

Mr Long ON SQL

Summary

Structured Query Language

SQL (Structured Query Language) is a special language used to manipulate and query data in a database.

Basic Structure of SELECT query

SELECT query is used to display specific data from one or tables in a database.

SELECT <field names to be displayed>
FROM <table(s) that contain those fields>
WHERE <criteria for the query>

*NOTE: Can use SELECT * if you want to list all fields in a table.*

Examples used in this document will reference this table (CD) that contains the following fields:

- Artist (text)
- CD_name (text)
- Genre (text)
- ReplacementValue (Currency)
- OwnerID (Number)

 CD

Artist	CD_name	Genre	ReplacementValu	OwnerID
LL Cool J	10	Jazz	R 130.00	1
Staind	14 shades of grey	Alternative Rock	R 120.00	8
Goo Goo Dolls	A Boy named Goo	Alternative Rock	R 100.00	60
Celine Dion	A New day has come	Pop	R 120.00	2
Coldplay	A Rush of Blood to the head	Rock	R 120.00	5
Muse	Absolution	Rock	R 180.00	65

Examples:

SQL Statement	Description
SELECT * FROM CD	Display all the fields of the CD table.
SELECT Artist, CD_Name FROM CD	Display only the Artist and CD name fields of the CD table.
SELECT * FROM CD WHERE Genre = "Rock"	Display all the fields of the CD table but only the records where the Genre is "Rock".

Different types of criteria

The following section will cover examples of criteria that are used in the WHERE clause.

*NOTE: Assume that these queries have a SELECT * FROM CD above the where clause.*

- WHERE clause with normal operators (= , <> , >= , <= , > and <)

Examples	Description
WHERE ReplacementValue = 100	Where the replacement value is EQUAL to R100.
WHERE Genre <> "Rock"	Where genre is NOT EQUAL to Rock <i>NOTE: Text values must have double quotes.</i>
WHERE ReplacementValue >= 120	Where the replacement value is GREATER THAN 120.
WHERE Artist < "F"	Where the Artist starts with a letter A, B, C, D or E only. (Just before F but doesn't include F).
WHERE HasPaid = TRUE	Where HasPaid is TRUE <i>(If HasPaid was a boolean field)</i>

- WHERE clause with multiple criteria using the **AND** and **OR** operators

Examples	Description
WHERE ReplacementValue = 100 AND Genre = "Rock"	All Rock CD's with a replacement value of R100.
WHERE ReplacementValue = 100 OR Genre = "Rock"	All Rock CD's (regardless of replacement value) or all CD's with a replacement value of R100 (regardless of genre).
WHERE ReplacementValue >= 120 AND ReplacementValue <= 150	Where the replacement value is between R120 and R150 (inclusive).
WHERE Genre = "Rock" OR Genre = "Jazz" AND ReplacementValue = 150	All Rock CD's (regardless of replacement value) as well as All Jazz CD's with a replacement value of R150. <i>NOTE: AND operator takes precedence over OR.</i>
WHERE (Genre = "Rock" OR Genre = "Jazz") AND ReplacementValue = 150	All Rock or Jazz CD's with a replacement value of R150. <i>NOTE: Brackets are evaluated first.</i>

- WHERE clause with the **NOT** operator

Examples	Description
WHERE NOT Genre = "Pop"	Where genre is NOT equal to Pop. (Same as Genre <> "Pop")
WHERE NOT ReplacementValue > 100	Where replacement value is NOT greater than R100. (Same as ReplacementValue <= 100)

- WHERE clause with the **BETWEEN** operator

Examples	Description
WHERE ReplacementValue BETWEEN 120 AND 150	Where the replacement value is between R120 and R150 (inclusive).
WHERE Artist BETWEEN "A" AND "F"	Where Artist starts with an A, B, C, D or E. (Just before F but doesn't include starts with F)

- WHERE clause with the **IS NULL** operator

Examples	Description
WHERE Artist IS NULL	Where the artist field is empty / blank.
WHERE Artist IS NOT NULL	Where the artist field contains something (not blank or empty).

- WHERE clause with the **IN** operator

Examples	Description
WHERE ReplacementValue IN (100, 120, 150)	Where the replacement value is either R100, or R120 or R150.
WHERE Genre IN ("Rock", "Jazz", "Pop")	Where the genre is either Rock or Jazz or Pop.

