DATABASE SYSTEMS

Dr. Noha Nagy

Lecture 6

SQL[DML]

SQL
Structured Query Language

- Data Definition Language (DDL)
  - Define relational *schemata*
  - Create/Alter/Drop tables and their attributes
- Data Manipulation Language (DML)
  - Insert/Delete/Update tuples in tables
  - Query one or more table
- Data Control Language (DCL)
  - Specify user permissions
  - Grant/revoke

DML

Insert Statment

Employee

Enum	Ename	phone	Pnum
123	Ahmed	01110025878	111
124	Ali	01225929785	
127	Ola	0102457896	111

Insert into Employee values (128, ‘Mahmoud’, 01113005581, 326);
Insert into Employee (Enum,Ename, Pnum)values (130, ‘Eyad’ , 327);

Employee

Enum	Ename	phone	Pnum
123	Ahmed	01110025878	111
124	Ali	01225929785	
127	Ola	0102457896	111
128	Mahmoud	01113005581	326
130	Eyad		327

Update Statment

Product

Pnum	Pname	Price	Quantity
123	Arial	200	20
124	Persil	180	50
127	OXI	100	11
128	Tide	150	32

Update Product Set Price=price*2

Product

Pnum	Pname	Price	Quantity
123	Arial	400	20
124	Persil	360	50
127	OXI	200	11
128	Tide	300	32

Update Statment

Product

Pnum	Pname	Price	Quantity
123	Arial	200	20
124	Persil	180	50
127	OXI	100	11
128	Tide	150	32

Update Product Set Quantity= Quantity – 1 Where Pnum= 123

Product

Pnum	Pname	Price	Quantity
123	Arial	400	19
124	Persil	360	50
127	OXI	200	11
128	Tide	300	32

Delete Statment

Employee

Enum	Ename	phone	Pnum
123	Ahmed	01110025878	111
124	Ali	01225929785	254
127	Ola	0102457896	111

```
Delete From Employee
Where Pnum = 254;
```

Employee

Enum	Ename	phone	Pnum
123	Ahmed	01110025878	111
127	Ola	0102457896	111

Truncate

Employee

Enum	Ename	phone
123	Ahmed	01110025878
124	Ali	01225929785
127	Ola	0102457896

Delete all data in the table
No where condition
Reset the identity

```
Truncate table Employee;
```

Employee

Enum	Ename	phone

SQL SYNTAX

- Basic form

```
SELECT <attributes>
FROM   <one or more relations>
WHERE  <conditions>
```

Call this a **SFW** query.

```
SELECT <Column list>
FROM <table names>
[WHERE <Condition>]
[GROUP BY <Column list>]
[HAVING <Condition>]
[ORDER BY <Column list>]
```

