PI OLEDB Basics, Learn How to Query PI

OSIsoft.

1.1 PI OLEDB Basics, Learn How to Query PI

1.1.1 Description

This tutorial will show how easy it is to get started and provides the minimum SQL knowledge to perform data queries.

1.1.2 Objectives

- Understand how the PI Server Catalogs and Tables are exposed via PI OLEDB
- Discover the SQL Language in the context of querying PI

1.1.3 Problem Description

The SQL connectivity of the PI Enterprise Server is very popular in middleware scenarios, e.g. to connect PI with Oracle or MS SQL Server. But it is also a very powerful end user tool. Unfortunately many users are scared of the SQL language and never look into this great opportunity. This tutorial will show how easy it is to get started and provides the minimum SQL knowledge to be able to formulate data queries. So step through the examples and learn how to write SQL queries for PI OLEDB.

Here is an introduction to the PI OLEDB Provider. This information is good to know but is not necessary to perform this lab using the Step-by-Step Instructions. At any time, feel free to jump to the "Approach" and "Step-by-Step Instructions" sections of this document.

The PI OLEDB data provider simply exposes data from PI Servers in the form of Tables, categorized in Catalogs. All of this is available through the standard way to communicate with relational databases: OLEDB.

The following catalogs exist in the current version:

piarchive	contains archive related tables
pibatch	contains batch data tables
pids	contains PI digital state tables
pifunction	contains tables representing PE functions
piheading	contains heading tables
pilog	contains the pimessagelog table
pimodule	contains tables representing the Module Database

pipoint	contains tag configuration tables (one per point class)
pisystem	contains product version information
piuser	contains user database tables

The user can also create new tables in the pipoint and pids catalog. This is equivalent to creating new point classes and new digital state sets, respectively.

Another function is to create Views. Large tables like the picomp table which represents all events in the PI archive may be unwieldy for end users. Also, it might be more convenient to virtually merge tables to a new table. This can be done with Views. A newly created View will be available to other users too.

This product is licensed with the PI Data Access (DA) server module and needs to be installed on the various computers from which one needs to access PI via OLEDB.

When installed on a user PC it may update the Microsoft MDAC components and will install or update the PI SDK. PI OLEDB is mainly a DLL, comparable to a printer driver or ODBC driver. It requires an application (in OLE DB terms called a consumer) to make use of it. One application that supports standard OLE DB providers by default is MS Excel (XP or higher). Also there are ActiveX controls that connect to OLE DB providers that can for example be inserted into ProcessBook displays. Finally PI OLEDB comes with a number of example applications that show how to develop your own OLE DB applications.

1.1.4 Suggested Approach

Part A

Use the PI OLEDB snap-in for MMC to explore the table structure.

- Starting from the example query of the *piarchive..piinterp* table, extract the values of the last day – with their timestamp – for the tag *ba:level.1* (only the *time* and *value* columns should be displayed)
- Create a new laboratory PI point in the *pipoint..classic* table, whose name is "LabTag1" and type is "Int32"
- Send two laboratory measurement values in the *LabTag1* point, using the *piarchive..picomp2* table, then make sure these values appear in the *pimin* and *pimax* tables:
- Value of -111, at the current time minus a few seconds
- Value of 222, at the current time
- Create a new digital state set (in the *pids* catalog) named "AlertStates", that contains 3 states: "Information", "Warning" and "Danger"
- Delete the point and the digital state set you just created

Part B

Use the PI OLEDB Tester to experiment with custom-built SQL statements.

- Read PI Archive values;
- Write PI Values to the Archive.
- Add a new user (with your name) in the piuser..piuser table using an INSERT INTO statement.
- Then modify the description of this new user to "User created with PI OLEDB", using an UPDATE statement.
- Make sure the user is created by issuing a SELECT statement.
- Finally, delete the new user with a DELETE statement.

Part C

Use the Sample Statements provided in the PI OLEDB Tester

- Execute the query example that extracts the points from the PI system with their description, when their pointsource attribute is equal to "R".
- Then modify the query so it extracts the creation date (creationdate column) as well, when the points' pointsource attribute is equal to "R" or "9".

