

Lecture 9

SQL : Schema Definition, Constraints, and Queries and Views

Specifying Updates in SQL

- ▶ There are three SQL commands to modify the database: **INSERT**, **DELETE**, and **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:

```
U1:INSERT INTO EMPLOYEE  
VALUES ("Richard",'K',"Marini", "653298653", "30-  
DEC-52", "98 Oak Forest,Katy,TX", 'M',  
37000,"987654321", 4 )
```

- ▶ An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple
 - Attributes with NULL values can be left out
- ▶ Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

```
U1A: INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)  
VALUES ("Richard", "Marini", "653298653")
```

INSERT

- ▶ Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database
 - Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation

INSERT

- ▶ Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department.
 - A table DEPTS_INFO is created by U3A, and is loaded with the summary information retrieved from the database by the query in U3B.

U3A: `CREATE TABLE DEPTS_INFO
(DEPT_NAME VARCHAR(10),
 NO_OF_EMPS INTEGER,
 TOTAL_SAL INTEGER);`

U3B: `INSERT INTO DEPTS_INFO (DEPT_NAME,
 NO_OF_EMPS, TOTAL_SAL)
 SELECT DNAME, COUNT (*), SUM (SALARY)
 FROM DEPARTMENT, EMPLOYEE
 WHERE DNUMBER=DNO
 GROUPBY DNAME ;`

INSERT

- ▶ Note: The DEPTS_INFO table may not be up-to-date if we change the tuples in either the DEPARTMENT or the EMPLOYEE relations *after* issuing U3B. We have to create a view (see later) to keep such a table up to date.

DELETE

- ▶ Removes tuples from a relation
 - Includes a WHERE-clause to select the tuples to be deleted
 - Referential integrity should be enforced
 - 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

DELETE

- ▶ Examples:

U4A: DELETE FROM EMPLOYEE
WHERE LNAME="Brown"

U4B: DELETE FROM EMPLOYEE
WHERE SSN="123456789"

U4C: DELETE FROM EMPLOYEE
WHERE DNO IN
(SELECT DNUMBER
FROM DEPARTMENT
WHERE DNAME="Research")

U4D: DELETE FROM EMPLOYEE

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: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

```
U5: UPDATE PROJECT  
      SET PLOCATION = "Bellaire",  
          DNUM = 5  
      WHERE PNUMBER=10
```

UPDATE

- ▶ Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6:UPDATE EMPLOYEE  
SET SALARY = SALARY *1.1  
WHERE DNO IN (SELECT DNUMBER  
FROM DEPARTMENT  
WHERE DNAME='Research')
```

- ▶ In this request, the modified SALARY value depends on the original SALARY value in each tuple
 - ▶ The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
 - ▶ The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Views in SQL

View is just a query for simplicity, any time you run a query on the view, the DB engines add your query to the view query.

- ▶ A view is a “virtual” table that is derived from other tables
- ▶ Allows for limited update operations
 - ▶ Since the table may not physically be stored
- ▶ Allows full query operations
- ▶ A convenience for expressing certain operations

A good to know concept is 'Materialized Views' which is a view stored on disk , so new queries can run with higher performance. But there is a trade-off for this, either refresh view with every commit, or with a cron job (scheduled tasks) for data correctness when reading from the view.

Specification of Views

- ▶ SQL command: **CREATE VIEW**
 - a table (view) name
 - a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)
 - a query to specify the table contents

SQL Views: An Example

- ▶ Specify a different WORKS_ON table

```
CREATE VIEW WORKS_ON_NEW AS  
SELECT FNAME, LNAME, PNAME, HOURS  
FROM EMPLOYEE, PROJECT, WORKS_ON  
WHERE SSN=ESSN AND PNO=PNUMBER  
GROUP BY PNAME;
```

Using a Virtual Table

- ▶ We can specify SQL queries on a newly create table (view):

```
SELECT FNAME , LNAME  
FROM WORKS_ON_NEW  
WHERE PNAME="Seena";
```

- ▶ When no longer needed, a view can be dropped:

```
DROP WORKS_ON_NEW;
```

Efficient View Implementation

- ▶ Query modification:
 - Present the view query in terms of a query on the underlying base tables
- ▶ Disadvantage:
 - Inefficient for views defined via complex queries
 - ❖ Especially if additional queries are to be applied to the view within a short time period

Update Views

- ▶ Update on a single view without aggregate operations:
 - Update may map to an update on the underlying base table
- ▶ Views involving joins:
 - An update *may* map to an update on the underlying base relations
 - ❖ Not always possible

Un-updatable Views

- ▶ Views defined using groups and aggregate functions are not updateable
- ▶ Views defined on multiple tables using joins are generally not updateable
- ▶ **WITH CHECK OPTION:** must be added to the definition of a view if the view is to be updated
 - To allow check for updatability and to plan for an execution strategy