International Institute of Information Technology Bangalore

Introduction to Basic SQL (Structured Query Language)

CS 301

“Goal of the chapter”

“Learn about a higher-level declarative language - SQL.”

1. Describe the SQL DDL commands for creating schemas and tables and provide overview of the basic data types.

2. Specify basic constraints such as key and referential integrity.

3. Describe SQL constructs and commands for retrieval, insert, delete and update.

SQL: Background

- User only specifies what the result is to be, leaving the actual optimization and decisions on how to execute the query to the DBMS.

- Originally, SQL was called SEQUEL (Structured English QUEry Language), designed and implemented by IBM Research.

- SQL:2011 is the updated version (a DDL and DML) that has statements for data definitions, queries, updates, with facilities for views, specifying security and authorization, for defining integrity constraints, and for specifying transaction controls. It can be embedded into Java, C and C++.

- Additionally, SQL has extensions for data mining, spatial data, temporal data, data warehousing, OLAP (online analytical processing), multimedia data, etc.

SQL data definition and data types

- The main SQL command for data definition is the CREATE statement that is used to create schemas, relations, types and domains as well as constructs such as views, assertions, and triggers.

- The concept of SQL schema is to group together "*tables and other constructs*" that belong to the same DB application.

- A SQL schema is identified by a *schema name* and includes an *authorization identifier* to indicate the user or account who owns the schema, as well as *descriptors* for each element in the schema.
  - Schema elements include relations, types, constraints, views, domains, authorization grants, etc.

- A schema is created via the CREATE SCHEMA statement, which can include all the schema element's definitions.

- **CREATE SCHEMA** COMPANY **AUTHORIZATION** 'Jsmith';

  Note: Each statement in SQL ends with a semicolon (;).

- DB have default environment and schema, so a user upon login can refer directly to tables and other constructs within that schema without specifying schema name.

- A catalog contains a special schema called INFORMATION_SCHEMA, which provides information on all the schemas in the catalog and all the element descriptors in these schemas.

- Schemas within the same catalog can also share certain elements, such as type and domain definitions.

CREATE TABLE command

Example:

CREATE TABLE EMPLOYEE

- It is used to specify a new relation by giving it a **name** and specifying its **attributes and initial constraints**.

- Each **attribute is given a name and datatype** to specify its domain of values, and attribute constraints, such as NOT NULL.

- Key, entity integrity and referential integrity constraints are also specified in the CREATE TABLE statement or can be added later using **ALTER TABLE** command.

Example

```
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) );
CREATE TABLE PROJECT
        ( Pname          VARCHAR(15)        NOT NULL,
          Pnumber        INT                NOT NULL,
          Plocation      VARCHAR(15),
          Dnum           INT                NOT NULL,
        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) );
```

- We can also attach the schema name to the relation name, separated by a period, example:

CREATE TABLE COMPANY.EMPLOYEE

- By this, we can explicitly make the EMPLOYEE table part of the COMPANY schema.

- The attributes are considered to be ordered in the sequence in which they are specified in the CREATE TABLE statement, however, tuples are not considered to be ordered within a table (relation).

Attribute data types and domains in SQL

- Numeric
- Character-string
- Bit-string
- Boolean
- DATE
- TIME
- TIMESTAMP

- Numeric – They include integer numbers of various sizes (INTEGER, INT and SMALLINT), floating-point (real) numbers of various precisions (FLOAT, REAL, DOUBLE PRECISION). Formatted numbers can also be declared using DECIMAL (*i*,*j*), where *i* is the precision (the total number of decimal digits and *j* is the scale is the number of digits after the decimal point).

- Character-string – They are either fixed length – CHAR(*n*) or CHARACTER(*n*), where *n* is the number of characters – or varying length – VARCHAR(*n*), where *n* is the maximum number of characters.

  - When specifying a literal string value, it is placed between single quotation marks (apostrophes) and is case sensitive.

  - For fixed length strings, a shorter string is padded with blank characters to the right. Padded blanks are generally ignored when strings are compared.

- If *str1* appears before string *str2* in alphabetic order, then *str1* is considered to be less than *str2*.

- Concatenation operator is denoted by "||". Example, 'abc' || 'XYZ' results in abcXYZ.

