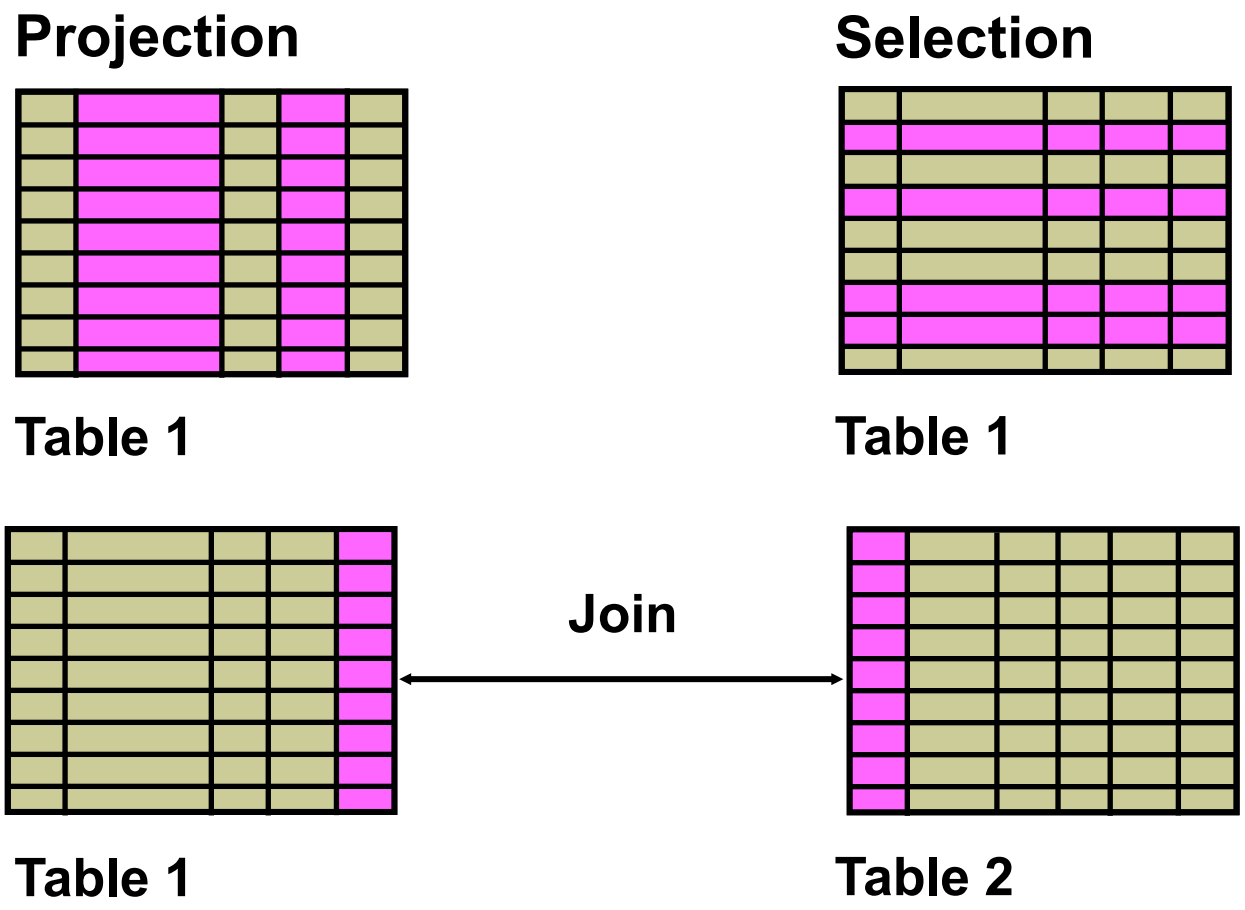
7BDIN006W
Big Data Theory and Practice

Lecture 3

Big Data and Relational Model. Retrieving Data Using SQL

UNIVERSITY OF WESTMINSTER

Capabilities of SQL `SELECT` Statements



Basic SELECT Statement

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM    table;
```

- **SELECT identifies the columns to be displayed.**
- **FROM identifies the table containing those columns.**

Selecting All Columns

```
SELECT *
FROM   dept;
```

Selecting Specific Columns

```
SELECT deptno,loc
FROM   dept;
```

Writing SQL Statements

- SQL statements are not case sensitive.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.

Arithmetic Expressions

Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

```
SELECT ename, sal, sal + 300
FROM   emp;
```

Operator Precedence

```
SELECT ename, sal, 12*sal+100
FROM   emp;
```

1

```
SELECT ename, sal, 12*(sal+100)
FROM   emp;
```

2

Defining a Null Value

- A null is a value that is unavailable, unassigned, unknown, or inapplicable.
- A null is not the same as a zero or a blank space.

