



7BUIS030W

Data System Concepts and Fundamentals

Lecture -7



Lecture-6 Outline

Physical database design and implementation, DML, SQL select, retrieve and sort the tables and the contents

Physical model

- 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 column is a vertical entity in a table that contains all information associated with a specific field in a table.

DML

➤ **Data Manipulation Language (DML)** is for retrieving and updating data.

| | |
|---|----------------------------------|
| SELECT INSERT UPDATE DELETE MERGE | Data Manipulation Language (DML) |
|---|----------------------------------|

Table Structure- Employees and departments

| Table Name | Null? | Data Type |
|----------------|----------|--------------|
| Employee_ID | NOT_NULL | INT(6) |
| First_Name | | VARCHAR(20) |
| Last_Name | NOT_NULL | VARCHAR(20) |
| Email | NOT_NULL | VARCHAR(25) |
| Phone_Number | | VARCHAR(20) |
| Hire_Date | NOT_NULL | DATE |
| Job_ID | NOT_NULL | VARCHAR(10) |
| SALARY | | DECIMAL(8,2) |
| Commission_pct | | DECIMAL(3,2) |
| Manager_ID | | INT(6) |
| Department_ID | | INT(6) |

SQL SELECT statement- Example

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

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

SELECT Syntax

```
SELECT column1, column2, ...  
FROM table_name;
```

Here, *column1, column2, ...* are the field names of the table you want to select data from.

SQL SELECT statement

If you want to select all the fields available in the table, use the following syntax:

SELECT Syntax

SELECT *

FROM *table_name*;

SQL SELECT statement

SELECT *
From departments;

| Department_ID | Department_Name | Manager_ID | Location_ID |
|---------------|-----------------|------------|-------------|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 50 | Shipping | 124 | 1500 |
| 60 | IT | 103 | 1400 |
| 80 | Sales | 149 | 2500 |
| 90 | Executive | 100 | 1700 |
| 110 | Accounting | 205 | 1700 |
| 190 | Contracting | | 1700 |

SQL SELECT statement

```
SELECT department_id, location_id  
From departments;
```

| Department_ID | Location_ID |
|---------------|-------------|
| 10 | 1700 |
| 20 | 1800 |
| 50 | 1500 |
| 60 | 1400 |
| 80 | 2500 |
| 90 | 1700 |
| 110 | 1700 |
| 190 | 1700 |

SQL 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.

SQL SELECT DISTINCT statement

```
SELECT department_id  
FROM departments;
```

| DEPARTMENT_ID |
|---------------|
| 90 |
| 90 |
| 90 |
| 10 |
| 20 |
| 50 |

```
SELECT DISTINCT department_id  
FROM departments;
```

| DEPARTMENT_ID |
|---------------|
| 90 |
| 10 |
| 20 |
| 50 |

SQL SELECT ALIASES

SQL 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 the query.

Alias Column Syntax

```
SELECT column_name AS alias_name  
FROM table_name;
```

Alias Table Syntax

- SELECT *column_name(s)*
FROM *table_name* AS *alias_name*;

SQL SELECT ALIASES

```
SELECT department_id as dept, location_id as Loc  
From departments;
```

| DEPT | LOC |
|------|------|
| 10 | 1700 |
| 20 | 1800 |
| 50 | 1500 |
| 60 | 1400 |
| 80 | 2500 |
| 90 | 1700 |
| 110 | 1700 |
| 190 | 1700 |

SQL SELECT ALIASES

```
SELECT department_id as “Dept”, location_id as “Loc”  
From departments;
```

| Dept | Loc |
|------|------|
| 10 | 1700 |
| 20 | 1800 |
| 50 | 1500 |
| 60 | 1400 |
| 80 | 2500 |
| 90 | 1700 |
| 110 | 1700 |
| 190 | 1700 |



The SQL WHERE Clause

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

WHERE Syntax

```
SELECT column1, column2,  
FROM table_name  
WHERE condition;
```

SQL WHERE clause

```
SELECT employee_id, last_name, job_id, department_id  
FROM employees  
WHERE department_id=90;
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | DEPARTMENT_ID |
|-------------|-----------|---------|---------------|
| 100 | King | AD_PRES | 90 |
| 101 | Giles | AD_VP | 90 |
| 102 | West | AD_VP | 90 |

The SQL WHERE Clause

Text Fields vs. Numeric Fields

- Numeric fields in SQL should not be enclosed in quotes as shown in the previous example
- SQL requires single quotes around text values (most database systems will also allow double quotes).

