Microsoft SQL Server

➢ **To communicate with Sybase or Microsoft SQL Server:**

1. Configure the Windows database client.
2. Connect to Sybase® or Microsoft® SQL Server by executing the **SQLConnect()** function in an InTouch QuickScript.

   For more information on the usage of **SQLConnect()**, see Chapter 4 - Using SQL Functions.

Configuring the Client

➢ **Connecting to the database:**

To connect to the database, the following client DLLs must be installed in the WINDOWS/SYSTEM directory:

- DBNMP3.DLL
- W3DBLIB.DLL

Note If the application is using a Winsock (TCP/IP) connection to the database, DBNMP3.DLL is not needed.

SQLConnect() Format

The **SQLConnect()** function is used to connect to Sybase or Microsoft SQL Server. Executing this function logs you onto the database server and opens a connection to allow other SQL functions to be executed. The connection string used by the **SQLConnect()** function is formatted as follows:

```
SQLConnect(ConnectionId,"<attribute>=<value>;<attribute>=<value>;
...");
```

The following describes the attributes used by Sybase or Microsoft SQL Server. They must be specified in the following order:

Attribute	Value
DSN	The name of the data source as configured in Microsoft ODBC Administrator, or
DRV	For compatibility with SQL Access for InTouch, this value is used if a data source name (DSN) is not present in the connection string. QELIB changes it to the data source name.
UID	Logon ID, case sensitive.
PWD	Password, case sensitive.
SRVR	Name of the server computer with the database tables to be accessed.
DB	The database name to be accessed.

Example:

```
SQLConnect(ConnectionId,"DSN=SQL_Data;UID=OPERATOR;PWD=XYZZ");
```

Data Types Supported

SQL Access Manager supports three data types for Sybase and Microsoft SQL Server databases. The char data type contains fixed length character strings. InTouch Message tagnames require a char data type. A field length must be specified. Sybase and Microsoft SQL Server databases support a char field with a maximum length of 255 characters. However, InTouch Message tagnames are limited to 131 characters. If a message variable contains more characters than the length specified for a database field, the string will be truncated when inserted into the database.

The int data type represents InTouch Integer tagnames. If a field length is not specified, the length is set to the default value of the database. If the length is specified, it will be in the form Width. The Width determines the maximum number of digits for the column.

The float data type represents InTouch Real tagnames. The field length setting is fixed by the database. A field length for this data type is not required.

Note When using the Sybase Database Server, the data types (char, int, float) are case sensitive and must be lowercase.

Sybase or Microsoft SQL Server supports only one active statement at a time per ConnectionId. When executing a **SQLSelect()** or **SQLNext()**, data can be browsed, but cannot be inserted, deleted or updated. To insert, delete or update, perform the **SQLSelect()** operations on a separate ConnectionId from **SQLInsert()**, **SQLUpdate()** and **SQLDelete()**. Column names and Table names are also case sensitive. Table names must be fully qualified with the table owner. For example, dbo.MyTable)

Using dBASE

To communicate with dBASE® you must connect to dBASE by executing the **SQLConnect()** function in an InTouch QuickScript.

SQLConnect() Format

The **SQLConnect()** function is used to connect to dBASE databases. Executing this function logs you onto the database server and opens a connection to allow other SQL functions to be executed. The connection string used by **SQLConnect()** is formatted as follows:

```
SQLConnect(ConnectionId,"<attribute>=<value>;<attribute>=<value>;
...");
```

The following describes the attributes used by dBASE. They are listed in the order that they must be specified:

Attribute	Value
DSN	The name of the data source as configured in the Microsoft ODBC Administrator or,
DRV	For compatibility with SQL Access for InTouch, this value is used if a data source name (DSN) is not present in the connection string. QELIB changes it to the data source name.
CS	The character set that tells the driver whether the data is stored in the IBM PC character set or the ANSI character set. The default is IBMPC.
DB	Specifies the directory in which the dBASE files are stored. If none is specified, the current working directory will be used.
FOC	The maximum number of unused file opens to cache. The default is 0. The dBASE driver supports file open caching for increased performance. When FOC is greater than 0, the most recently used files are left open. If you try to open these files exclusively using another application, an error stating that the file is in use will be displayed.
LCK	Determines the level of record locking for the database file. Valid values are FILE, RECORD (default), or NONE (for example, LCK=FILE).
CSZ	The number of 64K blocks that are used by the driver to cache database records. The higher the number of blocks, the better the performance. The default is 4 blocks. The maximum number of blocks that can be set is determined by the available system memory. When browsing backwards, updates made by other users will not be visible until the **SQLSelect()** statement is re-executed.
USF	Determines when the driver updates DOS directory entries. If this option is set to 1, the driver updates directory entries at each COMMIT. This decreases performance. The default is 0. This means that the driver updates the directory entry when the file is closed. In this case, a machine "crash" prior to closing the file causes newly inserted records to be lost.

Attribute	Value
MS	This option is provided for backward compatibility with previous Q+E software products. Specify MS=0 to have the driver understand SQL dialects found in earlier Q+E software drivers. The default is 1.
LCOMP	Determines whether to use dBASE-compatible locking or Q+E-compatible locking. Specify LCOMP=DBASE to have the driver implement dBASE-compatible locking; specify LCOMP=Q+E for Q+E-compatible locking. The default is dBASE. Q+E-compatible locking may be used if the locking is to be compatible with earlier versions of Q+E.
COMP	Provides backward compatibility with previous Q+E software products. Use COMP=DBASE for backward compatibility; use COMP=ANSI for portability. The default is ANSI.

Example:

```
SQLConnect(ConnectionId,"DSN=DBASE_FILES;CS=ANSI;DB=C:\ODBC\EMP;
LCK=NONE");
```

Data Types Supported

SQL Access Manager supports three data types for dBASE databases. The char data type contains fixed length character strings. InTouch Message tagnames require a char data type. A length must be specified. dBASE databases support a char field with a maximum length of 254 characters. However, InTouch Message tagnames are limited to 131 characters. If a message variable contains more characters than the length specified for a database field, the string will be truncated when inserted into the database.

The numeric and float data types represent either InTouch Integer or Real tagnames. The length must be specified. The format is Width.Decimal. The Width value determines the maximum number of digits for the column. The Decimal value specifies the number of digits to the right of the decimal point. dBASE databases can store analog values with up to 19 digits of precision.

Using Microsoft Access

To communicate with Microsoft Access, you must connect to it by executing the **SQLConnect()** function in an InTouch QuickScript.

SQLConnect() Format

The **SQLConnect()** function is used to connect to Microsoft Access databases. Executing this function logs you on to the database server and opens a connection to allow other SQL functions to be executed. The connection string used by **SQLConnect()** is formatted as follows:

```
SQLConnect(ConnectionId,"<attribute>=<value>;<attribute>=<value>;
...");
```

The following describes the attributes used by Microsoft Access. They are listed in the order that they must be specified:

Attribute	Value
DSN	The name of the data source as configured in the Microsoft ODBC Administrator, or
DRV	For compatibility with SQL Access for InTouch, this value is used if a data source name (DSN) is not present in the connection string. QELIB changes it to the data source name.