- CLOB (character large object) allows to specify columns that <u>have large text values</u> such as documents (of any size). CLOB(20M) means maximum length of 20 MB.

- Bit-string – They are either of fixed length *n* – BIT(*n*) – or varying length – BIT VARYING(*n*), where *n* is the maximum number of bits. Default for *n* is 1.

  - Literal bit strings are placed between single quotes but preceded by a B to distinguish them from character strings, example, B'10101'.

  - Binary Large Object (BLOB) specify columns that <u>have large binary values</u>. Example, BLOB(30G) specifies a maximum length of 30 gigabits.

- Boolean – This data type has the values of TRUE OR FALSE. In SQL, a three-valued logic is used to include another BOOLEAN data type – UNKNOWN (i.e. presence of NULL values).

- DATE – This has ten positions with components YEAR, MONTH, DAY in the form YYYY-MM-DD. The TIME data type has at least eight positions with components HOUR, MINUTE and SECOND in the form HH:MM:SS.

  - SQL allows months to be between 1 and 12 and days between 01 and 31.

  - The < (less than) comparison can be used with dates or times – an earlier date is considered to be smaller than a later date and similarly with time.

- Literal values are represented by single-quoted strings preceded by the keyword DATE or TIME, example, DATE ‘2014-09-27’ or TIME ’09:12:47’.

- TIMESTAMP – This <u>includes the DATE and TIME fields</u>, and a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier. Literal values are represented by single-quoted strings preceded by the keyword TIMESTAMP with a blank space between date and time, example,

  ‘2014-09-27 09:12:47.648302’.

- INTERVAL – This specifies an interval which is a relative value that can be used to increment or decrement an absolute value of a date, time or timestamp. Intervals are either YEAR/MONTH or DAY/TIME.

- It is possible to specify the data type of each attribute through a domain i.e. a domain can be declared, and the domain name can be used with the attribute specification.

- This allows changing data type for a domain easily that is used by numerous attributes in a schema and improve schema readability. Example:

```
CREATE DOMAIN SSN_TYPE AS CHAR(9)
```

- SSN_TYPE can be used in place of CHAR(9) for attributes: Ssn and Super_ssn of EMPLOYEE, Mgr_ssn of DEPARTMENT, Essn of WORKS_ON, and Essn of DEPENDENT.

- A domain can also have an optional default specification via a DEFAULT clause.

- CREATE TYPE command – creates user defined types (UDTs) that can be used as data types for attributes or for creating tables.

Specifying constraints in SQL

- These include key and referential integrity constraints, restrictions on attribute domains and NULLS, and constraints on individual tuples within a relation using "CHECK" clause.

- Attribute constraints:

  - A constraint NOT NULL is specified if NULL is not permitted for a particular attribute (see next slide).

Example

```
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) );
CREATE TABLE PROJECT
        ( Pname          VARCHAR(15)      NOT NULL,
          Pnumber        INT              NOT NULL,
          Plocation      VARCHAR(15),
          Dnum           INT              NOT NULL,
        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) );
```

Specifying constraints in SQL

- A default value is defined for an attribute by appending the clause DEFAULT <value> to an attribute definition. This is included in any tuple if an explicit value is not provided for that attribute (see next slide).

- Example, default manager for a new department and a default department for a new employee.

- If no default clause is specified, then the default "*default value*" is NULL for attributes that do not have the NOT NULL constraint.

```
CREATE TABLE EMPLOYEE
    ( … ,
      Dno          INT          NOT NULL      DEFAULT 1,
    CONSTRAINT EMPPK
      PRIMARY KEY (Ssn),
    CONSTRAINT EMPSUPERFK
      FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET NULL          ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
      FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
    ( … ,
      Mgr_ssn CHAR(9)          NOT NULL      DEFAULT '888665555',
      … ,
    CONSTRAINT DEPTPK
      PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
      UNIQUE (Dname),
    CONSTRAINT DEPTMGRFK
      FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                    ON DELETE SET DEFAULT       ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
    ( … ,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE CASCADE           ON UPDATE CASCADE);
```

Default attribute values and referential integrity triggered actions are specified in SQL.

Specifying constraints in SQL

- Another type of constraint can restrict attribute or domain values using the **CHECK** clause following an attribute or domain definition.

