
CS3402: Chapter 5

SQL: Structured Query Language

SQL: Structured Query Language

- Data Definition Language(DDL): define the structures in a database, i.e. create, modify, and remove database objects such as tables, indexes, and users. Commands include **CREATE**, **ALTER**, and **DROP**.
- Data Manipulation Language (DML): deal with the manipulation of data present in database. Commands include : **INSERT**, **UPDATE** , and **DELETE**.
- Data Query Language (DQL): make queries for data in databases. Commands include : **SELECT**.

Note:

- Each statement in SQL ends with a semicolon (;)
- SQL commands are case insensitive
- Despite the existence of the standard, SQL code is not completely portable among different database systems without adjustments.

DDL: Define database

CREATE Table

- Create Database:

CREATE SCHEMA database_name AUTHORIZATION user-name;

- Create Table: CREATE TABLE table_name
(column_name1 data_type(size),
column_name2 data_type(size),....);

- Delete Table: DROP TABLE table-name;

- Update Table:

ALTER TABLE table-name
ADD Aj, Dj;

(to add new attribute Aj with
domain Dj to an existing table)

ALTER TABLE table-name
DROP Aj;

(to delete attribute Aj from
an existing table)

SQL CREATE TABLE data definition statements for defining the COMPANY schema

Standard data types

CREATE TABLE EMPLOYEE

(Fname	VARCHAR(15)	NOT NULL,
Minit	CHAR,	
Lname	VARCHAR(15)	NOT NULL,
Ssn	CHAR(9)	NOT NULL,
Bdate	DATE,	
Address	VARCHAR(30),	
Sex	CHAR,	
Salary	DECIMAL(10,2),	
Super_ssn	CHAR(9),	
Dno	INT	NOT NULL,

PRIMARY KEY (Ssn),

CREATE TABLE DEPARTMENT

(Dname	VARCHAR(15)	NOT NULL,
Dnumber	INT	NOT NULL,
Mgr_ssn	CHAR(9)	NOT NULL,
Mgr_start_date	DATE,	

PRIMARY KEY (Dnumber),

UNIQUE (Dname),

FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) ;

CREATE TABLE DEPT_LOCATIONS

(Dnumber	INT	NOT NULL,
Dlocation	VARCHAR(15)	NOT NULL,

PRIMARY KEY (Dnumber, Dlocation),

FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) ;

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Data Types and Domains: Numeric

- **Numeric** data types include integer numbers of various sizes (**INTEGER** or **INT**, and **SMALLINT**) and floating-point numbers of various precision (**FLOAT** or **REAL**, and **DOUBLE PRECISION**).
- Formatted numbers can be declared by using **DECIMAL(i, j)** , where i (i.e., the precision) is the total number of decimal digits and j (i.e., the scale) is the number of digits after the decimal point. The default for scale is zero, and the default for precision is implementation-defined.

Data Types and Domains: Character string

- **Character string** data types are either fixed length **CHAR(n)** or **CHARACTER (n)**, where n is the number of characters or varying length **VARCHAR (n)** or **CHAR VARYING(n)** or **CHARACTER VARYING(n)**, where n is the maximum number of characters.
- For fixed length strings , a shorter string is **padded with blank characters** to the right. For example, if the value 'Smith' is for an attribute of type CHAR(10), it is padded with five blank characters to become 'Smith' if needed. Padded blanks are generally ignored when strings are compared.
- When specifying a string value, it is placed between single quotation marks (apostrophes), and it is case sensitive (a distinction is made between uppercase and lowercase)

Data Types and Domains: Date and Time

- The **DATE** data type has ten positions, and its components are YEAR, MONTH, and DAY in the form **YYYY-MM-DD**.
- The **TIME** data type has at least eight positions, with the components HOUR, MINUTE, and SECOND in the form **HH:MM:SS**.
- Literal values are represented by single-quoted strings preceded by the keyword **DATE** or **TIME**; for example, DATE '2014-09-27' or TIME '09:12:47'.
- Only valid dates and times should be allowed by the SQL implementation.

Data Types and Domains: Boolean

- A **Boolean** data type is defined as **Boolean** and has the traditional values of TRUE or FALSE. In SQL, because of the presence of NULL values, a three-valued logic is used, so a third possible value for a Boolean data type is UNKNOWN.

