UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FAKULTAS ILMU KOMPUTER

Slide 4

SQL: Advanced Query

CSF2600700 - BASIS DATA

SEMESTER GENAP 2019/2020

These slides are a modification to the supplementary slide of "Database System", 7th edition, Elmasri/Navathe, 2015: **Chapter 7 More SQL: Complex Queries, Triggers, Views, and Schema Modification**

Review: SQL yang Sudah Di Pelajari

- DDL
- Basic SQL Query, cartesian product.

Outline

Join SQL

More Complex SQL Retrieval Queries

Meanings of NULL values

Unknown value

- A person's date of birth is not known

Unavailable

- A person has a home phone but does not want it to be listed

Not applicable attribute

- Passport number

SQL does not distinguish between the different meanings of NULL

Operations on NULL value

Table 5.1 Logical Connectives in Three-Valued Logic

(a)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

(b)

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

(c)

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Operations on NULL value

SQL allows queries that check whether an attribute value is NULL

- **IS** or **IS NOT NULL**

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 18. Retrieve the names of all employees who do not have supervisors.

```
Q18:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Super_ssn IS NULL;
```

Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result

Arithmetic Operations

The standard arithmetic operators '+', '-'. '*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result

Query 13: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
Q13: SELECT   FNAME, LNAME, 1.1*SALARY
             AS INCREASED_SAL
     FROM     EMPLOYEE, WORKS_ON, PROJECT
     WHERE    SSN=ESSN AND PNO=PNUMBER AND
     PNAME='ProductX;
```

Arithmetic Operations

Query 14: Retrieve all employees in department 5 whose salary is between $30,000 and $40,000

```
Q14: SELECT   *
     FROM     EMPLOYEE
     WHERE    (SALARY BETWEEN 30000 AND 40000)
              AND DNO=5;

Q14A: SELECT  *
      FROM    EMPLOYEE
      WHERE   (SALARY >= 30000 AND SALARY <=40000)
              AND DNO=5;
```

The EXCEPT Function

Equal to **minus** operation

A except B means set of data in A **without** data that appears in B

```
(SELECT … FROM …. WHERE…. ) EXCEPT
(SELECT … FROM … WHERE …)
```

A

ID
12345
67890
54321
98760

EXCEPT

B

ID
12345
67890

C

ID
54321
98760

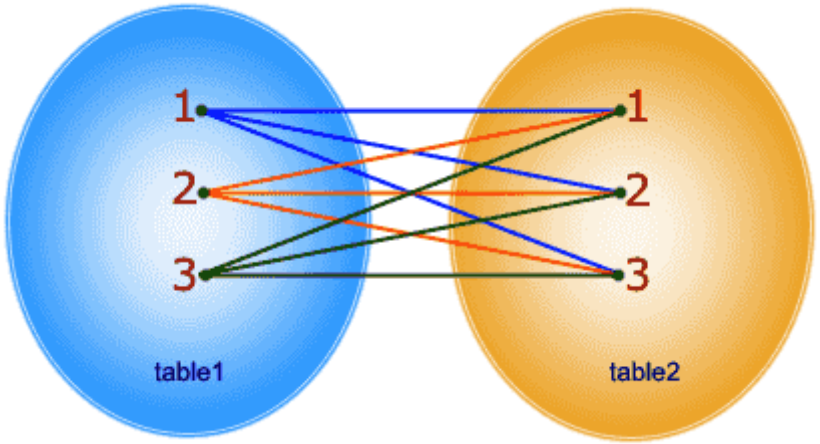
Joined Relations Feature in SQL

Can specify a “joined relation” in the FROM-clause

Looks like any other relation but is the result of a join

Allows the user to specify different types of joins (reguler “theta” JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, CROSS JOIN, etc)

Example CROSS-JOIN



Foods

Name	Cafe
Food 1	XYZ
Food 2	ABC
Food 3	ABC

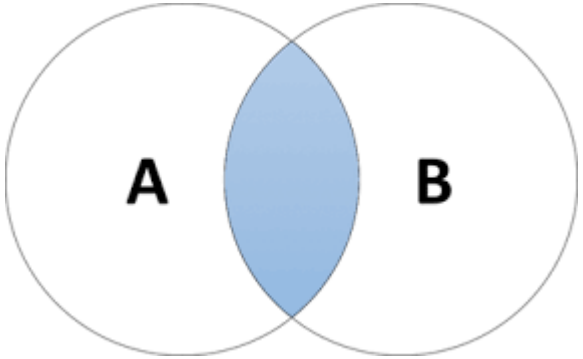
Likes

Person	Food
Narpati	Food 1
Nizar	Food 1
Danu	Food 3

