SQL – Part 2

Sub-queries and Set Predicates

The relational model

- Our running example.

Student

Sid	Sname	Major	Byear
s1	John	CS	1990
s2	Ellen	Math	1995
s3	Eric	CS	1990
s4	Ann	Biology	2001

Course

Cno	Cname	Dept
c1	Dbs	CS
c2	Calc1	Math
c3	Calc2	Math
c4	AI	Info

Enroll

Sid	Cno	Grade
s1	c1	B
s1	c2	A
s2	c3	B
s3	c1	A
s3	c2	C

The IN and NOT IN predicates

- Consider the query “Find the sids of students who are enrolled in CS courses.”
- In SQL with join conditions:

```
SELECT DISTINCT E.Sid
FROM   Enroll E, Course C
WHERE E.Cno = C.Cno AND C.Dept = ‘CS’;
```

The IN predicate

- “Find the sids of students who are enrolled in CS courses.”
- In SQL with the IN predicate,

```
SELECT DISTINCT E.Sid
FROM   Enroll E
WHERE E.Cno IN (SELECT C.Cno
                FROM   Course C
                WHERE C.Dept = ‘CS’);
```

- The IN predicate corresponds to the set-membership predicate $\in$.
- For example, $a \in \{a, b, c\}$ is true, but $d \in \{a, b, c\}$ is false.

The NOT IN predicate

- Now consider the SQL query

```
SELECT DISTINCT E.Sid
FROM   Enroll E
WHERE E.Cno NOT IN (SELECT C.Cno
                    FROM   Course C
                    WHERE C.Dept = ‘CS’);
```

- This is the query “Find the sids of students who are enrolled in a course that is not offered by the CS department.”
- Note that this is not the query “Find the sids of students who take no CS courses.”

The SOME predicate

- Consider again the query “Find the sids of students who are enrolled in a CS course”:

```
SELECT DISTINCT E.Sid
FROM   Enroll E
WHERE E.Cno IN (SELECT C.Cno FROM Course C WHERE C.Dept = ‘CS’);
```

- This query can also be written using the SOME predicate as follows:

```
SELECT DISTINCT E.Sid
FROM   Enroll E
WHERE E.Cno = SOME (SELECT C.Cno FROM Course C WHERE C.Dept = ‘CS’);
```

- The = SOME predicate checks if E.Cno is equal to some (i.e., at least one) number of a CS course.

The ALL predicate

- Consider the relation

Pid	Age
p1	10
p2	9
p3	12
p4	9

- Consider the query “Find the pids of the youngest persons.”
- The answer to this query consists of p2 and p4 since their age (i.e., 9) is smaller than or equal to all ages, i.e., (10, 9, 12, 9):

Pid
p2
p4

- In SQL with the ALL predicate,

```
SELECT P.Pid
FROM   Person P
WHERE P.Age <= ALL (SELECT P1.Age FROM Person P1);
```

More SOME and ALL predicates

- Consider the SQL query

```
SELECT P.Pid
FROM   Person P
WHERE P.Age < ALL (SELECT P1.Age
                   FROM Person P1);
```

- This query will return nothing since there is no person whose age is strictly smaller than all ages.
- What are the answers of similar queries with the following predicates?

SOME	ALL
= SOME	= ALL
<> SOME	<> ALL
< SOME	< ALL
<= SOME	<= ALL
> SOME	> ALL
>= SOME	>= ALL

The EXISTS predicate

- The EXISTS predicate takes as argument a relation that is the answer of a query.
- If there exists a tuple in that relation, then the EXISTS predicate evaluates to true.
- Otherwise, if there does not exist a tuple in that relation, then the EXISTS predicate evaluates to false.

The EXISTS predicate: example

- Assume that there exist students who major in CS, then

```
EXISTS (SELECT S.Sid
        FROM   Student S
        WHERE S.Major = 'CS')
```

evaluates to true.

- If there are no students who major in biology, then

```
EXISTS (SELECT S.Sid
        FROM  Student S
        WHERE S.Major = 'Biology')
```

evaluates to false.

The NOT EXISTS predicate

- The NOT EXISTS predicate takes as argument a relation that is the answer of a query.
- If there exists a tuple in that relation, then the NOT EXISTS predicate evaluates to false.
- Otherwise, i.e., if there does not exist a tuple in that relation, then the NOT EXISTS predicate evaluates to true.