- Example, suppose that department numbers are restricted to integer numbers between 1 and 20 then we can write:

Dnumber INT **NOT NULL CHECK** (Dnumber > 0 **AND** Dnumber < 21);

- The CHECK clause can also be used in conjunction with the CREATE DOMAIN statement. Example:

CREATE DOMAIN D_NUM **AS** INTEGER **CHECK** (D_NUM > 0 **AND** D_NUM < 21);

We can then use D_NUM as the attribute type for all attributes such as Dnumber of DEPARTMENT, Dnum of PROJECT, Dno of EMPLOYEE.

Specifying key and referential integrity constraints

- CREATE TABLE statement can specify keys and referential integrity constraints.

- The "PRIMARY KEY" clause specifies one or more attributes that make up the primary key of a relation.

- Example, the primary key of DEPARTMENT can be specified as

Dnumber INT **PRIMARY KEY**

- The UNIQUE clause specifies alternate (unique) keys/candidate keys. Example, in DEPARTMENT and PROJECT table declarations:

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

- The UNIQUE clause can also be specified directly for a unique key if it is a single attribute as follows:

Dname VARCHAR(15) **UNIQUE**

- Referential integrity is <u>specified via the "FOREIGN KEY"</u> clause:

```
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) );
CREATE TABLE PROJECT
        ( Pname          VARCHAR(15)      NOT NULL,
          Pnumber        INT              NOT NULL,
          Plocation      VARCHAR(15),
          Dnum           INT              NOT NULL,
        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) );
```

Quiz

- When a referential integrity constraint is violated (due to insert or delete operation) or when a foreign or primary key attribute is updated, SQL rejects the update.
  - This is known as RESTRICT option.
- Here, the schema designer can specify an alternate action to be taken by attaching a *referential triggered action* clause to any foreign key constraint. The options include SET NULL, CASCADE, and set DEFAULT and an option must be qualified with either ON DELETE or ON UPDATE.

Example, ON DELETE SET NULL and ON UPDATE CASCADE for the foreign key Super_ssn of EMPLOYEE.

- This means that if the tuple for a *supervising employee* is *deleted*, the value of Super_ssn is automatically *set to NULL* for all employee tuples that were referencing the deleted employee tuple.

- When the Ssn value for a supervising employee is *updated*, the new value *is cascaded* to Super_ssn for all employee tuples referencing the updated employee tuple.

- So, generally the action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE <u>for the value of the affected referencing attributes</u>.

- The action for CASCADE on DELETE is to delete all the referencing tuples, whereas the action for CASCADE on UPDATE is to change the value of the referencing foreign key attribute(s) to the updated primary key value for all the referencing tuples.

- So, the CASCADE option is suitable for "relationship" relations (example: WORKS_ON); for relations that represent multivalued attributes (such as DEPT_LOCATIONS); and for relations that represent weak entity types (such as DEPENDENT).

Giving names to constraints

- A constraint may be given a name following the keyword CONSTRAINT.

- A name is used to identify a particular constraint in case the constraint must be dropped later and replaced with another constraint.

CHECK clause

- This "**row based constraint**" is specified at the end of a CREATE TABLE statement because they apply to each row *individually* and are checked whenever a row is inserted or modified.

- Example, if there is an attribute Dept_create_date, which stores the "date" when the department was established, then it can be checked to make sure that a manager's start date is later than the department establishment date as

CHECK(Dept_create_date <= Mgr_start_date);

Basic retrieval queries in SQL

- SQL has one basic statement for retrieving information from a DB: the SELECT statement.

- While SQL allows a relation to have two or more tuples that are identical in all their attribute values, a relational model does not.

The SELECT-FROM-WHERE Structure of Basic SQL Queries

- The basic form of the SELECT statement is also called **select-from-where block**, is formed of three clauses SELECT, FROM and WHERE:

```
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 statement (Boolean) expression that identifies the tuples to be retrieved by the query.

- The logical comparison operators for comparing attribute values in SQL are =, <, <=, >, >=, and <>.

COMPANY database schema

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

Schema diagram for the COMPANY relational DB schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

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

DB state for the COMPANY relational DB schema.

Sample queries

- Query 1: Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

```
SELECT   Bdate, Address
FROM     EMPLOYEE
WHERE    Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
```

- The query selects the individual EMPLOYEE tuples that satisfy the condition of the WHERE clause, then projects the result on the Bdate and Address attributes listed in the SELECT clause.

- SELECT clause specifies the attributes whose values are to be retrieved and WHERE clause specifies the Boolean condition that must be true for any retrieved tuple.

- Only those tuples that satisfy the condition – that is, those tuples for which the condition evaluates to TRUE after substituting their corresponding attribute values are selected.

