W22
CSC 430/530 DBMS/DT

Lab 2 - Examples

8

Basic SQL Query

```
SELECT    [DISTINCT] <attribute list>
FROM      <table list>
WHERE     <condition>;
```

- *table-list or relation-list*: List of relation names
- *Attribute list or target-list* : List of attributes of tables in *relation-list*
- *qualification* : Comparisons combined using AND, OR and NOT.
- *DISTINCT* : optional keyword indicating that the answer should not contain duplicates.

9

Query Semantics

1. FROM : compute *cross product* of tables.
2. WHERE : Check conditions, discard tuples that fail.
3. SELECT : Delete unwanted fields.
4. DISTINCT *(optional)* : eliminate duplicate rows.

Note: Probably the least efficient way to compute a query! *Query optimizer* will find more efficient ways to get the *same answer*.

10

Example of Conceptual Evaluation (1)

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103;

(1) Compute the cross-product of relation-list.

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

x

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

11

Example of Conceptual Evaluation (1)

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103;

(1) Compute the cross-product of relation-list.

Sailors

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

X

Reserves

sid	bid	day
22	101	10/10/96
58	103	11/12/96

12

Example of Conceptual Evaluation (2)

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103;

(2) Discard tuples if they fail qualifications.

Sailors X Reserves

S.sid	sname	rating	age	R.sid	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

13

Example of Conceptual Evaluation (2)

SELECT S.sname
FROM Sailors S, Reserves R
WHERE ***S.sid=R.sid AND R.bid=103****;*

(2) Discard tuples if they fail qualifications.

Sailors X Reserves

S.sid	sname	rating	age	R.sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~22~~	~~101~~	~~10/10/96~~
22	dustin	7	45.0	58	103	11/12/96
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~10/10/96~~
31	lubber	8	55.5	58	103	11/12/96
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35.0	58	103	11/12/96

14

Example of Conceptual Evaluation (2)

SELECT S.sname
FROM Sailors S, Reserves R
WHERE ***S.sid=R.sid AND R.bid=103****;*

(2) Discard tuples if they fail qualifications.

Sailors X Reserves

S.sid	sname	rating	age	R.sid	bid	day
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~22~~	~~101~~	~~10/10/96~~
~~22~~	~~dustin~~	~~7~~	~~45.0~~	~~58~~	~~103~~	~~11/12/96~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~22~~	~~101~~	~~10/10/96~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~	~~58~~	~~103~~	~~11/12/96~~
~~58~~	~~rusty~~	~~10~~	~~35.0~~	~~22~~	~~101~~	~~10/10/96~~
58	rusty	10	35.0	58	103	11/12/96

15

Example of Conceptual Evaluation (3)

SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103;

(3) Delete attribute columns that are not in target-list.

sname
rusty

Sailors X Reserves

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

16

Aliasing

Consider the following SALESREPS relation

Empl_num	name	age	Rep_office	manager
105	Bill	37	13	104
104	Bob	33	12	106
106	Sam	52	11	NULL

Query: Determine the name of Bob's manager?

HINT: Use the ρ operator from Relational Algebra

17

Aliasing in SQL

```
SELECT s2.name
FROM   SALESREPS s1, SALESREPS s2
WHERE  s1.name='Bob' AND
       s1.manager=s2.empl_num;
```

- Aliases must be used here.
- The row referenced by s1 is intended to be Bob (employee role)…while s2 will be his manager (role).
- Keep in mind SQL executes its queries: first FROM, then WHERE, then SELECT

18

Important naming of variables

- Needed when ambiguity could arise.
  - e.g., same table used multiple times in FROM ("self-join")

```
SELECT  x.sname, x.age, y.sname, y.age
FROM    Sailors x, Sailors y
WHERE   x.age > y.age
```

Sailors x

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Sailors y

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

21

Arithmetic Expressions

```
SELECT S.age, S.age-5 AS age1, 2*S.age AS age2
FROM   Sailors S
WHERE  S.sname = 'dustin'
```

```
SELECT  S1.sname AS name1, S2.sname AS name2
FROM   Sailors S1, Sailors S2
WHERE  2*S1.rating = S2.rating - 1
```

22

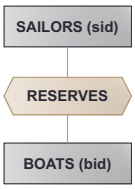
String Comparisons

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sname LIKE 'B_%B'
```

'_' stands for any one character and '%' stands for 0 or more arbitrary characters.

23

Find sid's of sailors who've reserved a red **or** a green boat



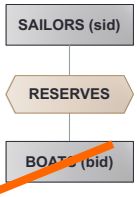

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND
              (B.color='red' OR B.color='green')
```

... or:

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color='red'
UNION
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color='green'
```

24

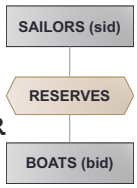
Find sid's of sailors who've reserved red **and** green boats




```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND (B.color='red' AND
            B.color='green')
```

25

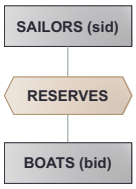
Find sid's of sailors who've reserved red and green boats




```
SELECT S.sid
FROM   Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid
         AND R.bid=B.bid
         AND B.color='red'
INTERSECT
SELECT S.sid
FROM   Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid
         AND R.bid=B.bid
         AND B.color='green'
```

26

Find sid's of sailors who've reserved red and green boats

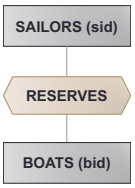


- Could use joins:

```
SELECT R1.sid
FROM   Boats B1, Reserves R1, Boats B2, Reserves R2
WHERE  R1.sid=R2.sid
       AND R1.bid=B1.bid
       AND R2.bid=B2.bid
       AND (B1.color='red' AND B2.color='green')
```

27

Find sid's of sailors who have <u>not</u> reserved a boat




```
SELECT S.sid
FROM   Sailors S
EXCEPT
SELECT S.sid
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid
```

28

Nested Queries: IN

Names of sailors who've reserved boat #103:

```
SELECT   S.sname
FROM     Sailors S
WHERE    S.sid IN (SELECT   R.sid
                   FROM     Reserves R
                   WHERE    R.bid=103)
```

29

Nested Queries: NOT IN

*Names of sailors who've **not** reserved boat #103:*

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid NOT IN (SELECT  R.sid
                     FROM    Reserves R
                     WHERE  R.bid=103)
```

30

Motivation for Subqueries

- Find the name of the professors who teach "CS 430."

```
SELECT Name
FROM   Professors, Teach
WHERE  (PID = ProfessorPID) AND (Number = '430')
       AND (DeptName = 'CS');
```

- Do we need to take the natural join of two big relations just to get a relation with few tuples?
- Can we rewrite the query without using a join?

32

Nesting

- A query can be put inside another query
- Most commonly in the WHERE clause
- Sometimes in the FROM clause (depending on the DBMS)
- This subquery is executed first (if possible)

33

Subquery Example

- Find the name of the professor who teaches "CS 430."

```
SELECT Name
FROM Professors
WHERE PID =
            (SELECT ProfessorPID
            FROM Teach
            WHERE (Number = 430) AND (DeptName = 'CS')
            );
```

- When using =, the sub-query must return a single tuple

34

Conditions Involving Relations

- SQL includes operators that can be applied to produce a Boolean result.
- These operators are useful while working with sub-queries.

- Let **R be a relation** and **t be a tuple** (**with the same set of attributes** as that of R).
  1. t **EXISTS** R is true if and only if R contains at least one tuple.
  2. t **IN** R is true if and only if t equals a tuple in R.
  3. t **> ALL** R is true if and only if R is *unary* (has one attribute) and t is greater than **every** value in R.
  4. t **> ANY** R (which is *unary*) is true if and only if t is greater than **at least** one value in R.

- We can use **NOT** to negate EXISTS, ALL, and ANY.

35

Subqueries Using Conditions

- Find the departments of the courses taken by the student with name 'Suri'.

```
SELECT DeptName
FROM Take
WHERE StudentPID IN
                ( SELECT PID
                  FROM Students
                  WHERE (Name = 'Suri')
                );
```

36

Correlated vs Uncorrelated

- The previous sub-queries did not depend on anything **outside the sub-query**
  - …and thus need to be executed just once.
  - These are called uncorrelated.

A correlated sub-query depends on data from the outer query … and thus must be executed for each row of the outer table(s)

37

Nested Queries with Correlation

Names of sailors who've reserved boat #103:

```
SELECT  S.sname
FROM  Sailors S
WHERE EXISTS
            (SELECT  *
             FROM  Reserves R
             WHERE R.bid=103 AND S.sid=R.sid)
```

- Sub-query must be **recomputed** for each Sailors tuple.
  - Think of sub-query as a function call that runs a query!
- The same holds for NOT EXISTS as well.

38

Correlated Subqueries

- Find course names that have been used for two or more courses.

```
SELECT CourseName
FROM Courses AS First
WHERE CourseName IN
                  (SELECT CourseName
                  FROM Courses
                  WHERE (Number <> First.Number)
                  AND (DeptName <> First.DeptName)
                  );
```

39

Evaluating Correlated Subqueries

```
SELECT CourseName
FROM   Courses AS First
WHERE  CourseName IN
            (SELECT CourseName
             FROM   Courses
             WHERE  (Number <> First.Number)
             AND (DeptName <> First.DeptName)
             );
```

- Evaluate query by looping over tuples of First, and for each tuple evaluate the subquery.
- **Scoping rules:** an attribute in a subquery belongs to one of the tuple variables in that subquery's FROM clause, or to the immediately surrounding subquery, and so on.

40

UNIQUE

Names of sailors who've reserved boat #103 ***exactly once****:*

```
SELECT  S.sname
FROM  Sailors S
WHERE UNIQUE
              (SELECT  *
               FROM  Reserves R
               WHERE R.bid=103 AND
                     S.sid=R.sid)
```

42

A Tough One

Find sailors who've reserved all boats.

```
SELECT  S.sname
FROM    Sailors S
WHERE   NOT EXISTS (SELECT  B.bid
                    FROM    Boats B
                    WHERE   NOT EXISTS(SELECT  R.bid
                                       FROM  Reserves R
                                       WHERE  R.bid=B.bid
                                       AND R.sid=S.sid))
```

Sailors S such that ...

there is no boat B without …

a Reserves tuple showing S reserved B

44

Summary

- Relational model has well-defined query semantics
- SQL provides functionality close to basic relational model
  (some differences in duplicate handling, null values, set operators, …)
- Typically, many ways to write a query
  - DBMS figures out a fast way to execute a query, regardless of how it is written.

45