- WHERE clause with the **LIKE** operator
 - * (star symbol) means nothing or anything (any amount of characters)
 - ? means any ONE character (Can not be empty)

Examples	Description
WHERE Artist LIKE "M*"	Where the artist <u>starts</u> with the letter 'M'.
WHERE Artist LIKE "*son"	Where the artist <u>ends</u> with an 'son'.
WHERE Artist LIKE "*be*"	Where the artist has an 'be' <u>somewhere</u> in the field.
WHERE Artist LIKE "M???"	Where the artist has M as a first character & then any 3 other characters after it. (In other words, where artist is any 4 letter beginning with an M)

Sorting and Unique records

When wanting to sort the data in your query, you use the ORDER BY clause.

```
SELECT <field names to be displayed>
FROM <table(s) that contain those fields>
WHERE <criteria for the query>
ORDER BY <field name> <ASC or DESC>
```

- The ORDER BY clause always comes after the WHERE clause.
(after the FROM clause if there is no where clause)
- If you just mention the field name, then it is assumed to be ascending.
- Use the code ASC for ascending and DESC for descending (must be capitals).
- If you sort by multiple fields, then you separate each field by a comma.

Examples	Description
ORDER BY ReplacementValue	Sort by replacement value in ascending order
ORDER BY Genre ASC	Sort by genre in ascending order
ORDER BY Genre DESC	Sort by genre in descending order
ORDER BY Genre ASC, Artist DESC	Sort first by genre in ascending order, then for genres that are the same, sort by Artist in descending order.

For unique values in a field of duplicates you use the keyword DISTINCT before the field names in the SELECT clause.

Examples	Description
SELECT DISTINCT Genre FROM CD	Display all the genres in the CD table with no duplicates.
SELECT DISTINCT Genre, Artist FROM CD	Display all the combinations of genres and artists in the CD table that are unique.

Calculations in a query

You may add a calculated field to your query in the SELECT clause.

- You may give this calculated field a name by following the calculation with the AS operator and then the name of the calculated field.
- If you use current fields in your calculations, make sure they are spelt as they are in the database table.
- If a field contains spaces in its name, use square brackets around the full field name when referring to it.

Examples	Description
SELECT Artist, ReplacementValue, 15/100 * ReplacementValue AS VAT	Displays a third field called VAT that is 15% of the replacement value.
SELECT Artist, CD_name, ReplacementValue, ReplacementValue + 30 AS [Final Cost]	Displays a fourth field called <i>Final Cost</i> that is the replacement value plus an additional R30.

You can't use the calculated fields name in the WHERE or ORDER BY clause.

- If you need a criteria based on the calculation field then restate the calculation in the WHERE clause.

Example: WHERE **15/100 * ReplacementValue** >= 25

- If you need to sort by the calculated field, you can refer to the calculated field by the number of that field in the query. Every field mentioned in the SELECT clause is numbered, starting at 1. So with the following select statement:

SELECT Artist, ReplacementValue, **15/100 * ReplacementValue AS VAT**

Artist is field 1, *ReplacementValue* is field 2 and *VAT* is field 3.

To sort by VAT you refer to it's number and whether you want ASC or DESC

Example: ORDER BY **3** ASC

Functions in the calculated fields

- FORMAT function displays the field in a specified format.
 - Parameters are first the field name or calculation followed by the required format *IN DOUBLE QUOTES*
 - This is considered a calculated field, so you need to provide a new name with the AS operator

Examples	Description
SELECT Artist, ReplacementValue, FORMAT(15/100 * ReplacementValue , "Currency") AS VAT	Displays the calculated field in a currency format.
SELECT Artist, CD_name, FORMAT(ReplacementValue , "Fixed") AS Value	Displays the replacement value to TWO decimal places with the field name: Value.
SELECT Artist, CD_name, FORMAT(ReplacementValue , "0.0") AS Value	Displays the replacement value to ONE decimal place with the field name: Value.

- ROUND function rounds off the field or calculation.
 - Parameters are first the field name or calculation followed by the number of decimal places to be returned.
 - ROUND (25.756 , 2) returns a value of 25.76
 - ROUND (25.753 , 2) returns a value of 25.75
 - ROUND (25.753 , 0) returns a value of 26

Examples	Description
SELECT Artist, ReplacementValue, ROUND(15/100 * ReplacementValue , 2) AS VAT	Displays the calculated field rounded to 2 decimal places.
SELECT Artist, CD_name, ROUND(ReplacementValue , 0) AS Value	Displays the replacement value rounded to zero decimal places.