Example:

```
SQLConnect(ConnectionId,"DSN=MSACC");
```

Data Types Supported

SQL Access Manager supports five data types for Microsoft Access databases. The valid data type depends on the version of the ODBC driver is being used. The text data type contains fixed length character strings and are used with InTouch Message tagnames. A length must be specified. Microsoft Access databases support text fields with a maximum length of 255 characters. InTouch Message tagnames are limited to 131 characters. If a message variable contains more characters than the length specified for a database field, the string will be truncated when inserted into the database. The Microsoft Access ODBC driver supports up to 17 characters per column name. The maximum number of columns supported when using SQLSetStatement(Select Col1, Col2, ...) is 40.

For programs using the "SIMBA.DLL" driver for Microsoft Access version 1.1 (including Excel 5.0, Microsoft Word 6.0, etc.), the valid numeric types are short or long for Integer tagnames and singlefloat or doublefloat for Real tagnames.

For programs using the "ODBCJT16.DLL" driver for Microsoft Access version 2.0, the valid numeric types are short or long for Integer tagnames and number with a modifier of Double or Single for Real tagnames. The length is automatically calculated based on the data type and must not be specified in the Table Template definition.

Using Paradox

In order to communicate with Paradox®, you must connect to it by executing the **SQLConnect()** function in an InTouch QuickScript.

SQLConnect() Format

The **SQLConnect()** function is used to connect to Paradox databases. Executing this function logs you on to the database server and opens a connection to allow other SQL functions to be executed. The connection string used by **SQLConnect()** is formatted as follows:

SQLConnect(ConnectionId,"<attribute>=<value>;<attribute>=<value>;...");

The following describes the attributes used by Paradox. They are listed in the order that they must be specified:

Attribute	Value
DSN	The name of the data source as configured in the Microsoft ODBC Administrator.
or	
DRV	For compatibility with SQL Access for InTouch, this value is used if a data source name (DSN) is not present in the connection string. QELIB changes it to the data source name.

Example:

```
SQLConnect(ConnectionId,"DSN=PARADOX_FILE");
```

Data Types Supported

InTouch SQL supports three data types for Paradox databases. The alphanumeric data type contains fixed length character strings and is used with InTouch Message tagnames. A field length must be specified. Paradox databases support an alphanumeric field with a maximum length of 255 characters. However, InTouch Message tagnames are limited to 131 characters. If a message variable contains more characters than the length specified for a database field, the string will be truncated when inserted into the database.

The valid numeric types are number for Integer, Real, or Discrete tagnames and short (whole values between -32,767 and 32,767) for Integer or Discrete tagnames only. A field length must not be specified.

Data Type Values for Supported Databases

Oracle

Data Type	Length	Default	Range	Tag Type
char	255 characters	1 character		Message
number	38 digits	38 digits		Integer

Sybase or Microsoft SQL Server

Data Type	Length	Default	Range	Tag Type
char	255 characters			Message
int			-2,147,483,647 to 2,147,483,647	Integer
float	15 digits		$1.7E^{-294}$ to $1.7E^{+308}$	Real

dBASE

Data Type	Length	Default	Range	Tag Type
char	254 characters			Message
numeric or float	19 digits			Integer or Real

Microsoft Access 1.1 (Simba.DLL)

Data Type	Length	Default	Range	Tag Type
text	254 characters			Message
int				Integer

Microsoft Access 2.0 (ODBCJT16.DLL), 7.0 (ODBCJT32.DLL)

Data Type	Length	Default	Range	Tag Type
text	254 characters			Message
number				Integer
number				Real

Paradox

Data Type	Length	Default	Range	Tag Type
alphanumeric number	255 characters			Message Integer, Real, or Discrete
short			-32,767 to 32,767	Integer or Discrete

CHAPTER 3

Configuring SQL Access Manager

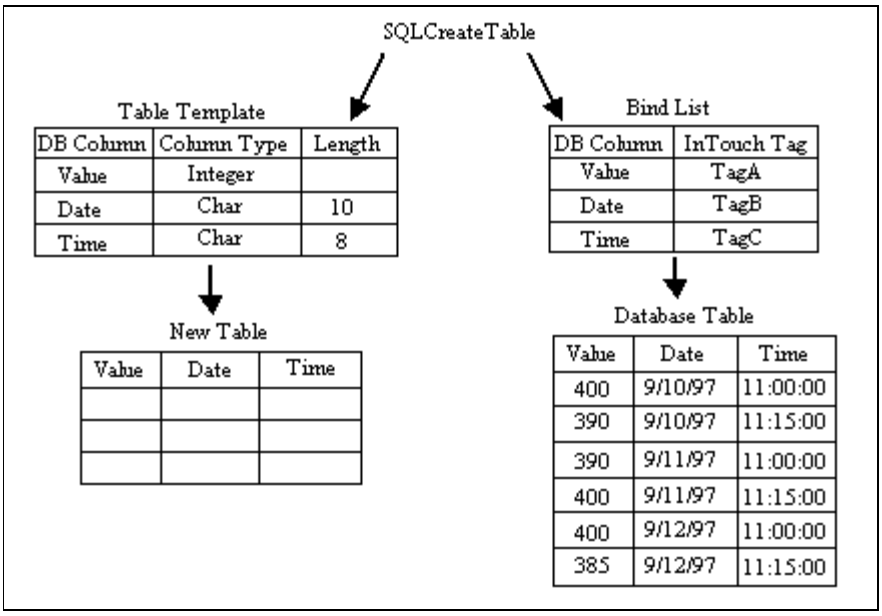
The SQL Access Manager utility program creates Bind Lists and Table Templates. The Bind List associates database columns with tagnames in the InTouch Tagname Data Dictionary. The Table Template defines the structure and format of a new table in the database.

Contents

SQL Access Manager Overview

When an InTouch application executes a **SQLCreateTable()** command, the Table Template argument defines the structure of the new database file.

When a **SQLInsert()**, **SQLSelect()** or **SQLUpdate()** is executed, the Bind List argument defines which InTouch tagnames are used and which database columns to associate.



Configuring a Bind List

The Bind List associates database columns with tagnames in the InTouch Data Dictionary.

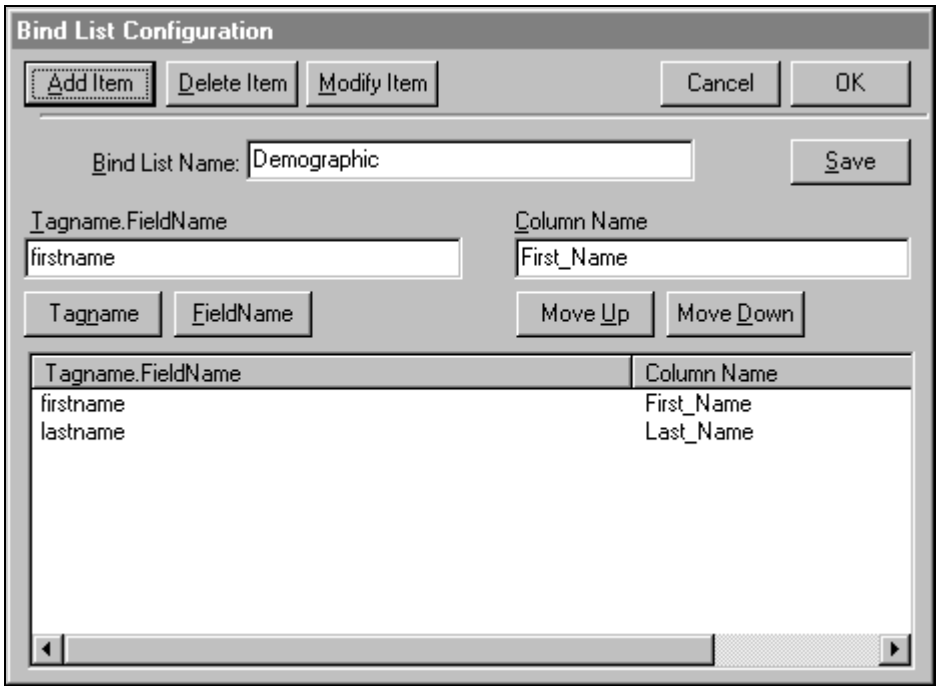
➢ **To create a new Bind List:**

1. On the **Special** menu, point to **SQL Access Manager**, and then click **Bind List**, or in the Application Explorer under **SQL Access Manager**, double-click **Bind List**.

