|  |
| --- |
| **C:\Users\faisal\Pictures\NSU_pic_download\n91267046457_2661.jpg**  **CSE 311L(Database Management System)**  **LAB-Week 03 (Part A)** |

Restricting and Sorting Data (Part A is based on Company2 schema)

Topics:

* Limiting the Rows Selected
* Restricting with Character Strings and Dates
* Comparison Conditions
* Other Comparison Conditions,

**Limiting the Rows Selected**

SELECT employee\_id, last\_name, job\_id, department\_id

FROM emps

WHERE department\_id = 90 ;

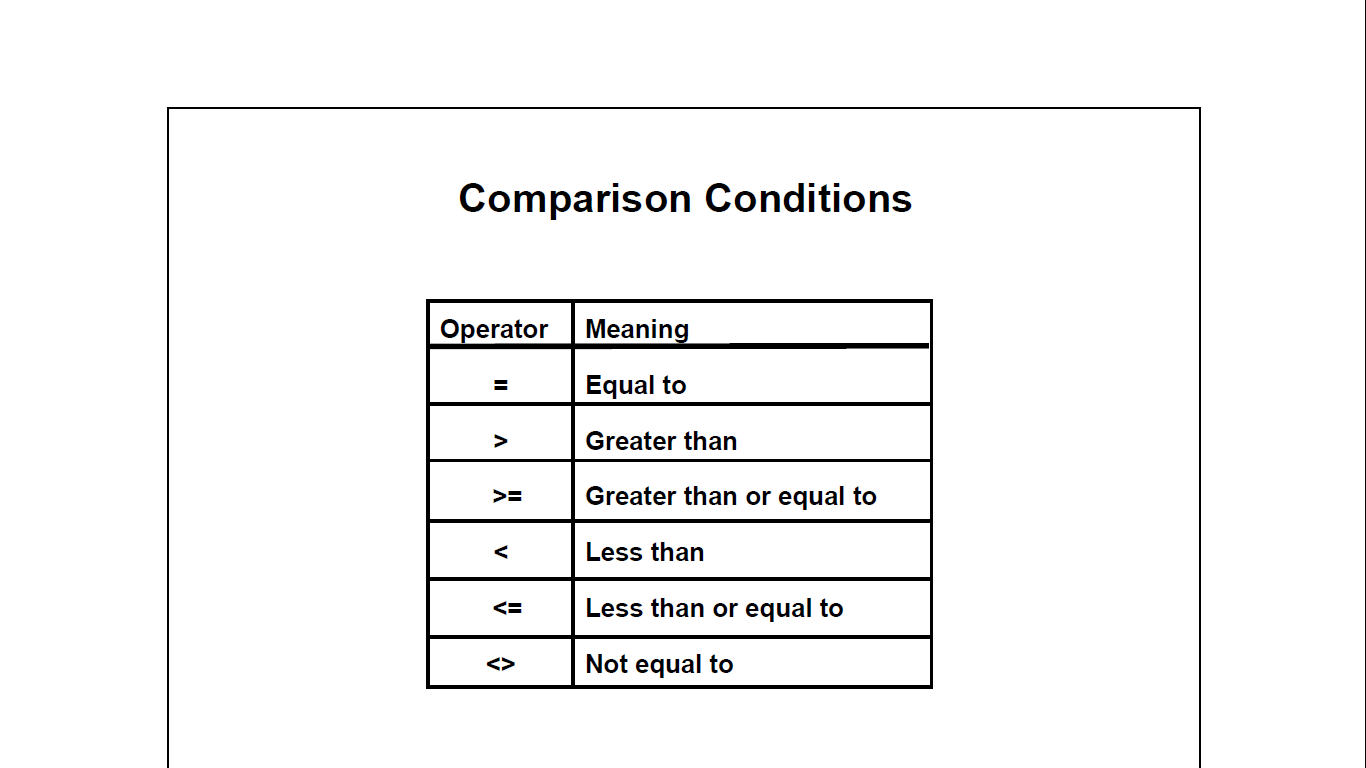
**Character Strings and Dates**

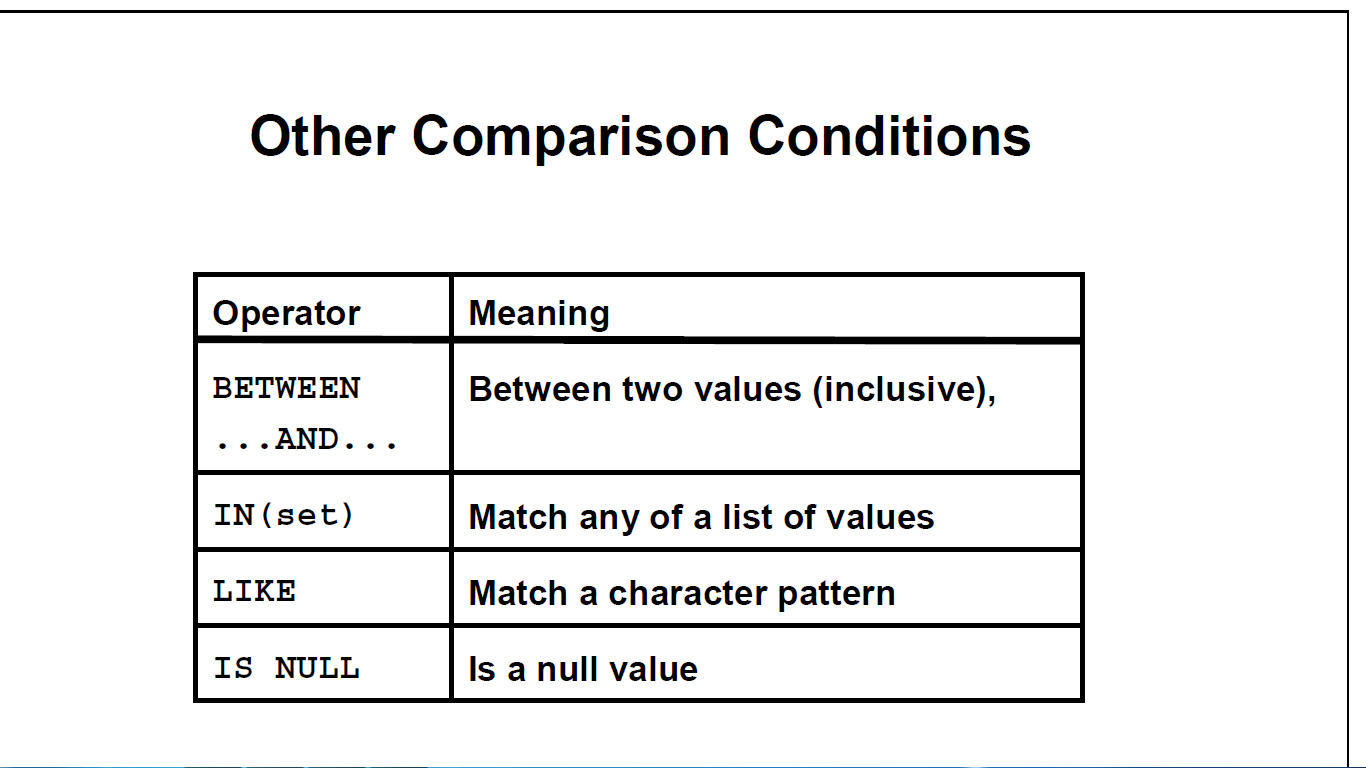
SELECT last\_name, job\_id, department\_id

FROM emps

WHERE last\_name = ’WHALEN’;

**Comparison Conditions**





SELECT last\_name, salary

FROM emps

WHERE salary <= 3000;

**Other Comparison Conditions**

SELECT last\_name, salary

FROM emps

WHERE salary BETWEEN 2500 AND 3500;

SELECT employee\_id, last\_name, salary, manager\_id

FROM emps

WHERE manager\_id IN (100, 101, 201);

**ORDER BY Clause**

SELECT last\_name, job\_id, department\_id, hire\_date

FROM emps

ORDER BY hire\_date DESC ;



**Sorting by Multiple Columns**

SELECT last\_name, department\_id, salary

FROM emps

ORDER BY department\_id, salary DESC;



**Activity 01:**

Display the employee last name, job ID, and start date of employees hired between February 20, 1998, and May 1, 1998. Order the query in ascending order by start date.

