SQL

Structured **Q**uery **L**anguage

SQL is a Standard - BUT....

SQL is an ANSI (American National Standards Institute) standard computer language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a database. SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, etc.

Unfortunately, there are many different versions of the SQL language, but to be in compliance with the ANSI standard, they must support the same major keywords in a similar manner (such as SELECT, UPDATE, DELETE, INSERT, WHERE, and others).

Note: Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard!

SQL Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.
Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

SQL Queries

With SQL, we can query a database and have a result set returned.
A query like this:

SELECT LastName FROM Persons

Gives a result set like this:

LastName
Hansen
Svendson
Pettersen

SQL Data Manipulation Language (DML)

SQL (Structured Query Language) is a syntax for executing queries. But the SQL language also includes a syntax to update, insert, and delete records.
These query and update commands together form the Data Manipulation Language (DML) part of SQL:

- **SELECT** - extracts data from a database table
- **UPDATE** - updates data in a database table
- **DELETE** - deletes data from a database table
- **INSERT INTO** - inserts new data into a database table

SQL Data Definition Language (DDL)

The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted. We can also define indexes (keys), specify links between tables, and impose constraints between database tables.

The most important DDL statements in SQL are:

- **CREATE TABLE** - creates a new database table
- **ALTER TABLE** - alters (changes) a database table
- **DROP TABLE** - deletes a database table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

SQL The SELECT Statement

The SELECT statement is used to select data from a table. The tabular result is stored in a result table (called the result-set).

Syntax

SELECT column_name(s)
FROM table_name

To select the columns named "LastName" and "FirstName", use a SELECT statement like this:

```
SELECT LastName, FirstName FROM Persons
```

Persons

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

výsledok

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

Select All Columns

To select all columns from the "Persons" table, use a * symbol instead of column names, like this:

```
SELECT * FROM Persons
```

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

The Result Set

The result from a SQL query is stored in a result-set. Most database software systems allow navigation of the result set with programming functions, like: Move-To-First-Record, Get-Record-Content, Move-To-Next-Record, etc.

Programming functions like these are not a part of this tutorial. To learn about accessing data with function calls, please visit our ADO tutorial.

Semicolon after SQL Statements?

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.

Some SQL tutorials end each SQL statement with a semicolon. Is this necessary? We are using MS Access and SQL Server 2000 and we do not have to put a semicolon after each SQL statement, but some database programs force you to use it.

The SELECT DISTINCT Statement

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

The SELECT statement returns information from table columns. But what if we only want to select distinct elements?

With SQL, all we need to do is to add a DISTINCT keyword to the SELECT statement:

Syntax

SELECT DISTINCT column_name(s)

FROM table_name

Using the DISTINCT keyword

To select ALL values from the column named "Company" we use a SELECT statement like this:

```
SELECT Company FROM Orders
```

Orders

Company	OrderNumber
Sega	3412
W3Schools	2312
Trio	4678
W3Schools	6798

Company
Sega
W3Schools
Trio
W3Schools

Note that "W3Schools" is listed twice in the result-set.

To select only DIFFERENT values from the column named "Company" we use a SELECT DISTINCT statement like this:

SELECT **DISTINCT** Company FROM Orders

Orders

Company	OrderNumber
Sega	3412
W3Schools	2312
Trio	4678
W3Schools	6798

Company
Sega
W3Schools
Trio

Select All Columns

The WHERE clause is used to specify a selection criterion.

The WHERE Clause

To conditionally select data from a table, a WHERE clause can be added to the SELECT statement.

Syntax

SELECT column FROM table

WHERE column operator value

With the WHERE clause, the following operators can be used:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern

Note: In some versions of SQL the <> operator may be written as !=

Using the WHERE Clause

To select only the persons living in the city "Sandnes", we add a WHERE clause to the SELECT statement:

```
SELECT * FROM Persons
WHERE City='Sandnes'
```

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951
Svendson	Tove	Borgvn 23	Sandnes	1978
Svendson	Stale	Kaivn 18	Sandnes	1980
Pettersen	Kari	Storgt 20	Stavanger	1960

LastName	FirstName	Address	City	Year
Hansen	Ola	Timoteivn 10	Sandnes	1951
Svendson	Tove	Borgvn 23	Sandnes	1978
Svendson	Stale	Kaivn 18	Sandnes	1980

Using Quotes

Note that we have used single quotes around the conditional values in the examples.

SQL uses single quotes around text values (most database systems will also accept double quotes). Numeric values should not be enclosed in quotes.