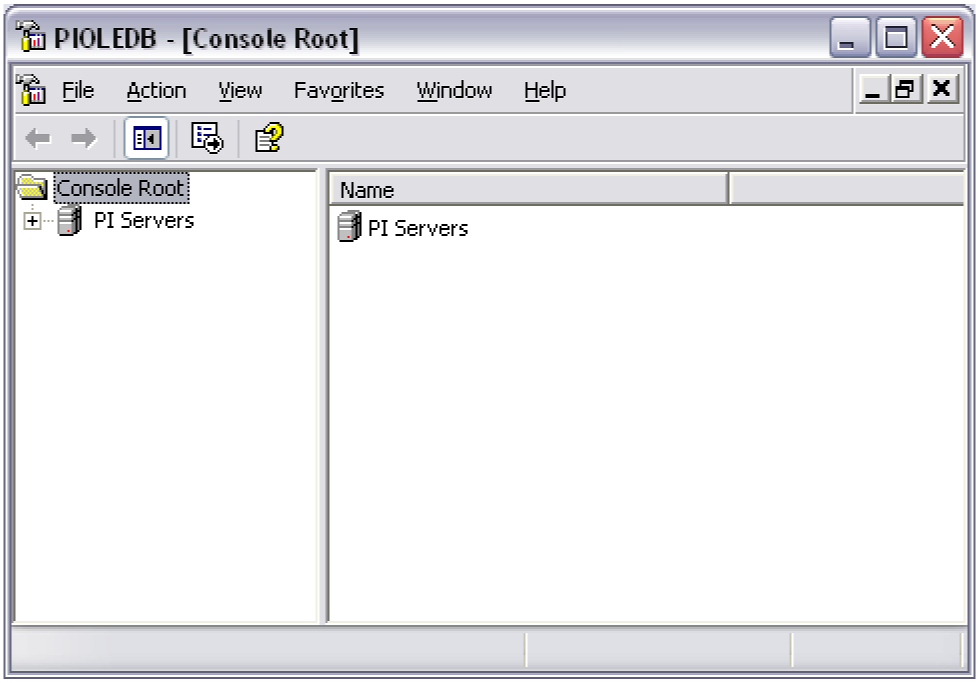
STOP

Try to do this exercise on your own before proceeding to the Step-by-Step Instructions.

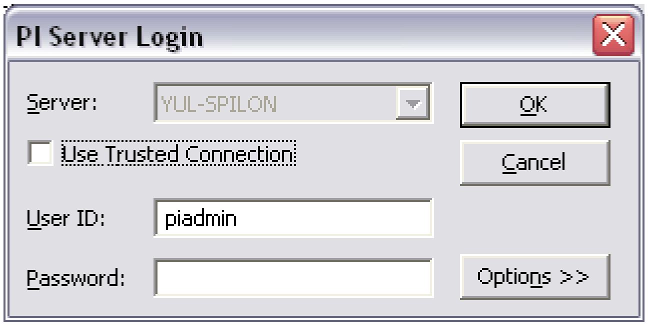
1.1.5 Step-by-Step Instructions

Part A

1. Open the Windows File Explorer and navigate to the following directory: *C:\Program Files\PIPC\OLEDB\Tools\MMC*
2. Double-click the file *PIOLEDB.msc*
3. You can alternatively run "mmc" from a Command Window and add the PIOLEDB snap-in to the Microsoft Management Console



4. Open the *PI Servers* branch and select one of the available PI Server
5. In the *PI Server Login* window that appears, check the *Use Trusted Connection.*