```
SELECT   Bdate, Address
FROM     EMPLOYEE
WHERE    Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
```

Result:

Bdate	Address
1965-01-09	731 Fondren, Houston, TX

- Query 2: 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;
```

a **select-project-join query**

- Dname = 'Research' is a "selection condition" and 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.

Result:

Fname	Lname	Address
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

- A query that involves only selection and join conditions plus projection attributes is known as a **select-project-join query**.

- The projection attributes are used to "choose the attributes" to be displayed from each combined tuple.

- Query 3: For every project located in 'Stafford', list the project number, the controlling department, and the department manager's last name, address, and birth date (See DB in next slide).

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

Result:

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

- The join condition Dnum = Dnumber relates a project tuple to its controlling department tuple, whereas the join condition Mgr_ssn = Ssn relates the controlling department tuple to the employee tuple who manages that department.

Outcome - Each tuple in the result will be a combination of one project, one department that controls the project, and one employee that manages the department.

EMPLOYEE

Fname	Minit	Lname	Ssn	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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

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

DB state for the COMPANY relational DB schema.

Ambiguous attribute names, aliasing and renaming

- The same name <u>can be used</u> for two (or more) attributes as long as the attributes are in different tables.

- Here, we must qualify the attribute name with the relation name to prevent ambiguity by *prefixing the relation name to the attribute name by a period*.

Ambiguous attribute names, aliasing and renaming

- Example, suppose Dno and Lname attributes of the EMPLOYEE relation are called Dnumber and Name, and Dname attribute of DEPARTMENT is also called Name, then

```
SELECT   Fname, Lname, Address
FROM     EMPLOYEE, DEPARTMENT
WHERE    Dname = 'Research' AND Dnumber = Dno;
```

is written as

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

So, we must prefix the attributes "Name" and "Dnumber" to specify which ones we are referring to, because the same attribute names are used in both the relations.

- Fully qualified attribute names can be used for clarity even if there is no ambiguity in attribute names. Example:

```
SELECT    Fname, Lname, Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dname = 'Research' AND Dnumber = Dno;
```

can be written as

```
SELECT    EMPLOYEE.Fname, EMPLOYEE.LName,
          EMPLOYEE.Address
FROM      EMPLOYEE, DEPARTMENT
WHERE     DEPARTMENT.DName = 'Research' AND
          DEPARTMENT.Dnumber = EMPLOYEE.Dno;
```

- We can also rename the table names to shorter names by creating "<u>an alias</u>" for each table name to avoid repeated typing of long table names as in Query 4 (see next slide).

- Ambiguity of attribute names also arise in the case of queries that refer to the same relation twice (Query 4).

- Query 4: For every 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;
```

- In this case, we are required to declare alternative relation names E and S called *aliases* or *tuple variables* for the EMPLOYEE relation.

- An alias follow the keyword **AS** or it can directly follow the relation name. Example, EMPLOYEE E, EMPLOYEE S. (see the FROM clause above in Query 4).

- It is also possible to rename the relation attributes by giving them alias as follows in the FROM clause:

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

Fn becomes an alias for Fname, Mi for Minit, Ln for Lname, etc.

- "E" and "S" are two different copies of the EMPLOYEE relation; E represents employees in the role of supervisees and S represents employees in the role of supervisors.

- We can now join the two copies i.e. the join condition is meant to join the relation with itself by matching the tuples that satisfy the join condition E.Super_ssn = S.Snn.

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

Result:

E.Fname	E.Lname	S.Fname	S.Lname
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	English	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

- We can use *aliasing* or *renaming* in any SQL query to specify tuple variables for every table in the WHERE clause, whether or not the same relation needs to be referenced more than once. Example,

```
SELECT   Fname, Lname, Address
FROM     EMPLOYEE, DEPARTMENT
WHERE    Dname = 'Research' AND Dnumber = Dno;
```

can be written as