3-valued
logic:

Not	
T	F
U	U
F	T

AND	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

OR	T	U	F
T	T	T	T
U	T	U	U
F	T	U	F

And

- Only T-T returns T
- And-ing F with anything results with F
- The rest is UNKNOWN

Or

- Only F-F returns F
- Or-ing T with anything results with T
- The rest is UNKNOWN

SQL CREATE TABLE data definition statements for defining the COMPANY schema

```
CREATE TABLE EMPLOYEE
( Fname VARCHAR(15),
  Minit CHAR,
  Lname VARCHAR(15),
  Ssn CHAR(9) NOT NULL,
  Bdate DATE,
  Address VARCHAR(30),
  Sex CHAR,
  Salary INT,
  Super_ssn INT,
  Dno INT );
PRIMARY KEY (Ssn);
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn);

CREATE TABLE DEPARTMENT
( Dname VARCHAR(15) NOT NULL,
  Dnumber INT NOT NULL,
  Mgr_ssn INT NOT NULL,
  Mgr_start_date DATE,
  Plocation VARCHAR(15) );
PRIMARY KEY (Dnumber),
UNIQUE (Dname),
FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn);

CREATE TABLE DEPT_LOCATIONS
( Dnumber INT NOT NULL,
  Dlocation VARCHAR(15) NOT NULL );
PRIMARY KEY (Dnumber, Dlocation),
FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber);
```

Define primary key

ensures that all values in a column are different

Indicate this column does not accept NULL value.

NOT NULL,

NOT NULL,

NOT NULL,

NOT NULL,

NOT NULL,

NOT NULL,

NOT NULL,

NOT NULL,

NOT NULL,

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	-----	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
-------	---------	---------	----------------

DEPT_LOCATIONS

Dnumber	Dlocation
---------	-----------

PROJECT

Pname	Pnumber	Plocation	Dnum
-------	---------	-----------	------

WORKS_ON

Essn	Pno	Hours
------	-----	-------

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
------	----------------	-----	-------	--------------

Define foreign key and its reference

Attribute Constraints

- Because SQL allows NULLs as attribute values, a constraint **NOT NULL** may be specified if NULL is not permitted for a particular attribute. This is always implicitly specified for the attributes that are part of the primary key, but it can be specified for any other attributes whose values are required not to be NULL
- The **PRIMARY KEY** clause specifies one or more attributes that make up the primary key of a relation.
- The **UNIQUE** clause specifies alternate (unique) keys, also known as candidate keys.
- Referential integrity is specified via the **FOREIGN KEY** clause.

Adding Constrains after creating table

```
ALTER TABLE COUNTRY ADD CONSTRAINT PK_country PRIMARY KEY (cntry_cd);
```

```
ALTER TABLE EXCHANGE ADD CONSTRAINT PK_exchange PRIMARY KEY  
(exchg_cd);
```

```
ALTER TABLE EXCHANGE ADD CONSTRAINT FK_exchg_cntry FOREIGN KEY  
(cntry_cd) REFERENCES COUNTRY (cntry_cd) ;
```