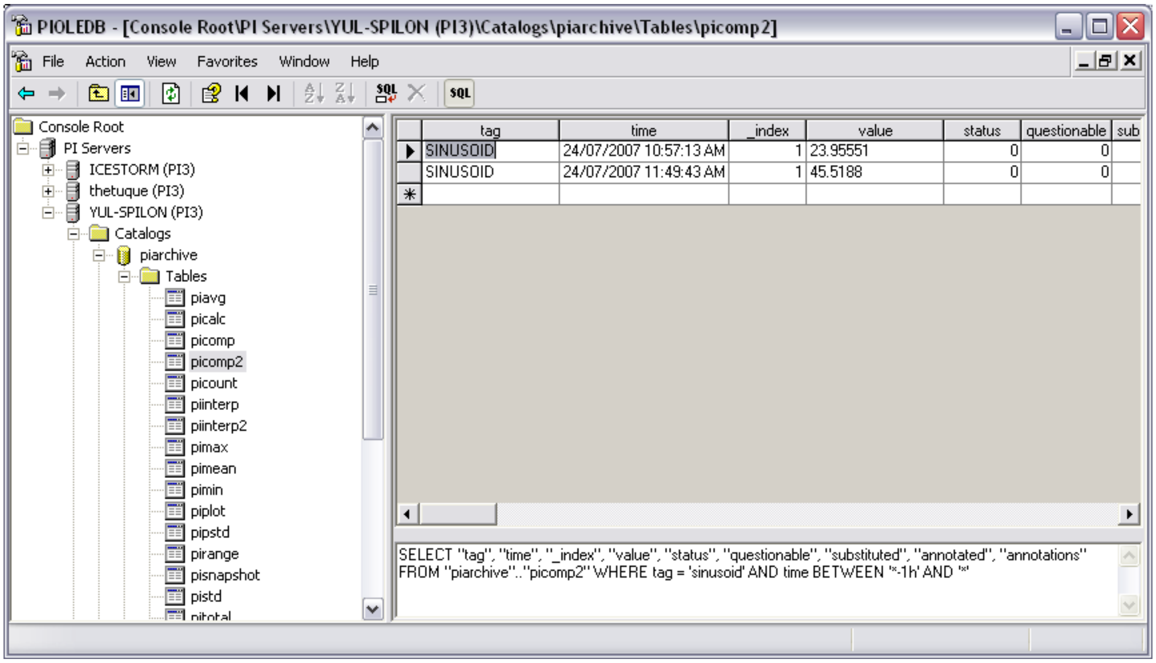
Server: YUL-SPILON

□ Use Trusted Connection

User ID: piadmin

Password:

6. Browse through the tree and discover the "Catalogs" and "Tables"

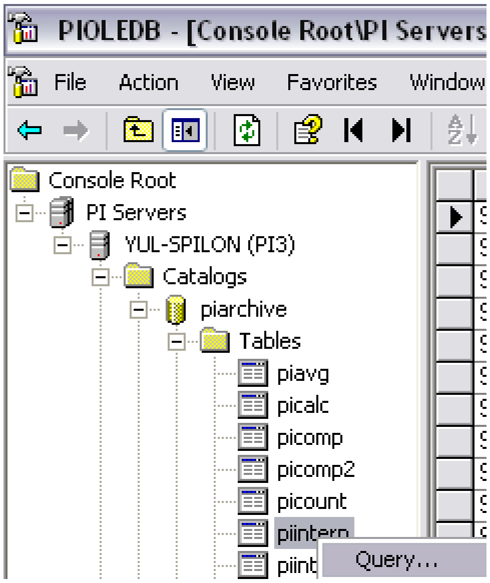


	tag	time	_index	value	status	questionable	sub
▶	SINUSOID	24/07/2007 10:57:13 AM	1	23.95551	0	0	
	SINUSOID	24/07/2007 11:49:43 AM	1	45.5188	0	0	
✳							

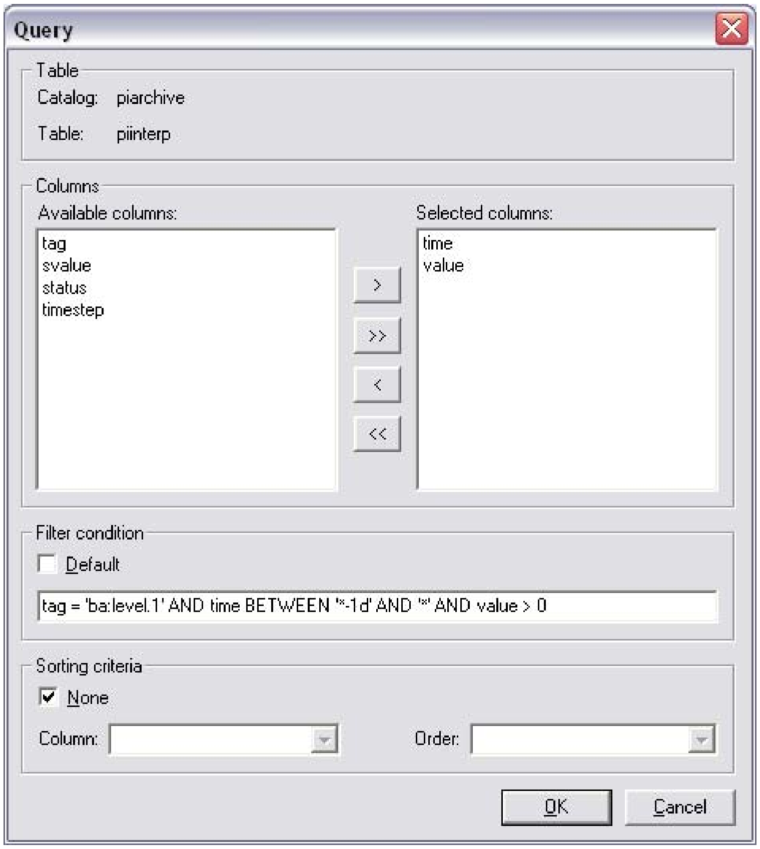
```
SELECT "tag", "time", "_index", "value", "status", "questionable", "substituted", "annotated", "annotations"
FROM "piarchive".."picomp2" WHERE tag = 'sinusoid' AND time BETWEEN '*-1h' AND '*'
```

Most tables have a predefined "WHERE" condition so that on a simple click you only see a subset of the available data in that table.