```
SELECT   E.Fname, E.LName, E.Address
FROM     EMPLOYEE AS E, DEPARTMENT AS D
WHERE    D.DName = 'Research' AND D.Dnumber = E.Dno;
```

Unspecified WHERE clause and the use of Asterisk

- A missing WHERE clause indicates <u>no condition</u> on tuple selection, therefore, <u>all tuples</u> specified in the FROM clause <u>qualify and are selected</u> for the result.

- If more than one relation is specified in the FROM clause and there is no WHERE clause, then the CROSS PRODUCT – <u>all possible tuple combinations</u> – of these relations is selected. Example:

Query 5: Select all EMPLOYEE Ssns in the database.

```
SELECT    Ssn
FROM      EMPLOYEE;
```

Result:

Ssn
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

Query 6: Select all combinations of EMPLOYEE Ssns and DEPARTMENT Dname in the database.

```
SELECT    Ssn, Dname
FROM      EMPLOYEE, DEPARTMENT;
```

Result:

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

- To retrieve <u>all the attribute values of the selected tuples</u>, we do not have to list the attribute names explicitly in SQL, we just specify an asterisk (*) which means for all the attributes.

- The * can also be prefixed by the relation name or alias. As such, EMPLOYEE * refers to <u>all attributes of the EMPLOYEE table</u>. Example:

- <u>Query 7</u>: Retrieve all the attribute values of any EMPLOYEE who works in DEPARTMENT 5.

```
SELECT   *
FROM     EMPLOYEE
WHERE    Dno = 5;
```

Result:

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
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

- Query 8: Retrieve all the attributes of an EMPLOYEE and the attributes of the DEPARTMENT in which he or she works for every employee of the "Research" department.

```
SELECT    *
FROM      EMPLOYEE, DEPARTMENT
WHERE     Dname = 'Research' AND Dno = Dnumber;
```

- Query 9: Retrieve all the possible attributes from EMPLOYEE and DEPARTMENT relations (CROSS PRODUCT).

```
SELECT    *
FROM      EMPLOYEE, DEPARTMENT;
```

Tables as sets in SQL

- SQL treats a table not as a set but rather as a multiset i.e. duplicate tuples can appear more than once in a table, and in the result of a query.

- So, SQL does not automatically eliminate duplicate tuples in the results of queries because

  1. Duplicate elimination is an expensive operation and can be done by sorting the tuples first and then eliminate duplicates.
  2. The user may want to see duplicate tuples in the result of a query.
  3. When an aggregate function is applied to tuples, we may not want to eliminate duplicates.

- If we *<u>want to eliminate</u>* duplicate tuples from the result of SQL query, we use **DISTINCT** in the SELECT clause – meaning that only distinct tuples should remain in the result.

- A query with SELECT DISTINCT <u>eliminates duplicates</u>, whereas a query with SELECT ALL <u>does not</u>. Just specifying SELECT is same as SELECT ALL. Example:

- <u>Query 10</u>: Retrieve the salary of every employee.

```
SELECT    ALL Salary
FROM      EMPLOYEE;
```

Result:

Salary
30000
40000
25000
43000
38000
25000
25000
55000

- <u>Query 11</u>: Retrieve all the distinct salary of every employee.

Result:

Salary
30000
40000
25000
43000
38000
55000

```
SELECT    DISTINCT Salary
FROM      EMPLOYEE;
```

- SQL has some of the set operations such as union (UNION), set difference (EXCEPT), and set intersection (INTERSECT) operations.

- They apply only to *type compatible* relations i.e. the two relations on which we apply the operations should have the same attributes and in same order in both relations. Duplicate tuples are eliminated from the result. Example:

Query 12: 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.

```
( 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' );
```

EMPLOYEE

Fname	Minit	Lname	Ssn	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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

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

DB state for the COMPANY relational DB schema.

