Chapter 6
Basic SQL

Content

- SQL Data Definition and Data Types
- Specifying Constraints in SQL
- Basic Retrieval Queries in SQL
- INSERT, DELETE, and UPDATE Statements in SQL
- Additional features in SQL

Introduction about SQL

- **SQL: Structured Query Language.**
- Originally, SQL was called SEQUEL (Structured English QUEry Language) and was designed and implemented at IBM Research as the interface for an experimental relational database system called SYSTEM R.
- SQL is now the **standard language** for commercial relational DBMSs.
- SQL is a comprehensive database language: It has statements for data definitions, queries, and updates. Hence, it is both a **DDL and a DML**.
- It has facilities for**:**
  - **Defining views** on the database,
  - specifying **security and authorization**,
  - defining **integrity constraints**, and
  - specifying **transaction controls**.

SQL Data Definition and Data Types

- SQL uses the terms table, row, and column for the formal relational model terms **relation, tuple, and attribute**.

- The main SQL command for data definition is the **CREATE** statement, which can be used to
  - create schemas,
  - tables (relations),
  - domains,
  - views,
  - assertions,
  - and triggers.

SQL Data Definition and Data Types

- **Schema and Catalog Concepts in SQL**
- An 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 **tables, constraints, views, domains, and other constructs (such as authorization grants**) that describe the schema.
- A schema is created via the **CREATE SCHEMA** statement, which can include all the schema elements' definitions.
- For example, the following statement creates a schema called "COMPANY" owned by the user with authorization identifier 'Jsmith'.
- CREATE SCHEMA COMPANY AUTHORIZATION 'Jsmith';
- In general, all users are not allowed to create schemas. This privilege is grant by system Admin or the DBA

SQL Data Definition and Data Types

- **Schema and Catalog Concepts in SQL**
- In addition to the concept of a schema, SQL uses the concept of a **catalog—a named collection of schemas**.
- A catalog always 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**.
- **Integrity constraints** such as referential integrity can be defined between relations only if they exist in schemas within the same catalog.

SQL Data Definition and Data Types

- The CREATE TABLE Command in SQL
- The CREATE TABLE command is used to **specify a new relation by giving it a name and specifying its attributes and initial constraints.**
- The **attributes** are specified first, and each attribute is given a name, **a data type** to specify its domain of values, and **possibly attribute constraints, such as NOT NULL.**
- The key, entity integrity, and referential integrity constraints can be specified within the CREATE TABLE statement after the attributes are declared, or they can be added later using the ALTER TABLE command.
- Figure 6.1 in the next slide shows sample data definition statements in SQL for the COMPANY relational database schema.

```
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) );
```

Figure 6.1
SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7.

SQL Data Definition and Data Types

- The CREATE TABLE Command in SQL
- The relations declared through CREATE TABLE statements are called base tables (or base relations) this means that the table and its rows are actually created and stored as a file by the DBMS.
- Base relations are distinguished from virtual relations, created through the CREATE VIEW statement, which may or may not correspond to an actual physical file.
- In SQL, the attributes in a base table are considered to be ordered in the sequence in which they are specified in the CREATE TABLE statement. However, rows (tuples) are not considered to be ordered within a table (relation).
- It is important to note that in Figure 6.1, there are some foreign keys that may cause errors because they are specified either via circular references(self join) or because they refer to a table that has not yet been created.

SQL Data Definition and Data Types

- **The CREATE TABLE Command in SQL**
- To deal with this type of problem, these constraints can be left out of the initial CREATE TABLE statement, and then **added later using the ALTER TABLE statement**.
  - For example, the foreign key Super_ssn in the EMPLOYEE table is a circular reference because it refers to the EMPLOYEE table itself. The foreign key Dno in the EMPLOYEE table refers to the DEPARTMENT table, which has not been created yet.
  - We create dept table first then add its FK to Employee table using ALTER TABLE command.