**Activity 02:**

Display the last name and department number of all employees in departments 20 and 50 in alphabetical order by name.

|  |
| --- |
| **C:\Users\faisal\Pictures\NSU_pic_download\n91267046457_2661.jpg**  **CSE 311L(Database Management System)**  **LAB-Week 03 (Part B)** |

Topics:

After completing this lesson, you should be able to restrict rows:

* Using the LIKE Condition
* Using the NULL Conditions
* Logical Conditions

**SET OPERATIONS (Based on Company.sql)**

SQL has directly incorporated some set operations such as union operation (UNION), set difference (MINUS) and intersection (INTERSECT) operations. The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result. The set operations apply only to union compatible relations; the two relations must have the same attributes and the attributes must appear in the same order

**Query 5: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.**

Q5: (SELECT PNAME FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')

**UNION**

(SELECT PNAME FROM PROJECT, WORKS\_ON, EMPLOYEE WHERE PNUMBER=PNO AND ESSN=SSN AND NAME='Smith')

**NESTING OF QUERIES**

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query. Many of the previous queries can be specified in an alternative form using nesting

**Query 6: Retrieve the name and address of all employees who work for the 'Research'**

**department.**

Q6: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE WHERE DNO **IN**

(SELECT DNUMBER FROM DEPARTMENT WHERE DNAME='Research' )

**Note:** The nested query selects the number of the 'Research' department. The outer query selectsan EMPLOYEE tuple if its DNO value is in the result of either nested query. The comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V

In general, we can have several levels of nested queries. A reference to an unqualified attribute refers to the relation declared in the innermost nested query. In this example, the nested query is not correlated with the outer query

**CORRELATED NESTED QUERIES**

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated. The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query

**Query 7: Retrieve the name of each employee who has a dependent with the same first**

**name as the employee**.

Q7: SELECT E.FNAME, E.LNAME FROM EMPLOYEE AS E WHERE E.SSN **IN**

(SELECT ESSN FROM DEPENDENT WHERE ESSN=E.SSN AND E.FNAME=DEPENDENT\_NAME)

In Q7, the nested query has a different result in the outer query. A query written with nested SELECT... FROM… WHERE... blocks and using the **= or IN** comparison operators can *always* be expressed as a single block query. For example, Q7 may be written as in Q7a

Q7a: SELECT E.FNAME, E.LNAME FROM EMPLOYEE E, DEPENDENT D WHERE E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT\_NAME

**THE EXISTS FUNCTION**

EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. We can formulate Query 7 in an alternative form that uses EXIST.

Q7b: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE **EXISTS** (SELECT \* FROM DEPENDENT WHERE SSN=ESSN AND FNAME=DEPENDENT\_NAME)

**Query 8: Retrieve the names of employees who have no dependents***.*

Q8: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE **NOT EXISTS**

(SELECT \* FROM DEPENDENT WHERE SSN=ESSN)

**Note:** In Q8, the correlated nested query retrieves all DEPENDENT tuples related to anEMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

**EXPLICIT SETS**

It is also possible to use an explicit (enumerated) set of values in the WHERE-clause rather than a nested query

**Query 9: Retrieve the social security numbers of all employees who work on project**

**number 1, 2, or 3.**

Q9: SELECT DISTINCT ESSN FROM WORKS\_ON WHERE PNO **IN (1, 2, 3)**

**NULLS IN SQL QUERIES**

SQL allows queries that check if a value is NULL (missing or undefined or not applicable). SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

**Query 10: Retrieve the names of all employees who do not have supervisors.**

Q10: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE SUPERSSN IS NULL

**Note:** If a join condition is specified, tuples with NULL values for the join attributes are notincluded in the result

**SUBSTRING COMPARISON**

The LIKE comparison operator is used to compare partial strings. Two reserved characters are used: **'%'** (or '\*' in some implementations) replaces an arbitrary number of characters, and **'\_'** replaces a single arbitrary character.

**Query 18: Retrieve all employees whose address is in Houston, Texas. Here, the value of the**

**ADDRESS attribute must contain the substring 'Houston,TX‘ in it.**

Q18: SELECT FNAME, LNAME

FROM EMPLOYEE WHERE ADDRESS LIKE '%Houston,TX%'

**Query 19: Retrieve all employees who were born during the 1950s.**

Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '\_\_\_\_\_\_\_5\_', with each underscore as a place holder for a single arbitrary character.