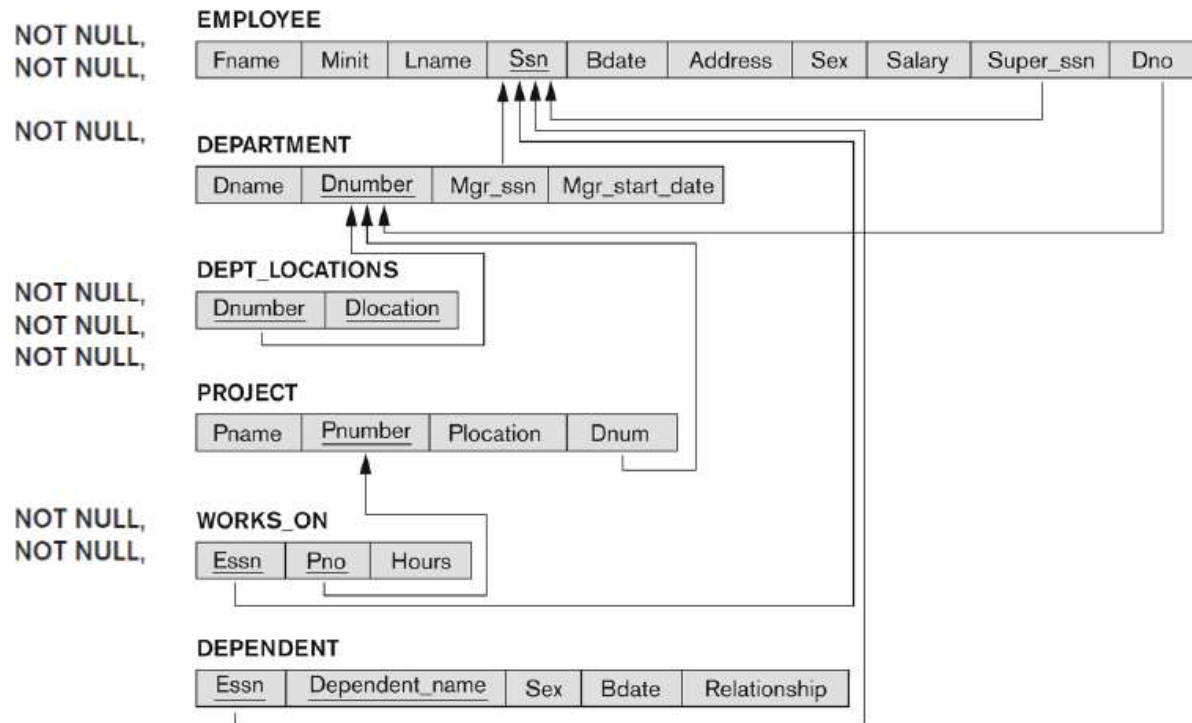
SQL CREATE TABLE data definition statements for defining the COMPANY schema

```

CREATE TABLE PROJECT
( Pname          VARCHAR(15)          NOT NULL,
  Pnumber        INT                  NOT NULL,
  Plocation      VARCHAR(15),
  Dnum           INT
  PRIMARY KEY (Pnumber),
  UNIQUE (Pname),
  FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );

CREATE TABLE WORKS_ON
( Essn          CHAR(9)          NOT NULL,
  Pno           INT              NOT NULL,
  Hours         DECIMAL(3,1)     NOT NULL,
  PRIMARY KEY (Essn, Pno),
  FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
  FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );

CREATE TABLE DEPENDENT
( Essn          CHAR(9)          NOT NULL,
  Dependent_name VARCHAR(15)     NOT NULL,
  Sex           CHAR,
  Bdate        DATE,
  Relationship   VARCHAR(8),
  PRIMARY KEY (Essn, Dependent_name),
  FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
  
```



One possible database state for the COMPANY relational database schema

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	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	1968-01-19	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

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

One possible database state for the COMPANY relational database schema

WORKS_ON

<u>Essn</u>	<u>Pno</u>	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

<u>Pname</u>	<u>Pnumber</u>	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

<u>Essn</u>	<u>Dependent_name</u>	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

DQL: make query of data

The *SELECT-FROM-WHERE* Structure of Basic SQL Queries

- Queries in SQL can be very complex. We will start with simple queries.
- The basic form of the **SELECT** statement formed of the three clauses **SELECT**, **FROM**, and **WHERE** and has the following form:

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

where

- <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.

Specified attributes



Satisfy the
conditions →

Title	Year	Length	Type
Star War	1977	124	Color
Mighty Duck	1991	104	Color
Wayne's World	1992	95	Color

Retrieval from a single table

Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

Q0: **SELECT** Bdate, Address ← Projection attributes
 FROM EMPLOYEE
 WHERE Fname='John' AND Minit='B' AND Lname='Smith'; ← Selection conditions

<u>Bdate</u>	<u>Address</u>
1965-01-09	731Fondren, Houston, TX

- In SQL, the basic logical comparison operators for comparing attribute are =, <, <=, >, >=, and <>.
- Logic operator (**AND**, **OR**, **NOT**) can be used to connect multiple conditions. Priority is : NOT, AND, OR.
E.g. A AND B OR NOT C equals to (A AND B) OR (NOT C)