SQL Data Definition and Data Types

- **Attribute Data Types and Domains in SQL**
- The **basic data types available for attributes include numeric, character string, bit string, Boolean, date, and time.**
- **Numeric data types** include integer numbers of various sizes (**INTEGER or INT, and SMALLINT**) and floating-point (real) numbers of various precision (**FLOAT or REAL, and DOUBLE PRECISION**). Formatted numbers can be declared by using **DECIMAL(i, j)—or DEC(i, j)** where i, the precision, is the total number of decimal digits and j, the scale, is the number of digits after the decimal point.
- **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. Another datatype CLOB (Character Large Object) can hold larger text values like in kilobytes, mega bytes of Gigabytes.
  - E.g., CLOB (20M)
- **Bit-string data types** are either of fixed length n—**BIT(n)—or varying length— BIT VARYING(n),** where n is the maximum number of bits. The default for n, the length of a character string or bit string, is 1. used to hold data like 1010101011.

SQL Data Definition and Data Types

- **Attribute Data Types and Domains in SQL**
- The basic data types available for attributes include numeric, character string, bit string, Boolean, date, and time.
- **A Boolean data type** 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.**
- **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.
- A **timestamp data type (TIMESTAMP)** includes the **DATE and TIME** fields, plus a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier.
- We can specify daatypes as in figure 6.1 bbut alternatively a Domain can be declared, and the domain name can be used with the attribute specification.
  - For example, we can create a domain SSN_TYPE by the following statement:
    - CREATE DOMAIN SSN_TYPE AS CHAR(9);

Specifying Constraints in SQL

- **Specifying Attribute Constraints and Attribute Defaults**
- Since SQL allows NULLs as attribute values, so a constraint **NOT NULL** may be specified if NULL is not permitted for a particular attribute.
- It is also possible to define a default value for an attribute by appending the clause **DEFAULT <value> to an attribute definition**. The default value is included in any new tuple if an explicit value is not provided for that attribute.
- Figure 6.2 illustrates an example of specifying a default manager for a new department and a default department for a new employee.
- **If no default clause is specified, the 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);
```

Figure 6.2
Example illustrating how default attribute values and referential integrity triggered actions are specified in SQL.

Specifying Constraints in SQL

- **Specifying Attribute Constraints and Attribute Defaults**
- Another type of constraint can restrict attribute or domain values using the CHECK clause following an attribute or domain definition.
- For example, suppose that department numbers are restricted to integer numbers between 1 and 20; then, we can change the attribute declaration of Dnumber in the DEPARTMENT table.

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

- The CHECK clause can also be used in conjunction with the CREATE DOMAIN statement.
- For example, we can write the following statement:

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

- Now, we can use this domain as attribute type in multiple relations where Dnum is used.

Specifying Constraints in SQL

- **Specifying Key and Referential Integrity Constraints**
- **PRIMARY KEY clause:**
  - specifies one or more attributes that make up the primary key of a relation.
  - If a primary key has a single attribute, the clause can follow the attribute directly.
  - For example, the primary key of DEPARTMENT can be specified as follows:

```
Dnumber INT PRIMARY KEY
```

- **UNIQUE Clause:**
  - specifies alternate (unique) keys, also known as candidate keys.
  - can also be specified directly for a unique key if it is a single attribute, as in the following example:

```
Dname VARCHAR(15) UNIQUE
```

Specifying Constraints in SQL

- **Specifying Key and Referential Integrity Constraints**
- **FOREIGN KEY clause:**
  - Referential integrity can be maintained using foreign key clause
  - That clause can be violated during insertion, deletion or updating.
  - So an instant action is required to deal with the violations
  - SQL by default uses the **RESTRICT** clause that rejects the operations in case of violation.
  - **But Schema designer can specify an alternative 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. An option must be qualified with either **ON DELETE or ON UPDATE**

Live SQL

SQL Worksheet