7. Select the *piinterp* table in the *piarchive* catalog and modify the example query by right-clicking on the table and choosing the *Query* option.



8. Move the *tag*, *status*, *svalue* and *timestep* columns to the left-hand pane so they do not appear in the results grid.
9. First clear the “Default” checkbox, then modify the conditions of the *WHERE* clause in order to extract only the positive values, for the tag *ba:level.1*, in the last day.



Catalog: piarchive

Table: piinterp

Available columns: tag, svalue, status, timestep

Selected columns: time, value

□ Default

tag = 'ba:level.1' AND time BETWEEN '*-1d' AND '*' AND value > 0

☑ None

Column:

Order:

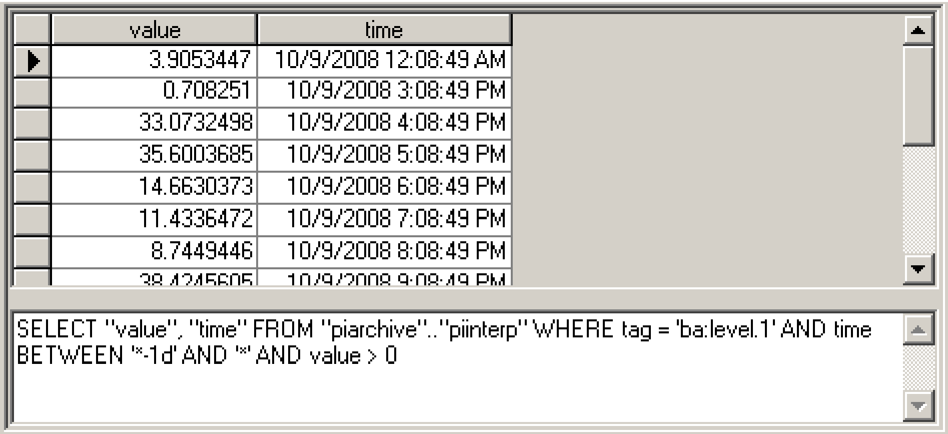
The complete filter expression should look like this:

```
tag = 'ba:level.1' AND time BETWEEN '*-1d' AND '*' AND value > 0
```

10. Click *OK* to execute the query.

This results in the following SQL expression with results as shown below.

```
SELECT time,value FROM piarchive..piinterp
WHERE tag = 'ba:level.1' AND time BETWEEN '*-1d' AND '*' AND value >
0
```



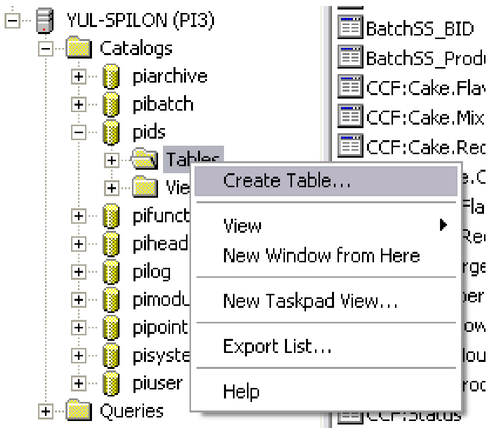
	value	time
▶	3.9053447	10/9/2008 12:08:49 AM
	0.708251	10/9/2008 3:08:49 PM
	33.0732498	10/9/2008 4:08:49 PM
	35.6003685	10/9/2008 5:08:49 PM
	14.6630373	10/9/2008 6:08:49 PM
	11.4336472	10/9/2008 7:08:49 PM
	8.7449446	10/9/2008 8:08:49 PM
	38.4245605	10/9/2008 9:08:49 PM

```
SELECT "value", "time" FROM "piarchive".."piinterp" WHERE tag = 'ba:level.1' AND time
BETWEEN '*-1d' AND '*' AND value > 0
```

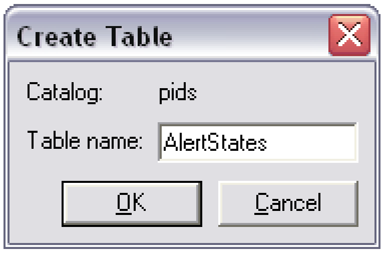
11. Click the *Refresh* button (or F5) to see what happens to the results.