Retrieval from two tables

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

Q1: **SELECT** Fname, Lname, Address
 FROM EMPLOYEE, DEPARTMENT ← Two tables
 WHERE Dname='Research' **AND** Dnumber=Dno; ← Join conditions

<u>Fname</u>	<u>Lname</u>	<u>Address</u>
John	Smith	731 Fondren, Houston, TX
Franklin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

- The condition Dname='Research' is a **selection condition** that chooses the particular tuple of interest in the DEPARTMENT table, because Dname is an attribute of DEPARTMENT.
- The condition Dnumber=Dno is called a **join condition**, because it combines two tuples: one from DEPARTMENT and one from EMPLOYEE, whenever the value of Dnumber in DEPARTMENT is equal to the value of Dno in EMPLOYEE.

Retrieval from two tables:Join Operation

A	B
ABC	DEF

X
JOIN

C	D	E	F
XYZ			
AAAA			
BBBB			
CCCC			

A	B	C	D	E	F
ABC	DEF	XYZ			
ABC	DEF	AAAA			
ABC	DEF	BBBB			
ABC	DEF	CCCC			

Retrieval from three tables

(c)

<u>Pnumber</u>	<u>Dnum</u>	<u>Lname</u>	<u>Address</u>	<u>Bdate</u>
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

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 birth date.

Q2: **SELECT** Pnumber, Dnum, Lname, Address, Bdate
 FROM PROJECT, DEPARTMENT, EMPLOYEE
 WHERE Dnum=Dnumber AND Mgr_ssn=Ssn AND
 Plocation='Stafford';

← Select-project-join query

Project joins
Department

Department joins
Employee

NULL Value Comparisons

- When a record with NULL in one of its attributes is involved in a comparison operation, the result is considered to be UNKNOWN (it may be TRUE or it may be FALSE).

```
SELECT Essn,  
FROM WORKS_ON  
WHERE HOURS > 0
```

(UNKNOWN result will not return!)

- In most SQL-based DBMSs, the special keyword NULL may be used to test for a NULL value.
- E.g.,

```
SELECT c-name  
FROM Deposit  
WHERE balance IS NULL;  
(or balance IS NOT NULL;)
```

NULL Values Comparisons


Condition	Value of a	Evaluation
a IS NULL	10	FALSE
a IS NOT NULL	10	TRUE
a IS NULL	NULL	TRUE
a IS NOT NULL	NULL	FALSE
a = NULL	10	UNKNOWN
a <> NULL	10	UNKNOWN
a = NULL	NULL	UNKNOWN
a <> NULL	NULL	UNKNOWN
a = 10	NULL	UNKNOWN
a = 10	NULL	UNKNOWN

Substring Pattern Matching

- The **LIKE** comparison allows comparison conditions on only parts of a character string, using operator. This can be used for string pattern matching.
- Partial strings are specified using two reserved characters: percentage (%) replaces an arbitrary number of zero or more characters, and the underscore () replaces a single character.

- Find the names of all employees whose first name has the substring 'mm' included

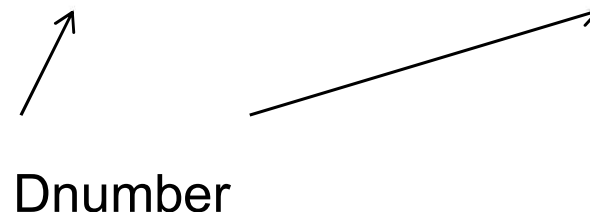
```
SELECT Fname, Minit, Lname
FROM EMPLOYEE
WHERE Fname LIKE '%mm%';
```

(Note: if we use ' __mm%', then it becomes a special case)

 3rd character (case sensitive)

Ambiguous Attribute Names

- Same name can be used for two (or more) attributes in different relations, which may cause ambiguity.
 - ◆ Must **qualify** the attribute name with the relation name to prevent ambiguity
 - ◆ This is done by prefixing the relation name to the attribute name and separating the two by a period, e.g.,
 - *Find the first name and address of employee who belongs to the Research department.*

```
Q1A:  SELECT  Fname, Address
        FROM    EMPLOYEE, DEPARTMENT
        WHERE   DEPARTMENT.Name='Research' AND
                DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```



Aliasing, and Renaming

- The ambiguity of attribute names also arises in the case of queries that refer to the same relation twice.
 - ◆ Must declare alternative relation, called aliases or tuple variables using **AS**
 - ◆ E.g. Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:

Query 8. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname  
      FROM EMPLOYEE AS E, EMPLOYEE AS S  
      WHERE E.Super_ssn=S.Ssn;
```

Aliasing, and Renaming

- We can use this mechanism to rename any table in the WHERE clause, whether or not the same relation needs to be referenced more than once. In fact, this practice is recommended since it results in queries that are easier to comprehend.