```
SELECT employee_id, last_name, job_id, department_id
FROM employees
WHERE last_name='King';
```

The SQL WHERE Clause

Operators in The WHERE Clause

| Operator | Meaning |
|----------------------|-----------------------------------|
| = | Equal to |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| <> | Not equal to |
| BETWEEN ...AND... | Between two values (inclusive) |
| IN (set) | Match any of a list of values |
| LIKE | Match a character pattern |
| IS NULL | Is a null value |

The SQL WHERE Clause

Operators in The WHERE Clause-Examples

```
SELECT *  
FROM Products  
WHERE Price = 18;
```

```
SELECT *  
FROM Products  
WHERE Price > 30;
```

```
SELECT *  
FROM Products  
WHERE Price <= 30;
```

The SQL WHERE Clause

Operators in The WHERE Clause-Examples

```
SELECT *
```

```
FROM Products
```

```
WHERE Price <> 18;
```

```
SELECT *
```

```
FROM Products
```

```
WHERE Price BETWEEN 50 AND 60;
```

```
SELECT *
```

```
FROM Customers
```

```
WHERE City LIKE 's%';
```

```
SELECT *
```

```
FROM Customers
```

```
WHERE City IN ('Paris','London');
```

The SQL WHERE Clause

Operators in The WHERE Clause-Examples

```
SELECT *
```

```
FROM Products
```

```
WHERE Price <> 18;
```

```
SELECT *
```

```
FROM Products
```

```
WHERE Price BETWEEN 50 AND 60;
```

```
SELECT *
```

```
FROM Customers
```

```
WHERE City LIKE 's%';
```

```
SELECT *
```

```
FROM Customers
```

```
WHERE City IN ('Paris','London');
```

SQL WHERE clause

```
SELECT last_name, salary  
FROM employees  
WHERE Salary<=3000;
```

| LAST_NAME | SALARY |
|-----------|--------|
| Smith | 2600 |
| Jenkins | 2900 |
| Gomez | 1800 |
| Silva | 3000 |

SQL WHERE clause- Using the between condition

- 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.

LOWER LIMIT

UPPER LIMIT

```
SELECT last_name, salary
FROM employees
WHERE Salary BETWEEN 2500 AND 3500
```

| LAST_NAME | SALARY |
|-----------|--------|
| Lowe | 3500 |
| Smith | 2600 |
| Jenkins | 2900 |
| Hussein | 2500 |

SQL WHERE clause- Using the IN condition

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

```
SELECT employee_id, last_name, salary, manager_id  
FROM employees  
WHERE manager_id IN (100, 101, 201)
```

| EMPLOYEE_ID | LAST_NAME | SALARY | MANAGER_ID |
|-------------|-----------|--------|------------|
| 202 | Stuart | 6000 | 201 |
| 200 | Smith | 2600 | 101 |
| 205 | Ahmed | 13000 | 101 |
| 102 | Hussein | 2500 | 100 |
| 101 | Lowe | 17000 | 100 |
| 124 | Hickens | 9000 | 201 |
| 149 | Jenkins | 2900 | 201 |

SQL WHERE clause- Using the 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 represents a single character
- The percent sign and the underscore can also be used in combinations!

SQL WHERE clause- Using the LIKE operator

```
SELECT first_name  
FROM employees  
WHERE first_name LIKE 'S%';
```

| FIRST_NAME |
|------------|
| Stephen |
| Sarah |
| Sukie |
| Sajid |

SQL WHERE clause- Using the LIKE operator

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE '_a%';
```

| LAST_NAME |
|-----------|
| Sarah |
| Sajid |
| Martin |
| Daniel |
| Barbara |



SQL NULL VALUES

- A field with a NULL value is a field with no value.
- 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!
- It is not possible to test for NULL values with comparison operators, such as =, <, or <>.
- Will have to use the IS NULL and IS NOT NULL operators instead.



SQL NULL VALUES

IS NULL Syntax

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column_names  
FROM table_name  
WHERE column_name IS NOT NULL;
```

SQL NULL VALUES

```
SELECT last_name,manager_id  
FROM employees  
WHERE manager_id IS NULL;
```

| LAST_NAME | MANAGER_ID |
|-----------|------------|
| King | |
| Sarah | |



SQL Logical Conditions

The SQL 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 Logical Conditions

The SQL AND, OR and NOT Operators

| Operator | Meaning |
|----------|---|
| AND | Returns TRUE if <i>both</i> component conditions are true |
| OR | Returns TRUE if <i>either</i> component condition is true |
| NOT | Returns TRUE if the following condition is false |



SQL Logical Conditions

The SQL AND, OR and NOT Operators

AND Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...  
FROM table_name  
WHERE NOT condition;
```