   The first time you choose this command the **Select a Bind List** dialog box appears.

2. Click **New.**

3. The **Bind List Configuration** dialog box appears:



   If you right click the mouse in any of the text entry boxes, a menu will appear displaying the commands that you can apply to the selected text.

4. In the **Bind List Name** box, type the Bind List Name.

   A Bind List Name can be up to 32 characters in length. The new Bind List links database columns to InTouch tagnames. For example, if an employee demographic list is being created, you would enter the Bind List Name that associates information on the employees here.

Note The **SQLInsert()**, **SQLSelect()**, and **SQLUpdate()** functions use the Bind List parameter. Also, when you are using the **SQLExecute()** function the order of the tagnames is important because the **SQLExecute()** function writes to the tagnames in the sequence listed.

5. In the **Tagname.FieldName** box, type an InTouch **tagname.field** name.

   - The Tagname Dictionary associates this **tagname.field** with the **Column Name** in the database. If this tagname is not currently defined in the Tagname Dictionary, double-click it to open the **Tagname Dictionary** dialog box and define it now.

6. Click **Tagname** to select an existing tagname. The Tag Browser will appear.

   - The Tag Browser will display the tagnames for the currently selected tag source. To select a tagname, double-click it or select it, and then click **OK**. To select a **.field** for the tagname click the **Dot Field** arrow, and select the **.field** that you want to use, and then click **OK**.

   - For more information on the Tag Browser, see your *InTouch User's Guide*.

7. Click **FieldName** to append a **.field** to the tagname. The **Choose field name** dialog box will appear.

8. Click the **.field** that you want to use. The dialog box will close and the **.field** will automatically be appended to the tagname in the **Tagname.FieldNam**e field.

   - For more information on tagname **.fields**, see Chapter 4 in your *InTouch User's Guide*.

9. In the **Column Name** box, type the name of the column.

   - A Column Name can be up to 30 characters in length. The column name is directly associated with the database column name. If the Column Name has a space, use square brackets around the Column Name in the Bind List and when used in a script. For example:

     ```
     WHERE EXPR= "[Pipe Flow} = " text (tagname,"#");
     ```

     Special Delimiters can also be used to associate your column name with your database.

     - For more information on special delimiters, see "Using Special Delimiters."

10. Click **Move Up** to move the selected tagname up one level in the list.

11. Click **Move Down** to move the selected tagname down one level in the list.

12. Click **Add Item** to add your new **Tagname.FieldName** and **Column Name** to the Bind List.

13. Click **Delete Item** to delete a selected **Tagname.FieldName** and **Column Name** from the Bind List.

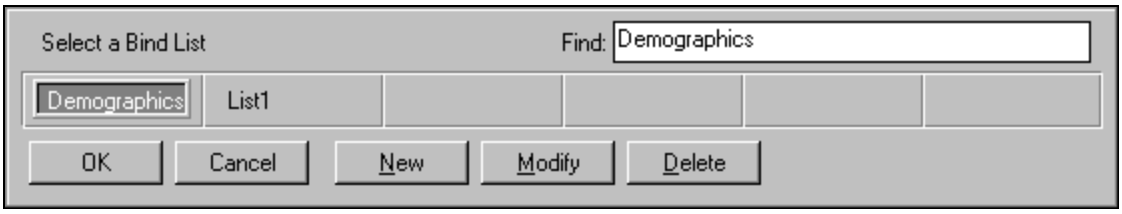
14. Click **Modify Item** to modify a selected **Tagname.FieldName** or **Column Name** for this Bind List.

15. Click **OK** to save your new Bind List configuration and close the dialog box.

    - You can click **Save** to save your settings without closing the dialog box.

➢ **To modify a Bind List:**

1. On the **Special** menu, point to **SQL Access Manager**, and then click **Bind List**, or in the Application Explorer under **SQL Access Manager**, double-click **Bind List**.
2. The **Select a Bind List** dialog box appears:

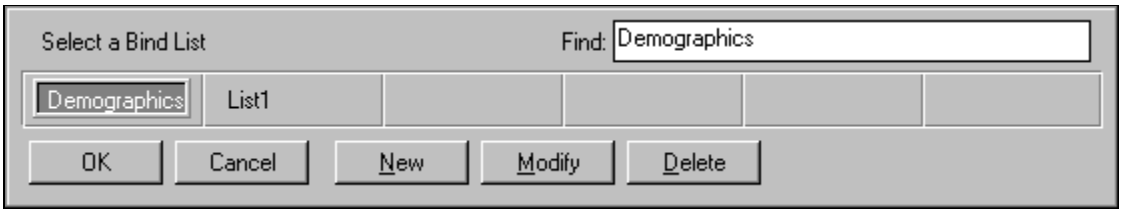


3. Select the Bind List name that you want to change, and then click **Modify**. The **Bind List Configuration** dialog box will appear.
4. Modify the required item(s).
5. Click **OK** to save your changes and close the dialog box.

☞ For more information on configuring a Bind List, see the "To create a new Bind List."

➢ **To delete a Bind List:**

1. On the **Special** menu, point to **SQL Access Manager**, and then click **Bind List**, or in the Application Explorer under **SQL Access Manager**, double-click **Bind List**.
2. The **Select a Bind List** dialog box appears:



3. Select the Bind List name that you want to delete.
4. Click **Delete**. A message box will appear asking you to confirm you deletion. Click **Yes** to delete the selected Bind List, or click **No** to cancel the deletion. The **Bind List Configuration** dialog box reappears.
5. Click **OK** to close the dialog box.