```
SELECT * FROM Foods
CROSS JOIN Likes
```

Name	Cafe	Person	Food
Food 1	XYZ	Narpati	Food 1
Food 1	XYZ	Nizar	Food 1
Food 1	XYZ	Danu	Food 3
Food 2	ABC	Narpati	Food 1
Food 2	ABC	Nizar	Food 1
Food 2	ABC	Danu	Food 3
Food 3	ABC	Narpati	Food 1
Food 3	ABC	Nizar	Food 1
Food 3	ABC	Danu	Food 3

EXAMPLE – THETA JOIN



Foods

Name	Cafe
Food 1	XYZ
Food 2	ABC
Food 3	ABC

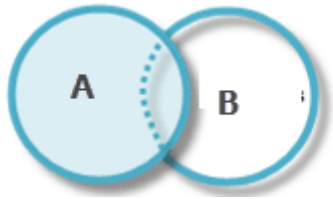
Likes

Person	Food
Narpati	Food 1
Nizar	Food 1
Danu	Food 3

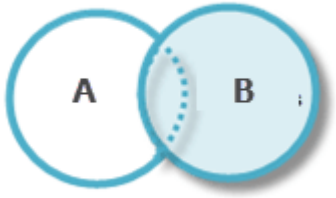
SELECT *
FROM Foods F
JOIN Likes L **ON**
F.name = L.food

Name	Cafe	Person	Food
Food 1	XYZ	Narpati	Food 1
Food 1	XYZ	Nizar	Food 1
Food 3	ABC	Danu	Food 3

EXAMPLE – OUTER JOIN



Left outer join



Right outer join

Foods

Name	Cafe
Food 1	XYZ
Food 2	ABC
Food 3	ABC

Likes

Person	Food
Narpati	Food 1
Nizar	Food 1
Danu	Food 3
Avi	Food 5

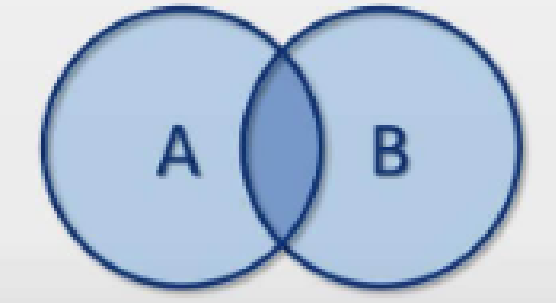
```
SELECT * FROM Foods B LEFT OUTER
JOIN Likes L ON B.name = L.Food
```

Name	Cafe	Person	Food
Food 1	XYZ	Narpati	Food 1
Food 1	XYZ	Nizar	Food 1
Food 2	ABC		
Food 3	ABC	Danu	Food 3

```
SELECT * FROM Foods B RIGHT OUTER
JOIN Likes L ON B.name = L.Food
```

Name	Cafe	Person	Food
Food 1	XYZ	Narpati	Food 1
Food 1	XYZ	Nizar	Food 1
Food 3	ABC	Danu	Food 3
		Avi	Food 5

Example – FULL OUTER JOIN




```
SELECT *
FROM Foods B
FULL OUTER JOIN Likes L ON
B.name = L.Food
```

Foods

Name	Cafe
Food 1	XYZ
Food 2	ABC
Food 3	ABC

Likes

Person	Food
Narpati	Food 1
Nizar	Food 1
Danu	Food 3
Avi	Food 5

Name	Cafe	Person	Food
Food 1	XYZ	Narpati	Food 1
Food 1	XYZ	Nizar	Food 1
Food 2	ABC		
Food 3	ABC	Danu	Food 3
		Avi	Food 5

EXAMPLE- NATURAL JOIN

Likes

Person	Food
Narpati	Food 1
Nizar	Food 1
Danu	Food 3
Harith	Food 2

Frequents

Person	Cafe
Avi	ABC
Danu	XYZ
Nizar	ABC
Jack	SB

