Chapter 8
SQL - The relational DB Standard

SQL1: ANSI Standard 1986
SQL2: ANSI Standard 1992
SQL3: Recently being developed
Data Definition in SQL

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, and specifying each of its attributes and their data types (INTEGER, FLOAT, Number(i,j), CHAR(n), VARCHAR2(n))
- A constraint NOT NULL may be specified on an attribute
Create Table (1)

CREATE TABLE DEPARTMENT
    ( DNAME VARCHAR2(10) NOT NULL,
      DNUMBER INTEGER NOT NULL,
      MGRSSN CHAR(9),
      MGRSTARTDATE CHAR(9) );

In SQL2, can use the CREATE TABLE command for specifying the primary key attributes, secondary keys, and referential integrity constraints (foreign keys)

- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases
CREATE TABLE DEPT
  (  DNAME VARCHAR2(10) NOT NULL,
      DNUMBER INTEGER NOT NULL,
      MGRSSN CHAR(9),
      MGRSTARTDATE CHAR(9),
      PRIMARY KEY (DNUMBER),
      UNIQUE (DNAME),
      FOREIGN KEY (MGRSSN)
      REFERENCES EMPLOYEE );
Drop Table

**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

**Example:**

```
DROP TABLE DEPENDENT;
DROP TABLE EMPLOYEE CASCADE CONSTRAINTS;
```
Alter Table

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

Example:
ALTER TABLE EMPLOYEE ADD JOB VARCHAR2(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.
Alter Table Drop Column:

ALTER TABLE employee
  DROP address CASCADE;
Removes all views and referential integrity constraints
  that refer to this column.

ALTER TABLE employee
  DROP address RETRICT;
Succeeds if no views or foreign keys refer to this
column.

Can also drop default clauses, change default values, and drop column constraints.
Referential Integrity Options

In SQL2, we can specify

  CASCADE
  SET NULL
  SET DEFAULT

on referential integrity constraints (foreign keys)
CREATE TABLE EMPLOYEE  
(  FNAME VARCHAR2(30) NOT NULL,  MINIT CHAR(1),  LNAME VARCHAR2(30),  SSN CHAR(9),  BDATE DATE,  ADDRESS VARCHAR2(100),  SEX CHAR(1) CHECK (SEX in ('M', 'F')),  SALARY NUMBER(10,2),  SUPERSSN CHAR(9),  DNO INTEGER NOT NULL DEFAULT 1,  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 ) ;
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
- Important distinction between SQL and the formal relational model; 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
Basic form of the SQL SELECT statement is called a mapping or a *SELECT-FROM-WHERE* block.

```sql
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.
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 0

 Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

 SELECT  BDATE, ADDRESS
 FROM    EMPLOYEE
 WHERE   FNAME='John' AND
       MINIT='B' AND
       LNAME='Smith'
Query 0 (2)

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
Query 1

Retrieve the name and address of all employees who work for the 'Research' department.

SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE, DEPARTMENT
WHERE DNAME='Research'
AND DNUMBER=DNO
Query 1 (2)

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)
Query 2

For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and bdate.

```
SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS
FROM   PROJECT, DEPARTMENT, EMPLOYEE
WHERE  DNUM=DNUMBER AND MGRSSN=SSN AND PLOCATION='Stafford'
```
Query 2 (2)

- In Q2, 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.
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

**Example:** EMPLOYEE.LNAME, DEPARTMENT.DNAME

Some queries need to refer to the same relation twice, In this case, **aliases** are given to the relation name
Query 8

For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME
FROM EMPLOYEE E S
WHERE E.SUPERSSN=S.SSN
```
A missing WHERE-clause indicates no condition; hence, \textit{all tuples} of the relations in the FROM-clause are selected.

This is equivalent to the condition WHERE TRUE.

Query 9: Retrieve the SSN values for all employees.

\textbf{SELECT SSN}

\textbf{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.

Example:

```
SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT
```
USE OF *:

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

**Examples:**

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

```sql
SELECT * 
FROM EMPLOYEE, DEPARTMENT 
WHERE DNAME='Research' AND DNO=DNUMBER
```
USE OF DISTINCT

SQL does not treat a relation as a set
To eliminate duplicate tuples in a query result, the keyword `DISTINCT` is used

For example,

Q11

```
SELECT SALARY
FROM EMPLOYEE
```

Q11A

```
SELECT DISTINCT SALARY
FROM EMPLOYEE
```
Set Operations

- SQL has directly incorporated some set operations.
- There is a union operation (UNION), and in some versions of SQL there are set difference (MINUS) and intersection (INTERSECT) operations.
- The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result.
- The set operations apply only to union compatible relations; the two relations must have the same attributes and the attributes must appear in the same order.
Query 4

Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

(SELECT PNAME
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')

UNION

(SELECT PNAME
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE PNUMBER=PNO AND ESSN=SSN AND LNAME='Smith')
NESTING OF QUERIES

A complete SELECT query, called a *nested query*, can be specified within the WHERE-clause of another query, called the *outer query*

Many of the previous queries can be specified in an alternative form using nesting
Query 1

Retrieve the name and address of all employees who work for the 'Research' department.

```
SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE
WHERE DNO IN (SELECT DNUMBER
               FROM DEPARTMENT
               WHERE DNAME='Research')
```
Query 1 (2)

- The nested query selects the number of the 'Research' department.
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query.
- The comparison operator **IN** compares a value \( v \) with a set (or multi-set) of values \( V \), and evaluates to **TRUE** if \( v \) is one of the elements in \( V \).
- A reference to an *unqualified attribute* refers to the relation declared in the *innermost nested query*.
- In this example, the nested query is *not correlated* with the outer query.
CORRELATED NESTED QUERIES

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated.

The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query
Query 12

Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
SELECT  E.FNAME, E.LNAME
FROM EMPLOYEE AS E
WHERE E.SSN IN (SELECT ESSN
    FROM DEPENDENT
    WHERE ESSN=E.SSN AND E.FNAME=DEPENDENT_NAME)
```
The EXISTS function

- EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not.
- We can formulate Query 12 in an alternative form that uses EXISTS as Q12B below.
Query 12

Retrieve the name of each employee who has a dependent with the same first name as the employee.

Q12B:

```sql
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE   EXISTS  (SELECT *
                         FROM    DEPENDENT
                         WHERE   SSN=ESSN AND
                                FNAME=DEPENDENT_NAME)
```
Query 6

Retrieves the names of employees who have no dependents

```
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE   NOT EXISTS (SELECT * 
                 FROM    DEPENDENT
                 WHERE   SSN=ESSN)
```
NULLS IN SQL QUERIES

SQL allows queries that check if a value is NULL (missing or undefined or not applicable).

SQL uses **IS** or **IS NOT** to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

Retrieve the names of all employees who do not have supervisors.

Q14:  

```
SELECT  FNAME, LNAME
FROM    EMPLOYEE
WHERE   SUPERSSN IS NULL
```
Aggregate Functions

- Include **COUNT, SUM, MAX, MIN, and AVG**

**Q(15)** Find the maximum salary, the minimum salary, and the average salary among all employees.

```sql
SELECT MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE
```

- Some SQL implementations *may not allow more than one function* in the SELECT-clause.
GROUPING

In many cases, we want to apply the aggregate functions to subgroups of tuples in a relation.

Each subgroup of tuples consists of the set of tuples that have the same value for the grouping attribute(s).

The function is applied to each subgroup independently.

SQL has a GROUP BY-clause for specifying the grouping attributes, which must also appear in the SELECT-clause.
Query 20

For each department, retrieve the department number, the number of employees in the department, and their average salary.

```sql
SELECT DNO, COUNT (*), AVG(SALARY)
FROM EMPLOYEE
GROUP BY DNO
```
Query 21

For each project, retrieve the project number, project name, and the number of employees who work on that project.

SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME

In this case, the grouping and functions are applied after the joining of the two relations.
THE HAVING-CLAUSE

Sometimes we want to retrieve the values of these functions for only those groups that satisfy certain conditions.

The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples).
Query 22

For each project *on which more than two employees work*, retrieve the project number, project name, and the number of employees who work on that project.

```
SELECT PNUMBER, PNAME, COUNT(*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT(*) > 2
```
SUBSTRING COMPARISON

The **LIKE** comparison operator is used to compare partial strings.

Two reserved characters are used: '%' (or '*' in some implementations) replaces an arbitrary number of characters, and '_' replaces a single arbitrary character.
SUBSTRING COMPARISON (2)

- Retrieve all employees whose address is in Houston, Texas. (i.e.'Houston,TX')
  ```sql
  SELECT FNAME, LNAME
  FROM EMPLOYEE
  WHERE ADDRESS LIKE '%Houston,TX%'
  ```

- Retrieve all employees who were born during the 1950s.
  ```sql
  SELECT FNAME, LNAME
  FROM EMPLOYEE
  WHERE BDATE LIKE '_________5_'
  ```

- The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible; hence, in SQL, character string attribute values are not atomic.
The standard arithmetic operators '+', '-', '*', and '/' can be applied to numeric values in an SQL query result.

Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
SELECT FNAME, LNAME, 1.1*SALARY
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE SSN=ESSN AND
PNO=PNUMBER AND
PNAME='ProductX'
```
ORDER BY

Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

```
SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
ORDER BY DNAME, LNAME
```