Using Special Delimiters

The **SQLInsert()** and the **SQLUpdate()** functions use a default format that encloses message strings with single quotes. Some SQL databases expect to receive message strings enclosed by another type of delimiter. For example, Oracle expects to receive a date string surrounded by brackets. When this occurs, the **Delim()** function must be used as follows:

In the **Bind List Configuration** dialog box **Column Name** field, after the column name, type the keyword "delim" (not case sensitive). The keyword "delim" must be entered followed by:

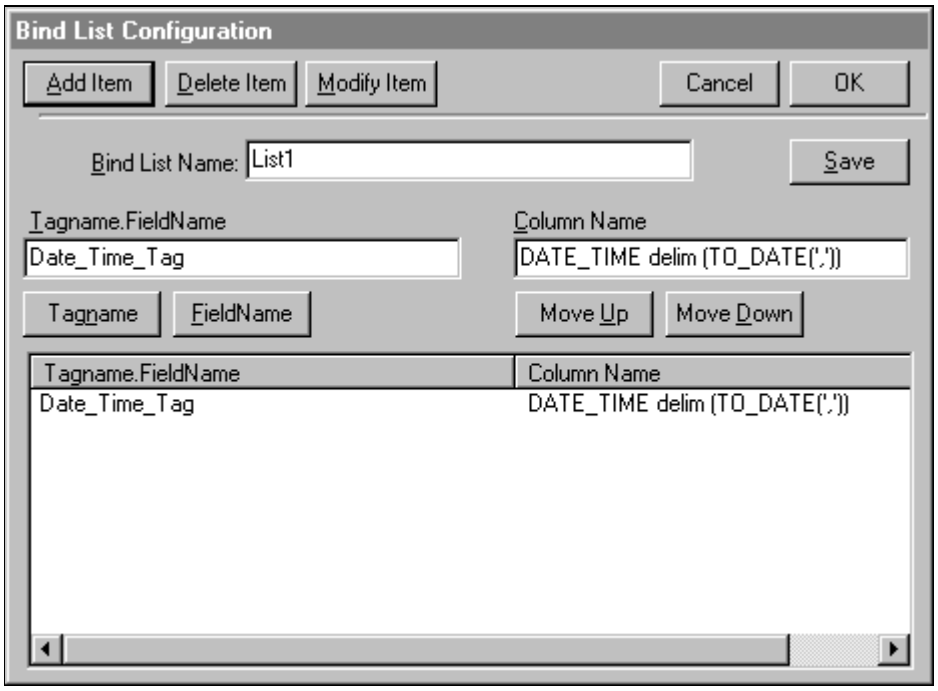
- a left parenthesis
- the left delimiter
- a comma
- the right delimiter
- a right parenthesis

Example: **datestring delim (‘,‘)**

To use the same delimiter for both left and right, just specify the delimiter in parentheses without the comma.

Example: **datestring delim (‘ ‘)**

The following example uses different left and right delimiters. Notice where **date delim (‘,’)** is entered in the **Column Name** field:



For more information on logging date and time to an Oracle date field, see Chapter 2 - Configuring and Connecting Databases

Configuring a Table Template

This command creates a Table Template defining the structure and format of a new table in the database.

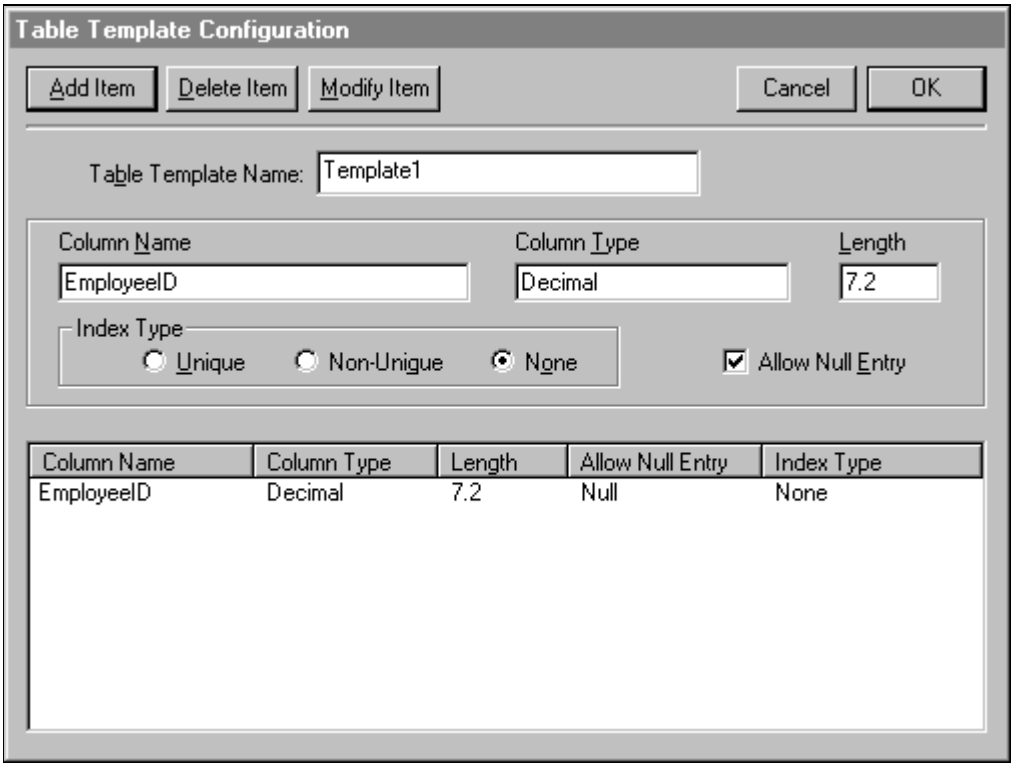
➢ **To create a new Table Template:**

1. On the **Special** menu, point to **SQL Access Manager**, and then click **Table Template**, or in the Application Explorer under **SQL Access Manager**, double-click **Table Template**.

   The first time you choose this command the **Select a Table Template** dialog box appears:

2. Click **New**.

3. The **Table Template Configuration** dialog box appears:



   If you right click the mouse in any of the text entry boxes, a menu will appear displaying the commands that you can apply to the selected text.

4. In the **Table Template Name** box, type the name of the Table Template.

   A Table Template Name can be up to 32 characters in length. The Table Template name is used to identify the structure of a database for the **SQLCreateTable()** function.

5. In the **Column Name** box, type the column name for the Table Template.

   - A Column Name can be up to 30 characters in length.

6. In the **Column Type** box, type the data type for the column.

   - Data type selections vary according to the database being used.

   - For more information on data types for a specific database, see Chapter 2, "Data Types Values for Supported Databases."

7. Select the **Index Type** as follows:

   Unique A column requires that each value in that column be unique.

   Non-Unique A column does not require that each value in that column be unique.

   None No Index.

   - When you execute a **SQLCreateTable()**, an index file is automatically created.

8. Select **Allow Null Entry** to allow null data to be entered in this column.

   - InTouch does not support null data. When inserting data, if a value has not been entered for a tagname, values will be:

     Discrete 0
     Integer 0
     Message Strings with no characters

     When selecting data, null values will be translated according to the data type as shown above.

9. Click **Add Item** to add your new Column Name, Column Type, Length and Index Type to the Table Template.

10. Click **Delete Item** to delete a selected Column Name, Column Type, Length and Index Type from the Table Template list.

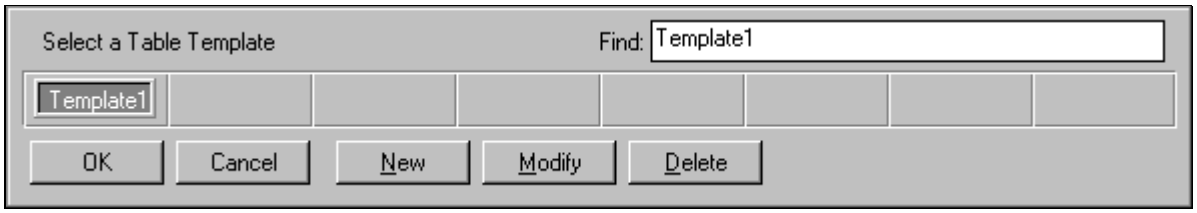
11. Click **Modify Item** to modify a selected Column Name, Column Type, Length and Index Type in the Table Template list.