- The attribute names can also be renamed

```
EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex,  
              Sal, Sssn, Dno)
```

- The “AS” may be dropped in most SQL implementations

```
EMPLOYEE E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex,  
           Sal, Sssn, Dno)
```

Unspecified WHERE Clause

- Missing **WHERE** clause

- ◆ Indicates no condition on tuple selection (select ALL)

Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

Q9: **SELECT** Ssn
 FROM EMPLOYEE;

Q10: **SELECT** Ssn, Dname
 FROM EMPLOYEE, DEPARTMENT;

Ssn
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

Values of all
tuples



All possible
combinations



Ssn	Dname
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

Use of the Asterisk

- Specify an asterisk (*)
 - ◆ Retrieve all the attribute values of the selected tuples

Q1C: **SELECT** *

FROM EMPLOYEE

WHERE Dno=5;

Q1D: **SELECT** *

FROM EMPLOYEE, DEPARTMENT

WHERE Dname='Research' **AND** Dno=Dnumber;

Q10A: **SELECT** *

FROM EMPLOYEE, DEPARTMENT;

←

EMPLOYEE: 100 tuples
DEPARTMENT: 10 tuples
Finally, 1000 tuples and all attributes

Ordering Tuples

- SQL allows the user to order the tuples in the result of a query by the values of one or more of the attributes that appear in the query result, by using the **ORDER BY** clause.

- List first name of all employees in alphabetic order

```
SELECT Fname  
FROM EMPLOYEE  
ORDER BY Fname;
```

By default, in ascending order.

- List the employee names in descending order of last name, and if several employees have the same last name, order them in ascending order by the first name

```
SELECT Fname, Minit, Lname  
FROM EMPLOYEE  
ORDER BY Lname DESC, Fname ASC;
```

Tables as Sets in SQL

- SELECT does not automatically eliminate duplicate tuples (the attributes of two tuples have same values) in query results (**NOT** a set)
- Use the keyword **DISTINCT** in the SELECT clause
 - ◆ Only distinct tuples should remain in the result

Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).

Q11: **SELECT** **ALL** Salary
 FROM EMPLOYEE;

Q11A: **SELECT** **DISTINCT** Salary
 FROM EMPLOYEE;

ALL:

Salary
30000
40000
25000
43000
38000
25000
25000
55000

Distinct:

Salary
30000
40000
25000
43000
38000
55000

- Specifying SELECT with neither ALL nor DISTINCT is equivalent to SELECT ALL.

Tables as Sets in SQL

■ Set operations

- ◆ UNION, INTERSECT, MINUS/EXCEPT These set operations apply only to type compatible relations, so we must make sure that the two relations on which we apply the operation have the same attributes and that the attributes appear in the same order in both relations.

Query 4. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

Q4A: (SELECT DISTINCT Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
AND Lname='Smith')

UNION
(SELECT DISTINCT Pnumber
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Essn=Ssn
AND Lname='Smith');

Same attribute

The projects in the dept
with "Smith" as manager

The projects with
"Smith" as the worker

Tables as Sets in SQL

- The relations resulting from these set operations `UNION`, `INTERSECT`, `MINUS/EXCEPT` are sets of tuples; that is, duplicate tuples are eliminated from the result.
- If we do not want to eliminate the duplicate tuples, we should use multiset operations `UNION ALL`, `INTERSECT ALL`, `MINUS/EXCEPT ALL`.