For text values:

This is correct:

```
SELECT * FROM Persons WHERE FirstName='Tove'
```

This is wrong:

```
SELECT * FROM Persons WHERE FirstName=Tove
```

The LIKE Condition

The LIKE condition is used to specify a search for a pattern in a column.

Syntax

SELECT column FROM table

WHERE column LIKE pattern

A "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

Using LIKE

The following SQL statement will return persons with first names that start with an 'O':

```
SELECT * FROM Persons
WHERE FirstName LIKE 'O%'
```

The following SQL statement will return persons with first names that end with an 'a':

```
SELECT * FROM Persons
WHERE FirstName LIKE '%a'
```

Using LIKE 2

The following SQL statement will return persons with first names that contain the pattern 'la':

```
SELECT * FROM Persons

WHERE FirstName LIKE '%la%'
```

SQL The INSERT INTO Statement

The INSERT INTO Statement

The INSERT INTO statement is used to insert new rows into a table.

Syntax

INSERT INTO table_name

VALUES (value1, value2,....)

You can also specify the columns for which you want to insert data:

INSERT INTO table_name (column1, column2,...)

VALUES (value1, value2,....)

Insert a New Row

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger

And this SQL statement:

```
INSERT INTO Persons
VALUES ('Hetland', 'Camilla', 'Hagabakka 24', 'Sandnes')
```

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

Insert Data in Specified Columns

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes

And This SQL statement:

```
INSERT INTO Persons (LastName, Address)
VALUES ('Rasmussen', 'Storgt 67')
```

LastName	FirstName	Address	City
Pettersen	Kari	Storgt 20	Stavanger
Hetland	Camilla	Hagabakka 24	Sandnes
Rasmussen		Storgt 67	

SQL The UPDATE Statement

The Update Statement

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

Syntax

```
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value
```

Update one Column in a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen		Storgt 67	

We want to add a first name to the person with a last name of "Rasmussen":

```
UPDATE Person SET FirstName = 'Nina'
WHERE LastName = 'Rasmussen'
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Storgt 67	

Update several Columns in a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen		Storgt 67	

We want to change the address and add the name of the city:

```
UPDATE Person
SET Address = 'Stien 12', City = 'Stavanger'
WHERE LastName = 'Rasmussen'
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

SQL The Delete Statement

The Delete Statement

The DELETE statement is used to delete rows in a table.

Syntax

```
DELETE FROM table_name
WHERE column_name = some_value
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

Delete a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger
Rasmussen	Nina	Stien 12	Stavanger

"Nina Rasmussen" is going to be deleted:

```
DELETE FROM Person WHERE LastName = 'Rasmussen'
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	Stavanger

Delete All Rows

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

Or

DELETE * FROM table_name

Miscellaneous Commands

- `show databases;`
  - Show all the databases on the server
- `show tables;`
  - Show all the tables of the present database
- `show columns from table EMPLOYEE;`
- `drop table t_name;`
  - Delete the entire table *t_name*
- `drop database db_name;`
  - Delete the entire database *db_name*
- `load data infile f_name into table t_name;`
  - To be discussed with the next homework.

DCL (Data Control Language) Commands

- The rights or permissions assigned to user(s) to use some or all of Oracle objects are known as privileges.

- **Granting Privileges:** It is used to assigning permissions to users. (Only DBA can assign)
- **Syntax:** grant <permissions> '*select, insert, delete, update*
- on <object_name>
- to <username>;

- **Eg.** grant insert on emp to user1; '*only user1 can insert*
- grant all on emp to public; '*assign all permissions to all users.*

- **Revoking Privileges:** It get back permissions from the users.

- **Syntax:** revoke <permission> on <object_name> from <username>;

- **Examples**
  revoke all on emp from user1; '*get back all permissions from user1*.

  revoke select on emp from public; '*get back select permission from all users.*

TCL

- **(<u>Transaction Control Language</u>):** It controls over transaction processing by specifying the beginning and ending of transactions.
- **Eg.** Commit, Rollback, Rollback to, Save point etc.

TCL Commands

- Oracle treat a transaction as a single entity & incase of successful termination of transaction the changes are made permanent. The commands used with transactions are:

- **COMMIT:** It ends the current transaction by saving database changes & starts a new transaction.
- **Eg.** commit; *'i.e end or start a transaction*

- **ROLLBACK:** It ends the current transaction by discarding database changes & starts a new transaction.
- **Eg.** rollback; *'i.e undo upto commit*

- **SAVEPOINT:** It defines breakpoints or bookmarks for the transaction to allow partial rollbacks.
- **Eg.** savepoint P1;

- **ROLLBACK TO:** Its undo up to given bookmark or breakpoint.
  Eg. rollback to P1;