- INT function removes the decimal part of the value.
 - INT (25.3) returns a value of 25
 - INT (25.7) returns a value of 25

Examples	Description
SELECT Artist, ReplacementValue, INT(15/100 * ReplacementValue) AS VAT	Displays the calculated field as a whole number (NOT rounded).

- STR function converts a numerical field or calculation into text to be used with text functions.

Examples	Description
SELECT Artist, STR(ReplacementValue) + " per order" AS Details	Displays the second field (called <i>Details</i>) in the format of the replacement value with text added at the end: Example: 130 per order

- LEN function returns the number of characters in the given text parameter.
 - LEN ("Hello") returns a value of 5
 - LEN ("Hello World") returns a value of 11

Examples	Description
SELECT Artist, LEN(Artist) AS [Chars In Name]	Displays the second field (called <i>Chars In Name</i>) with the number of characters in each artist field

- UCASE function returns all the letters in the text field as upper case letters.
- LCASE function returns all the letters in the text field as lower case letters.
 - UCASE ("Hello") returns a value of *HELLO*
 - LCASE ("Pop Rock") returns a value of *pop rock*

Examples	Description
SELECT Artist, UCASE (CD_name)	Displays the CD names in all capital letters.
SELECT Artist, LCASE (Genre)	Displays the genres in all small letters.

- LEFT function copies a specific number of characters from the left side of the given text parameter.
- RIGHT function copies a specific number of characters from the right side of the given text parameter.
 - Parameters are first the text field name or calculation followed by the number of characters to copy.
 - LEFT ("Hello World" , 4) returns a value of *Hell*
 - RIGHT ("Hello World" , 4) returns a value of *orld*

Examples	Description
SELECT Artist, LEFT (Artist , 1) AS [Artist Initial]	Displays the second field (called <i>Artist Initial</i>) with the first character of the artist field.
SELECT Artist, RIGHT (Genre , 4) AS Last_Value	Displays the second field (called <i>Last_Value</i>) with the last 4 characters of the genre field.

- MID function copies a specific number of characters from a specified position in the given text parameter.
 - Parameters are first the text field name or calculation followed by the character's position start coping from followed by the number of characters to copy.
 - MID ("Hello World" , 4 , 5) returns a value of *lo Wo*
 - MID ("Hello World" , 7 , 3) returns a value of *Wor*

Examples	Description
SELECT Artist, MID (Artist , 3, 2) AS [Random Text]	Displays the second field (called <i>Random Text</i>) with the third and fourth characters of the artist field.

Aggregate functions

These functions can be used to perform statistical or aggregate summaries on multiple records in a database table.

- They are used in the SELECT clause and need a field name with the AS operator.
- Use the function name with a field name as a parameter.
- The following functions can ONLY be used on numerical fields (Fields that contain numbers: integers, currency, etc):
 - SUM – returns the sum of the values in the specified field.
 - AVG – returns the average of the values in a specified field.
 - MIN – returns the smallest value in a specified field.
 - MAX – returns the largest value in a specified field.

Examples	Description
SELECT SUM (ReplacementValue) AS Total FROM CD	Displays the total replacement value of ALL CD's in the table.
SELECT AVG (ReplacementValue) AS [Average Muse] FROM CD WHERE Artist = "Muse"	Displays the average replacement value of just the CD's from the artist: Muse.
SELECT MIN (ReplacementValue) AS [Cheapest Rock] FROM CD WHERE Genre = "Rock"	Displays the cheapest (smallest) replacement value of just the Rock CD's.
SELECT MAX (ReplacementValue) AS [Most Expensive] FROM CD	Displays the most expensive (largest) replacement value of ALL the CD's in the table.

- The COUNT functions can be used on any field.
- COUNT(*) can also be used if you do not want to specify a field name.
- This function returns the number of records in the query.

Examples	Description
SELECT COUNT (*) AS TotalRecords FROM CD	Displays the total number (how many) of records in the CD table.
SELECT COUNT (*) AS [Total Rock] FROM CD WHERE Genre = "Rock"	Displays the total number (how many) of Rock CD's.
SELECT MIN (ReplacementValue) AS [Cheapest Rock] FROM CD WHERE Genre = "Rock"	Displays the cheapest (smallest) replacement value of just the Rock CD's.
SELECT COUNT (ReplacementValue) AS [Number of Prices] FROM CD	Displays the total number (how many) of records that have a replacement value entered.

NOTE: The examples shown above will only display ONE block with ONE value in it.