Note that the timestamps (and the values) change according to the current time. This table offers interpolated values at regular intervals (the *timestep* column), which is set to '1h' by default

12. Open the *classic* table in the *pipoint* catalog.
13. In the last line (the one preceded by a star), add the new point by typing "LabTag1" in the *tag* columns and "int32" in the *pointtypex* columns
14. Open the *picomp2* table in the *piarchive* catalog
15. In the last line (the one preceded by a star), add the value by typing "LabTag1" in the *tag* column, "-111" in the *value* column and the timestamp (current time, minus a few seconds) in the *time* column, according to the regional format (see other values shown in this same table).
16. Repeat the last step with a value of 222, at the current time
17. Open the *pimin* table and verify that in the last hour, only the smallest value is extracted for the tag *LabTag1* (you have to modify the query for tag *LabTag1*)
18. Open the *pimax* table and verify that in the last hour, only the largest value is extracted for the tag *LabTag1* (you have to modify the query for tag *LabTag1*).
19. Select the *pids* catalog, right-click on its *Table* collection and choose the *Create Table…* option



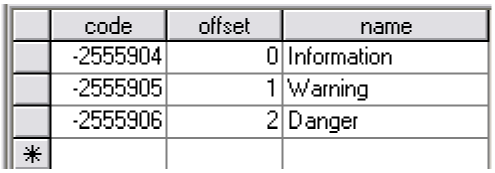
20. Name the table "AlertStates" and click on *OK*.



Catalog: pids

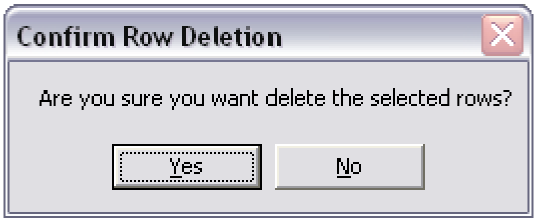
Table name: AlertStates

21. Open the new *AlertStates* table that now appears in the *pids* catalog.
22. Add the first state by typing "Information" in the *name* column
23. In the last line (the one preceded by a star), type "Warning" in the *name* column
24. Repeat the previous step for the "Danger" state



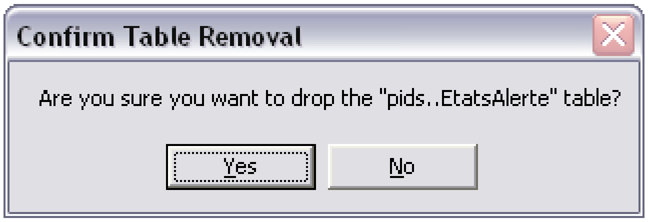
	code	offset	name
	-2555904	0	Information
	-2555905	1	Warning
	-2555906	2	Danger
*			

25. Open the *classic* table in the *pipoint* catalog.
26. Select the line of the point *LabTag1* (by clicking on the gray box at the beginning of the line) and press the *Delete* key on your keyboard or click on the ✕ button in the toolbar.
27. Click on *Yes* in the window that appears to delete the point.


Confirm Row Deletion

Are you sure you want delete the selected rows?

28. Right-click on the *AlertStates* table in the *pids* catalog, then choose the *Drop Table* option.
29. In the window that appears, click on *Yes*.


Confirm Table Removal

Are you sure you want to drop the "pids..EtatsAlerte" table?