(a)	<table><tr><th>R</th></tr><tr><td>A</td></tr><tr><td>a1</td></tr><tr><td>a2</td></tr><tr><td>a2</td></tr><tr><td>a3</td></tr></table>	R	A	a1	a2	a2	a3	<table><tr><th>S</th></tr><tr><td>A</td></tr><tr><td>a1</td></tr><tr><td>a2</td></tr><tr><td>a4</td></tr><tr><td>a5</td></tr></table>	S	A	a1	a2	a4	a5
R														
A														
a1														
a2														
a2														
a3														
S														
A														
a1														
a2														
a4														
a5														
(b)	<table><tr><th>T</th></tr><tr><td>A</td></tr><tr><td>a1</td></tr><tr><td>a1</td></tr><tr><td>a2</td></tr><tr><td>a2</td></tr><tr><td>a2</td></tr><tr><td>a3</td></tr><tr><td>a4</td></tr><tr><td>a5</td></tr></table>	T	A	a1	a1	a2	a2	a2	a3	a4	a5			
T														
A														
a1														
a1														
a2														
a2														
a2														
a3														
a4														
a5														
(c)	<table><tr><th>T</th></tr><tr><td>A</td></tr><tr><td>a2</td></tr><tr><td>a3</td></tr></table>	T	A	a2	a3									
T														
A														
a2														
a3														
(d)	<table><tr><th>T</th></tr><tr><td>A</td></tr><tr><td>a1</td></tr><tr><td>a2</td></tr></table>	T	A	a1	a2									
T														
A														
a1														
a2														

Figure 6.5

The results of SQL multiset operations. (a) Two tables, R(A) and S(A). (b) R(A) UNION ALL S(A). (c) R(A) EXCEPT ALL S(A). (d) R(A) INTERSECT ALL S(A).

DML: Manipulate data

INSERT Command

- **INSERT** is used to add a single tuple (row) to a relation (table). We must specify the relation name and a list of values for the tuple.

```
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 values for the attributes are obtained from the results of the SELECT statement

DELETE Command

- **DELETE** command removes tuples from a relation.

DELETE FROM table-name;

- ◆ (Note: this operation only deletes all tuples from the table and the table is still there)
- ◆ Includes a **WHERE** clause to select the tuples 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;

UPDATE Command

- **UPDATE** command is used to modify attribute values of one or more selected tuples.
- Additional **SET** clause in the **UPDATE** command
 - ◆ Specifies attributes to be modified and new values

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

Views (Virtual Tables)

- Base table (base relation)
 - ◆ Relation and its tuples are **actually** (physically) created and stored as a **file** by the DBMS
 - ◆ The tables are stored in the secondary storage in the **specified** format

- Virtual table (view)
 - ◆ Single table **derived** from other base tables temporarily.
 - ◆ A view does not necessarily exist in physical form; it is just presented to the user through reconstruction (view) of base tables.

Views (Virtual Tables)

- CREATE VIEW command

- ◆ In V1, attributes retain the names from base tables. In V2, attributes are assigned new names

```
V1:  CREATE VIEW  WORKS_ON1
      AS SELECT   Fname, Lname, Pname, Hours
          FROM     EMPLOYEE, PROJECT, WORKS_ON
          WHERE    Ssn=Essn AND Pno=Pnumber;
```

```
V2:  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;
```

New attribute
names



- We can think of a view as a way of specifying a table that we need to reference frequently, even though it may not exist physically.

Views (Virtual Tables)

- Once a View is defined, SQL queries can use the View relation in the FROM clause. Views can be regarded and retrieved as ordinary tables.

E.g.,

```
SELECT Fname  
FROM WORKS_ON1  
WHERE HOURS > 5.0;
```

- View is always up-to-date
 - ◆ Responsibility of the DBMS and not the user
 - ◆ Change in base table will be reflected in the views
- DROP VIEW command
 - ◆ DROP VIEW WORKS_ON1;

Views (Virtual Tables)

- View has limits on data modification operations.

e.g. Suppose we insert a tuple (“Mary”, “Black”, “ProjectX”, 10.0) into WORKS_ON1, it will cause other attribute has NULL value in the base table: (“Mary”, NULL, “Black”, NULL, ...) in EMPLOYEE,

- To avoid such problems and to simplify implementation, most SQL-based DBMSs restrict the following condition:

“A modification is permitted through a view ONLY IF the view is defined in terms of ONE base relation.”