Retrieve All Columns and All Rows

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

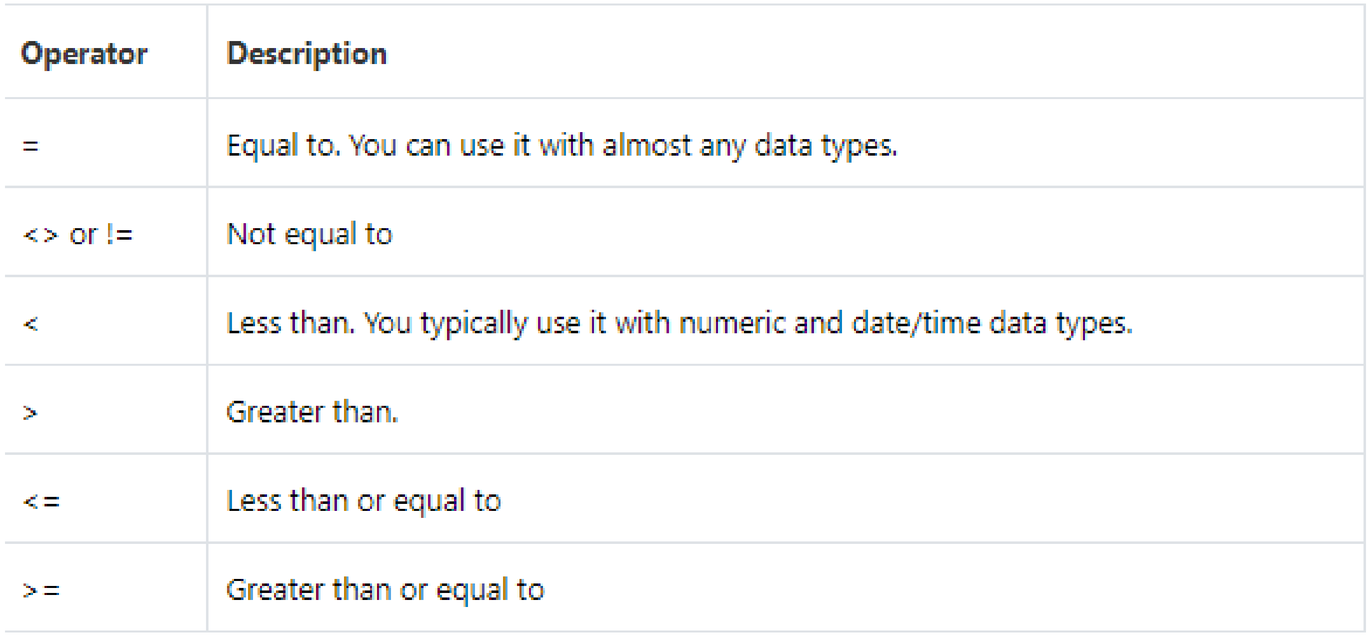
```
SELECT   Pname, Price, Category, Manufacturer
FROM     Product
```

OR

```
SELECT   *
FROM     Product
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

SQL WHERE Clause - Operators



Operator	Description
=	Equal to. You can use it with almost any data types.
<> or !=	Not equal to
<	Less than. You typically use it with numeric and date/time data types.
>	Greater than.
<=	Less than or equal to
>=	Greater than or equal to

Retrieve Specific Columns

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   PName, Price
FROM     Product
```

PName	Price
Gizmo	$19.99
Powergizmo	$29.99
SingleTouch	$149.99
MultiTouch	$203.99

“projection”

Retrieve Specific Rows

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   *
FROM     Product
WHERE    category='Gadgets'
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

"selection"

Retrieve Specific Columns and Rows

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100
```

“selection” and “projection”

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

Conditions with AND

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100 and Manufacturer= ‘Canon’
```

PName	Price	Manufacturer
SingleTouch	$149.99	Canon

Conditions with OR

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	Samsung
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Price > 100 OR Manufacturer= ‘Samsung’
```

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi
Gizmo	$19.99	Samsung

LIKE: Simple String Pattern Matching

```
SELECT *
FROM   Products
WHERE  PName LIKE '%gizmo%'
```

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
  - % = zero, one, or multiple characters
  - _ = any single character

LIKE Cont,

Employee

Enum	Ename	phone
123	Ahmed	01110025878
124	Ali	01225929785
127	Ola	0102457896

- selects all Employees with a Name that start with “A“

```
SELECT *

FROM Employee
WHERE Ename LIKE ’a%';
```

Enum	Ename	phone
123	Ahmed	01110025878
124	Ali	01225929785

- selects all Employees with a Name that does NOT start with “A"

```
SELECT *

FROM Employee
WHERE Ename NOTLIKE ’a%';
```

Enum	Ename	phone
127	Ola	0102457896

ORDER BY: Sorting the Results

- The column specified in the ORDER BY clause does not need to be included in the SELECT clause
- Null values are ordered as the lowest value

Employee

Enum	Ename	Salary	Gender
123	Ahmed	10000	M
124	Ali	5000	M
127	Ola	30000	F

```
SELECT   Ename, Salary
FROM     Employee
WHERE    gender='M'
ORDER BY Salary DESC
```

Ename	Salary
Ahmed	10000
Ali	5000

