SQL: SCHEMA DEFINITION, BASIC CONSTRAINTS, AND QUERIES 1

Week 11

Outline

- Data Definition, Constraints, and Schema Changes
- Updates
- Data retrieval

CREATE DATABASE

- Creates a database with the given name (in SQL command line).

- **CREATE DATABASE *database_name*;**
- **CREATE DATABASE Department;**

- An error occurs if the database exists . So check if the database is existing
- **DROP DATABASE IF EXISTS Department;**
- **CREATE DATABASE Department;**

- Or
- **CREATE DATABASE IF NOT EXISTS Department;**

Data Definition, Constraints, and Schema Changes

- Used to CREATE, DROP, and ALTER the descriptions of the tables (relations) of a database

CREATE TABLE

- Specifies a new base **relation** by
  - Giving it a name
  - Specifying each of its attributes and their data types (*INTEGER*, *FLOAT*, *DECIMAL(i, j)*, *CHAR(n)*, *VARCHAR(n)*)

- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE   DEPARTMENT
(    DNAME VARCHAR(10)      NOT NULL,
     DNUMBER    INTEGER     NOT NULL,
     MGRSSN     CHAR(9),
     MGRSTARTDATE     CHAR(9)  );
```

CREATE TABLE

- In SQL, can use the CREATE TABLE command for specifying the primary key attributes, and referential integrity constraints (foreign keys).

- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
CREATE TABLE   DEPT
(   DNAME   VARCHAR(10)  NOT NULL,
     DNUMBER       INTEGER       NOT NULL,
     MGRSSN CHAR(9),
     MGRSTARTDATE CHAR(9),
     PRIMARY KEY (DNUMBER),
     UNIQUE (DNAME),
     FOREIGN KEY (MGRSSN) REFERENCES EMP  );
```

CREATE TABLE – Primary Key

```
CREATE TABLE   DEPT
(   DNAME VARCHAR(10) NOT NULL,
      DNUMBER INTEGER NOT NULL,
      MGRSSN CHAR(9),
      MGRSTARTDATE  CHAR(9),
      PRIMARY KEY (DNUMBER)
      );
```

or

```
CREATE TABLE   DEPT
(   DNAME VARCHAR(10) NOT NULL,
      DNUMBER INTEGER PRIMARY KEY,
      MGRSSN CHAR(9),
      MGRSTARTDATE  CHAR(9)
      );
```

CREATE TABLE – Multiple Primary Keys

```
CREATE TABLE PROJECT_HOURS
(    PNUMBER INTEGER NOT NULL,
     EMPNO   INTEGER NOT NULL,
     WORKED_HOURS  INTEGER,
     PRIMARY KEY (PNUMBER, EMPNO)
     . . . .
     );
```

REFERENTIAL INTEGRITY OPTIONS

- We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE   DEPT
 (   DNAME VARCHAR(10)      NOT NULL,
     DNUMBER     INTEGER    NOT NULL,
     MGRSSN      CHAR(9),
     MGRSTARTDATE     CHAR(9),
     PRIMARY KEY (DNUMBER),
     UNIQUE (DNAME),
     FOREIGN KEY (MGRSSN) REFERENCES EMP
ON DELETE SET DEFAULT ON UPDATE CASCADE  );
```

REFERENTIAL INTEGRITY OPTIONS (continued)

```
CREATE TABLE   EMP
(ENAME  VARCHAR(30)  NOT NULL,
 ESSN   CHAR(9),
 BDATE  DATE,
 DNO    INTEGER  DEFAULT 1,
 SUPERSSN     CHAR(9),
 PRIMARY KEY (ESSN),
 FOREIGN KEY (DNO) REFERENCES DEPT
    ON DELETE SET DEFAULT ON UPDATE CASCADE,
 FOREIGN KEY (SUPERSSN) REFERENCES EMP
    ON DELETE SET NULL ON UPDATE CASCADE  );
```

Additional Data Types

Has DATE, TIME, and TIMESTAMP data types

- **DATE:**
  - Made up of year-month-day in the format yyyy-mm-dd
- **TIME:**
  - Made up of hour:minute:second in the format hh:mm:ss
- **TIME(i):**
  - Made up of hour:minute:second plus i additional digits specifying fractions of a second
  - format is hh:mm:ss:ii...i
- **TIMESTAMP:**
  - Has both DATE and TIME components

Additional Data Types

- **INTERVAL:**
  - Specifies a relative value rather than an absolute value
  - Can be DAY/TIME intervals or YEAR/MONTH intervals
  - Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value

DROP TABLE

- Used to remove a relation (base table) *and its definition*
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists

```
DROP TABLE DEPENDENT;
```

ALTER TABLE

- Used to add an attribute to one of the base relations

- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is *not allowed* for such an attribute

```
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12);
```

- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

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

- **U1:**

```
INSERT INTO EMPLOYEE
    VALUES ('Richard','K','Marini', '653298653',
'30-DEC-92', '98 Oak Forest, Katy,TX', 'M',
37000,'987654321', 4);
```

INSERT (cont.)

- **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')
```

DELETE

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

DELETE (cont.)