```
SELECT ename, job, sal, comm
FROM   emp;
```

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

```
SELECT ename, 12*sal * comm
FROM   emp;
```

Defining a Column Alias

A column alias:

- Renames a column heading
- Is useful with calculations
- Immediately follows the column name (There can also be the optional `AS` keyword between the column name and alias.)
- Requires double quotation marks if it contains spaces or special characters or if it is case sensitive

Using Column Aliases

```
SELECT ename AS last_name, comm commission
FROM   emp;
```

```
SELECT ename AS “Last Name”, sal*12 "Annual Salary"
FROM   emp;
```

Concatenation Operator

A concatenation operator:

- Links columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a resultant column that is a character expression

```
SELECT   ename||job AS "Employees"
FROM     emp;
```

Literal Character Strings

- A literal is a character, a number, or a date that is included in the `SELECT` statement.
- Date and character literal values must be enclosed by single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings

```
SELECT ename ||' is a `|| job
       AS "Employee Details"
FROM   emp;
```

Alternative Quote (q) Operator

- Specify your own quotation mark delimiter
- Choose any delimiter
- Increase readability and usability

```
SELECT ename ||
       q'[ this employee's assigned Manager Id is ]'
       || mgr
       AS "Employee and Manager"
FROM emp;
```

Duplicate Rows

The default display of queries is all rows, including duplicate rows.

```
SELECT deptno
FROM   emp;
```

1

```
SELECT DISTINCT deptno
FROM   emp;
```

2

Restricting and Sorting Data

Limiting the Rows That Are Selected

- Restrict the rows that are returned by using the `WHERE` clause:

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM   table
[WHERE condition(s)];
```

- The `WHERE` clause follows the `FROM` clause.

Using the WHERE Clause

```
SELECT empno, ename, job, deptno
FROM   emp
WHERE  deptno = 20 ;
```

Character Strings and Dates

- Character strings and date values are enclosed in single quotation marks.
- Character values are case sensitive, and date values are format sensitive.
- The default date format is DD-MON-RR.

```
SELECT ename, job, deptno
FROM   emp
WHERE  ename = 'JONES' ;
```

Comparison Conditions

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

Using Comparison Conditions

```
SELECT ename, sal
FROM   emp
WHERE  sal <= 3000 ;
```

Using the BETWEEN Condition

Use the BETWEEN condition to display rows based on a range of values:

```
SELECT ename, sal
FROM   emp
WHERE  sal BETWEEN 2500 AND 3500 ;
```

Lower limit Upper limit

Using the IN Condition

Use the IN membership condition to test for values in a list:

```
SELECT empno, ename, sal, mgr
FROM   emp
WHERE  deptno (10, 20, 50) ;
```

8 rows selected.

Using the LIKE Condition

- Use the LIKE condition to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
  - % denotes zero or many characters.
  - _ denotes one character.

```
SELECT    ename
FROM      emp
WHERE     ename LIKE 'S%' ;
```

Using the LIKE Condition

- You can combine pattern-matching characters:

```
SELECT ename
FROM   emp
WHERE  ename LIKE '_L%' ;
```

- You can use the ESCAPE identifier to search for the actual % and _ symbols.

Using the NULL Conditions

Test for nulls with the IS NULL operator.

```
SELECT ename, mgr
FROM   emp
WHERE  mgr IS NULL ;
```

Logical Conditions

Operator	Meaning
`AND`	Returns `TRUE` if *both* component conditions are true
`OR`	Returns `TRUE` if *either* component condition is true
`NOT`	Returns `TRUE` if the following condition is false

Using the AND Operator

AND requires both conditions to be true:

```
SELECT empno, ename, job, sal
FROM   emp
WHERE  sal >= 2500
AND    job LIKE '%MAN%' ;
```

AND Truth Table

The following table shows the results of combining two expressions with AND:

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Using the OR Operator

OR requires either condition to be true:

```
SELECT empno, ename, job, sal
FROM   emp
WHERE  sal >= 2500
  OR   job LIKE '%MAN%' ;
```

OR Truth Table

The following table shows the results of combining two expressions with OR:

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Using the NOT Operator

NOT negates the condition:

```
SELECT ename, job
FROM   emp
WHERE  job
       NOT IN (ANALYST', CLERK', 'PRESIDENT') ;
```

NOT Truth Table

The following table shows the results of combining an expression with NOT

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Rules of Precedence

Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	`IS [NOT] NULL`, `LIKE`, `[NOT] IN`
5	`[NOT] BETWEEN`
6	Not equal to
7	`NOT` logical condition
8	`AND` logical condition
9	`OR` logical condition

You can use parentheses to override rules of precedence.

Rules of Precedence

```
SELECT ename, job, sal
FROM   emp
WHERE  job = 'CLERK'
OR     job = 'MANAGER'
AND    sal > 15000;
```

1

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

2