12. Click **OK** to save your new Table Template configuration and close the dialog box.

    - You can click **Save** to save your settings without closing the dialog box.

➢ **To modify a Table Template:**

1. On the **Special** menu, point to **SQL Access Manager**, and then click **Table Template**, or in the Application Explorer under **SQL Access Manager**, double-click **Table Template**.
2. The **Select a Table Template** dialog box appears:

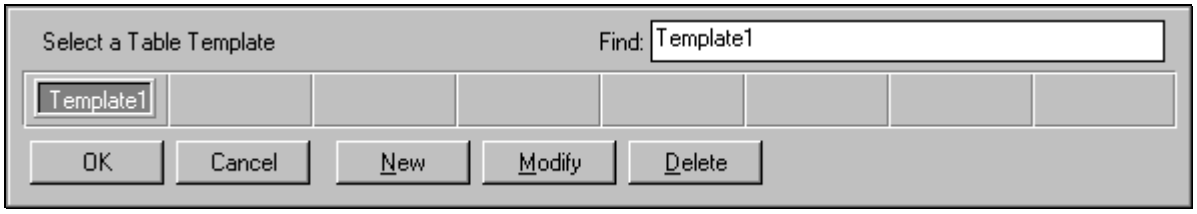


3. Select the Table Template name that you want to modify, and then click **Modify**. The **Table Template Configuration** dialog box appears.
4. Modify the required item(s).
5. Click **OK** to save your changes and close the dialog box.

   ✍ For more information on configuring a Table Template, see "To create a New Table Template."

➢ **To delete a Table Template:**

1. On the **Special** menu, point to **SQL Access Manager**, and then click **Table Template**, or in the Application Explorer under **SQL Access Manager**, double-click **Table Template**.
2. The **Select a Table Template** dialog box appears:



3. Select the Table Template name that you want to delete
4. Click **Delete**. A message box will appear asking you to confirm you deletion. Click **Yes** to delete the selected Bind List, or click **No** to cancel the deletion. The **Table Template Configuration** dialog box reappears.
5. Click **OK** to close the dialog box.

The SQL.DEF File

The SQL Access Manager saves the configuration information for the Bind Lists and Table Templates to a file named "SQL.DEF." This file is formatted as a comma-separated variable (.CSV) type file. The SQL.DEF file can be viewed or modified using SQL Access Manager or any text editor, such as Notepad. The data will appear in the file as follows:

:BindListName,*BindListName*
Tagname1.FieldName,ColumnName1
Tagname2.FieldName,ColumnName2
Tagname3.FieldName,ColumnName3

:TableTemplateName,*TableTemplateName*
ColumnName1,ColumnType,[ColumnLength],Null,Index
ColumnName2,ColumnType,[ColumnLength],Null,Index
ColumnName3,ColumnType,[ColumnLength],Null,Index

CHAPTER 4

Using SQL Functions

InTouch uses SQL Functions to interact with information in the database. These functions are an extension of the standard InTouch QuickScript functions and can be used in any script. They allow you to select, modify, insert or delete records in the tables you choose to access.

Contents

SQL Functions

This section lists each SQL Function. Keep in mind that SQL actions are synchronous. Control is not returned to InTouch until the SQL activity is complete (InTouch trending, polling, etc. are suspended).

All SQL Functions (with the exception of **SQLNumRows()**) return a *ResultCode*. If the *ResultCode* is non-zero, the function failed and other actions should be taken. The *ResultCode* can be used by the **SQLErrorMsg()** function.

The general format for SQL Functions is as follows:

```
SQLFunction(Parameter1, Parameter2,...)
```

📖 For complete details on each SQL function and examples of how you use each function, see your *InTouch Reference Guide*.

Function	Description
SQLAppendStatement	Continues a SQL statement using the contents of *SQLStatement*.
SQLClearParam	Clears the value of the specified parameter.
SQLClearStatement	Releases the resources associated with the statement specified by *SQLHandle*.
SQLClearTable	Deletes all records in a database table, but keeps the table.
SQLCommit	The **SQLCommit()**command defines the end of a group of transaction commands.
SQLConnect	Connects InTouch to the database specified in the *ConnectString*.
SQLCreateTable	Creates a table in the database using the parameters in the named Table Template.
SQLDelete	Deletes a record or multiple records.
SQLDisconnect	Disconnects the user from the database.
SQLDropTable	Destroys a table.
SQLEnd	Use this function after a **SQLSelect()** to free resources that were being used to store the Results Table.
SQLErrorMsg	Retrieves the text error message associated with a specific *ResultCode*. *ErrorMsg* is the InTouch memory message tag (maximum 131 characters) associated with *ResultCode*.
SQLExecute	Executes the SQL statement. If the statement is a select statement, the *BindList* parameter designates the name of the *BindList* to use for binding the database columns with tagnames. If the *BindList* is NULL, no tagnames relationships are formed.
SQLFirst	Selects the first record of the Results Table created by the last **SQLSelect()**.
SQLGetRecord	Retrieves the record specified by *RecordNumber* from the current selection buffer.

Function	Description
SQLInsert	Inserts a new record into the referenced table using the values of the tagnames in the supplied *BindList*. The *BindList* parameter defines which InTouch tagnames are used and which database columns they are associated.
SQLInsertEnd	Releases the statement.
SQLInsertExecute	Execute the already prepared statement.
SQLInsertPrepare	Creates and prepares an Insert statement for execution.
SQLLast	Selects the last record of the Results Table created by the last **SQLSelect()**.
SQLLoadStatement	Reads the statement contained in *FileName.* At this point the statement is similar to one created by **SQLSetStatement()**, and can be appended to via **SQLAppendStatement()**, or executed by **SQLExecute()**. There can be only one statement per file. However, **SQLAppendStatement()** can be used to append something to the statement if **SQLPrepareStatement()** or **SQLExecute()** has not been called.
SQLManageDSN	Runs the Microsoft ODBC Manager setup program. This can be used to add, delete and modify all data source names.
SQLNext	Selects the next record of the Results Table created by the last **SQLSelect()** function.
SQLNumRows	Indicates how many rows met the criteria specified in the last **SQLSelect()** function.
SQLPrepareStatement	A **SQLPrepareStatement()** prepares an existing SQL statement for use by the **SQLSetParam()** function. A statement can be created by using either a **SQLSetStatement()**, or **SQLLoadStatement()**.
SQLPrev	Selects the previous record of the Results Table created by the last **SQLSelect()** function.
SQLRollback	The **SQLRollback()** command reverses, or "rolls back," the most recently issued transaction set. A transaction set is a group of commands issued between the **SQLTransact()** command and the **SQLCommit()** command or the **SQLRollback()** command. A transaction set is handled like a single transaction. After the **SQLTransact()** command is issued, all subsequent operations are not committed to the database until the **SQLCommit()** command is issued.

