Yet More SQL

CS157A
Chris Pollett
Oct. 19, 2005.

Outline

- NULLs and 3-valued Logic
- Nested Queries
- Joins
- Aggregate Functions in SQL
- GROUP BY and HAVING
- INSERT,DELETE, and UPDATE

NULLs and 3-valued Logic

- NULL has three potential meanings: An unknown value, an unavailable or withheld value, and an attribute which is not applicable.
- NULL in SQL does not distinguish between these three different meanings.
- If comparisons <, =, etc are done with NULL the result is FALSE.
- To check with an expression is something is/is not NULL, you can use the syntax like:

  WHERE SUPERSSN IS NULL;

  or

  WHERE SUPERSSN IS NOT NULL:

Nested Queries

- Sometimes it is useful to be able to create results of a query by computing intermediate query results first.
- Doing this suggests it would be useful to nest queries.
- Consider the query: List all the project numbers for projects that involve an employee whose last name is 'Smith' where this 'Smith' is either a worker on the project or is a manager of the department that controls the project

```
SELECT DISTINCT PNUMBER FROM PROJECT
    WHERE PNUMBER IN
      (SELECT PNUMBER FROM PROJECT, DEPARTMENT,
                EMPLOYEE WHERE DNUM=DNUMBER AND
                MGRSSN=SSN AND LNAME='SMITH')
      OR
      (SELECT PNUMBER FROM WORKS_ON,
                EMPLOYEE WHERE ESSN=SSN AND LNAME='SMITH')
```

More on Nested Queries

- If the nested query only returns one attribute of one row, then rather than use 'IN' one can use '='.
- If one wants to check if multiple columns come from the result of a query one can do queries like:

```
SELECT DISTINCT ESSN
    FROM WORKS_ON
    WHERE (PNO, HOURS) in (SELECT PNO, HOURS FROM
      WORKS_ON WHERE SSN='123456789');
```

- Besides 'IN' one can also do checks like 'op ANY', 'op SOME', 'op ALL', where op is one of >, >=, <=, <, <>.

```
SELECT LNAME, FNAME FROM EMPLOYEE
      WHERE SALARY >ALL (SELECT SALARY FROM EMPLOYEE WHERE
        DNO=5);
```

- It is legal to have several nestings of queries. To make this work, it is convenient to be able to refer attributes of the outer query in the inner query such as:

```
SELECT E.FNAME, E.LNAME FROM EMPLOYEE AS E WHERE E.SSN
    IN ( SELECT ESSN FROM DEPENDENT WHERE
    E.FNAME=DEPENDENT_NAME AND E.SEX=SEX);
```

Correlated Nested Queries

- Whenever a WHERE clause of a nested query references some attribute of a relation declared in the outer query the two queries are said to be **correlated**.
- The last query of the previous slide is a correlated query.
- Such queries, if they use = or IN to nest, can always be expressed as a single block query. Fo example,

  SELECT E.FNAME, E.LNAME FROM EMPLOYEE AS E,
  DEPENDENT AS D WHERE E.SSN=D.ESSN AND
  E.SEX=D.SEX AND E.FNAME=D.DEPENDENT_NAME;

EXISTS and UNIQUE in SQL

- SQL supports the keywords EXISTS and NOT EXISTS to check whether or not the result of a correlated nested query is empty.
- For example, suppose we want to list the names of employees who have at least one dependent:

```
SELECT FNAME, LNAME FROM EMPLOYEE WHERE EXISTS
   (SELECT * FROM DEPENDENT WHERE SSN=ESSN);
```

- In addition to EXISTS and NOT EXISTS there is a keyword UNIQUE which returns true if there are no duplicates in the result of a query.

Explicit Sets and Renaming of Attributes

- The ‘IN’ clause can also be used with explicitly listed sets rather than nested queries:

```
SELECT DISTINCT ESSN FROM WORKS_ON
   WHERE PNO IN (1,2,3);
```

- It is also sometimes useful to rename the output columns of a query. This can be done using ‘AS’:

```
SELECT E.LNAME AS EMPLOYEE_NAME,
   S.LNAME AS SUPERVISOR_NAME
   FROM EMPLOYEE AS E, EMPLOYEE AS S
   WHERE E.SUPERSSN = S.SSN;
```

JOIN

- SQL supports the keywords: JOIN (aka INNER JOIN), and OUTER JOIN.
- The keywords LEFT, RIGHT, and FULL may precede OUTER JOIN.
- The keyword NATURAL can precede any of the above.
- The basic syntax for a join looks like:

```
TABLE1 JOIN TABLE2 ON TABLE1.A=TABLE2.B;
```

Aggregate Functions in SQL

- The aggregates COUNT, SUM, MAX, MIN, AVG can appear in the SELECT clause of an SQL query:

```
SELECT COUNT(SSN), SUM(SALARY), AVG(SALARY),
    MAX(SALARY), MIN(SALARY), COUNT(DISTINCT
    SALARY), COUNT(*) FROM EMPLOYEE;
```

- COUNT(*) is used to count number of tuples.
- Notice we can also use the keyword DISTINCT before an attribute.
- COUNT and other aggregates ignore NULLs.
- Aggregates can we used with nested queries:

```
SELECT LNAME, FNAME FROM EMPLOYEE WHERE
    (SELECT COUNT(*) FROM DEPENDENT
      WHERE SSN=ESSN) >= 2;
```

GROUP BY and HAVING

- To completely simulate the relational algebra aggregates, we need to be able to break down the aggregation by some attribute.
- This can be done using GROUP BY:

```
SELECT DNO, COUNT(*), AVG(SALARY)
FROM EMPLOYEE
GROUP BY DNO;
```

- It is also useful to be able to place conditions on the groups which are output. This can be done with a HAVING clause:

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT(*) > 2 AND PNAME LIKE ‘R%’;
```

INSERT

- To insert a row into a database the basic syntax is like:
  - INSERT INTO MY_TABLE VALUES (‘A’, ‘B’, ‘C’, ‘D’, ‘E’);
  - If one doesn’t want to specify all the columns or wants to specify a permutation of the columns, but instead wants to rely on default values:
  - INSERT INTO MY_TABLE(COL1, COL3, COL4) VALUES (‘A’, ‘C’, ‘D’);

DELETE

- To delete a row in SQL the basic syntax is:

  DELETE FROM MY_TABLE WHERE SOME_CONDITION;

- For example,

  DELETE FROM EMPLOYEE WHERE SSN=‘123456789’;

- DELETE FROM EMPLOYEE; would delete all rows.

UPDATE

- To change the value of an existing row one can use the UPDATE command:

```
UPDATE PROJECT
SET PLOCATION='Bellaire', DNUM=5
WHERE PNUMBER=10;
```