SQL:
A COMMERCIAL DATABASE LANGUAGE

Data Change Statements, Views

Outline of Chapters 8, 9

1. Introduction
2. Data Definition, Basic Constraints, and Schema Changes
3. Basic Queries
4. More complex Queries
5. Aggregate Functions and Grouping
6. Summary of SQL queries
7. Data Change statements
8. Views

7. Data change statements

- SQL has three commands to change data in a database: INSERT, DELETE, and UPDATE.

7.1 The INSERT command (1)

- 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
- Example**:**
  U1:

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

- It is also possible to insert into a relation *multiple tuples* separated by commas in a single INSERT command.

7.1 The INSERT command (2)

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple.
- The attributes are listed in the same order as the values are listed in the command.
- Example:
  U1A:

  INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)
  VALUES ('Richard', 'Marini', '653298653');

- Non-listed attributes are set to their *DEFAULT value or to NULL*.
- *Thus, the listed attributes* must include *all attributes with NOT NULL specification and no default value*.

7.1 The INSERT command (3)

- The constraints specified in the DDL commands are *automatically enforced by the DBMS when updates are applied to the database*.
- It is the responsibility of the user to check that any constraints whose check is not enforced by the system are not violated.
- Example**:**
  The following insertion is rejected if *referential integrity constraint checking* is provided by the DBMS.

  INSERT INTO EMPLOYEE (FNAME, LNAME, SSN, DNO)
  VALUES (‘Robert'**, ‘**Hatcher'**, ‘**980760540‘, 2**);**

  The following insertion is rejected if *NOT NULL checking* is provided by the DBMS.

  INSERT INTO EMPLOYEE (FNAME, LNAME, DNO)
  VALUES (‘Robert'**, ‘**Hatcher'**,** 5**);**

7.1 The INSERT command (4)

- Another variation of INSERT allows insertion of *multiple tuples* in conjunction with *creating the relation and loading it with the result of a query*.
- Example: Suppose we want to create a temporary table that has the name, number of employees, and total salaries for each department.

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

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

7.1 The INSERT command (5)

- 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 the insert command. We have to create a view (see later) to keep such a table up to date.

7.2 The DELETE command (1)

- The DELETE command removes tuples from a relation.
- It 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).
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause.
- Example**:**

```
DELETE FROM      EMPLOYEE
WHERE   LNAME='Brown' ;
```

7.2 The DELETE command (2)

- Example:
  The following DELETE command will delete at most one tuple. Why?

```
DELETE FROM     EMPLOYEE
WHERE   SSN='123456789' ;
```

- Example:
  The following DELETE commands deletes multiple tuples from the EMPLOYEE relation (and possibly from other relations). Why?

```
DELETE FROM     EMPLOYEE
WHERE   DNO  IN
                (SELECT    DNUMBER
                FROM       DEPARTMENT
                WHERE      DNAME='Research');
```

7.2 The DELETE command (3)

- A missing WHERE-clause specifies that all the tuples of the relation are to be deleted.
- Example:

DELETE FROM EMPLOYEE ;

- Note that EMPLOYEE table remains in the database as an empty table.

7.3 The UPDATE command (1)

- The UPDATE command is 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
- Example**:** The following UPDATE command changes the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

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

7.3 The UPDATE command (2)

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

```
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.

8. Views in SQL

- A view is a single *table* that is derived from other tables.
- The other tables could be base tables or previously defined views.
- A view does not necessarily exist in physical form; it is generally considered a virtual table in contrast to base tables whose tuples are actually stored in the database.
- This limits the possible update operations that can be applied to views.
- There are no limitations on querying a view.

8.1 View specification in SQL (1)

- The **CREATE VIEW** command is used to specify a view by specifying a view name, a list of attribute names, and a defining query.
- If none of the view attributes result from applying functions or arithmetic operations, we do not have to specify attribute names for the view. The view attribute names can be *inherited from* the attribute names of the tables in the defining query.
- Example:

```
CREATE VIEW     WORKS_ON1   AS
    SELECT   FNAME, LNAME, PNAME, HOURS
    FROM     EMPLOYEE, PROJECT, WORKS_ON
    WHERE    SSN=ESSN AND PNO=PNUMBER;
```

WORKS_ON1

FNAME	LNAME	PNAME	HOURS

8.1 View specification in SQL (2)

- Example:

  In the following view, the view attribute names are listed using a one-to-one correspondence with the entries in the SELECT-clause of the defining query.

```
CREATE VIEW   DEPT_INFO (DEPT_NAME, NO_OF_EMPS,
                         TOTAL_SAL) AS
SELECT        DNAME, COUNT (*), SUM(SALARY)
FROM          DEPARTMENT, EMPLOYEE
WHERE         DNUMBER=DNO
GROUP BY      DNAME;
```

DEPT_INFO

DEPT_NAME	NO_OF_EMPS	TOTAL_SAL

8.2 Queries on views (1)

- We can specify SQL queries on a view in the same way we specify queries involving base tables.

- A view can be defined to simplify frequently occurring queries

- Example**:**

  Retrieve the last name and first name of all employees who work on 'ProjectX'.

```
SELECT    FNAME, LNAME
FROM      WORKS_ON1
WHERE     PNAME = 'ProjectX' ;
```

  Without the view WORKS_ON1, this query specification would require two join conditions

8.2 Queries on views (2)

- The DBMS is responsible for keeping the view always up-to-date if the base tables on which the view is defined are modified.

- Hence, the view is *not* realized at the time of *view definition*, but rather at the time we specify a query on the view.

- A view is removed using the DROP view command.

- Example**:**

  DROP VIEW WORKS_ON1 **;**

  DROP VIEW DEPT_INFO **;**

- Views can also be used as a *security and authorization mechanism*.

8.3 View implementation (1)

- Two main approaches have been suggested for implementing a view.

- One approach, called **query modification**, involves rewriting the query that involves a view into a query on the underlying base tables.

  This approach is also called **virtual view approach** because the (answer of the view) view does not have a physical existence.

- The disadvantage of this approach is that it is inefficient for views defined via complex queries.

8.3 View implementation (2)

• Example:

The query
```
SELECT  FNAME, LNAME
FROM    WORKS_ON1
WHERE   PNAME='ProjectX' ;
```

on the view

```
CREATE VIEW    WORKS_ON1   AS
   SELECT  FNAME, LNAME, PNAME, HOURS
   FROM    EMPLOYEE, PROJECT, WORKS_ON
   WHERE   SSN=ESSN AND PNO=PNUMBER;
```

can be rewritten on the base relations:

```
SELECT     FNAME, LNAME
FROM       EMPLOYEE, PROJECT, WORKS_ON
WHERE      SSN=ESSN AND PNO=PNUMBER AND
           PNAME='ProjectX';
```

8.3 View implementation (3)

- The other approach, called **view materialization**, involves physically creating a temporary view table when the view is first queried.

  This table is kept on the assumption that other queries on the view will follow. If it is not queried for a certain period of time, it is automatically dropped by the system.

  If this approach is adopted, an efficient strategy for automatically updating the view table when the base tables are updated must be developed in order to keep the view up to date.