```
( 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' );
```

- The first SELECT query retrieves the projects that involve a 'Smith' as manager of the department that controls the project, and the second retrieves the projects that involve a 'Smith' as a worker on the project.
- Note: If several employees have the last name 'Smith', the project names involving any of them will be retrieved.
- Applying the UNION operation to the two SELECT queries gives the desired result.

- SQL also has corresponding multiset operations, which are followed by the keyword ALL (UNION ALL, EXCEPT ALL, INTERSECT ALL). <u>Multiset means duplicates are not eliminated</u>. Example:

R

A
a1
a2
a2
a3

S

A
a1
a2
a4
a5

TWO TABLES

Result:

T

A
a1
a1
a2
a2
a2
a3
a4
a5

UNION ALL

T

A
a2
a3

EXCEPT ALL

T

A
a1
a2

INTERSECT ALL

Substring pattern matching and arithmetic operations

- "LIKE" allows ***comparison conditions*** on "only parts" of a character string. This is used for string pattern matching.

- Partial strings are specified using two reserved characters: "%" replaces an arbitrary number of zero or more characters, and the underscore "_" replaces a single character.

Example: Query 13: Retrieve all employees whose address is in Houston, Texas (see DB in next slide).

```
SELECT   Fname, Lname
FROM     EMPLOYEE
WHERE    Address LIKE '%Houston,TX%';
```

Query 14: Find all employees who were born during 1970s.

```
SELECT   Fname, Lname
FROM     EMPLOYEE
WHERE    Bdate LIKE '_ _ 7 _ _ _ _ _ _ _';
```

Each underscore serves as a place holder for an arbitrary character.

EMPLOYEE

Fname	Minit	Lname	Ssn	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	Dnumber	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

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

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

DB state for the COMPANY relational DB schema.

- If an underscore "_" or "%" is needed as a literal character in the string, the character should be preceded by an "escape character", specified after the string using the keyword "ESCAPE". Example:

- 'AB_CD\%EF' ESCAPE '\' represents the literal string 'AB_CD%EF'.

- If apostrophes or single quotation marks (' ') needs to be included (as they are also used to begin and end strings), it is represented as two consecutive apostrophes ('') so that it is not interpreted as ending the string.

- Arithmetic operations such as +, -, * and / can also be applied to numeric values/attributes. Example:

<u>Query 15</u>: Show the resulting salaries if every employee working on the 'ProductX' project is given a 10% raise.

```
SELECT   E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM     EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE    E.Ssn = W.Essn AND W.Pno = P.Pnumber AND
         P.Pname = 'ProductX';
```

Notice that we <u>can rename an attribute in the query result using AS in the SELECT clause</u>.

- For string data types, the concatenate operator "||" can be used in a query to append two string values.

- For date, time, timestamp and interval data types, incrementing (+) or decrementing (-) a date, time, or timestamp by an interval is also possible.

- Additionally, an interval value is the result of the difference between two date, time or timestamp.

- Another comparison operator is BETWEEN. Example:

Query 16: Retrieve all employees in department 5 whose salary is between $30000 and $40000 (i.e. Salary >= 30000) AND (Salary <= 40000)).

```
SELECT    *
FROM      EMPLOYEE
WHERE     (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Ordering of query results

- SQL allows "to order the tuples" in the result of a query <u>by the values of one or more of the attributes</u> that appear in the result by using ORDER BY clause.

<u>Query 17</u>: Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

```
SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W,
          PROJECT AS P
WHERE     D.Dnumber = E.Dno AND E.Ssn = W.Essn AND W.Pno =
          P.Pnumber
ORDER BY  D.Dname, E.Lname, E.Fname;
```

Ordering of query results

Query 17: Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

```
SELECT    D.Dname, E.Lname, E.Fname, P.Pname
FROM      DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W,
          PROJECT AS P
WHERE     D.Dnumber = E.Dno AND E.Ssn = W.Essn AND W.Pno =
          P.Pnumber
ORDER BY  D.Dname, E.Lname, E.Fname;
```

- The default is in ascending order of values. We can specify the keyword DESC if we want to see the results in descending order of values and ASC to specify ascending order explicitly.

- Example, if we want descending alphabetical order on Dname and ascending order on Lname, Fname, the ORDER BY clause can be

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

Insert

- Insert is used to add a single tuple to a relation. The values should be listed *in the same order* in which the corresponding attributes were specified in the CREATE TABLE command. Example:

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

- Another form of INSERT statement allows the user to specify explicit attribute names that correspond to the values provided in the INSERT command. <u>It is useful when the relation has many attributes but only a few are assigned values in the new tuple</u>. Example:

```
INSERT INTO    EMPLOYEE (Fname, Lname, Dno, Ssn)
VALUES         ('Richard', 'Marini', 4, '653298653');
```

Insert

- Attributes not specified in the INSERT statement are set to DEFAULT or to NULL and the values are listed in the same order as the attributes are listed in the INSERT command.

