

## Activity 46

Try it yourself

Try executing a similar JOIN query in the GAMERESERVE database by checking where the GateID field from the Gates table is equal to the Entrance field in the Visitors table.

Display all the fields from the Visitors table and the actual name of the entrance/gate (the GateName field from the Gates table).

## Allowing the user to enter SQL code

As you should have gathered by now, SQL means that you do not have to write lines of code in order to work with the data in your database. All you need to do is construct (or even load from file) a text string that holds the correct SQL statement, put the SQL statement into the SQL property and execute it!

This is very powerful and exciting, because it makes it easy to change your program without having to write a lot of code. You can even add a Memo, allow the user to type in their own SQL and then run that.

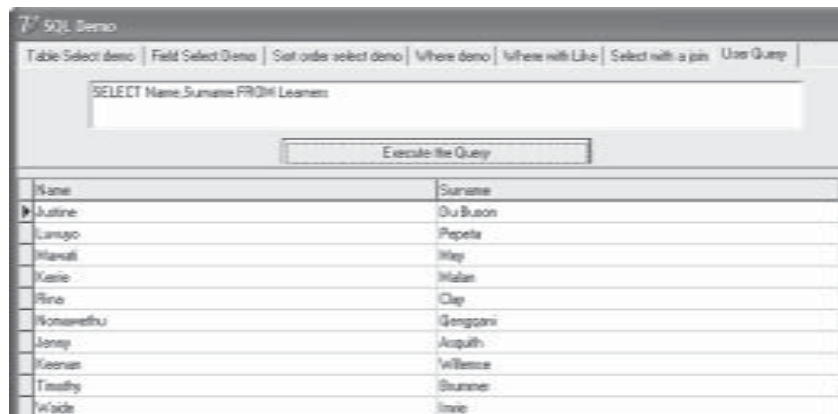
Let's give that a try:

## Activity 47

Explore

We will be adding to our current program, MusicDB\_SQLDemo.

- Create a new TabSheet [User Query].
- Place a Memo and a Button on the TabSheet.
- Format it to look as follows:



- Write the following code for the Button's OnClick Event handler:

```
procedure TfrmMusic.btnExecuteQueryClick(Sender: TObject);
begin
    qryMusicdb.Active := false;
    qryMusicdb.SQL.Text := memoQuery.Lines.Text;
    qryMusicdb.Active := true;
end;
```
- Run the program, type in an SQL statement, click the Button and watch the magic!

## Using SQL to delete records

SQL also allows us to delete records from a table. It looks and works as follows:

DELETE FROM Music	This will delete all records from the table.
DELETE FROM Music WHERE artist = 'Blue'	This will delete all songs from the artist 'Blue' from the table.
DELETE FROM Music WHERE artist = 'Eminem' and title= 'Superman'	This refines the delete even more – it deletes records where the artist is 'Eminem' and the title is 'Superman'.

DELETE also works with the LIKE command.

DELETE is an SQL statement which does not return a result set (i.e. it does not result in a table that can be displayed).

Therefore the Open method or setting the Active property to true can not be used. The ExecSQL method has to be called, and an additional SQL query needs to be executed to display the changed dataset.

```
procedure TForm1.btnDelRockClick(Sender: TObject);
begin
    qryDel.SQL.Text := 'DELETE FROM Music WHERE genre = ''Rock''';
    qryDel.ExecSQL ;
    qryDel.SQL.Text := 'SELECT * FROM Music' ;
    qryDel.Open ;
end;
```

**NB:** Be careful when you use DELETE. It can't be undone and the data will be lost!

### Note:

- The WHERE clause is vital, otherwise all records are deleted from the table!
- Deleting can be tricky and fail! This might happen if Access is open at the same time as your Delphi program – or even if your program source code is open (with an active table open) at the same time that you are trying to delete the data. To fix it make sure that the only application working with the table is your program.

## Activity 48

Try it yourself

- Open the program you worked on in Activity 47.
- Type some DELETE SQL statements into the Memo on the [User Query] TabSheet of the SQL demo program you created earlier. Then type in the display commands to check that the data has been deleted!

**NB** Make sure that you have a copy of the original database safely stored somewhere!

Beware – since you do not know if the user entered a SELECT or a DELETE statement, you will not know if the ExecSQL or the Open method should be called. An error message will be displayed if you use the Open method (or set Active property true) when a DELETE statement was entered. You need to change the Event Handler to make provision for both cases.

```
procedure TForm1.btnExecuteQueryClick(Sender: TObject);
begin
  qryMusicdb.SQL.Text := memoQuery.Lines.Text;
  try
    qryMusicdb.Open ;
  except //it was not a SELECT query
    qryMusicdb.ExecSQL ;
    //do a query to display the data
    qryMusicdb.SQL.Text := 'SELECT * FROM Music' ;
    qryMusicdb.Open ;
  end ;
end;
```

## Using SQL to change records

SQL can change records as easily as it deletes them. To tell SQL you are changing a record you start the command with the keyword **UPDATE**. You then use the keyword SET to show what needs to be changed.

The general format of the UPDATE command is:

```
UPDATE < table >
SET    < Field 1 > = < Value >
         < Field 2 > = < Value >
         < WHERE > < Criteria >
```

where table is the name of the table where the record(s) are to be updated and the WHERE clause and criteria are what we encountered in the SELECT examples. Here's an example that will show you how easy it is:

```
UPDATE Music SET Artist = 'Punk Rapper' WHERE Artist = 'Eminem'
UPDATE Music SET Year = '2000',Artist = 'Boy Band' WHERE Artist = 'Blue'
                                                AND Title = 'Curtain Falls'
UPDATE Music SET Cost = Cost * 1.10
```

*Note:*

- UPDATE also does not return a result set, therefore the ExecSQL method needs to be called.
- The SET command defines which field(s) you want to change, whilst the WHERE part of the SQL statement defines which records will be affected.
- Using UPDATE and SET without a WHERE command affects all the records in a table!
- The first example changes all songs by 'Eminem' into songs by 'Punk Rapper'.
- The second example will only affect songs called 'Curtain Falls' by the artist 'Blue'.