```
1  create table people(
2  ID int primary key,
3  name varchar(50)
4  );
5
6  create table cars (
7  LicenseNo char(6) Primary key,
8  make varchar(10),
9  ownerID int references people(ID)
10 );
11
12 insert into people values (1, 'hajra');
13 insert into people  values (2, 'Ahmed');
14 insert into people  values (3, 'Maryam');
15
16
17 insert into cars values ('AX1030','HONDA',1);
18 insert into cars  values ('AX1031','HONDA',2);
19 insert into cars  values ('AX1032','HONDA',2);
20
21 select * from people;
22 select * from cars;
23
24
25
26 delete from people where id=1;
```

```
ORA-02292: integrity constraint (SQL_LSDNLMOGZEJALZTQTYFFARKQW.SYS_C0095449876) violated - child record found ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Live SQL

SQL Worksheet

```
1  create table people(
2  ID int primary key,
3  name varchar(50)
4  );
5
6  create table cars (
7  LicenseNo char(6) Primary key,
8  make varchar(10),
9  ownerID int references people(ID) on delete cascade
10 );
11
12 insert into people values (1, 'hajra');
13 insert into people  values (2, 'Ahmed');
14 insert into people  values (3, 'Maryam');
15
16
17 insert into cars values ('AX1030','HONDA',1);
18 insert into cars  values ('AX1031','HONDA',2);
19 insert into cars  values ('AX1032','HONDA',2);
20
21 select * from people;
22 select * from cars;
23
24 delete from people where id=1;
25
26
```

LICENSENO	MAKE	OWNERID
AX1031	HONDA	2
AX1032	HONDA	2

Download CSV

Specifying Constraints in SQL

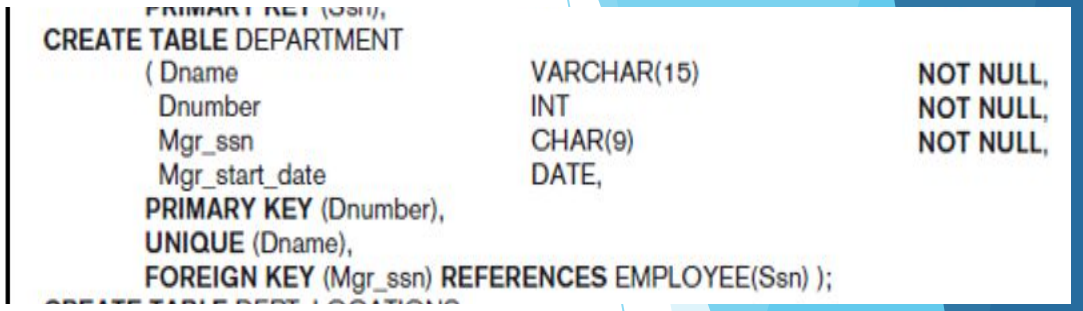
- **Specifying Key and Referential Integrity Constraints**
- **FOREIGN KEY clause:**
  - The options include SET NULL, CASCADE, and SET DEFAULT. An option must be qualified with either **ON DELETE or ON UPDATE**
    - **Consider the following schema:**
      - **People(id, name)**
      - **Cars(licenceNo, model, make, ownerID)**
    - **Here we can imply the FK as :**
      - **ownerID int references people(id) on delete set default**
      - **ownerID int references people(id) on delete cascade**
      - **ownerID int references people(id) on delete no action**

Specifying Constraints in SQL

- **Giving Names to Constraints**
- A constraint may be given a constraint name, following the keyword **CONSTRAINT**.
- The names of all constraints within a particular schema must be **unique**.
- A constraint name is used to identify a particular constraint in case the constraint must be dropped later and replaced with another 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);
```

Specifying Constraints in SQL



