SQL (Structured Query Language)

***Paradigm****- Declarative*

***Family****- Query Language*

***Designed by****- Donald D. Chamberlin & Raymond F. Boyce*

**SQL** is a **standard query language** for **storing, manipulating,** and **retrieving** **data in databases**.

**SQL became** a **standard** of the **American National Standards Institute (ANSI) in 1986,** and of the **International Organization for Standardization (ISO) in 1987.**

SQL **keywords** are **NOT case sensitive**: select is the same as SELECT.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

The data returned is stored in a **result table**, called the **result-set.**

RDBMS

RDBMS stands for **Relational Database Management System**, **invented** by **Edgar Frank Codd.**

**RDBMS is the basis for SQL**, **and for all modern database systems** such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

**The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries, and it consists of columns and rows.**

Every table is broken up into smaller entities called fields. A field is a column in a table that is designed to maintain specific information about every record in the table.

A record, also called a row, is each individual entry that exists in a table. A record is a horizontal entity in a table.

Field 1 Field 2 Field 3 Field 4

|  |  |  |  |
| --- | --- | --- | --- |
| Record 1 |  |  |  |
| Record 2 |  |  |  |
| Record 3 |  |  |  |

Table

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

***Note****: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!*

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

SELECT Statement

The SELECT statement is used to select data from a database.

SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

WHERE Clause

The WHERE clause is used to filter records.

It is used to extract only those records that fulfil a specified condition.

The following operators can be used in the WHERE clause:

|  |  |
| --- | --- |
| Operator | Description |
| = | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| <> | Not equal. Note: In some versions of SQL this operator may be written as != |
| BETWEEN | Between a certain range |
| LIKE | Search for a pattern |
| IN | To specify multiple possible values for a column |

AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if ***all*** the conditions separated by AND are TRUE.
* The OR operator displays a record if ***any*** of the conditions separated by OR is TRUE.
* The NOT operator displays a record if the condition(s) is NOT TRUE.

SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records **in ascending order by default**. To sort the records in descending order, use the DESC keyword.

The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and customer with same country name are sorted descending by the "CustomerName" column:

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

*INSERT INTO Syntax*

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

**Note:** If you omit the WHERE clause, all records in the table will be updated!

DELETE Statement

The DELETE statement is used to delete existing records in a table.

DELETE FROM table\_name WHERE condition;

**Note:** If you omit the WHERE clause, all records in the table will be deleted!

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM *table\_name*;

## SELECT TOP Clause

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

***Note:****Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses FETCH FIRST*n*ROWS ONLY and ROWNUM.*

## MIN(), MAX(), COUNT(), AVG() and SUM() Functions

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

The COUNT() function returns the number of rows that matches a specified criterion.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

***Note:****NULL values are not counted and get ignored.*

## LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

* The percent sign (%) represents zero, one, or multiple characters
* The underscore sign (\_) represents one, single character

***Note:****MS Access uses an asterisk (\*) instead of the percent sign (%), and a question mark (?) instead of the underscore (\_).*

## Wildcard Characters

A wildcard character is used to substitute one or more characters in a string.



Wildcard characters are used with the [LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

All the wildcards can also be used in combinations!

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| LIKE Operator | Description |
| WHERE CustomerName LIKE 'a%' | Finds any values that starts with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that ends with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that starts with "a" and are at least 3 characters in  length |
| WHERE ContactName LIKE 'a%o' | Finds any values that starts with "a" and ends with "o" |

## IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

## BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

Dates

#07/01/1996# | '1996-07-01'

## Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

Aliases can be useful when:

* There are more than one table involved in a query
* Functions are used in the query
* Column names are big or not very readable
* Two or more columns are combined together

## Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

      

## UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

* Every SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in every SELECT statement must also be in the same order

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

## GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

## HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

## EXISTS Operator

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records

## ANY Operator

* returns a boolean value as a result
* returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for *any* of the values in the range

## ALL Operator

* returns a boolean value as a result
* returns TRUE if ALL of the subquery values meet the condition
* is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for *all* values in the range.

## SELECT INTO Statement

The SELECT INTO statement copies data from one table into a new table.

**Tip:** SELECT INTO can also be used to create a new, empty table using the schema of another. Just add a WHERE clause that causes the query to return no data:

SELECT \* INTO *newtable*  
FROM *oldtable*  
WHERE 1 = 0;

## INSERT INTO SELECT Statement

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

The INSERT INTO SELECT statement requires that the data types in source and target tables match.

**Note:** The existing records in the target table are unaffected.

## CASE Statement

The CASE statement goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

## What is a Stored Procedure?

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

### **Stored Procedure Syntax**

CREATE PROCEDURE procedure\_name  
AS  
sql\_statement  
GO;

### **Execute a Stored Procedure**

EXEC procedure\_name;

## Comments

Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.

Single line comments start with --.

Multi-line comments start with /\* and end with \*/.

Queries are generally processed in the follow order

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. HAVING
7. SELECT
8. ORDER BY
9. LIMIT | TOP