Function	Description
SQLSelect	Instructs the database to retrieve information from a table. When a **SQLSelect()** function is executed, a temporary Results Table is created in memory, containing records that can be browsed using **SQLFirst()**, **SQLLast()**, **SQLNext()** and **SQLPrev()**.
SQLSetParamChar	Sets the value of the specified parameter to the specified string. **SQLSetParamChar()** may be call multiple times before executing, resulting in the parameter value being set to the concatenation of all values sent.
SQLSetParamDate	Sets the value of the specified date parameter to the specified string.
SQLSetParamDateTime	Sets the value of the specified date-time parameter to the specified string.
SQLSetParamDecimal	Sets the value of the specified decimal parameter to the specified string.
SQLSetParamFloat	Sets the value of the specified parameter to the specified *ParameterValue*.
SQLSetParamInt	Sets the value of the specified parameter to the specified *ParameterValue*.
SQLSetParamLong	Sets the value of the specified parameter to the specified *ParameterValue*.
SQLSetParamNull	Sets the value of the specified parameter to the NULL.
SQLSetParamTime	Sets the value of the specified time parameter to the specified string.
SQLSetStatement	Starts a SQL statement buffer using the contents of *SQLStatement*, on the established connection, *ConnectionID*. There can be one SQL Statement buffer per *ConnectionID*.
SQLTransact	The **SQLTransact()** command defines the beginning of a group of transaction commands. The group of commands performed between the **SQLTransact()** command and the **SQLCommit()** command is called a transaction set. A transaction set is handled like a single transaction. After the **SQLTransact()** command is issued, all subsequent operations will not be committed to the database until the **SQLCommit()** command is issued.
SQLUpdate	Modifies a record to update the record with the current tagname values.
SQLUpdateCurrent	Takes the currently selected record and updates it with any new InTouch values.

SQL Parameters

The following describes the parameters required for each SQL function. When a parameter is entered in a script surrounded by quotation marks, e.g., "Parameter1", that exact string will be used. If no quotation marks are used, Parameter1 is assumed to be a tagname and the system will access the InTouch tagname dictionary for the value of the tagname, Parameter1.

Example:

```
"c:\main\file" vs.  Location
```

where: location is an InTouch message tagname
"c:\main\file" is a literal string

The parameters for most of the SQL functions will be one or more of the following:

Parameter	Description
BindList	Corresponds to one of the Bind List names in the SQL.DEF file.
ConnectionID	Is a memory integer tagname created by the user to hold the number (ID) assigned by the SQLConnect function to each database connection.
ConnectString	String that identifies the database and any additional logon information used in **SQLConnect()**.
ErrorMsg	Message variable containing a text description of the error message. For more information on error message descriptions, see Chapter 5 - Troubleshooting.
FileName	The name of the file name in which the information is contained.
MaxLen	Maximum size of the column with which this parameter is associated. This setting determines whether the parameter is of varying character or long varying character type. If *MaxLen* is less than or equal to the largest character string allowed by the database, then the parameter is varying character type. If greater, long varying character type.
OrderByExpression	Defines the columns and direction for sorting. Only column names can be used to sort. The expression must be formatted: ColumnName [ASC|DESC] To sort the selected table by a column name (e.g., manager) in ascending order: **"manager ASC"** To sort by multi-columns, the expression is formatted: **ColumnName [ASC|DESC],** **ColumnName [ASC|DESC]** To sort a selected table by one column name (e.g., temperature) in ascending order and another column name (e.g., time) in descending order: **"temperature ASC, time DESC"**

ParameterNumber Actual parameter number in the statement.

ParameterType Data type of the specified parameter. Valid values:

Type	Value	Description
Char	1	Blank Padded fixed length string
Var Char	2	Variable Length String
Decimal	3	BCD Number
Integer	4	4-byte Signed integer
Small integer	5	2-byte signed integer
Float	6	4-byte floating point
Double Precision Float	7	8-byte floating point
DateTime	8	26-byte date time value
Date	111	26-byte date time value
Time	112	26-byte date time value
No Type	0	No Data Type

ParameterValue Actual value to set.

Precision Is the decimal value's precision, the max. size of the character, or the length in bytes of the date-time value.

RecordNumber Actual record number to retrieve.

ResultCode Integer variable returned from most SQL functions. *ResultCode* is returned as zero (0) if the function is successful and a negative integer if it fails.

For more information, see Chapter 5 - Troubleshooting.

Scale Is the decimal value's scale. This value is required only if applicable to the parameter being set to null.

SQLHandle When using the advanced functionality statements, SQL returns an *SQLHandle*, which it uses internally.

SQLStatement Actual statement, for example:

```
ResultCode=SQLSetStatement(ConnectionID,"Select
LotNo, LotName from LotInfo");
```

TableName The database table name you want to access.

TemplateName The name of the template definition you want to use.

WhereExpression Defines a condition that can be either true or false for any row of the table. The command extracts only those rows from the table for which the condition is true. The expression must be in the following format:

ColumnName *comparison_operator* expression

Note If the column is a character data type, the expression must be in single quotes.

The following example will select all rows whose name column contains the value **EmployeeID**:

name='EmployeeID'

The following example will select all rows containing part numbers from 100 to 199:

partno>=100 and partno<200

The following example will select all rows whose temperature column contains a value that is greater than 350:

temperature>350

Using SQL Functions in InTouch QuickScripts

SQL functions can be automatically inserted into InTouch QuickScripts by clicking on the **Add-ons** button within the QuickScript editor dialog. The SQL function will be automatically inserted into the script at the current cursor position.

📖 For complete details on InTouch QuickScripts see your *InTouch User's Guide*, Chapter 6 "Creating QuickScripts in InTouch."

Specifying Complex Queries

SQL Access Manager allows you to specify complex queries and SQL statements of your own design. These queries may either be built dynamically or be contained in external files. Additionally, these queries may contain parameters that need to be "passed" into the query at runtime. These queries must then be executed and possibly have result sets returned. The SQL Access Manager API allows you to execute whatever SQL statement your database can handle, and retrieve the result of that query. As a by-product, stored procedures are also available for execution by you. (Stored procedures are not fully supported.)

For more information on stored procedures, see "Supporting Stored Procedures."

The SQLSetStatement function must be used for complex queries and string expressions greater than 131 characters. When the string expression exceeds 131 characters use the SQLAppend. For example:

```
SQLSetStatement( Connect_Id, “Select  Speed, Ser_No from
tablename where Ser_No =’” +   Serial_input  + “’”);
```

In the following example the *SQLHandle* is set to zero so the statement does not have to call **SQLPepare**(*Connect_Id*,*SQLhandle*) before the execute statement. Because the *SQLHandle* was not created by the **SQLPepare()** to properly end this select use the **SQLEnd()** function instead of the **SQLClearStatement()**.

```
SQLExceute(Connect_Id,0);
```

```
SQLSetStatement( Connect_Id, “Select  Speed, Ser_No from
tablename where Ser_No =’” +   Serial_input  + “’”);
```

```
SQLPrepareStatement(Connect_Id,SQlHandle);
```

In the following example the *SQLHandle* is created by the **SQLPrepareStatement()** and used in the **SQLExceute()** function. To end this select statement use **SQLClearStatment()** to free up resources and free the *SQLHandle*.

```
SQLExceute(Connect_Id,SqlHandle);
```

Building Queries Dynamically

To build queries dynamically, two additional functions are required: **SQLSetStatement()** and **SQLAppendStatement()**. **SQLSetStatement()** starts a new SQL statement. This can be any valid SQL statement, including the name of a stored procedure. Since InTouch only supports character strings of 131 characters, **SQLAppendStatement()** is provided to concatenate additional strings onto the statement.

Note **Bold** text refers to SQL Query language commands.