- It is also possible to insert multiple tuples separated by commas in a single INSERT command. The attribute values forming each tuple are enclosed in parentheses.

Insert

- A variation of the INSERT command inserts multiple tuples into a relation in conjunction with creating the relation and loading it with the result of a query.

- Example, to create a temporary table that has the employee last name, project name, and hours per week for each employee working on a project:

```
CREATE TABLE      WORKS_ON_INFO
( Emp_name        VARCHAR(15),
  Proj_name       VARCHAR(15),
  Hours_per_week  DECIMAL(3,1) );

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 <u>first query creates a table</u> WORKS_ON_INFO and <u>is loaded with the joined information retrieved by the second query</u>.

- DROP TABLE command is used to remove a table.

- WORKS_ON_INFO may not be “up to date” if any of the related tables such as PROJECT, WORKS_ON, EMPLOYEE is updated. Therefore, VIEW is useful here to keep such a table up to date.

- Formatted data can also be bulk loaded “from a file” into a table without using a large number of INSERT commands.

- Another variation of loading data is to create a new table TNEW that has the same attributes as an existing table T and load some data currently in T into TNEW using "LIKE" clause.

- Example: create a table D5EMPS with a similar structure to the EMPLOYEE table and load it with the rows of employees who work in department 5.

```
CREATE TABLE    D5EMPS LIKE EMPLOYEE
(SELECT         E.*
FROM            EMPLOYEE AS E
WHERE           E.Dno = 5) WITH DATA;
```

- The clause "WITH DATA" above specifies that <u>the table will be created and loaded with the data specified in the query</u>.

Delete

- It removes tuples from a relation using a WHERE clause to select the tuples to be deleted. Examples:

```
DELETE FROM    EMPLOYEE
WHERE          Lname = 'Brown';
DELETE FROM    EMPLOYEE
WHERE          Ssn = '123456789';
DELETE FROM    EMPLOYEE
WHERE          Dno = 5;
```

- Deletion may propagate to tuples in other relations if *referential triggered actions* are specified in the referential integrity constraints.

- A missing WHERE clause specifies that "<u>all tuples in the relation are to be deleted</u>", however an empty table remains. Example:

```
DELETE FROM    EMPLOYEE;
```

Update

- It "modifies" attribute values of one or more selected tuples using a WHERE clause to select the tuples to be modified from a single relation.

- SET clause in the UPDATE command specifies the attributes to be modified and their new values.

- Example: change the location and controlling department number of project number 10 to 'Bellaire' and 5.

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

- Several tuples can be modified with a single UPDATE command.

- Example, give all employees in the 'Research' department a 10% raise in salary.

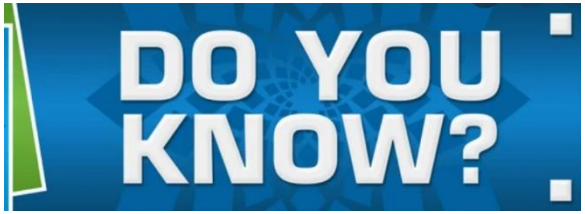
```
UPDATE    EMPLOYEE
SET       Salary = Salary * 1.1
WHERE     Dno = 5;
```

- Each UPDATE command explicitly refers to a single relation only. Multiple relations can be modified using several UPDATE commands.

Summary

- Introduced Basic SQL for data definition, queries, updates, constraint specification and view definition.

- The following SQL features were discussed:
  - Data definition commands for creating tables.
  - Basic data types.
  - Commands for constraint specification.
  - Retrieval queries and aliasing.
  - WHERE clause and use of asterisk.
  - Pattern matching and ordering.
  - Insert, delete and update commands.



Practice questions (check yourself)

1. What are the different uses of CREATE statement?
2. Explain ON DELETE SET NULL and ON UPDATE CASCADE.
3. What is the use of CHECK clause?
4. Explain SELECT-FROM-WHERE structure.
5. What is the difference between UNIQUE and DISTINCT?