DATA AND SCHEMA MODIFICATIONS

CHAPTERS 4,5 (6/E)

CHAPTER 8 (5/E)

LECTURE OUTLINE

- Updating Databases Using SQL
- Specifying Constraints as Assertions and Actions as Triggers
- Schema Change Statements in SQL

THE INSERT COMMAND

- Adds tuple(s) to a relation
- Needs relation name and a list of values for the tuple(s)
  - Union-compatible
  - Two options for specifying values:
    - Explicit list
    - Result from a `SELECT` statement

```
U1:   INSERT INTO   EMPLOYEE
      VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                      Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );

U3B:  INSERT INTO   WORKS_ON_INFO ( Emp_name, Proj_name,
                    Hours_per_week )
      SELECT        E.Lname, P.Pname, W.Hours
      FROM          PROJECT P, WORKS_ON W, EMPLOYEE E
      WHERE         P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

THE DELETE COMMAND

- Removes tuple(s) from a relation
- Needs relation name and (optionally) a `WHERE` clause to select tuple(s) to be deleted

```
U4A:  DELETE FROM   EMPLOYEE
      WHERE         Lname='Brown';

U4B:  DELETE FROM   EMPLOYEE
      WHERE         Ssn='123456789';

U4C:  DELETE FROM   EMPLOYEE
      WHERE         Dno=5;

U4D:  DELETE FROM   EMPLOYEE;
```

- Where clause can be arbitrarily complex (like for SELECT), including the use of nested SELECT statements

THE UPDATE COMMAND

- Modifies column value(s) in one or more selected tuples
- Needs relation name, column(s) to be modified and new values, and (optionally) `WHERE` clause to select tuple(s) to be modified
  - Required **`SET`** clause in the `UPDATE` command

```
U5:   UPDATE   PROJECT
      SET      Plocation = 'Bellaire', Dnum = 5
      WHERE    Pnumber=10;
```

  - May use old value(s) and relations to determine new value(s)

```
UPDATE EMPLOYEE
SET    Salary = Salary*1.03
WHERE  Dno IN ( SELECT Dnumber
                FROM DEPARTMENT
                WHERE Dname LIKE '%Research%');
```

UPDATES MIGHT FAIL

- Recall: constraints specified in schema declaration (recall DDL)
  1. Inserted tuples might violate domain, uniqueness, referential, or check constraints
  2. Deleted tuples might violate referential constraints

     (why not domain, uniqueness, or check constraints?)
     - Instead of failing, might cause cascaded deletes
  3. Modifications might fail (or cascade) like deletions or insertions

ASSERTIONS

- Other constraints can be declared as **assertions**

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK ( NOT EXISTS  ( SELECT     *
                      FROM       EMPLOYEE E, EMPLOYEE M,
                                 DEPARTMENT D
                      WHERE      E.Salary>M.Salary
                                 AND E.Dno=D.Dnumber
                                 AND D.Mgr_ssn=M.Ssn ) );
```

  - Query that selects tuple(s) that violate the desired condition
    - Non-empty result implies constraint violation
  - Only to be used for cases not otherwise covered

TRIGGERS

- Generalization of cascading deletions
  - Used to monitor the database and enforce business rules
    - Might update derived data in (possibly some other) table
    - Might enforce constraint (e.g., by first updating related data)
    - Might raise an alarm
- Typical trigger has three components:
  - *Event(s)*: Which updates are being monitored? Before/after/instead?
  - *Condition*: What specific data values are of concern?
  - *Action*: What should the system do when the conditions are met?
- Example: *Nobody's salary should be increased by more than 10%.*

```
CREATE TRIGGER Limit_sal
AFTER UPDATE OF Salary ON EMPLOYEE          (event)
REFERENCING OLD ROW AS O, NEW ROW AS N
FOR EACH ROW
WHEN (N.Salary > 1.1*O.Salary)              (condition)
UPDATE EMPLOYEE                             (action)
SET Salary = 1.1*O.Salary;
```

SCHEMA EVOLUTION COMMANDS

- Revise schema declaration as business needs evolve
  - Change set of tables
  - Change attributes within tables
  - Change set of constraints
- Part of DDL rather than DML
  - Contrast to database update commands
- Can be done while the database is operational
- Does not require recompilation of the database schema

THE DROP COMMAND

- `DROP` command
  - Used to drop named schema elements, such as tables, domains, or constraints
- Drop behavior options:
  - `CASCADE` and `RESTRICT`
  - Latter means no ripple-on effects allowed
- Example:

```
DROP SCHEMA COMPANY CASCADE;
```

  - Causes tables, domains, and constraints in schema to be dropped as well
  - With `RESTRICT`, command would only succeed if schema is empty

THE ALTER COMMAND

- Can add a column to a table

```
ALTER TABLE COMPANY.EMPLOYEE
    ADD COLUMN Job VARCHAR(12);
```

- Can drop a column
  - Choose either `CASCADE` or `RESTRICT`
  - `CASCADE` permits constraints on columns to be dropped automatically
- Can alter a column definition
  - Change type, nullability, or default value
- Can add or drop a named table constraint

```
ALTER TABLE COMPANY.EMPLOYEE
    DROP CONSTRAINT EMPSUPERFK;
```

LECTURE SUMMARY

- Database modification commands
- Assertions
- Triggers
- Schema modification commands