```
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) );
```

- **Specifying Constraints on Tuples Using CHECK**
- Table constraints can be specified through additional CHECK clauses at the end of a CREATE TABLE statement.
- These can be called **row-based constraints** because they apply to each row individually and are checked whenever a row is inserted or modified.
- For example, suppose that the DEPARTMENT table in Figure 6.1 had an additional attribute Dept_create_date, which stores the date when the department was created.
- Then we could add the following CHECK clause at the end of the CREATE TABLE statement for the DEPARTMENT table to make sure that a manager's start date is later than the department creation date.
- CHECK (Dept_create_date <= Mgr_start_date);

Basic Retrieval Queries in SQL

- **The SELECT-FROM-WHERE Structure of Basic SQL Queries**
- The basic form of the SELECT statement, sometimes called a mapping or a select-from-where block, is 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.

Basic Retrieval Queries in SQL

- **The SELECT-FROM-WHERE Structure of Basic SQL Queries**
- Query 0. Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.
- Q0: SELECT Bdate, Address
- FROM EMPLOYEE
- WHERE Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
- The **SELECT clause of SQL** specifies the attributes whose values are to be retrieved, which are called the **projection attributes** in relational algebra
- and the **WHERE clause** specifies the Boolean condition that must be true for any retrieved tuple, which is known as the **selection condition** in relational algebra.

```
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),
```

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

Basic Retrieval Queries in SQL

- **The SELECT-FROM-WHERE Structure of Basic SQL Queries**
- SQL **EQUI JOIN** performs a JOIN against equality or matching column(s) values of the associated tables.
- An equal sign (=) is used as comparison operator in the where clause to refer equality.

Syntax:

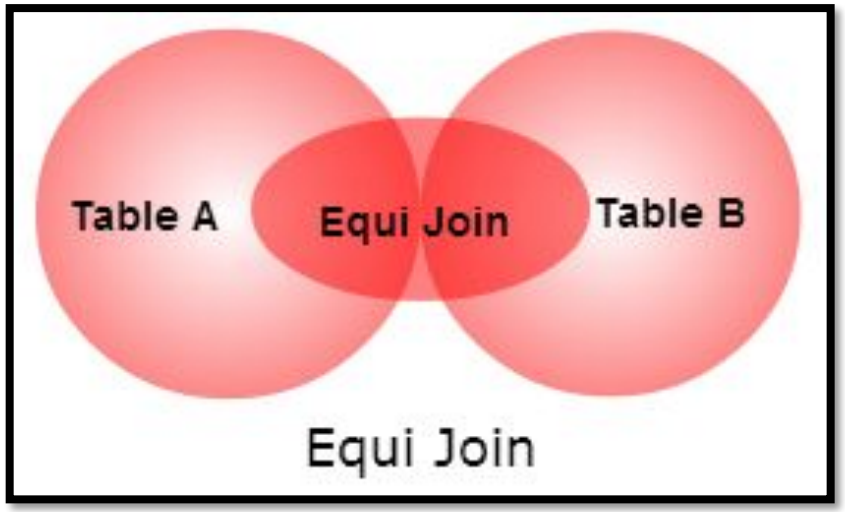
```
SELECT column_list
FROM table1, table2....
WHERE table1.column_name =
table2.column_name;
```

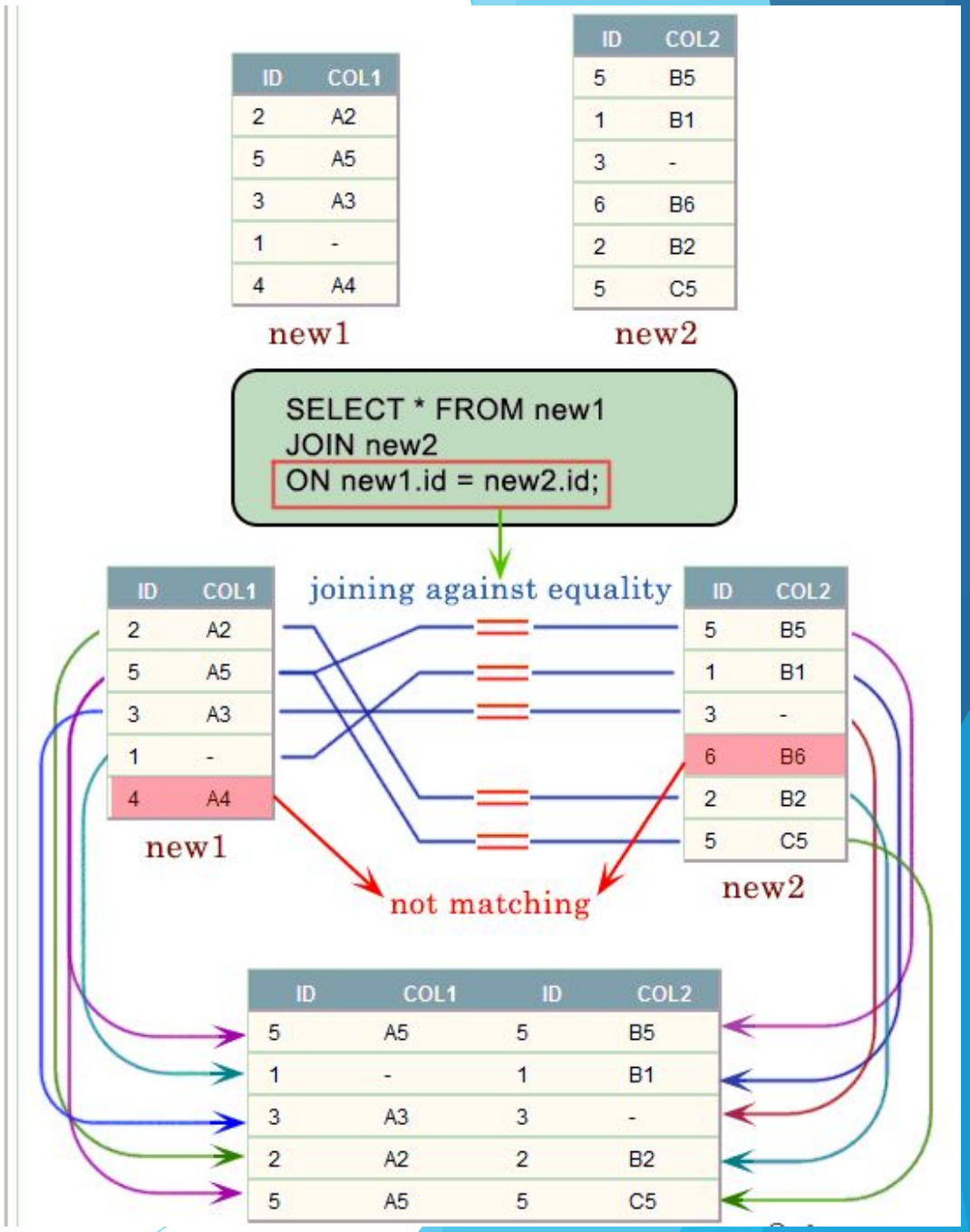
or

