

## PowerStar 5 SQL

The recent addition of SQL support has expanded PowerStar 5's ability to interact with third party software. SQL (Structured Query Language) provides a means of communicating with and interrogation of compatible databases.

Many database products support SQL, this means if you learn how to apply this Structured Query Language you can communicate with an array of databases such as MS Access, an SQL Server, Oracle, Ingres, Excel etc. By including support for SQL, PowerStar 5 is now able to perform many extremely powerful tasks by communicating directly with customers existing databases.

According to ANSI, SQL is now the standard language for relational database management systems. A relational database consists of one or more tables in which data is stored. Tables are uniquely identified by their names and consist of a list of records. Each record in a table has the same structure, each has a fixed number of "fields" of a given type.

SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. In summary SQL is a non-procedural database language that facilitates:

- Database definition and object creation
- Complex queries
- Updating
- The handling of security and privacy constraints

**ATE Systems**

### Typical Applications

The integrated SQL interface may be used for applications such as:

- Automatic test program generation
- Characterisation
- Process control
- Real-time trend analysis
- Custom results logging

## SQL a brief Lesson

Queries, updates and inserting new data are various kinds of tasks performed on databases, all of which can now be performed from within PowerStar 5 via a suitable Virtual Application.

**Query:** A query is used to retrieve specific data from a table or combination of tables that meet a certain set of conditions.

**Insert new data:** If new data is available the user can insert a new row containing the additional information for future reference.

**Update data:** If the data or value of a field/record changes, the old information must be overwritten by the new.

This application note will concentrate on the query and updating aspect of SQL.

The following example shows a typical 'product' table from a customer database.

Model	Output Voltage	Output Current	Input Voltage	Input Type
105	5	25	220	AC
106	12	10	110	AC
107	12	3	48	DC
108	3.3	12	220	AC
109	3.3	12	110	AC
110	48	55	75	DC
111	24	28	48	DC
112	5	2	12	DC
113	-12	5	220	AC
114	-5	12	220	AC

**Example database table 'Product'**

To make a query on the 'product' table listing all information contained would use the SELECT statement.

```
SELECT * FROM Products
```

105	5	25	220	AC
106	12	10	110	AC
107	12	3	48	DC
108	3.3	12	220	AC
109	3.3	12	110	AC
110	48	55	75	DC
111	24	28	48	DC
112	5	2	12	DC
113	-12	5	220	AC
114	-5	12	220	AC

Display all information for all AC models

```
SELECT * FROM Products WHERE [Input Type] = 'AC'
```

105	5	25	220	AC
106	12	10	110	AC
108	3.3	12	220	AC
109	3.3	12	110	AC
113	-12	5	220	AC
114	-5	12	220	AC

Display all information for all AC models where the input is 220

```
SELECT * FROM Products WHERE [Input Type] = 'AC' AND [Input Voltage] = 220
```

105	5	25	220	AC
108	3.3	12	220	AC
113	-12	5	220	AC
114	-5	12	220	AC

Display only the model name for all AC models where the input is 220

```
SELECT Model FROM Products WHERE [Input Type] = 'AC' AND [Input Voltage] = 220
```

105
108
113
114

Display the Model and Power for all DC Models

```
SELECT Model, [Output Voltage] * [Output Current] FROM product WHERE [Input Type] = 'DC'
```

107	36
110	2640
111	672
112	10

To append data to a database we use the INSERT statement

To add a new Model 115, 110V AC with a 15V 12A output

```
INSERT INTO Product (Model, [Output Voltage], [Output Current], [Input Voltage], [Input Type]) VALUES ('115', 3.3, 12,110,'AC')
```

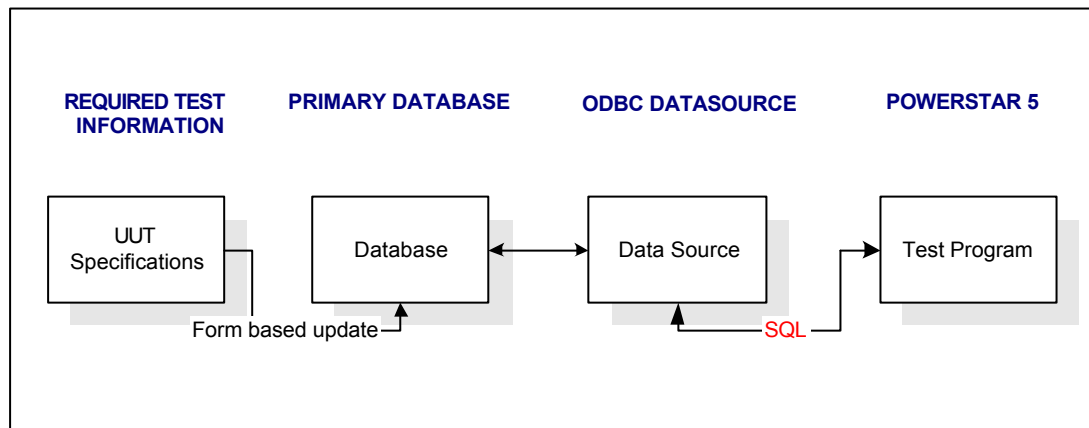


VA Sample illustrating the insertion of a new model as described above

## Case study - Automatic Test Program Generation

Automatic test program generation is a means to enable future products to be tested without the requirement of writing new test programs or the requirement of restructuring any existing product specification databases. The actual type of database used for the product specification is irrelevant as long as an SQL driver exists for it. The database may reside locally or be accessed via LAN or WAN.