SQL Logical Conditions

The SQL AND

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
AND job_id LIKE '%MAN%';
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|-------------|-----------|--------|--------|
| 205 | Ahmed | SA_MAN | 13000 |
| 101 | Lowe | MK_MAN | 17000 |

SQL Logical Conditions

The SQL OR

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary > 10000
OR job_id LIKE '%MAN%';
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|-------------|-----------|--------|--------|
| 101 | Lowe | MK_MAN | 17000 |
| 124 | Hickens | AC_MAN | 9000 |
| 205 | Ahmed | SA_MAN | 13000 |
| 101 | Giles | AD_VP | 15000 |

SQL Logical Conditions

The SQL NOT

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE job_id NOT IN ('IT_PROG', 'SA_MAN', 'AD_VP');
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | SALARY |
|-------------|-----------|---------|--------|
| 202 | Stuart | AD_MAN | 6000 |
| 200 | Smith | AD_ASST | 2600 |
| 101 | Lowe | MK_MAN | 17000 |
| 124 | Hickens | AC_MAN | 9000 |

SQL Logical Conditions

Combining AND, OR and NOT Operators

RULES of PRECEDENCE

| Operator | Priority |
|----------|----------|
| NOT | 1 |
| AND | 2 |
| OR | 3 |

Parenthesis can be used to override rules of precedence

SQL Logical Conditions

The SQL AND, OR and NOT Operators

```
SELECT last_name, job_id, salary  
FROM employees  
WHERE job_id= 'SA_REP'  
OR job_id='AD_PRES'  
AND salary>15000;
```

| LAST_NAME | JOB_ID | SALARY |
|-----------|---------|--------|
| Parker | AD_PRES | 16000 |
| Silva | SA_REP | 11000 |
| Caryer | SA_REP | 7000 |
| Horton | SA_REP | 24000 |

SQL Logical Conditions

The SQL AND, OR and NOT Operators

```
SELECT last_name, job_id, salary  
FROM employees  
WHERE (job_id= 'SA_REP'  
OR job_id='AD_PRES')  
AND salary>15000;
```

| LAST_NAME | JOB_ID | SALARY |
|-----------|---------|--------|
| Parker | AD_PRES | 16000 |
| Horton | SA_REP | 24000 |

SQL Logical Conditions

The SQL AND, OR and NOT Operators

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id= 'SA_REP'
OR job_id='AD_PRES')
AND salary >15000
AND NOT salary = 16000;
```

| LAST_NAME | JOB_ID | SALARY |
|-----------|--------|--------|
| Horton | SA_REP | 24000 |



SQL ORDER BY conditions

- 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.

SQL ORDER BY conditions

ORDER BY Syntax

```
SELECT column1, column2, ...  
FROM table_name  
ORDER BY column1, column2,  
... ASC|DESC;
```

The ORDER BY statement come last is in the SELECT statement

SQL ORDER BY conditions

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date;
```

| LAST_NAME | JOB_ID | DEPARTMENT_ID | HIRE_DATE |
|-----------|--------|----------------|-------------|
| Low | MK_MAN | Marketing | 17-JUN-2009 |
| Hickens | AC_MAN | Accounting | 5-SEPT-2011 |
| Ahmed | SA_MAN | Sales | 9-MAR-2012 |
| Giles | AD_VP | Administration | 25-MAY-2012 |

SQL ORDER BY

SQL ORDER BY descending order

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date DESC;
```

| LAST_NAME | JOB_ID | DEPARTMENT_ID | HIRE_DATE |
|-----------|--------|----------------|-------------|
| Giles | AD_VP | Administration | 25-MAY-2012 |
| Ahmed | SA_MAN | Sales | 9-MAR-2012 |
| Hickens | AC_MAN | Accounting | 5-SEPT-2011 |
| Lowe | MK_MAN | Marketing | 17-JUN-2009 |

SQL ORDER BY

SQL ORDER BY multiple columns

```
SELECT last_name, job_id, department_id, hire_date  
FROM employees  
ORDER BY hire_date , department_id;
```

This means that it orders by `hire_date` , but if some rows have the same `hire_date`, it orders them by `department_id`:

| LAST_NAME | JOB_ID | DEPARTMENT_ID | HIRE_DATE |
|-----------|--------|----------------|-------------|
| Lowe | MK_MAN | Marketing | 17-JUN-2009 |
| Hickens | AC_MAN | Accounting | 5-SEPT-2011 |
| Turner | SA_REP | Sales | 5-SEPT-2011 |
| Ahmed | SA_MAN | Sales | 9-MAR-2012 |
| Giles | AD_VP | Administration | 25-MAY-2012 |