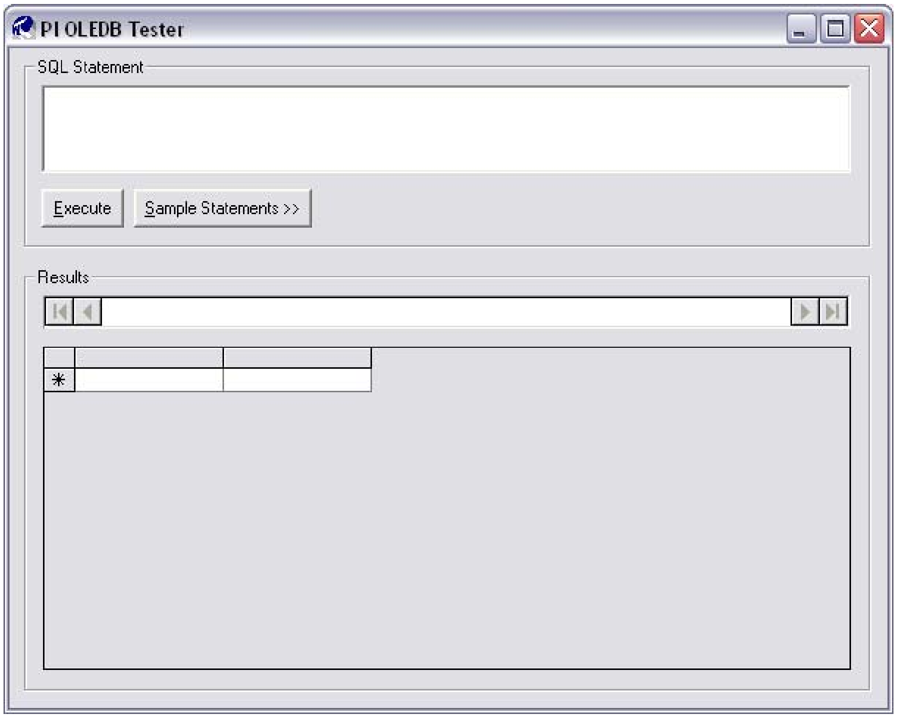
For more information on using the PI OLEDB snap-in for MMC, refer to Appendix C in the PI OLEDB Documentation:

C:\Program Files\PIPC\OLEDB\Doc\PIOLEDB.doc

Part B

As a first step in PI SQL we provide a simple Consumer application called PI OLEDB Tester.
It allows executing SQL statements against PI OLEDB and displays results in the Microsoft
Data Grid ActiveX control.

1. Open the Windows File Explorer and navigate to the following directory: *C:\Program Files\PIPC\OLEDB\Tools\PI OLEDB Tester*
2. Double-click the file *PIOLEDBTester.exe*
3. In the *PI Server Login* window that appears, either check the *Use Trusted Connection* checkbox or use "piadmin" as the *User ID* with a blank *Password*
4. You can now enter PI SQL commands into the *SQL Statement* text field



The way to **retrieve data** from a table is to use a **SELECT** query. A simple SELECT query looks as follows:

```
SELECT column_1, column_2, column_N

FROM table1

WHERE column_1 = 'text' AND column_2 > 0
```

A SELECT query returns a table (called result set). The columns of the result set are the ones that were specified between the SELECT and FROM keywords. As columns you can specify any or all of the columns that exist on the table (table1) that you query data from. Instead of specifying all columns you can also use the '*' as shortcut:

SELECT * FROM table1

After the WHERE keyword you specify a condition (expression) that each row of the originating table must meet in order to be copied to your result set. If you omit the WHERE part (it is optional) you will receive all rows of the originating table.

5. Type the following command and click *Execute* to read data from PI:

```
SELECT tag, descriptor, zero, span FROM pipoint..classic
```

Note the "2 dots" notation in PIPoint..Classic.

To **insert new rows** into a table this table must be fully updatable or at least support inserts. The picomp2 table for example is fully updatable and inserting a new row into the picomp2 table is equivalent to sending a new value into a certain PI Point.

The related query looks as follows:

INSERT piarchive..picomp2 (tag, time, value, status)

VALUES ('somePIPoint', 'y+8h', 12.5, 0)

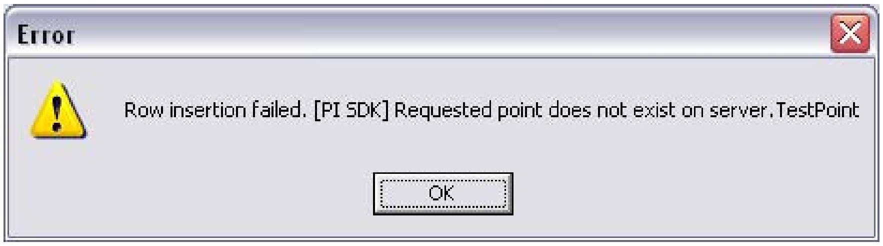
After the INSERT keyword you find the target table name followed by a list of column names set in parentheses. Then after the VALUES keyword you provide the values for each given column (again in parentheses) in the same order as the columns were specified before. String values have to be surrounded by single quotes.

It is not required that all columns be present in the query. Columns that are left out will get default values. However, some tables have columns that are mandatory to be contained in the INSERT query.