```
SELECT   Ename
FROM     Employee
ORDER BY Salary DES
```

Ename
Ola
Ahmed
Ali

ORDER BY Cont

- Order by several columns

```
SELECT   Lname, Fname, Salary
FROM     Employee
WHERE    Sex='F'
ORDER BY Fname, Lname
```

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Category='gizmo' AND Price > 50
ORDER BY Price ASC, Pname DESC
```

SELECT : with ALIAS

Employee

Enum	Ename	phone
123	Ahmed	01110025878
124	Ali	01225929785
127	Ola	0102457896

- Alias is an alternative column or table name

```
SELECT Ename as Name, Enum as Employee ID
FROM Employee
```

Name	Employee ID
Ahmed	123
Ali	124
Ola	127

```
SELECT Cust.Customer_Name as Name, Cust.Customer_address
FROM Customer Cust
WHERE Customer_Name= 'Home Furnishings';
```

Comparisons Involving NULL

- SQL allows queries that check whether an attribute value is `NULL`
  - `IS NULL` or `IS NOT NULL`

Retrieve the names of all employees who don't have supervisors.

```
Select Fname,Lname
From Employee
Where Super_ssn is null;
```

Employee

Fname	Lname	ID	Super_ssn
Ahmed	Fahmy	111	113
Ali	Zidan	112	114
Mark	Antony	113	114
Amr	Moussa	114	Null

Fname	Lname
Amr	Moussa

Constants and Arithmetic

- As well as column names, you can select constants, compute arithmetic expressions and evaluate functions in a **`SELECT`** statement

```
SELECT Name,Code,Mark/100
  FROM Grades
```

```
SELECT 1.175*Price
  FROM Products
```

Grades

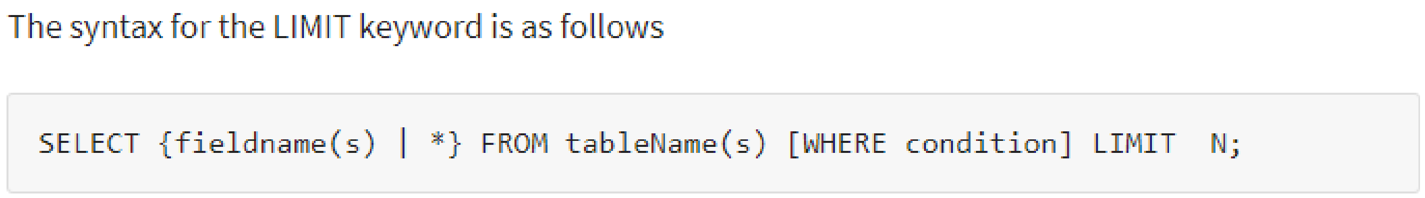
Name	Code	Mark
John	DBS	56
John	IAI	72
Mary	DBS	60
Mark	PR1	43
Mark	PR2	35
Jane	IAI	54

Grades

Name	Code	Mark
John	DBS	0.56
John	IAI	0.72
Mary	DBS	0.60
Mark	PR1	0.43
Mark	PR2	0.35
Jane	IAI	0.54

LIMIT Keyword

- The limit keyword is used to limit the number of rows returned in a query result.


The syntax for the LIMIT keyword is as follows

```
SELECT {fieldname(s) | *} FROM tableName(s) [WHERE condition] LIMIT  N;
```

•**"LIMIT N"** is the keyword and **N** is any number starting from 0, putting 0 as the limit does not return any records in the query. Putting a number say 5 will return five records. If the records in the specified table are less than N, then all the records from the queried table are returned in the result set.