Q19: SELECT FNAME, LNAME

FROM EMPLOYEE WHERE BDATE **LIKE** '**\_\_\_\_\_\_\_**5**\_**‘

**Note:** The LIKE operator allows us to get around the fact that each value is considered atomicand indivisible. Hence, in SQL, character string attribute values are not atomic

**Using the LIKE Condition (based on company2.schema)**

* 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 last\_name

FROM emps

WHERE last\_name LIKE ’\_o%’;

**The ESCAPE Option (based on company2.schema)**

SELECT employee\_id, last\_name, job\_id

FROM emps

WHERE job\_id LIKE ’%SA\\_%’ ESCAPE ’\’;



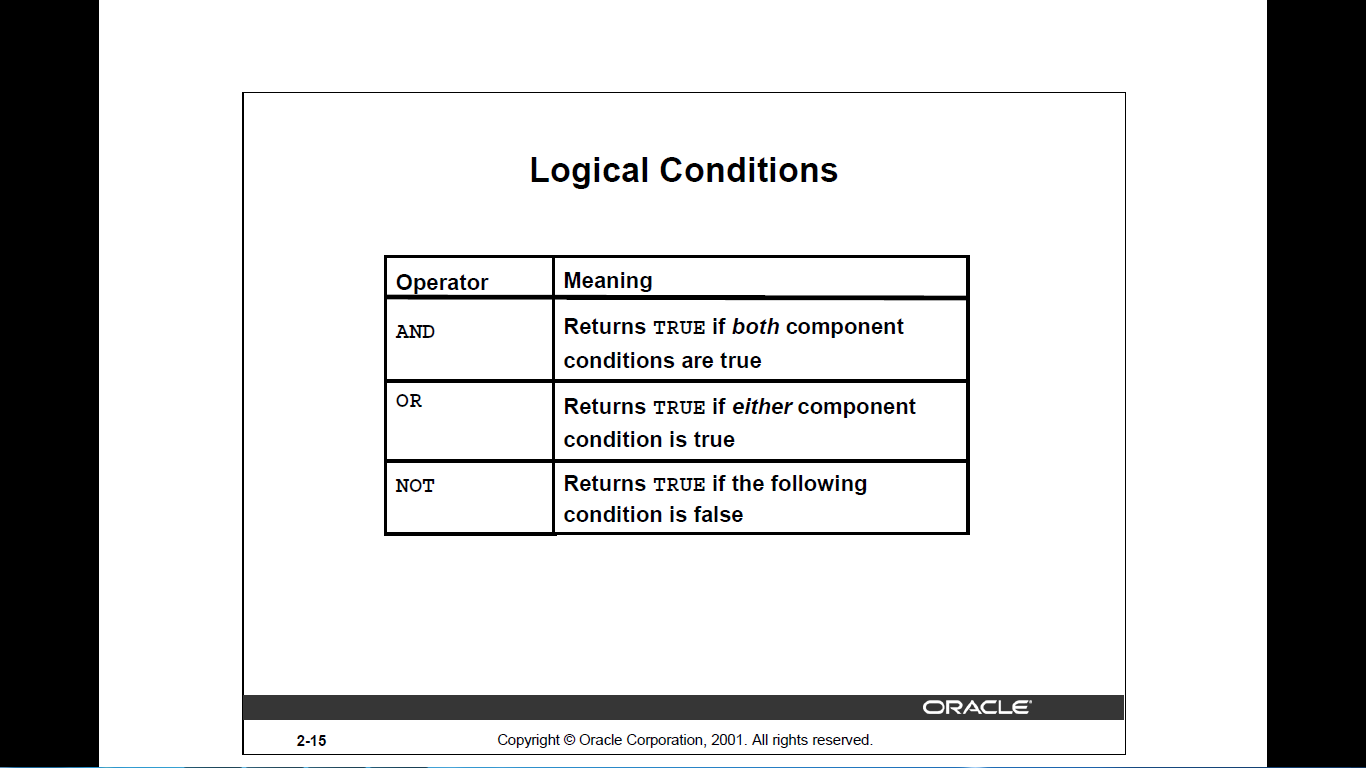
**Using the NULL Conditions (based on company2.schema)**

SELECT last\_name, manager\_id

FROM emps

WHERE manager\_id IS NULL;

**Logical Conditions**

****

SELECT employee\_id, last\_name, job\_id, salary

FROM emps

WHERE salary >=10000

AND job\_id LIKE ’%MAN%’;

****

**Using the NOT Operator**

SELECT last\_name, job\_id

FROM emps

WHERE job\_id

NOT IN (’IT\_PROG’, ’ST\_CLERK’, ’SA\_REP’);



All Activities are based on Company2 Schema

**Activity 01:**

Display the last name and hire date of every employee who was hired in 1994.

**Activity 02:**

Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.Title.

**Activity 03:**

Display the last name of all employees who have an *a* and an *e* in their last name.