Grouping results

There aggregate functions mentioned above only display ONE result. If you want to display multiple results, based on groups of records, then you add the GROUP BY clause after the WHERE clause.

- If you use the ORDER BY clause, then it will appear AFTER the GROUP BY clause.
- Rule: the moment you have more than one field in the SELECT clause with an aggregate function, then you will always use the GROUP BY clause with the fields that are NOT in the aggregate function.

```

SELECT    <field name(s)> , <Aggregate function with field>
FROM      <table(s) that contain those fields>
WHERE     <criteria for the query>
GROUP BY <field name(s)>
ORDER BY <field name> <ASC or DESC>
  
```

Examples	Description
SELECT Genre , AVG(ReplacementValue) AS Average_Cost FROM CD GROUP BY Genre	Displays the average replacement value for each genre. Example: Rock R123.00 Jazz R150.00.....
SELECT Artist , MIN (ReplacementValue) AS Cheapest_CD FROM CD GROUP BY Artist	Displays the cheapest replacement value for each artist. Example: Aaliyah R125.00 Akon R160.00....
SELECT Genre , MAX(ReplacementValue) AS MostExpensive FROM CD GROUP BY Genre ORDER BY MAX(ReplacementValue) DESC	Displays the most expensive replacement value for each genre (ordered from most expensive to least expensive). Example: Pop R200.00 Rap R190.00....

NOTE: For the last query, you could also have used ORDER BY 2 DESC
 (Order by 2nd field listed in SELECT clause)

If you want to apply a criteria to the aggregate function, then you can't use that aggregate function in the WHERE clause. Instead you have to make use of the HAVING clause.

- The HAVING clause comes after the GROUP BY clause but before the ORDER BY clause.


```

SELECT <field name(s)> , <Aggregate function with field>
FROM <table(s) that contain those fields>
WHERE <criteria for the query>
GROUP BY <field name(s)>
HAVING <criteria of aggregate function>
ORDER BY <field name> <ASC or DESC>
  
```

Examples	Description
SELECT Genre , AVG(ReplacementValue) AS Average_Cost FROM CD GROUP BY Genre HAVING AVG(ReplacementValue) >= 150	Displays the average replacement value for each genre but only the ones where the average replacement value is R150 or more.
SELECT Artist , MAX(ReplacementValue) AS MostExpensive FROM CD GROUP BY Artist HAVING MAX(ReplacementValue) < 150 ORDER BY MAX(ReplacementValue) DESC	Displays the most expensive replacement value for each artist (ordered from most expensive to least expensive) but only those that have a most expensive replacement value less than R150.

Using Date fields

Examples used in this section will reference this table (Owner) that contains a DateOfBirth field, which is a date field.

 Owner

OwnerID	OwnerName	ContactDetails	Grade	Class	Email	DiscJockey	OnDuty	DateOfBirth
1	Vuyolwetu Ngcaba	0831514512	11	A	ngcabav@mweb.co.za	<input checked="" type="checkbox"/>	6	1990/03/23
2	Peter Donovan	0825569851	12	C	peter07@absamail.co.za	<input checked="" type="checkbox"/>	4	1989/10/02
3	Mickey Curry	0123072008	12	C	mickeyc@webmail.co.za	<input type="checkbox"/>	0	1989/06/04
4	Kevin Roets	0725648765	10	B	roetsk@polka.net	<input checked="" type="checkbox"/>	2	1991/02/09

When using a Date/Time data type, you refer to the values with a hash symbol (#) on either side of the data type.

- Remember to include the year, month and day when referring to a date.
Can NOT have DateOfBirth = 1990 wrong>

Examples used below will refer to queries using the *DateOfBirth* field from the Owner table.

Examples	Description
SELECT * FROM Owner WHERE DateOfBirth = #1990/01/01#	Display all the owners that were born on the 1 January 1990.
WHERE DateOfBirth >= #2000/01/01#	Display the owners that were born in 2000 or later
WHERE DateOfBirth >= #2001/01/01# AND DateOfBirth <= #2002/12/31#	Display the owners that were born in 2001 or 2002.
WHERE DateOfBirth BETWEEN #2001/01/01# AND #2001/06/30#	Display the owners that were born in the first 6 months of 2001.
WHERE DateOfBirth LIKE "2000/03/*"	Displays the owners that were born in March of 2000.

NOTE: When using the LIKE operator, you do NOT need the hash symbols. We can use double quotes however you must use the format of the computer you are using. In other words, this computer uses a / to separate the year, month and day. Other computers may use a dash (-). When using LIKE you must use the correct one for that computer.