```
SELECT *
FROM table1
JOIN table2
[ON (join_condition)]
```

Basic Retrieval Queries in SQL

- EQUIJOIN





Basic Retrieval Queries in SQL

- **The SELECT-FROM-WHERE Structure of Basic SQL Queries**
- Query 1. Retrieve the name and address of all employees who work for the 'Research' department.
- Q1: SELECT Fname, Lname, Address
- FROM EMPLOYEE, DEPARTMENT
- WHERE Dname = 'Research' AND Dnumber = Dno;
- In the WHERE clause of Q1, **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.
- **SELECT clause -> projection operation**
- **It's a select-project-join query**

```
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),
```

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

Basic Retrieval Queries in SQL

- **The SELECT-FROM-WHERE Structure of Basic SQL Queries**
- 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'
- 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.
- 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).

```
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) );
```

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

Basic Retrieval Queries in SQL

- **Ambiguous Attribute Names, Aliasing, Renaming, and Tuple Variables**
- In SQL, the same name can be used for two (or more) attributes as long as the attributes are in different tables.
- If this is the case, and a multi table query refers to two or more attributes with the same name, we must **use 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. Like employee.ID or Department.ID

Basic Retrieval Queries in SQL

- **Ambiguous Attribute Names, Aliasing, Renaming, and Tuple Variables**
- To illustrate this, suppose that the Dno and Lname attributes of the EMPLOYEE relation were called Dnumber and Name, and the Dname attribute of DEPARTMENT was also called Name; then, to prevent ambiguity, query Q1 would be rephrased as shown in Q1A.
- We must prefix the attributes Name and Dnumber in Q1A to specify which ones we are referring to, because the same attribute names are used in both relations:
- Query 1. Retrieve the name and address of all employees who work for the 'Research' department.
- Q1A: SELECT Fname, EMPLOYEE.Name, Address
- FROM EMPLOYEE, DEPARTMENT
- WHERE DEPARTMENT.Name = 'Research' AND
- DEPARTMENT.Dnumber = EMPLOYEE.Dnumber;

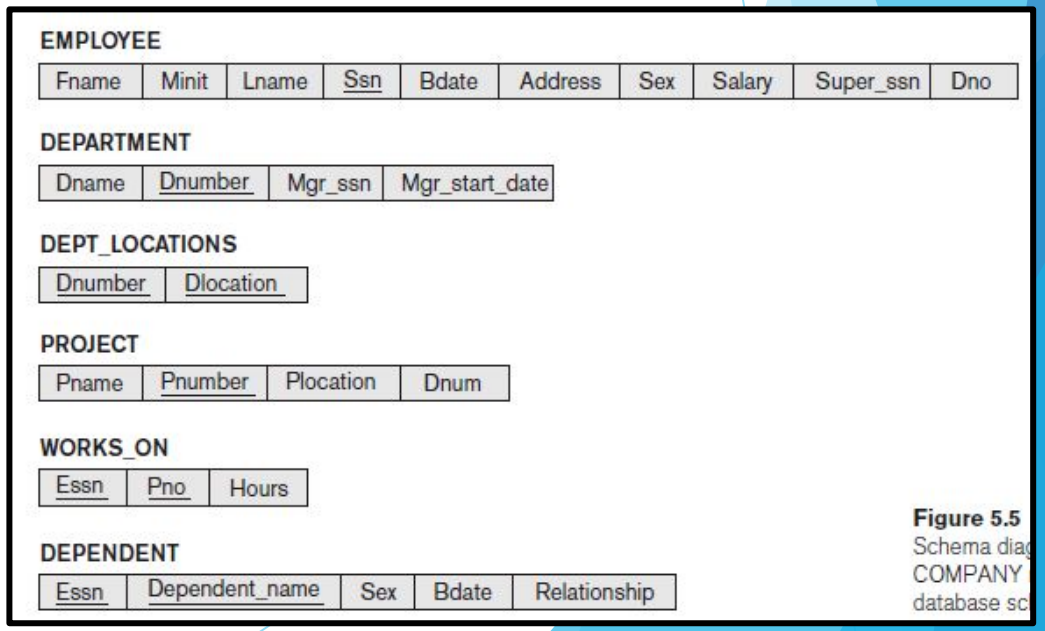


Figure 5.5
Schema diag
COMPANY
database sc

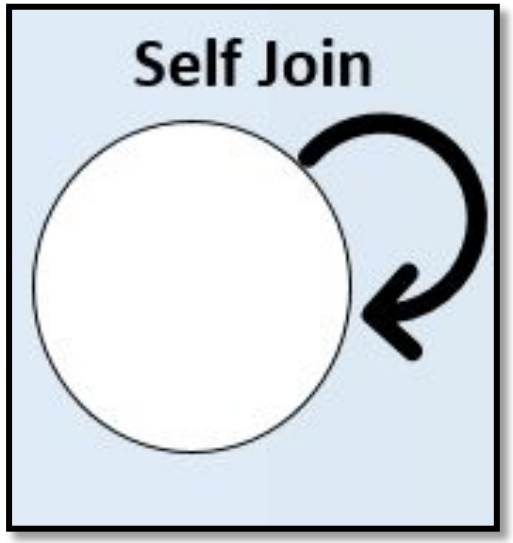
Basic Retrieval Queries in SQL

- **Ambiguous Attribute Names, Aliasing, Renaming, and Tuple Variables**
- A **self join** is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a **FOREIGN KEY which references its own PRIMARY KEY**.
- To join a table itself means that each row of the table is combined with itself and with every other row of the table.
- The syntax of the command for joining a table to itself is almost same as that for joining two different tables.
- To distinguish the column names from one another, **aliases for the actual the table name are used, since both the tables have the same name**.
- Table name aliases are defined in the FROM clause of the SELECT statement.

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_filed = b.common_field;
```