LIMIT Example

- Do you have any employee with last names “Fadi” ?
- select * from employees where lastName=‘Fadi’ limit 1;

Aggregate Functions

- Min
- Max
- Count
- Avg
- Sum

Products

PID	Pname	Price	Qty	SupplierID
1	Apple	20	200	22
2	Banana	10	100	10
3	Orange	4	400	6

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

```
SmallestPrice
4
```

```
SELECT MAX(Price) AS HighestPrice
FROM Products;
```

```
HighestPrice
20
```

```
SELECT COUNT(PID) AS Count
FROM Products;
```

```
Count
3
```

```
SELECT AVG(Price) AS AveragePrice
FROM Products;
```

```
AveragePrice
11.33
```

Using Aggregate Function

- Using the COUNT ***aggregate function*** to find totals
- Find number of customers who live in Rome

```
SELECT COUNT(*)
FROM Customer
WHERE City = ‘Rome’
```

Categorizing Results

- For use with aggregate functions
  - ***Scalar aggregate***: single value returned from SQL query with aggregate function
  - ***Vector aggregate***: multiple values returned from SQL query with aggregate function (via GROUP BY)

Customer

Fname	Lname	ID	City
Ahmed	Fahmy	111	Cairo
Ali	Zidan	112	Cairo
Mark	Antony	113	Cairo
Amr	Moussa	114	Giza

```
SELECT City, Count(City)
FROM Customer
GROUP BY City
```

```
SELECT City, Count(City)
FROM Customer
WHERE City='Cairo'
```

What is the Difference between them?

Group by

SQL has a **GROUP BY**-clause for specifying the grouping attributes, which *must also appear in the SELECT-clause*

For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
SELECT DNO, COUNT (*), AVG (SALARY)
        FROM EMPLOYEE
        GROUP BY DNO
```

```
1   2   1000
2   2   5500
```

Employee

Name	DNO	ID	Salary
Ahmed	1	111	1000
Ali	1	112	1000
Mark	2	113	5000
Amr	2	114	6000

Example

Purchase

Product	Date	Price	Quantity
Apple	10/21	1	20
Apple	10/25	1.50	20
Banana	10/3	0.5	10
Banana	10/10	1	10

Product	TotalSales
Apple	50
Banana	15

```
SELECT      product as Product, Sum(price*quantity) AS TotalSales
FROM        Purchase
WHERE       date > ‘10/1/2005’
GROUP BY    product
```

GROUP BY

Grades

Name	**Code**	**Mark**
John	DBS	56
John	IAI	72
Mary	DBS	60
Mark	PR1	43
Mark	PR2	35
Jane	IAI	54

```
SELECT Name,
   AVG(Mark) AS Average
   FROM Grades
   GROUP BY Name
```

Name	Average
John	64
Mary	60
Mark	39
Jane	54

Calculate the average marks of each Student

GROUP BY

Sales

Month	Department	Value
March	Fiction	20
March	Travel	30
March	Technical	40
April	Fiction	10
April	Fiction	30
April	Travel	25
April	Fiction	20
May	Fiction	20
May	Technical	50

- Find the total value of the sales for each department in each month
  - Can group by Month then Department or Department then Month
  - Same results, but in a different order

GROUP BY

```
SELECT Month, Department,
   SUM(Value) AS Total
  FROM Sales
 GROUP BY Month, Department
```

Month	Department	Total
April	Fiction	60
April	Travel	25
March	Fiction	20
March	Technical	40
March	Travel	30
May	Fiction	20
May	Technical	50

```
SELECT Month, Department,
   SUM(Value) AS Total
  FROM Sales
 GROUP BY Department, Month
```

Month	Department	Total
April	Fiction	60
March	Fiction	20
May	Fiction	20
March	Technical	40
May	Technical	50
April	Travel	25
March	Travel	30

Qualifying Results by Categories Using the HAVING Clause

- Return all Order IDs that include more than 3 products in their OrderLines.

Orders

OrderID	ProductID	Quantity
100	1	10
100	2	17
102	2	2
100	5	9
103	3	3
103	4	4
103	5	5
103	6	6

```
SELECT OrderID, Count(ProductID)
FROM Orders
GROUP BY OrderID
HAVING Count(productID) > 3;
```

Like a WHERE clause, but it operates on groups (categories), not on individual rows. Here, only those groups with total numbers greater than 3 will be included in final result. **HAVING is considered a SECOND WHERE.**