6. Type the following command and click *Execute* to write data to PI:

```
INSERT piarchive..picomp2 (tag, time, value, status)
VALUES ('TestPoint', '*', 12.5, 0)
```

7. Because the specified tag does not exist (TestPoint) the query returns an error:



Row insertion failed. [PI SDK] Requested point does not exist on server.TestPoint

8. Type the following command and click *Execute* to create it into PI:

```
INSERT pipoint..classic (tag, pointtypex)
VALUES ('TestTag', 'float32')
```

9. Try steps 10 and 11 again; that will write a value of 12.5 at the current time
10. Verify that the value was written by typing this command clicking *Execute*:

```
SELECT * FROM piarchive..picomp2
WHERE Tag = 'TestTag' AND time > '*-1h'
```

11. To create the new user (with your name), type the following **INSERT INTO** statement in the *SQL Statement* field, then click on the *Execute* button:

```
INSERT INTO piuser..piuser (name) VALUES ('YourName')
```

12. To modify the description of the new user, execute the following **UDPATE** statement:

```
UPDATE piuser..piuser SET description = 'User created with PI OLEDB'
WHERE name = 'YourName'
```

13. To make sure the user is correctly created, list all users by executing the following **SELECT** query:

```
SELECT name, description FROM piuser..piuser
```

14. Delete the new user from the table using the following **DELETE** statement. Execute the query. A

message will inform you the operation was completed successfully.

```
        DELETE FROM piuser..piuser WHERE name='YourName'
```

Make sure the user with your name has been deleted from the *piuser..piuser* table.

It is possible to modify the tables directly in the grid:

- Add new users in the last line.

- Modify users directly in the grid.

- Delete a user by choosing a line and then pressing the *Suppr* key

In order for the changes to be effective, you must click in another line of the grid.

Here are small examples that demonstrate the power of PI SQL. They perform tasks that cannot easily be done with other tools. Feel free to execute them in PI OLEDB Tester.

Count Tags by Point Source Attribute

```
SELECT pointsource, COUNT(*)

FROM pipoint..classic GROUP BY pointsource
```

How Many Events since Beginning of This Month

```
SELECT COUNT(*) FROM piarchive..picomp2

WHERE tag = 'sinusoid' AND time >= BOM('*')
```

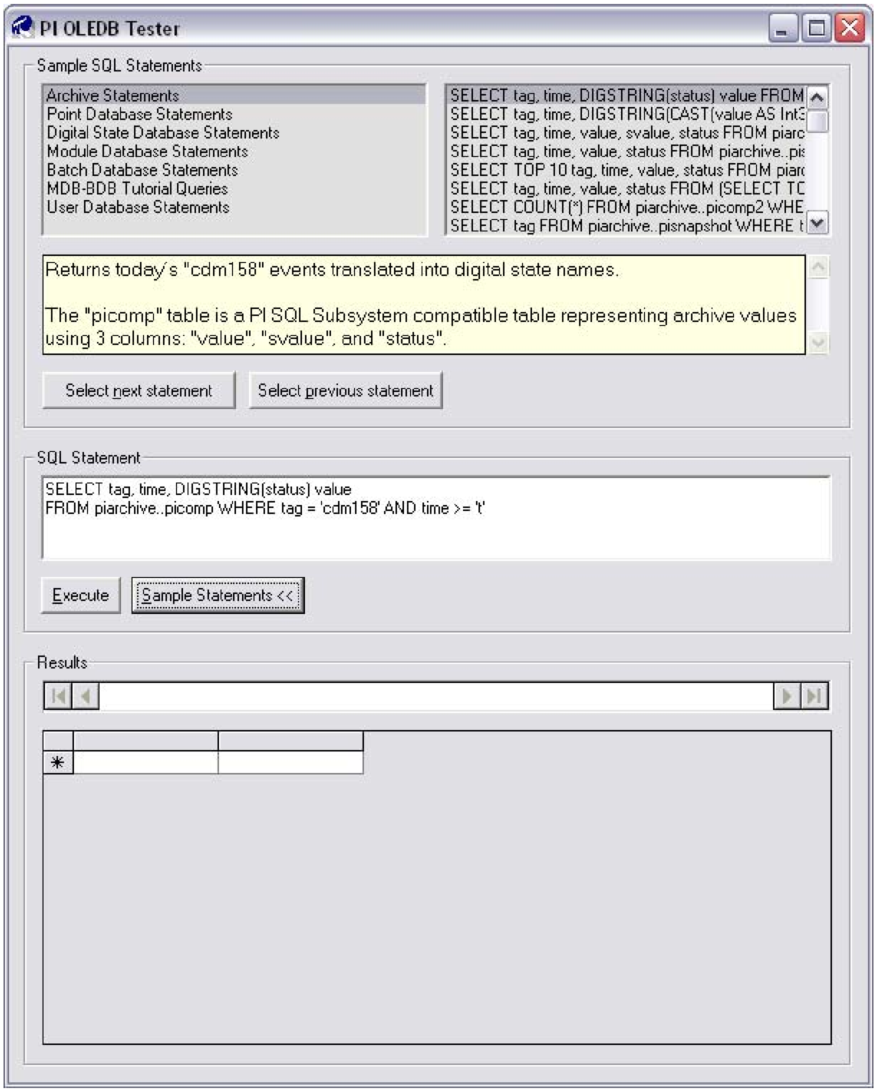
Get Annotated Events

```
SELECT time, value, annotations

FROM piarchive..picomp2

WHERE tag = 'mytag' AND time >= 't' AND annotated = TRUE
```