The NOT EXISTS predicate: example

- Assume that there exist students who major in CS, then

  ```
  NOT EXISTS (SELECT S.Sid
              FROM   Student S
              WHERE S.Major = 'CS')
  ```

  evaluates to false.

- If there are no students who major in biology, then

  ```
  NOT EXISTS (SELECT S.Sid
              FROM  Student S
              WHERE S.Major = 'Biology')
  ```

  evaluates to true.

Emptiness check of a relation

- The EXISTS predicate determines whether or not a relation is not empty.
- The NOT EXISTS predicates determines whether or not a relation is empty.

Predicate	Relation	Value
EXISTS	$\neq \emptyset$	true
EXISTS	$= \emptyset$	false
NOT EXISTS	$\neq \emptyset$	false
NOT EXISTS	$= \emptyset$	true

EXISTS and NOT EXISTS in WHERE clause

- Since EXISTS and NOT EXISTS are boolean predicates, they can be used in the WHERE clause of a SQL query.
- Example: “Find the sids of students whose name is John provided that there exist students who major in CS.”

```
SELECT S.Sid
FROM  Student S
WHERE S.Sname = ‘John’ AND EXISTS (SELECT S1.Sid
                                   FROM  Student S1
                                   WHERE S1.Major = ‘CS’);
```

- This query will return the sids of all students with name John, but only if there are student majoring in CS.
- If, however, there are no such students, then the result of this query is the empty relation.

Sub-queries with parameters

- The power of the EXISTS and NOT EXISTS predicates really emerges when the argument query has parameters.
- Consider the query “Find the sids of student who are enrolled in a course.”
- In SQL with the EXISTS predicate,

```
SELECT S.Sid
FROM  Student S
WHERE EXISTS (SELECT E.Cno
              FROM  Enroll E
              WHERE S.Sid = E.Sid);
```

- Notice how the Student tuple variable S is a parameter of the inner sub-query.

Global and local variables in SQL

- Consider

```
EXISTS (SELECT E.Cno
        FROM   Enroll E
        WHERE S.Sid = E.Sid)
```

- In this predicate, S is a global variable, whereas E is a local variable.
- The possible values for the parameter S.Sid are coming from the outside: for each tuple S in the Student relation, S.Sid takes on the sid value of that tuple.

- Now, if the value of S.Sid is the sid of a student who takes a course, then

```
EXISTS (SELECT E.Cno
        FROM   Enroll E
        WHERE S.Sid = E.Sid)
```

evaluates to true and, therefore, the sid of such a student is returned by the (outer) query.

- If, however, the value of S.Sid is the sid of a student who does not takes any course, then

```
EXISTS (SELECT E.Cno
        FROM   Enroll E
        WHERE S.Sid = E.Sid)
```

evaluates to false and, therefore, this sid value is not returned by the outer query.

Global and local variables: example

- We now want to find the sids of students who do not take any courses.
- For that purpose, we can write the following query:

```
SELECT S.Sid
FROM   Student S
WHERE NOT EXISTS (SELECT E.Cno
                  FROM   Enroll E
                  WHERE S.Sid = E.Sid);
```

Putting it all together

- Fairly complex queries can be composed.
- Example: find the majors of students named Ellen who do not take any CS course.

```
SELECT S.Major
FROM   Student S
WHERE S.Sname = ‘Ellen’ AND
               NOT EXISTS (SELECT C.Cno
                                FROM   Course C
                                WHERE C.dept = ‘CS’ AND
                                          C.cno IN (SELECT  E.Cno
                                                        FROM   Enroll E
                                                        WHERE E.sid = S.Sid));
```

- Next consider the query
“Find the sids of students who take all CS courses.”
- This query can be reformulated as follows:
“Find the sids of students for whom there does not exist a CS course they are not enrolled in.”
- The not exist and not in the latter statement should suggest a way to write this query in SQL.

- This SQL query is as follows:

```
SELECT S.Sid
FROM   Student S
WHERE NOT EXISTS (SELECT C.Cno
                  FROM   Course C
                  WHERE C.Dept = ‘CS’ AND
                        C.Cno NOT IN (SELECT  E.Cno
                                      FROM   Enroll E
                                      WHERE S.Sid = E.Sid))
```

- Notice that a parameter like S.Sid can occur inside a (NOT) EXISTS as well as inside a (NOT) IN predicate.