- FORMAT function with date fields

Examples	Description
SELECT OwnerName, Grade, Class, FORMAT (DateOfBirth , "dd-mmm-yyyy") AS Birthdate	Displays the date of birth like 01-Jan-1990

- Calculations with date fields
 - Subtracting TWO dates returns the number of DAYS between the TWO dates.
 - A Date plus or minus an integer will return that date plus/minus the integer's value in days.
 Example: Date field + 30 = That date but in 30 days' time.
 Date field – 60 = That date but 60 days ago.

- **DATE()** function returns the current date, set on the computer.

Examples	Description
SELECT OwnerName, Grade, Class, DATE() + 7 AS ReturnDate	Displays the <i>Return Date</i> as one week from TODAY.
SELECT #2021/01/01# - DATE() AS DaysTilNewYear	Displays the number of days from NOW till New Year 2021 (<i>This is if today is a day in 2020</i>).

- **DATEVALUE** function converts a string parameter into a date format.

Examples	Description
SELECT OwnerName FROM Owner WHERE DateOfBirth >= DATEVALUE ("2001/01/01")	Converts string "2001/01/01" into a date format that can now be used in a calculation.

- DAY, MONTH and YEAR function returns the numerical value of day, month or year for a date respectively.
 - Parameter is a date field.

Examples	Description
SELECT OwnerName, Grade, Class, DAY (DateOfBirth) AS Day_Of_BirthDate	Displays the day (number value) that the owner was born.
SELECT OwnerName, YEAR (DATE()) – YEAR (DateOfBirth) AS Age	Displays the age that the owner is turning that year (<i>Year value of today – year value of the owners birth date: Example 2020 – 2001 = 19</i>).
SELECT OwnerName, MONTH (DateOfBirth) AS BirthMonth FROM Owner WHERE MONTH (DateOfBirth) = 6	Displays the month (number value) that the owner was born but only for the owners born in the month of June (Month is 6)

Queries using multiple tables

When multiple tables are linked by a common field (normally the primary key of one table is used as a foreign key in another table) you can query information from both tables as long as you take note of the following:

- All the tables used must be mentioned in the FROM clause separated by commas.
- You must specify which fields link to each other in the WHERE clause.
- If you have fields with the same name in both tables, you refer to a field by using the table name <dot> field name. Example *CD.OwnerID*

For the following examples, we will use the CD and Owner table. The common field is the *OwnerID* field in the Owner table (primary key) links to the *OwnerID* field in the CD table (foreign key).

Example (<i>Description below</i>)
SELECT Artist, CD_name, OwnerName, ContactDetails FROM Owner, CD WHERE Owner.OwnerID = CD.OwnerID AND Artist = "Muse"
<i>Display the CD details and Owner details of all the Muse CD's. (Can see the CD name and artist and who owns that CD as well as what their contact number is).</i>
NOTE THE FOLLOWING: <ul style="list-style-type: none"> • The Artist and CD_name fields came from the CD table, the OwnerName and ContactDetails from the Owner table. • Both tables are listed in the FROM clause. • The common link is listed (in bold) in the WHERE clause together with any other criteria. • Because both tables have an OwnerID field, you refer to the fields by their table name <dot> field name.

Example (Description below)

```
SELECT CD.Artist, CD.CD_name, Owner.OwnerName, Owner.ContactDetails
FROM Owner, CD
WHERE Owner.OwnerID = CD.OwnerID AND CD.Artist = "Muse"
```

You can refer to each field in the format: table name <dot> field name (see bold).

Example (Description below)

```
SELECT B.Artist, B.CD_name, A.OwnerName, A.ContactDetails
FROM Owner AS A, CD AS B
WHERE A.OwnerID = B.OwnerID AND B.Artist = "Muse"
```

You can use the AS operator to give the tables a short name to refer to, which then minimises typing when referring to the fields with the table name <dot> field name format.

Subqueries

When you want to use a value in a criteria for a query that is based on another query's result from an aggregate function, you can use that second query as a subquery in the main query.

