Chapter 4

SQL DML – continued from Week 4 Monday's class

Week 4 Readings (SQL):

4.4
5.1

Week	Topic	Labs (Due Fridays)	Quizzes (Due Sundays)	Assignments (Due ~~Wednesdays~~)	Worksheets (Due Saturdays)
Week 0 (Sept 9th)	Introduction to the course			Thursdays	
Week 1 (Sept 13th)	Introduction, Relational Model				W1
Week 2 (Sept 20th)	Relational Algebra				W2
Week 3 (Sept 27th)	SQL DML	L1	Q1		W3
Week 4 (Oct 4th)	SQL DML, Constraints in SQL	L2			W4
Week 5 (Oct 11th)	No Class on Oct 11th	~~L3~~		~~A1~~	W5 (Optional and Ungraded)
	Design I (ER Modeling)				
Week 6 (Oct 18th)	Design I (ER Modeling)	L3		A1	W6
Week 7 (Oct 25th)	Design II (Normalization)		Q2		W7
Week 8 (Nov 1st)	Design II (Normalization), SQL DDL				W8
Week 9 (Nov 8th)	Stored procedures, triggers			A2	W9
Week 10 (Nov 15th)	Transaction Management	L4			W10
Week 11 (Nov 22nd)	NoSQL		Q3		W11
Week 12 (Nov 29th)	Final Exam preparation			A3	

Correlated Subqueries

Correlated Subqueries

- If a subquery refers to any name defined outside of itself, it must be evaluated once for each tuple in the outer query.
- These are called correlated subqueries
- EXISTS and NOT EXISTS

EXISTS - example

List all suppliers who supply a part.

```
SELECT SNAME
FROM S
WHERE EXISTS
    (SELECT *
    FROM SP
    WHERE S.SNO=SP.SNO)
```

NOT EXISTS - example

- Get supplier names for suppliers who do not supply part P2.

```
SELECT SNAME
FROM S
WHERE NOT EXISTS
    (SELECT *
    FROM SP
    WHERE S.SNO=SP.SNO
    AND       SP.PNO = 'P2')
```

Divide By

- In SQL, there is no 'for all' predicate (done by division operator in RA). However, a 'for all' predicate can be converted to :

  ∀, WHERE p(x) is true

  =>

  There doesn't exist an x, where p(x) is not true .

- SQL does not support ∀. But we can use logical tautology of converting a ∀ to 2 negations – explained through examples in the following slides.

Divide By in SQL

Get supplier names for suppliers who supply all parts.

Rewrite as 2 negations:

Get suppler names such that there does not exist a part that the supplier does not supply.

Divide By in SQL

- $\forall\, x \Rightarrow \nexists \nexists\, x$

Worksheet 2 Q 4 Get supplier names for suppliers who supply at least those parts supplied by S2.

Get suppler names such that there does not exist a part supplied by S2 that the supplier does not supply.

Divide By in SQL

- $\forall x \Rightarrow \nexists \nexists x$

Worksheet 2 Q 9. Get supplier names for suppliers who supply all red parts.

Get suppler names such that there does not exist a part that is RED that the supplier does not supply.

$\pi_{sno, pno}(SP) \div \pi_{pno}$

Divide from RA to SQL

~~Worksheet 2 Q 3.~~ Get supplier names for suppliers who supply all parts.

In Relational Algebra:

R1: = $\pi_{sno,pno}SP \div \pi_{pno}P$

R2 := π_{sname} R1 ∞ S

```
1  SELECT SNAME
2  FROM S
3  WHERE NOT EXISTS (SELECT *
4                    FROM P
5                    WHERE NOT EXISTS (SELECT *
6                                      FROM SP
7                                      WHERE S.SNO=SP.SNO
8                                      AND P.PNO=SP.PNO));
9
```

Get suppler names such that there does not exist a part that the supplier does not supply.

Divide from RA to SQL

~~Worksheet 2 Q 3~~. Get supplier names for suppliers who supply at least all those parts supplied by S2.

In Relational Algebra:

R0 := $\pi_{sno}(\sigma_{sno='S2'}(SP))$

R1: = $\pi_{sno,pno}SP \div$ R0

R2 := π_{sname} R1∞S

```
1  SELECT SNAME
2  FROM S
3  WHERE NOT EXISTS (SELECT  *
4                    FROM SP SP1
5                    WHERE SNO = 'S2'
6                    AND NOT EXISTS (SELECT  *
7                                    FROM SP SP2
8                                    WHERE  SP1.PNO = SP2.PNO
9                                    AND SP2.SNO = S.SNO))
```

Division operator – Method 2

- Get supplier names for suppliers who supply all parts.

You can use COUNT and GROUP BY clause to emulate the behavior of divide of RA. You can plan it as follows for the given example:

Step 1: Find a count of all parts

Step 2: group SP on sno and use HAVING to match the count.

Division operator – Method 2

- Get supplier names for suppliers who supply all parts.

```
SELECT sname
FROM S
WHERE sno IN (SELECT sno
                    FROM SP
                    GROUP BY sno
                    HAVING COUNT(pno) = (SELECT COUNT(*)
                                         FROM P));
```

Division operator – Method 2

- Get supplier names for suppliers who supply all parts supplied by ‘S2’.

```
SELECT sname
FROM S
WHERE sno IN (SELECT sno
              FROM SP
              WHERE pno IN (select pno from sp where sno='S2')
              GROUP BY sno
              HAVING COUNT(pno) = (SELECT COUNT(*)
                                   FROM SP
                                   WHERE Sno = 'S2'));
```

Specifying Updates in SQL

- There are three SQL commands to modify the database;

- INSERT
- DELETE
- UPDATE

INSERT

- In its simplest form, it is used to add one or more tuples to a relation

- Attribute values should be listed in the same order as the attributes were specified in the CREATE TABLE command

INSERT - example

- INSERT INTO S VALUES('S10', 'Natalie', 23, 'VANCOUVER');
- INSERT INTO S VALUES('S101', 'Natalie', NULL, NULL);

INSERT

- Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation
- Example:

```
INSERT INTO S_COPY (SELECT * FROM S);
```

DELETE

- Removes tuples from a relation
- Includes a WHERE-clause to select the tuples to be deleted
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause
- Referential integrity should be enforced

```
DELETE
FROM S
WHERE sno = 'S1';
```

DELETE

- Delete tuples satisfying a condition:

- `DELETE FROM` *«relation»*
  `WHERE` *«condition»*`;`

- Delete all tuples:
  `DELETE FROM` *«relation»;*

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

UPDATE - example

- To change the value of certain attributes in certain tuples to given values:

  `UPDATE` *«relation»*
  `SET` *«list of attribute assignments»*
  `WHERE` *«condition on tuples»*`;`

UPDATE SP
SET QTY = 150
WHERE QTY = 100;

UPDATE - example

- To change the value of certain attributes in certain tuples to given values:

  `UPDATE` *«relation»*
  `SET` *«list of attribute assignments»*
  `WHERE` *«condition on tuples»*`;`

UPDATE SP
SET QTY = 150;

- The above query updates all tuples (since WHERE is missing).