```
SELECT * FROM Likes
NATURAL JOIN Frequents
```

Person	Food	Cafe
Nizar	Food 1	ABC
Danu	Food 3	XYZ

Nested Queries

Some queries require that existing values in the database be fetched and then used in a comparison condition -> using **nested query**

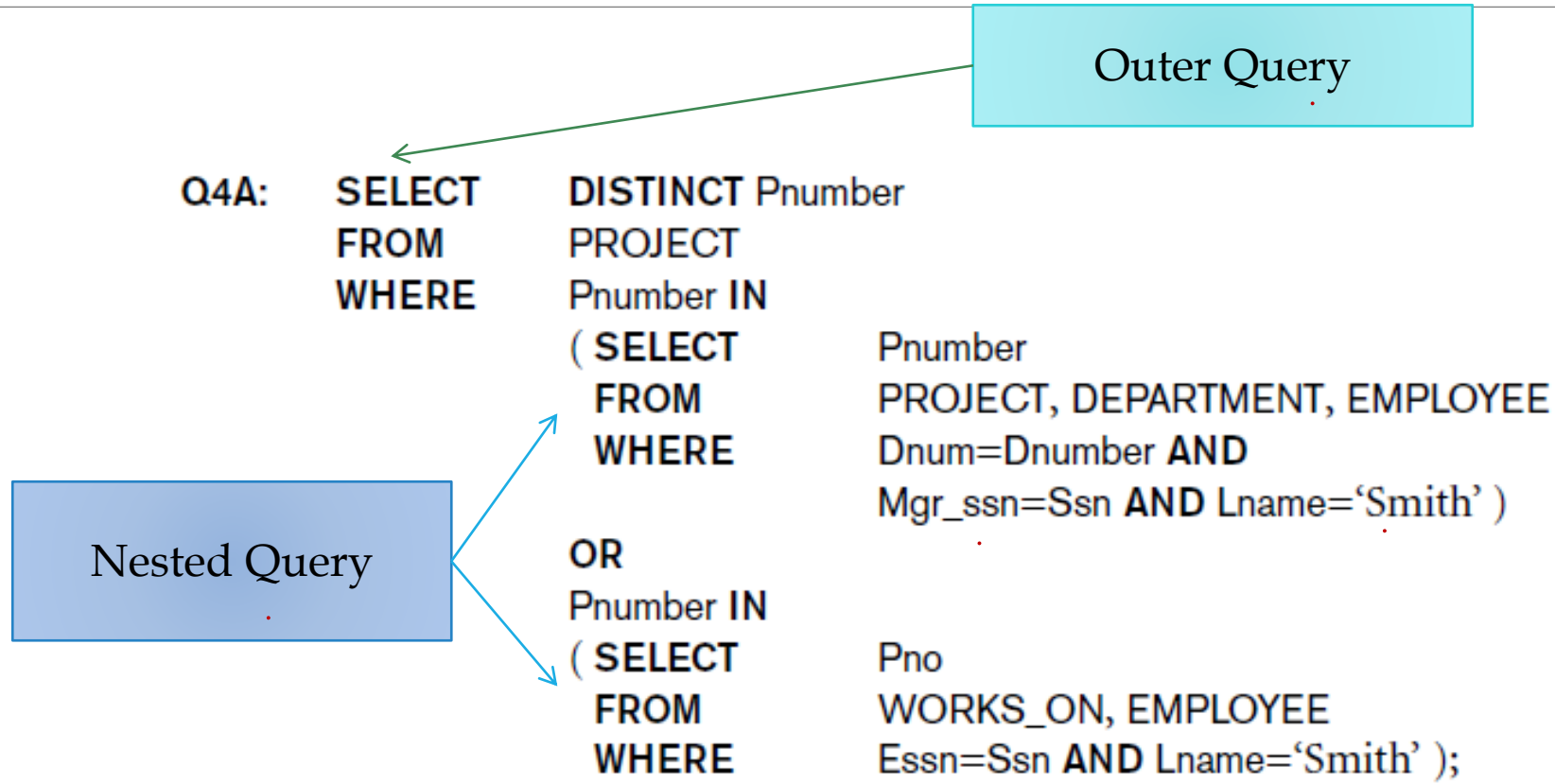
A nested query is a complete SELECT-FROM-WHERE block, within in the WHERE-clause of another query

That other query is called the *outer query*

Comparison operator IN

- Compares value *v* with a set (or multiset) of values *V*
- Evaluates to `TRUE` if *v* is one of the elements in *V*

Nested Queries



Nested Queries

Use tuples of values in comparisons

- Place them within parentheses

Query: retrieve the SSN from all employees who work the same (project,hours) combination on same project that employee 'John Smith' (ESSN = '123456789') works on.