Consider the following query:

Examples	Description
SELECT AVG(ReplacementValue) FROM CD	Displays the average replacement value of ALL the CD's. (Display one value)

Now we can use that query as a subquery in another query:

Example (Description below)

```
SELECT Artist, CD_name, Genre, ReplacementValue
FROM CD
WHERE ReplacementValue > ( SELECT AVG( ReplacementValue )
                           FROM CD )
```

*Display the CD details of all the CD's that have a replacement value that is **above the average replacement value**.*

Example (Description below)

```
SELECT OwnerName, Grade, Class
FROM Owner
WHERE Class = ( SELECT Class FROM Owner WHERE OwnerID = 7 )
```

*Display the Owner details of all the Owner's that are in **the same class as Owner number 7**.*

SELECT queries in Delphi

When running an SQL query in Delphi, you make use of the **ADOQuery** component instead of the **ADOTable** component.

- **ADOQuery** connects to an **ADOConnection** or uses a connection string exactly like an **ADOTable**.
- Do not need to specify a table name as the **ADOQuery** can interact with all tables based on its SQL statement.
- You can view the results of the query in a **DBGrid** as long as the relevant **Data Source** component connects to the ADOQuery (similar to if it connects to the **ADOTable**).

The SQL statement is written in the **SQL** string property of the **ADOQuery**:

- This can either be done in one line using the **Text** sub property
Example: `AdoQuery1.SQL.Text := 'SELECT * FROM CD';`
- Or it can be used like a memo control where you add multiple lines:
Example: `AdoQuery1.Clear;`
`AdoQuery1.SQL.Add('SELECT *');`
`AdoQuery1.SQL.Add('FROM CD');`

To execute the SQL statement you can:

- Call the **Open** method of the **ADOQuery**
Example: `AdoQuery1.Open;`
- or you set the **Active** property to **TRUE**
Example: `AdoQuery1.Active := TRUE;`
If using this method, then set the **Active** property to **FALSE** first, then edit the SQL property, then set the **Active** property to **TRUE**.

Example

```
qryCDDatabase.SQL.Clear ;  
qryCDDatabase.SQL.Add ( 'SELECT CD_Name, Artist, ReplacementValue FROM CD' );  
qryCDDatabase.SQL.Add ( 'WHERE Genre = "Rock" ORDER BY Artist ASC ' );  
qryCDDatabase.Open ;  
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' );
```

*In the example above the ADOQuery component is called qryCDDatabase.
RecordCount property returns the number of records in the query.*

or

Example

```
qryCDDatabase.Active := FALSE ;  
qryCDDatabase.SQL.Text := 'SELECT CD_Name, Artist, ReplacementValue FROM CD  
WHERE Genre = "Rock" ORDER BY Artist ASC ' ;  
qryCDDatabase.Active := TRUE ;  
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' );
```

You can use data from Delphi components or variables in your SQL queries.

- The SQL property is a string field.
- All your text fields in the database must be contained in double quotes (") and all the date fields must be contained in hash symbols (#).

Example

```
iValue := 150 ;           //integer variable
```

```
qryCDDatabase.SQL.Clear ;
qryCDDatabase.SQL.Add ( 'SELECT CD_Name, Artist, ReplacementValue FROM CD' ) ;
qryCDDatabase.SQL.Add ( 'WHERE Replacement Value >= ' + IntToStr( iValue ) ) ;
qryCDDatabase.Open ;
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' ) ;
```

Added the iValue integer value to the query but converted it to a string because the SQL property requires a string value.

Example

```
sName := edtInput.Text ;   //string variable
```

```
qryCDDatabase.SQL.Clear ;
qryCDDatabase.SQL.Add ( 'SELECT CD_Name, Artist, ReplacementValue FROM CD' ) ;
qryCDDatabase.SQL.Add ( 'WHERE Artist = "' + sName + '"' ) ;
qryCDDatabase.Open ;
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' ) ;
```

Added the sName string value to the query so no converting is needed, however the string variable needs to be contained in double quotes, hence the double quotes before the first string is concluded, then the string variable followed by the ending double quotes as a string.

‘WHERE Artist = " + sName + "'

*If the use of double quotes is confusing then you can make use of the **QuotedStr** function:*

‘WHERE Artist = ' + **QuotedStr**(sName) ;

Special Rule: When using the LIKE operator in Delphi

- Use the % symbol in place of the * (star symbol)
- Use the _ symbol (underscore) in place of the ?

