Specifying Updates in SQL

- There are three SQL commands to modify the database
  - INSERT,
  - DELETE, and
  - UPDATE
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 )

- 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')
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

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

and is loaded with the summary information retrieved from the database by a query

```sql
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 ;
```
Modification of the Database – Insertion

- Add a new tuple to `account`
  
  ```
  insert into account
  values (‘A-9732’, ‘Perryridge’,1200)
  ```

  or equivalently

  ```
  insert into account (branch_name, balance, account_number)
  values (‘Perryridge’, 1200, ‘A-9732’)
  ```

- Add a new tuple to `account` with `balance` set to null

  ```
  insert into account
  values (‘A-777’, ‘Perryridge’, null )
  ```
Modification of the Database – Insertion

- Gift for all loan customers of the Perryridge branch: a $200 savings account. Let the loan number serve as the account number for the new savings account.

```sql
insert into account
    select loan_number, branch_name, 200
from loan
where branch_name = 'Perryridge'
insert into depositor
    select customer_name, loan_number
from loan, borrower
where branch_name = 'Perryridge'
    and loan.account_number = borrower.account_number
```
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 (cont.)

- Examples:

  DELETE FROM EMPLOYEE
  WHERE LNAME='Brown'

  DELETE FROM EMPLOYEE
  WHERE SSN='123456789'

  DELETE FROM EMPLOYEE
  WHERE DNO IN
    (SELECT DNUMBER
     FROM DEPARTMENT
     WHERE DNAME='Research')
Modification of the Database – Deletion

- Delete all account tuples at the Perryridge branch
  
  ```sql
  delete from account
  where branch_name = 'Perryridge'
  ```

- Delete all accounts at every branch located in the city ‘Needham’.
  
  ```sql
  delete from account
  where branch_name in (select branch_name
                         from branch
                         where branch_city = 'Needham')
  ```
Deletions and Functions

Delete the record of all accounts with balances below the average at the bank.

```
delete from account
where balance < (select avg (balance )
  from account )
```

- Problem: as we delete tuples from deposit, the average balance changes
- Solution used in SQL:
  1. First, compute `avg` balance and find all tuples to delete
  2. Next, delete all tuples found above (without recomputing `avg` or retesting the tuples)
**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
Example: Change 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
```
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.
Modification of the Database – Updates

- Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.
  - Write two `update` statements:
    
    ```sql
    update account
    set balance = balance * 1.06
    where balance > 10000
    
    then
    
    update account
    set balance = balance * 1.05
    where balance <= 10000
    
    The order is important
    
    May use the case statement`
Case Statement for Conditional Updates

Same query as before: Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.

```sql
update account
set balance = case
  when balance <= 10000
    then balance * 1.05
  else balance * 1.06
end
```