Most companies already have product specification databases with all the required test information at hand. Generally the product specification database will contain all the information required to test the product, it's simply a matter of querying the database and using the relevant information in the test program.



### Automatic test program generation using PowerStar 5 SQL support

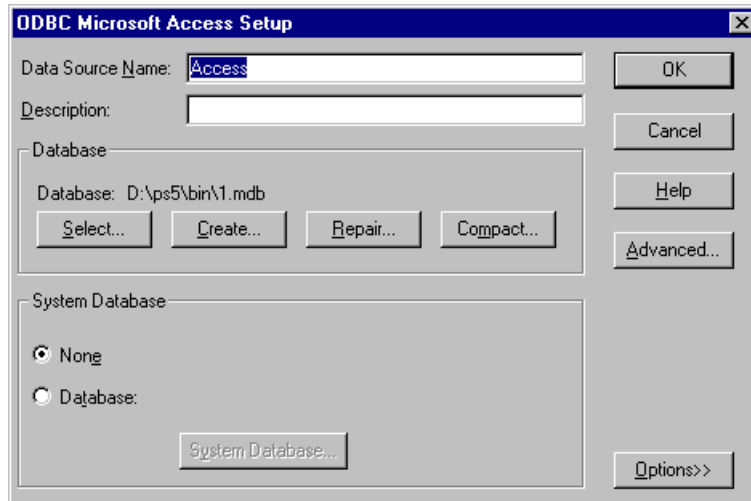
In the scenario shown above the relevant UUT specifications for a new product are integrated into the product specification database (primary database) using a form based update procedure. An ODBC data source is setup so that PowerStar 5 can read from and write to the primary database. The PowerStar 5 test program includes VA's, which call up SQL queries to the ODBC data source. At runtime view, the test specifications are read into the PowerStar 5 test program where they become the test parameters.

When new or additional UUT's are required to be tested their specifications are added to the primary database and the model name is entered in the runtime screen of PowerStar 5. Writing a new test program for a recently added UUT becomes as simple as commanding PowerStar 5 to read the bar code on the UUT.

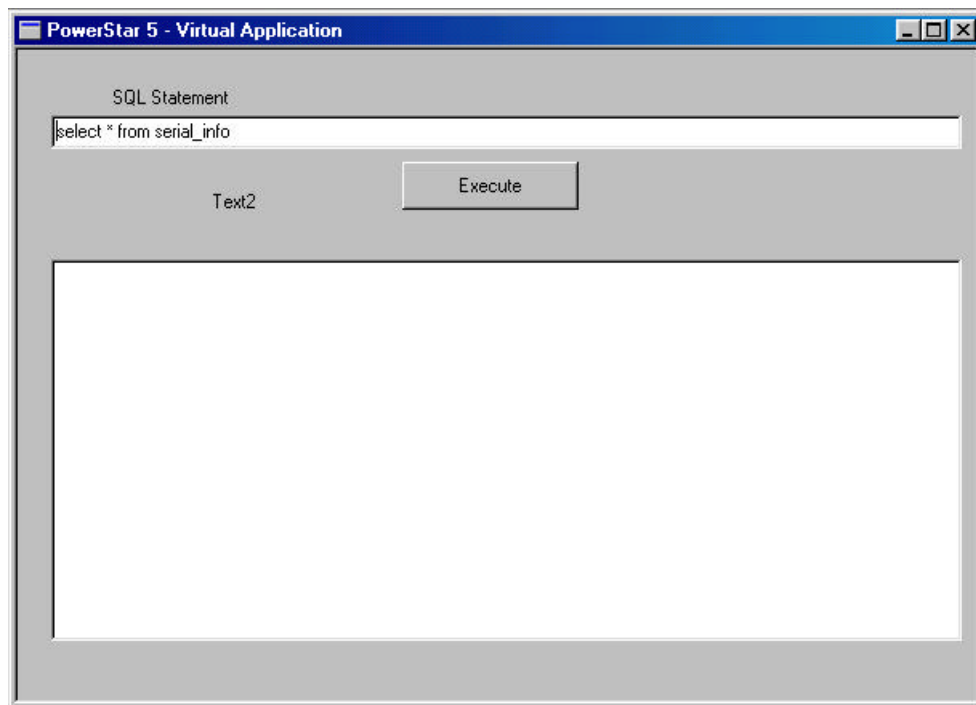
## PowerStar 5 SQL Interface

PowerStar 5 has integrated SQL support via its SQL interface.

