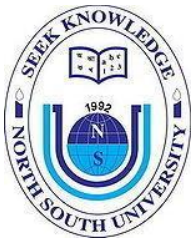

CSE 311L(Database Management System)

LAB-Week 03 (Part A)

Lab Instructor: Marilyn Dip Peris

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

Query1: Retrieve the employee ID, last name, job ID, and department ID for all employees who belong to department 90.

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

Character Strings and Dates

Query2: Retrieve the last name, job ID, and department ID for the employee whose last name is 'WHALEN'.

```
SELECT last_name, job_id, department_id
FROM emps
WHERE last_name = 'WHALEN';
```

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Operator	Meaning
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

Query3: Retrieve the last name and salary for all employees whose salary is less than or equal to 3000.

```
SELECT last_name, salary
FROM emps
WHERE salary <= 3000;
```

Other Comparison Conditions

Query4: Retrieve the last name and salary for all employees whose salary is between 2500 and 3500 (inclusive).

```
SELECT last_name, salary
FROM emps
WHERE salary BETWEEN 2500 AND 3500;
```

Query5: Retrieve the employee ID, last name, salary, and manager ID for all employees whose manager ID is either 100, 101, or 201.

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

ORDER BY Clause

Query6: Retrieve the last name, job ID, department ID, and hire date for all employees, ordered from the most recently hired to the least recently hired.

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

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-98

Sorting by Multiple Columns

```
SELECT last_name, department_id, salary
FROM emps
ORDER BY department_id, salary DESC;
```

LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000
Mourgos	50	5800
Rajs	50	3500
Davies	50	3100
Matos	50	2600
Vargas	50	2500

Activity 01:

Display the employee's last name, job ID, and start date of employees hired between September 17, 2003 and June 17, 2006. Order the query in descending order by start date.

Activity 02:

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

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

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 7: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q7: 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 selects an 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 8: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q8: SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE SSN IN  (SELECT ESSN
        FROM DEPENDENT
        WHERE FNAME=DEPENDENT_NAME)
```

In Q8, 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, Q8 may be written as in Q8a

```
Q8a: 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 8 in an alternative form that uses EXIST.

```
Q8b: SELECT FNAME, LNAME
     FROM EMPLOYEE
     WHERE EXISTS (SELECT *
              FROM DEPENDENT
              WHERE SSN=ESSN AND FNAME=DEPENDENT_NAME)
```

Query 9: Retrieve the names of employees who have no dependents.

```
Q9: SELECT FNAME, LNAME FROM EMPLOYEE
WHERE NOT EXISTS
(SELECT * FROM DEPENDENT WHERE SSN=ESSN)
```

Note: In Q9, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple.

EXPLICIT SETS

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

Query 10: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q10: 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 11: Retrieve the names of all employees who do not have supervisors.

Q11: 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 not included 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 12: Retrieve all employees whose address is in Houston, Texas. Here, the value of the ADDRESS attribute must contain the substring 'Houston TX‘ in it.

Q12: SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE ADDRESS LIKE '%Houston TX%'

Query 13: Retrieve all employees who were born during the 1960s.

Here, '6' must be the 3rd character of the string (according to our format for date), so the BDATE value is '__6_-__-__, with each underscore as a place holder for a single arbitrary character.

Q13: SELECT FNAME, LNAME
FROM EMPLOYEE WHERE BDATE LIKE '__6_-__-__'

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.

Query 14: Retrieve the last name for all employees whose last name has 'o' as the second

letter.

```
Q14: SELECT last_name
FROM emps
WHERE last_name LIKE '_o%';
```

Query 15: Retrieve the employee ID, last name, and job ID for all employees whose job ID starts with the string 'ST'

```
Q15: SELECT employee_id, last_name, job_id
FROM emps
WHERE job_id LIKE 'ST%';
```

Using the NULL Conditions

```
SELECT last_name, manager_id
FROM emps
WHERE manager_id 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

Query16: Retrieve the employee ID, last name, job ID, and salary for all employees who have a salary of at least 10,000 AND whose job ID contains the string 'MAN'.

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Using the NOT Operator

Query17: Retrieve the last name and job ID for all employees whose job ID is NOT one of 'IT_PROG', 'ST_CLERK', or 'SA_REP'.

```
SELECT last_name, job_id
FROM emps
WHERE job_id
```

NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP

All Activities are based on Company2 Schema

Activity 01:
Display the last name and hire date of every employee who was hired in 2008.

Activity 02:
Display the last name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

Activity 03:
Display the last name of all employees who have an *a* and an *e* in their last name.

Activity 04:
Display the employee ID, last name, salary, and department ID for all employees who have a salary between 3000 and 10000 (inclusive) AND who do NOT work in department 80 or 90. Order the result by salary in descending order.

Activity 05:
Display the last name and job ID for all employees whose job ID contains the letters 'SA' anywhere in the string OR whose last name has 'e' as the third letter.