Example

```
SELECT * FROM CD WHERE Genre LIKE "**Rock**"
```

```
qryCDDatabase.SQL.Clear ;
qryCDDatabase.SQL.Add ( 'SELECT * FROM CD WHERE Genre LIKE "%Rock%" ' ) ;
qryCDDatabase.Open ;
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' ) ;
```

Example

```
SELECT * FROM CD WHERE Artist LIKE "M???"
```

```
qryCDDatabase.SQL.Clear ;  
qryCDDatabase.SQL.Add ( 'SELECT * FROM CD WHERE Artist LIKE "M_ _ _"' );  
qryCDDatabase.Open ;  
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' );
```

Example

```
sInput := edtInput.Text ; //string variable
```

```
qryCDDatabase.SQL.Clear ;  
qryCDDatabase.SQL.Add ( 'SELECT * FROM CD WHERE Genre LIKE "%' + sInput + '%"') ;  
qryCDDatabase.Open ;  
showmessage( IntToStr( qryCDDatabase.RecordCount ) + ' records' );
```

Note: The % symbol is inside the string quotes when using a variable.

'WHERE Genre LIKE "%' + sInput + '%"'

*Or you can make use of the **QuotedStr** function:*

'WHERE Genre LIKE ' + **QuotedStr**('%' + sInput + '%') ;

INSERT statements

INSERT statement is used to add a new record in one of the tables in a database.

```
INSERT INTO <table name>  
( <field names of the table separated by commas> )  
VALUES  
( <values to go into the corresponding fields separated by commas> )
```

NOTE: Values must follow the rules for their data type: text to be contained in double quotes and dates to be contained in hash symbols. Numbers and Booleans can be used as is.

Example (Description below)

```
INSERT INTO Owner (OwnerID, OwnerName, Grade, Class, DiscJockey, DateOfBirth )  
VALUES ( 55, "Mister Long", 12, "B", TRUE, #2000/11/21#)
```

Add the following details into the corresponding fields as a new record in the Owner table.

NOTE THE FOLLOWING:

- The values are spaced out to show you that each value must correspond with the same field name in the correct order.
- Not all fields have been added to the table (ContactDetails and Email have been left out). You don't need to insert a value for every field except for required field.
- An error will occur if there is already an OwnerID of 55 in the Owner table because OwnerID is the primary key and it MUST BE UNIQUE and NOT BLANK.
- Note how the number and Boolean fields (OwnerID, Grade, DiscJockey) are listed as is, the text fields (OwnerName, Class) have double quotes and the date field (DateOfBirth) has hash symbols.

UPDATE statements

UPDATE statement is used to make changes to values in a record or records that currently exist in a database table.

```
UPDATE <table name>  
SET <field name> = <new value>  
WHERE <criteria>
```

NOTE THE FOLLOWING:

- Only records that match the criteria will be updated.
- If no WHERE clause is used, then all the records in the table will be updated.
- If you want to update multiple fields, then use commas to separate the different <field name> = <new value> parts.

Examples	Description
UPDATE Owner SET Email = "mrlong@gmail.com" WHERE OwnerID = 10	Change the e-mail address of the record with an Owner ID of 10 in the Owner table.
UPDATE CD SET ReplacementValue = ReplacementValue+15 WHERE Genre = "Rock"	Increase the replacement value by 15 of all the Rock CD's in the CD table.
UPDATE Owner SET Grade = 12, Class = "A" WHERE Grade = 11 AND DiscJockey = TRUE	Change all the Disc Jockeys in Grade 11 to Grade 12 class A in the Owner table.

DELETE statements

DELETE statement is used to remove records from a table in a database.

DELETE FROM <table name>
WHERE <criteria>

NOTE THE FOLLOWING:

- Only records that match the criteria will be deleted.
- If no WHERE clause is used, then all the records in the table will be deleted.
- *HINT: This can not be undone so make a backup copy of your database in case you unintentionally delete records you didn't want to.*

Examples	Description
DELETE FROM Owner WHERE OwnerID = 10	Delete record where Owner ID is 10 in the Owner table.
DELETE FROM CD WHERE Genre = "Rock"	Delete all the Rock CD's from the CD table.
DELETE FROM CD WHERE ReplacementValue > 200 AND Artist LIKE "B*"	Delete all the CD's that cost over R200 to replace and where the artist starts with the letter "B" in the CD table.
DELETE FROM CD	Delete ALL the records in the CD table.

INSERT, UPDATE & DELETE in Delphi

INSERT, UPDATE and DELETE statements in Delphi also make use of the **ADOQuery** component as described in the **SELECT queries in Delphi** section.

- The SQL statement will still be written in the SQL string property of the **ADOQuery** component using the **Text** sub property OR **Add** method.
- To execute the SQL is different.
 - You do not use the Active or Open methods to run an INSERT, UPDATE or DELETE statement as no result set is returned.
 - Simply call the **ExecSQL** method after you have added your SQL to the SQL property.