Before using the SQL interface a data source must first be added. To add a data source select ODBC Data Sources from the control panel. In the example shown a data source called Access has been created which connects to the database d:\ps5\bin\1.mdb



Below is a VA which uses the SQL.Connect method to connect to the data source, in the SQL statement field any SQL statement can be added and executed, for queries the result will be displayed in the large text box. If the default statement 'select \* from serial\_info' is executed it will list all the data from the serial\_info table of your primary database using the ODBC drivers.



SQL object List:

**Connect <data source name>**

Connect to the specified data source.

```
nStatus = SQL.Connect("DSN=Access")
```

**Execute <SQL Statement>**

Executes the specified SQL statement.

```
nStatus = SQL.Execute("SELECT *")
```

**FieldCount**

Returns the number of fields in the record set.

```
nNumFields = SQL.FieldCount
```

**FieldByNum <field number>**

Returns the data contained in specified field number within the recordset.

```
For q = 0 To sql.FieldCount - 1  
    sMsg = sMsg & " " & sql.FieldByNum(q)  
Next
```

**FieldByName <Field Name>**

Returns the data contained in the specified field name within the recordset.

```
msgbox(SQL.FieldByName("Serial_num"))
```

**MoveFirst**

Moves to the first record within the rowset.

```
SQL.MoveFirst
```

**MoveNext**

Moves to the next record within the rowset.

```
While Not sql.IsEof  
    For q = 0 To sql.FieldCount - 1  
        sMsg = sMsg & " " & sql.FieldByNum(q)  
    Next  
    sql.MoveNext  
  
    ListBox1.AddString sMsg  
    sMsg = ""  
Wend
```

**IsEof**

Returns true when the last row of a row set has been reached.

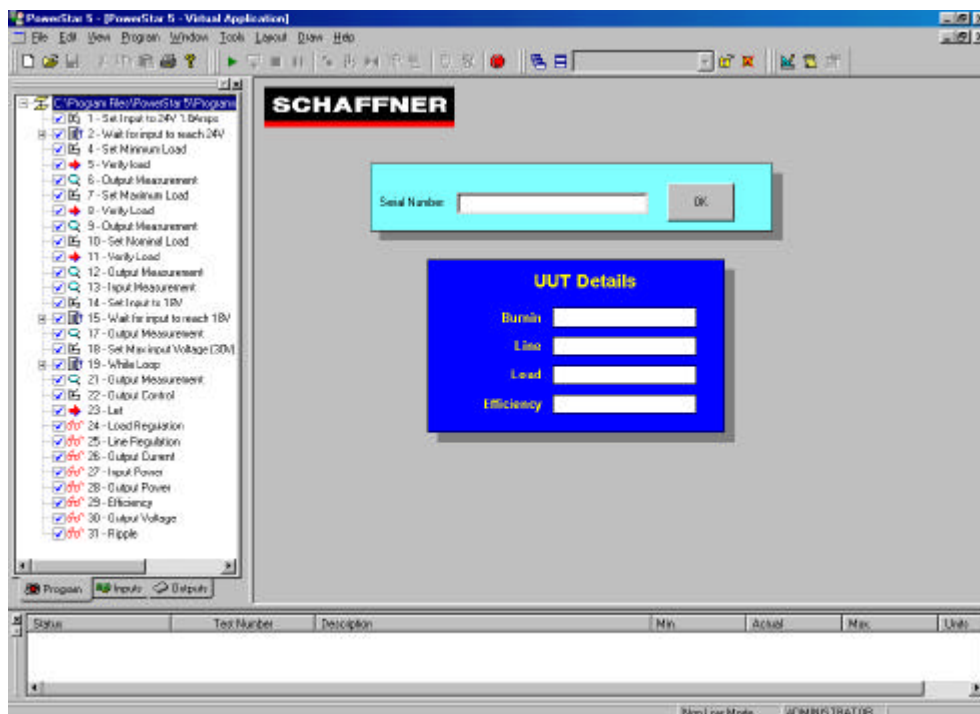
The following are two examples of how SQL could be applied:

### Example A – Automatic Selection of one of a Family of Programs

Customer has a wide range of products differing only in power rating, details of which are stored on the factory database. They use the same generic program by means of the Family Testing feature on PowerStar 5.

When a product is tested, the barcode is scanned in, and from this information PowerStar 5 automatically loads up the appropriate version of the program.

After the program is run, the test results are logged back as new data into the database to be stored and linked to the serial number.



### Example B – SQL Test Unit Tracking

SQL can be used to track units through production for real time analysis of product status. When a UUT goes to a functional test a serial no. is scanned in. The number is logged into a central database and the status of that number is queried using SQL. SQL can be used to determine if the unit has passed a test prior to functional test/burn-in. For example from the SQL it should be possible to determine if the unit is ready for final functional test and also to determine the program to use. At the end of program execution the central database can be automatically updated to say that the UUT has passed or failed.