- Examples:

U4A:
```
DELETE FROM     EMPLOYEE
WHERE           LNAME='Brown’
```

U4B:
```
DELETE FROM     EMPLOYEE
WHERE           SSN='123456789’
```

U4C:
```
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 (cont.)

- ***Example:*** ***Change the location and controlling department number of project number 10 to 'Galle' and 5, respectively.***

U5:

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

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the SELECT statement

- This is *not the same as* the SELECT operation of the relational algebra
  - SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values

- Hence, an SQL relation (table) is a *multi-set* (sometimes called a bag) of tuples; it *is not* a set of tuples

- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL (cont.)

- Basic form of the SQL SELECT statement is called a *mapping* or a *SELECT-FROM-WHERE block*

```
SELECT      <attribute list>
FROM        <table list>
WHERE       <condition>
```

  - <attribute list> is a list of attribute names whose values are to be retrieved by the query
  - <table list> is a list of the relation names required to process the query
  - <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Relational Database Schema

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP

Populated Database

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

Simple SQL Queries

- Basic SQL queries correspond to using the SELECT, PROJECT, and JOIN operations of the relational algebra

- All subsequent examples use the COMPANY database
- Example of a simple query on *one* relation

- ***Query 1:* Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.**

```
Q1:  SELECT       BDATE, ADDRESS
     FROM         EMPLOYEE
     WHERE  FNAME='John' AND MINIT='B' AND LNAME='Smith'
```

  - Similar to a SELECT-PROJECT pair of relational algebra operations; the SELECT-clause specifies the *projection attributes* and the WHERE-clause specifies the *selection condition*
  - However, the result of the query *may contain* duplicate tuples

Simple SQL Queries (cont.)

- ***Query 2:*** ***Retrieve the name and address of all employees who work for the 'Research' department.***

```
Q2:  SELECT FNAME, LNAME, ADDRESS
     FROM   EMPLOYEE, DEPARTMENT
     WHERE  DNAME='Research' AND DNUMBER=DNO
```

  - Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations
  - (DNAME='Research') is a *selection condition* (corresponds to a SELECT operation in relational algebra)
  - (DNUMBER=DNO) is a *join condition* (corresponds to a JOIN operation in relational algebra)

Simple SQL Queries (cont.)

- ***Query 3:*** ***For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.***

Q3:
```
SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS
FROM   PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND
       PLOCATION='Stafford'
```

  - In Q3, there are *two* join conditions
  - The join condition DNUM=DNUMBER relates a project to its controlling department
  - The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in *different relations*

- A query that refers to two or more attributes with the same name must *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name

- **EMPLOYEE.LNAME, DEPARTMENT.DNAME**

ALIASES

- Some queries need to refer to the same relation twice
- In this case, *aliases* are given to the relation name

- ***Query 4:* For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.**

Q4:
```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM   EMPLOYEE E S
WHERE  E.SUPERSSN=S.SSN
```

  - In Q4, the alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation

  - We can think of E and S as two *different copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*

ALIASES (cont.)

- Aliasing can also be used in any SQL query for convenience
  Can also use the **AS** keyword to specify aliases

Q4:
```
SELECT    E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM      EMPLOYEE AS E, EMPLOYEE AS S
WHERE     E.SUPERSSN=S.SSN
```

UNSPECIFIED WHERE-clause

- A *missing WHERE-clause* indicates no condition; hence, *all tuples* of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE

- ***Query 5:*** ***Retrieve the SSN values for all employees.***

```
Q5: SELECT    SSN
    FROM      EMPLOYEE
```

- If more than one relation is specified in the FROM-clause *and* there is no join condition, then the *CARTESIAN PRODUCT* of tuples is selected

UNSPECIFIED WHERE-clause (cont.)

Q6:
```
SELECT      SSN, DNAME
FROM        EMPLOYEE, DEPARTMENT
```

- It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*

Q7A:

```
SELECT      *
FROM  EMPLOYEE
WHERE DNO=5
```

Q7B:

```
SELECT      *
FROM  EMPLOYEE, DEPARTMENT
WHERE DNAME='Research' AND DNO=DNUMBER
```

USE OF DISTINCT

- SQL does not treat a relation as a set; *duplicate tuples can appear*
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used

- For example, the result of Q8 may have duplicate SALARY values whereas Q8A does not have any duplicate values

Q8:

```
SELECT     SALARY
FROM  EMPLOYEE
```

Q8A:

```
SELECT DISTINCT SALARY
FROM  EMPLOYEE
```

INSERT WITH SELECT

- 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
GROUP BY     DNAME ;
```

INSERT WITH SELECT

- 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 WITH SELECT

- Examples:

- **Delete employees in Research department**

U4C:

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

UPDATE WITH SELECT

- ***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