Example

```
qryCDDatabase.SQL.Clear ;  
qryCDDatabase.SQL.Add ( 'DELETE FROM CD WHERE Genre = "Rock" ' ) ;  
qryCDDatabase.ExecSQL ;  
//Once DELETE is complete, then run a SELECT query so user can observe the results of the  
//delete  
qryCDDatabase.SQL.Clear ;  
qryCDDatabase.SQL.Add ( 'SELECT * FROM CD ' ) ;  
qryCDDatabase.SQL.Open ;
```

You can use data from Delphi components or variables in your SQL queries.

- The SQL property is a string field.
- All your text fields in the database must be contained in double quotes (") and all the date fields must be contained in hash symbols (#).
- Remember the special rule: When using the LIKE operator in Delphi
 - Use the % symbol in place of the * (star symbol)
 - Use the _ symbol (underscore) in place of the ?

Example

```
iValue := StrToInt( InputBox( 'ID', 'Enter your Owner ID', '10' ) ); //integer var for Owner ID  
sNewEmail := InputBox( 'E-Mail', 'Enter NEW e-mail:', '' ); //string var for new email  
  
qryCDDatabase.SQL.Clear ;  
qryCDDatabase.SQL.Add ( 'UPDATE Owner SET Email = " + sNewEmail + "' ) ;  
qryCDDatabase.SQL.Add ( 'WHERE OwnerID = ' + IntToStr( iValue ) ) ;  
qryCDDatabase.ExecSQL ;
```

- Added the *iValue* integer value to the query but converted it to a string because the SQL property requires a string value.
- Added the *sNewEmail* string value to the query so no converting is needed, however the string variable needs to be contained in double quotes, hence the double quotes before the first string is concluded, then the string variable followed by the ending double quotes as a string.

Example

```
sOwnID := edtOwnerID.Text ; //ALL are string variables
sName := edtOwnerName.Text ;
sGrade := edtGrade.Text ;
sClass := edtClass.Text ;
sDJockey := edtDiscJockey.Text ;
sDOB := edtDOB.Text ;

qryCDDatabase.SQL.Clear ;
qryCDDatabase.SQL.Add ( 'INSERT INTO Owner ( OwnerID, OwnerName, Grade, Class, ' ) ;
qryCDDatabase.SQL.Add ( DiscJockey, DateOfBirth ) VALUES ( ' +sOwnID+ ' , "" +sName+ "" , ' ) ;
qryCDDatabase.SQL.Add ( sGrade + ' , "" +sClass + "" , ' +sDJockey+ ' , # +sDOB+ '#' ) ;
qryCDDatabase.ExecSQL ;
```

NOTE:

- Inputs are stored in string values because the SQL property is a string. If you had variables of other types, they would need to be converted to string (e.g. IntToStr, BoolToStr, etc)
- **String commas** are added between each field in the VALUES brackets.
- Each variable for number fields (sOwnID & sGrade) are added as is.
- Variable for Boolean field (sDJockey) is added as is.
- Each variable for string fields (sName & sClass) are added with **string double quotes**.
- Variable for date field (sDOB) is added with **string hash symbols**.

sGrade + ' , "" +sClass + "" , ' +sDJockey+ ' , # ' +sDOB+ '#')'

If the use of double quotes is confusing then you can make use of the **QuotedStr** function:

sGrade+ ' , ' +QuotedStr(sClass) + ' , ' +sDJockey+ ' , #' +sDOB+ '#')'

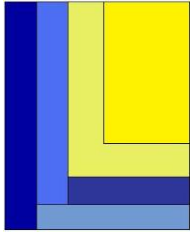
SUGGESTION: If you get an error saying your SQL statement is incorrect, then use the following line of code before the ExecSQL / Open / Active := TRUE statement:

showmessage(qryCDDatabase.SQL.Text) ;

It will display your SQL statement and you can analyse it to see if it is correct or if you used double quotes, commas, brackets, spaces etc correctly.

Additional Links:

- Youtube video playlist:
<https://www.youtube.com/watch?v=ZgNdtFp4yME&list=PLxAS51iVMjv9OrVc4wlZM0sFyorFg7ZBI>
- Google drive resource activities:
<https://tinyurl.com/MLE-G12IT-SQL>



For more IT related material find us on:



youtube.com/user/MrLongEducation



facebook.com/MrLongEducation



@MrLongEdu

SUBSCRIBE