Example

```
ResultCode  =  SQLSetStatement ( ConnectionID, "Select LotNo,
LotName, LotDescription, LotQuantity from LotInfo,
ProductionInfo"  );

ResultCode  =  SQLAppendStatement (ConnectionID, " where
LotInfo.LotNo = ProductionInfo.LotNo" );

ResultCode  =  SQLAppendStatement (ConnectionID, " order by
LotNo,NotName,LotQuantity" );
```

The statement is now ready for execution.

Note Many database column and table names are case sensitive. For the above script to function properly, the column and database names must be typed exactly as used in the database tables.

Reading SQL Statements from a File

You can model your query in other packages such as, Microsoft Access and other 3rd party database tools, then use SQL Access for InTouch to perform your query. As several of these packages will generate the SQL statement, it's a simple matter to take that SQL statement and store it into a file by using the **SQLLoadStatement()** function will be added to the SQL Access API.

Example

```
ResultCode  =  SQLLoadStatement ( ConnectionID,
"c:\myappdir\lotquery.sql"  );
```

The statement is now ready for execution.

Modifying Extended SQL Statements

To provide full SQL functionality, SQL Access Manager allows you to specify a where clause that contains a value of an InTouch tagname. To allow runtime specification of SQL parameters, the following functions are provided:

- **SQLPrepareStatement()**
- **SQLSetParam*Type*()**
- **SQLClearStatement()**
- **SQLClearParam()**

To perform parameter substitution on a SQL statement, put a "?" in the SQL statement where you want to specify a parameter at a later date. The statement is "prepared," parameters are "set" into the statement, and then the statement is executed.

SQLPrepareStatement() prepares the statement for execution. It does not execute the statement, it just makes the statement active so you can set parameter values. **SQLSetParamType()** is a set of functions that allow you to set values into parameters in the SQL statement.

Example

```
ResultCode  =  SQLSetStatement ( ConnectionID, "Select LotNo,
LotName, LotDescription, LotQuantity from LotInfo,
ProductionInfo"  );

ResultCode  =  SQLAppendStatement (ConnectionID, " where
LotInfo.LotNo = ?");

ResultCode  =  SQLAppendStatement (ConnectionID, " order by
LotNo,NotName,LotQuantity" );

ResultCode  =  SQLPrepareStatement (ConnectionID, SQLHandle );

{return the statement handle into tag 'SQLHandle'}

ResultCode  =  SQLSetParamInt ( SQLHandle, 1, tagLotNumber );

{put the value of tagLotNumber into param}
```

Since the statement only has one parameter, it is now ready for execution.

Once the statement is executed and you are finished with the prepared statement, **SQLClearStatement()** can be called to free the resources associated with that statement.

Note **SQLEnd()** is called to free "unnamed" SQL statements (those generated by existing SQL Access functions), and those statements created by **SQLSetStatement()** and **SQLLoadStatement()** and not prepared.

Executing Extended SQL Statements

Now that the statement has been either built dynamically or read from a file, and has been optionally prepared and modified, it's time to execute it. The SQL Access Manager API uses the **SQLExecute()** function to accomplish this. **SQLExecute()** will either execute the currently active statement (i.e., the one created by **SQLSetStatement()** or **SQLLoadStatement()**) or the statement that has been previously prepared and is specified by the statement handle parameter.

Example #1

```
ResultCode  =  SQLLoadStatement ( ConnectionID,
"c:\myappdir\lotquery.sql"  );

ResultCode  =  SQLExecute (ConnectionID, "BindList", 0);

{put the results of the select into the tags specified in
BindList.  prepared statement handle is zero}

ResultCode =  SQLNext ( ConnectionID );

{Get results of Select}
```

Example #2

```
ResultCode  =  SQLSetStatement ( ConnectionID, "Select LotNo,
LotName, LotDescription, LotQuantity from LotInfo,
ProductionInfo"  );

ResultCode  =  SQLAppendStatement (ConnectionID, " where
LotInfo.LotNo = ?");

{question mark means I'll  get back to you}

ResultCode  =  SQLAppendStatement (ConnectionID, " order by
LotNo,NotName,LotQuantity" );

ResultCode  =  SQLPrepareStatement (ConnectionID, SQLHandle );

{return the statement handle into tag 'SQLHandle'}

ResultCode  =  SQLSetParamInt ( SQLHandle, 1, tagLotNumber );

{put the value of tagLotNumber into param}

ResultCode  =  SQLExecute (ConnectionID, "BindList", SQLHandle);
{put the results of the Select into the tags specified in

BindList prepared statement handle is in SQLHandle}

ResultCode =   SQLNext ( ConnectionID );

{Get results of Select }
```

Example #3

SQLSetStatement – This statement must be used for complex queries and string expressions greater than 131 characters. When the string expression exceeds 131 characters use the SQLAppend

```
SQLSetStatement( Connect_Id, "Select  Speed, Ser_No from
tablename where Ser_No ='" +   Serial_input  + "'");

SQLExceute(Connect_Id,0);
```

In the above example the SQLhandle is set to zero so the statement does not have to call SQLPepare(Connect_Id,SQLhandle) before the execute statement. Because the SQLhandle was not created by the SQLPepare to properly end this select use the SQLEnd function instead of the SQLClearStatement().

```
SQLSetStatement( Connect_Id, "Select  Speed, Ser_No from
tablename where Ser_No ='" +   Serial_input  + "'");
SQLPrepareStatement(Connect_Id,SQlHandle);

SQLExceute(Connect_Id,SqlHandle);
```

In the above example the SQLhandle is created by the SqlPrepareStatement and used in the SQLExceute function. To end this select statement use SQLClearStatment to free up resources and free the SqlHandle.

Supporting Stored Procedures

The **SQLExecute()** function supports the execution of some stored procedures. For example, suppose you create a stored procedure on the database server named "LotInfoProc," that contains the following select statement: "Select LotNo, LotName from LotInfo." You would write the following InTouch QuickScript to execute the procedure and get the results:

Using Microsoft SQL Server

```
ResultCode  =  SQLSetStatement ( ConnectionID, "LotInfoProc"  );

ResultCode  =  SQLExecute(ConnectionID, "BindList", 0);

ResultCode =   SQLNext ( ConnectionID );

{ Get results of Select}
```

Using Oracle or Microsoft Access

```
ResultCode  =  SQLSetStatement ( ConnectionID, "{CALL
LotInfoProc}"  );

ResultCode  =  SQLExecute(ConnectionID, "BindList", 0);

ResultCode =   SQLNext ( ConnectionID );

{ Get results of Select}
```

CHAPTER 5

Troubleshooting

This chapter explains how to troubleshoot SQL applications using the Result Codes returned by SQL functions. The first section describes the **SQLErrorMsg()** function and includes a table of SQL Result Codes with their corresponding Error Messages. The second section includes tables with specific database Error Messages.

Contents

Troubleshooting Functions

All SQL Functions return a *ResultCode* that can be used for troubleshooting. The **SQLErrorMsg()** function returns the Error Message associated with the *ResultCode*.

Example:

```
ErrorMsg=SQLErrorMsg(ResultCode);
```

where: **ErrorMsg** is a memory message tag.
ResultCode is an integer value obtained from a previous SQL function.

Result Code Error Messages

For Result Codes that are not documented here, please refer to your specific database documentation and be sure to check the Wonderware Logger for any additional information.

