SQL

Dr. Parimala M

SCORE, VIT

DML STATEMENT

1. 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	

insert into emp values(&eno, &ename ,
&dno);
enter the value of eno: 10
enter the value ename: a
enter the value dno: 2
/
enter the value eno:

2. 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 emp set sal=sal+100
Where empno=107;
```

2. 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

3. The Delete Statement

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

Syntax

DELETE FROM table_name

WHERE column_name = some_value

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

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).

Select * from <tablename>;

Select * from employee;

Syntax- to display single column as a result

SELECT column_name(s)
FROM table_name
Select empno, ename from emp;

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.

You should not use DISTINCT keyword twice in the query.

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

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

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 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
```

And /or

- Select city from weather where humidity>60 and humidity<80 or temp>60;

- Select * from weather where temp!=80;

Order of clause

- Select city from weather order by temp;

(by default it is ascending order otherwise we can put asc.)

Select city,temp from weather order by temp desc;

Select * from weather order by humidity,temp desc;

Select * from weather where humidity>70 order by temp;

Order of clause

select--------→from----------→where-------→order by

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

The following SQL statement will return persons with first names that contain atleast 3 characters

```
SELECT * FROM Persons
WHERE FirstName LIKE '___%';
```

The following SQL statement will return persons with first names that contain third character to be 's'

```
SELECT * FROM Persons
WHERE FirstName LIKE '__s%';
```

mismatching

- To find the mismatching characters we will use, not like operator.
- Select * from weather where city not like ‘%ad%’;
- We can use like operator to compare with numbers.
- Select * from weather where temp like 50;

Null / not null

- Find the name of employees who do not get commission.

- Select ename from emp where comm is null;

****
****

- Find the name of employees who get commission.

- Select ename from emp where comm is not null;

****
******

- Find the name of president.

- Select ename from emp where mgr is null