Basic Retrieval Queries in SQL

- **SELF JOIN**



Column Name	Data Type	Nullable	Default	Primary Key
EMP_ID	VARCHAR2(5)	No	-	1
EMP_NAME	VARCHAR2(20)	Yes	-	-
DT_OF_JOIN	DATE	Yes	-	-
EMP_SUPV	VARCHAR2(5)	Yes	-	-

1 - 4

Constraint	Type	Table
SYS_C004074	C	EMPLOYEE
EMP_ID	P	EMPLOYEE
EMP_SUPV	R	EMPLOYEE

Primary key

Foreign key
Referencing EMP_ID of this table

Basic Retrieval Queries in SQL

- **SELF JOIN**

```
SELECT a.emp_id AS "Emp_ID",a.emp_name AS "Employee Name",
b.emp_id AS "Supervisor ID",b.emp_name AS "Supervisor Name"
FROM employee a, employee b
WHERE a.emp_supv = b.emp_id;
```

Emp_ID	Employee Name	Supervisor ID	Supervisor Name
20055	Vinod Rathor	20051	Vijes Setthi
20069	Anant Kumar	20051	Vijes Setthi
20073	Unnath Nayar	20051	Vijes Setthi
20075	Mukesh Singh	20073	Unnath Nayar
20064	Rakesh Patel	20073	Unnath Nayar

Basic Retrieval Queries in SQL

- **Ambiguous Attribute Names, Aliasing, Renaming, and Tuple Variables**
- We can also rename the table names to shorter names by creating an alias for each table name to avoid repeated typing of long table names.
- 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.
- Q8: SELECT E.Fname, E.Lname, S.Fname, S.Lname
- FROM EMPLOYEE AS E, EMPLOYEE AS S
- WHERE E.Super_ssn = S.Ssn;
- It is also possible to rename the relation attributes within the query in SQL by giving them aliases.
- For example, if we write **EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)** in the FROM clause, Fn becomes an alias for Fname, Mi for Minit, Ln for Lname, and so on.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

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

Basic Retrieval Queries in SQL

- **Unspecified WHERE Clause and Use of the Asterisk**
- A **missing WHERE clause** indicates no condition on tuple selection; hence, all tuples of the relation specified in the FROM clause qualify and are selected for the query result.
- If more than one relation is specified in the FROM clause and there is no WHERE clause, then the **CROSS PRODUCT—all possible tuple combinations—of these relations is selected**.
- 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	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

Basic Retrieval Queries in SQL

- **Unspecified WHERE Clause and Use of the Asterisk**
- To retrieve all the attribute values of the selected tuples, we do not have to list the attribute names explicitly in SQL; we just specify an asterisk (*), which stands for all the attributes.
- Query Q1C retrieves all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5 (Figure 6.3(g)), query Q1D retrieves 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, and Q10A specifies the CROSS PRODUCT of the EMPLOYEE and DEPARTMENT relations.
- Q1C: SELECT *
- FROM EMPLOYEE
- WHERE Dno = 5;

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

- Q1D: SELECT *
- FROM EMPLOYEE, DEPARTMENT
- WHERE Dname = 'Research' AND Dno = Dnumber;
- Q10A: SELECT *
- FROM EMPLOYEE, DEPARTMENT;