The **SQLErrorMsg()** function will set the value of the InTouch message tagname *ErrorMsg*. The following is a listing of some of the possible SQL Result Codes and their corresponding error messages and descriptions:

Result Code	Error Message	Description
0	**No errors occurred**	The command was successful
-5	**No more rows to fetch**	The last record in the table has been reached
-1001	**Out of memory**	There is insufficient memory to perform this function
-1002	**Invalid connection**	The ConnectionId passed to the function is not valid
-1003	**No bind list found**	The specified Bind List name does not exist
-1004	**No template found**	The specified Table Template name does not exist.
-1005	**Internal Error**	An internal error occurred. Call Technical Support.
-1006	**String is null**	Warning - the string read from the database is null.
-1007	**String is truncated**	Warning - the string read from the database is longer than 131 characters and is truncated on a select.
-1008	**No Where clause**	There is no Where clause on Delete.

Result Code	Error Message	Description
-1009	**Connection failed**	Check Wonderware Logger for a more detailed description of the failed connect.
-1010	**The database specified on the DB= portion of the connect string does not exist**	The specified database does not exist.
-1011	**No rows were selected**	A **SQLNumRows()**, **SQLFirst()**, **SQLNext()**, or **SQLPrev()** command was attempted without executing a **SQLSelect()** command first.
-4149	**The connection, statement, or query handle you provided is not valid**	Column type may be incorrectly defined. For example, if a Table Template is defined with a column type of character instead of char for a dBASE file, an error will be returned.

Specific Database Error Messages

Oracle

Error Message	Solution
ORA-03112 - Host String Syntax error	If you are not running **NETINIT.EXE**, place it in the Windows Startup group. If you are running **NETINIT.EXE** and want to establish more than one connection or session, you will need to allocate memory. Do this by setting the **WIN_REMOTE_SESSIONS** parameter in the **CONFIG.ORA** file equal to the number of required connections. The following example will allocate memory for 4 connections: **WIN_REMOTE_SESSIONS=4**
ORA-3121 - No interface driver connected	Start the SQL*NET TSR appropriate for your networking system **before** entering Windows and using InTouch SQL Access.
ORA-6435 - NetBIOS: Unable to add local name to name table	The network software (Novell, LAN Manager, etc.) must be running.
ORA-09301 - Local kernel only supported in standard mode	Specify the server name (**"SRVR="**) in the connection string.
ORA-06430 - Unable to make connection	Verify that the attributes in the connection string are accurate and in correct order.

Sybase or Microsoft SQL Server

Error Message	Solution
You cannot have more than one statement active at a time	You are trying to execute a SQL command after executing a **SQLSelect()**. Execute a **SQLEnd()** to free system resources from the **SQLSelect()** or; Use a separate ConnectionId for the second statement.
There is not enough memory available to process the command	Try rebooting the client workstation.
Invalid object name table name	The table name does not exist in the database you are using. Try DB=database name.

dBASE

Error Message	Solution
File or DLL not found	For Windows, the QEDBF.DLL must either be in your current directory or in the \windows\system directory in your DOS path.
Invalid connection	Make sure you have appropriate DLLs in your path. For DBF support, you will need QLDBF.DLL.
The connection or statement handle you provided is not valid	For more information, check the Wonderware Logger. There may be a syntax error in your SQL statement.

APPENDIX A

Reserved Keywords for SQL Access and ODBC

This appendix lists the keywords that are excluded from use for the SQL Access Bind List and the Table Template, and the Open Database Connectivity (ODBC) interface.

If a reserved keyword is used as the Column Name in a Bind List or Table Template, an error message is generated in the Wonderware Logger. The type of error generated depends upon the ODBC driver being used and the location in which the keyword is found. For example, one of the most common errors made is using DATE and TIME for Column Names in a Bind List or Table Template. To avoid this error, use a slightly different name, e.g., "aDATE" and "aTIME."

The reserved keywords define the Structured Query Language (SQL) used by InTouch SQL Access. The keywords are also recognized by the specific ODBC driver being used. SQL Access passes the SQL command containing one or more reserved keywords to the ODBC.DLL file. If the SQL command cannot be interpreted correctly, SQL Access generates an error message in the Wonderware Logger.

The reserved keywords are listed alphabetically below:

ABSOLUTE
ADA
ADD
ALL
ALLOCATE
ALTER
AND
ANY
ARE
AS
ASC
ASSERTION
AT
AUTHORIZATION
AVG
BEGIN
BETWEEN
BIT
BIT_LENGTH
BY
CASCADE
CASCADED
CASE
CAST
CATALOG
CHAR
CHAR_LENGTH
CHARACTER
CHARACTER_LENGTH
CHECK
CLOSE COALESCE
COBOL
COLLATE
COLLATION
COLUMN
COMMIT
CONNECT
CONNECTION
CONSTRAINT
CONSTRAINTS
CONTINUE
CONVERT
CORRESPONDING
COUNT
CREATE
CURRENT
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
CURSOR
DATE
DAY
DEALLOCATE
DEC
DECIMAL
DECLARE
DEFERRABLE
DEFERRED
DELETE
DESC
DESCRIBE
DESCRIPTOR
DIAGNOSTICS
DICTIONARY
DISCONNECT
DISPLACEMENT
DISTINCT
DOMAIN
DOUBLE
DROP
ELSE
END
ESCAPE
EXCEPT
EXCEPTION
EXEC
EXECUTE
EXISTS
EXTERNAL
EXTRACT
FALSE
FETCH
FIRST
FLOAT
FOR FOREIGN
FORTRAN
FOUND
FROM FULL
GET
GLOBAL
GO
GOTO
GRANT
GROUP
HAVING
HOUR
IDENTITY
IGNORE
IMMEDIATE
IN
INCLUDE
INDEX
INDICATOR
INITIALLY
INNER
INPUT
INSENSITIVE
INSERT
INTEGER
INTERSECT
INTERVAL
INTO
IS
ISOLATION

JOIN
KEY
LANGUAGE
LAST
LEFT
LEVEL
LIKE
LOCAL
LOWER
MATCH
MAX
MIN
MINUTE
MODULE
MONTH
MUMPS
NAMES
NATIONAL
NCHAR
NEXT
NONE
NOT
NULL
NULLIF
NUMERIC
OCTET_LENGTH
OF
OFF
ON
ONLY
OPEN
OPTION
OR
ORDER
OUTER
OUTPUT
OVERLAPS
PARTIAL
PASCAL
PLI
POSITION
PRECISION
PREPARE
PRESERVE
PRIMARY
PRIOR
PRIVILEGES
PROCEDURE
PUBLIC
RESTRICT
REVOKE
RIGHT
ROLLBACK
ROWS
SCHEMA
SCROLL
SECOND
SECTION
SELECT
SEQUENCE
SET
SIZE
SMALLINT
SOME
SQL
SQLCA
SQLCODE
SQLERROR
SQLSTATE
SQLWARNING
SUBSTRING
SUM
SYSTEM
TABLE
TEMPORARY
THEN
TIME
TIMESTAMP
TIMEZONE_HOUR
TIMEZONE_MINU
TO
TRANSACTION
TRANSLATE
TRANSLATION
TRUE
UNION
UNIQUE
UNKNOWN
UPDATE
UPPER
USAGE
USER
USING
VALUE
VALUES
VARCHAR
VARING
VIEW
WHEN
WHENEVER
WHERE
WITH
WORK
YEAR

Index

A

B

C

D

E

F

T

U

W