More example queries can be found in the PI OLEDB manual, Compendium of SQL Statements.

Part C

1. Still in *the PI OLEDB Tester* application, click the *Sample Statements* >> button
2. You should see the following:



Archive Statements
Point Database Statements
Digital State Database Statements
Module Database Statements
Batch Database Statements
MDB-BDB Tutorial Queries
User Database Statements

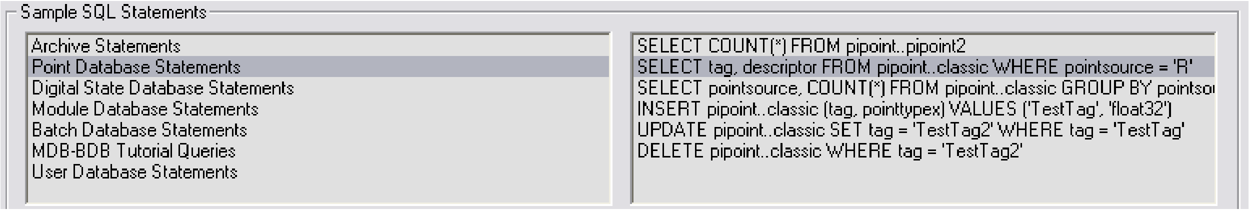
```
SELECT tag, time, DIGSTRING(status) value FROM
SELECT tag, time, DIGSTRING(CAST(value AS Int3
SELECT tag, time, value, svalue, status FROM piarc
SELECT tag, time, value, status FROM piarchive..pis
SELECT TOP 10 tag, time, value, status FROM piarc
SELECT tag, time, value, status FROM (SELECT TC
SELECT COUNT(*) FROM piarchive..picomp2 WHE
SELECT tag FROM piarchive..pisnapshot WHERE t
```

Returns today's "cdm158" events translated into digital state names.

The "picomp" table is a PI SQL Subsystem compatible table representing archive values using 3 columns: "value", "svalue", and "status".

```
SELECT tag, time, DIGSTRING(status) value
FROM piarchive..picomp WHERE tag = 'cdm158' AND time >= 't'
```

3. Browse through the available categories of statements (**top-left corner list**), select one of the available statements (**top-right corner list**) and click on the *Execute* button.
4. In the left-hand list, select the *Point Database Statements* category
5. In the right-hand list, select the query that extracts the points for which the *pointsource* attribute is equal to "R", and then click on the *Execute* button.



Sample SQL Statements

- Archive Statements
- Point Database Statements
- Digital State Database Statements
- Module Database Statements
- Batch Database Statements
- MDB-BDB Tutorial Queries
- User Database Statements

```
SELECT COUNT(*) FROM pipoint..pipoint2
SELECT tag, descriptor FROM pipoint..classic WHERE pointsource = 'R'
SELECT pointsource, COUNT(*) FROM pipoint..classic GROUP BY pointso
INSERT pipoint..classic (tag, pointtypex) VALUES ('TestTag', 'float32')
UPDATE pipoint..classic SET tag = 'TestTag2' WHERE tag = 'TestTag'
DELETE pipoint..classic WHERE tag = 'TestTag2'
```

6. In the *SQL Statement* field, modify the query as follows, and then click on the *Execute* button:

```
SELECT tag, descriptor, creationdate
FROM pipoint..classic WHERE pointsource = 'R' OR pointsource = '9'
```

Note that some queries may depend on previous queries. For example, in order to return with success inserting a value into a PI Point depends on the previous query that created that PI Point.