```
SELECT    DISTINCT Essn
FROM      WORKS_ON
WHERE     (Pno, Hours) IN ( SELECT    Pno, Hours
                            FROM      WORKS_ON
                            WHERE     Essn='123456789' );
```

Nested Queries

Use other comparison operators to compare a single value v

- = ANY (or = SOME) operator
  - Returns TRUE if the value v is equal to some value in the set V and is hence **equivalent** to IN
- Other operators that can be combined with ANY (or SOME): >, >=, <, <=, and <>

```
SELECT   Lname, Fname
FROM     EMPLOYEE
WHERE    Salary > ALL ( SELECT   Salary
                        FROM     EMPLOYEE
                        WHERE    Dno=5 );
```

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 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

```
Q16:  SELECT   E.Fname, E.Lname
      FROM     EMPLOYEE AS E
      WHERE    E.Ssn IN ( SELECT   Essn
                          FROM     DEPENDENT AS D
                          WHERE    E.Fname=D.Dependent_name
                          AND E.Sex=D.Sex );
```

Refer to sex attribute in outer query (EMPLOYEE)

Correlated Nested Queries

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, Q16 may be written as in Q12A

```
Q12A:   SELECT   E.FNAME, E.LNAME
        FROM     EMPLOYEE E, DEPENDENT D
        WHERE    E.SSN=D.ESSN AND
                 E.FNAME=D.DEPENDENT_NAME
                 AND
                 E.SEX = D.SEX
```

The EXISTS Functions

Check whether the result of a correlated nested query is empty (contains no tuples) or not

EXISTS and NOT EXISTS are **usually used in conjunction** with a correlated nested query

The EXISTS Function

Query 12: Retrieve the name of each employee who has a dependent with the same first name and same sex as the employee.

Q12B:

```
SELECT    Fname, Lname
FROM      EMPLOYEE E
WHERE     EXISTS (SELECT * FROM DEPENDENT
          WHERE SSN = ESSN AND
          Fname = DEPENDENT_NAME AND
          E.Sex = Sex);
```

The EXISTS Function

Query 6: Retrieve the names of employees who have no dependents

Q6:

```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   NOT EXISTS (
        SELECT * FROM DEPENDENT
        WHERE SSN = ESSN)
```

The correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

The EXISTS Function

Query 7: List the names of managers who have at least one dependent.

```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   EXISTS (
   SELECT * FROM DEPENDENT WHERE SSN = ESSN)
AND     EXISTS (
   SELECT * FROM DEPARTMENT WHERE SSN = MGRSSN);
```

- The first nested query select all DEPENDENT tuples related to an EMPLOYEE
- The second nested query select all DEPARTMENT tuples managed by the EMPLOYEE
- If at least one of the first and at least one of the second exists, we select the EMPLOYEE tuple.

Can you rewrite that query using only on a nested query or no nested query?

Alternative of Sample Query 7

List the names of managers who have at least one dependent without nested.

```
SELECT e.Fname, e.Lname
FROM EMPLOYEE e
JOIN DEPENDENT d on e.ssn = d.essn
JOIN DEPARTMENT dp on e.ssn = dp.mgr_ssn;
```

The EXISTS Function

Query 3: Retrieve the name of each employee who works on **all** the projects controlled by department number 5

Can be used: (S1 CONTAINS S2) that logically equivalent to (S2 EXCEPT S1) is empty.

```
SELECT  Fname, Lname
FROM    EMPLOYEE
WHERE   NOT EXISTS (
    (SELECT Pnumber FROM PROJECT WHERE DNUM = 5)
    EXCEPT
    (SELECT Pno FROM WORKS_ON WHERE SSN = ESSN)
);
```

- The first subquery select all projects controlled by dept 5
- The second subquery select all projects that particular employee being considered works on.
- If the set difference of the first subquery MINUS (EXCEPT) the second subquery is empty, it means that the employee works on all the projects and is hence selected

Exercise

Gunakan data state COMPANY untuk menuliskan query berdasarkan permintaan berikut.

1. Tampilkan nama depan dan gaji employee yang terlibat pada project namun memiliki jam kerja null.
2. Tampilkan nama depan manager dan nama department manager tersebut bekerja dimana project pada departemen tersebut dikerjakan terdapat karyawan yang memiliki jam kerja null.
3. Tampilkan nama depan dan ssn employee yang mempunyai departemen dan jenis kelamin yang sama dengan Franklin Wong.
4. Tampilkan nama employee dan nama departmentnya dimana employee tersebut minimal terlibat pada satu project.
5. Tampilkan nama belakang dan alamat employee yang tidak memiliki tanggungan anak (Son atau Daughter)
6. Tampilkan nama belakang department manager yang tidak mempunyai tanggungan.
7. Tampilkan nama depan dan ssn employee dimana project yang employee tersebut kerjakan selalu sama dengan yang dikerjakan oleh James Borg.