UNIT-III
Advanced normalization concepts and Basic SQL

Advanced normalization concepts and Basic SQL

Sessions to be Covered in Unit-III

- Advanced normalization concepts.
- Basic SQL

SESSION-I: TOPICS TO BE COVERED

Advanced normalization concepts.

- Boyce – Code normal form
- Multi-valued dependencies and fourth normal form
- Join dependencies and fifth normal form.

SESSION-II: TOPICS TO BE COVERED

Basic SQL

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

SESSION-I: TOPICS TO BE COVERED

Advanced normalization concepts.

- Boyce – Code normal form
- Multi-valued dependencies and fourth normal form
- Join dependencies and fifth normal form.

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X ->A** holds in R, then **X is a super key** of R
- There can be exist relations that are in 3NF but not in BCNF.

- **Difference between 3NF & Boyc Codd**
- **3NF**-> A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.
- **Boyc Codd->** A relation is in BCNF, if and only if every determinant is a candidate key.

https://www.youtube.com/watch?v=b39IhcO7qsA

Example : Boyce-Codd Normal Form

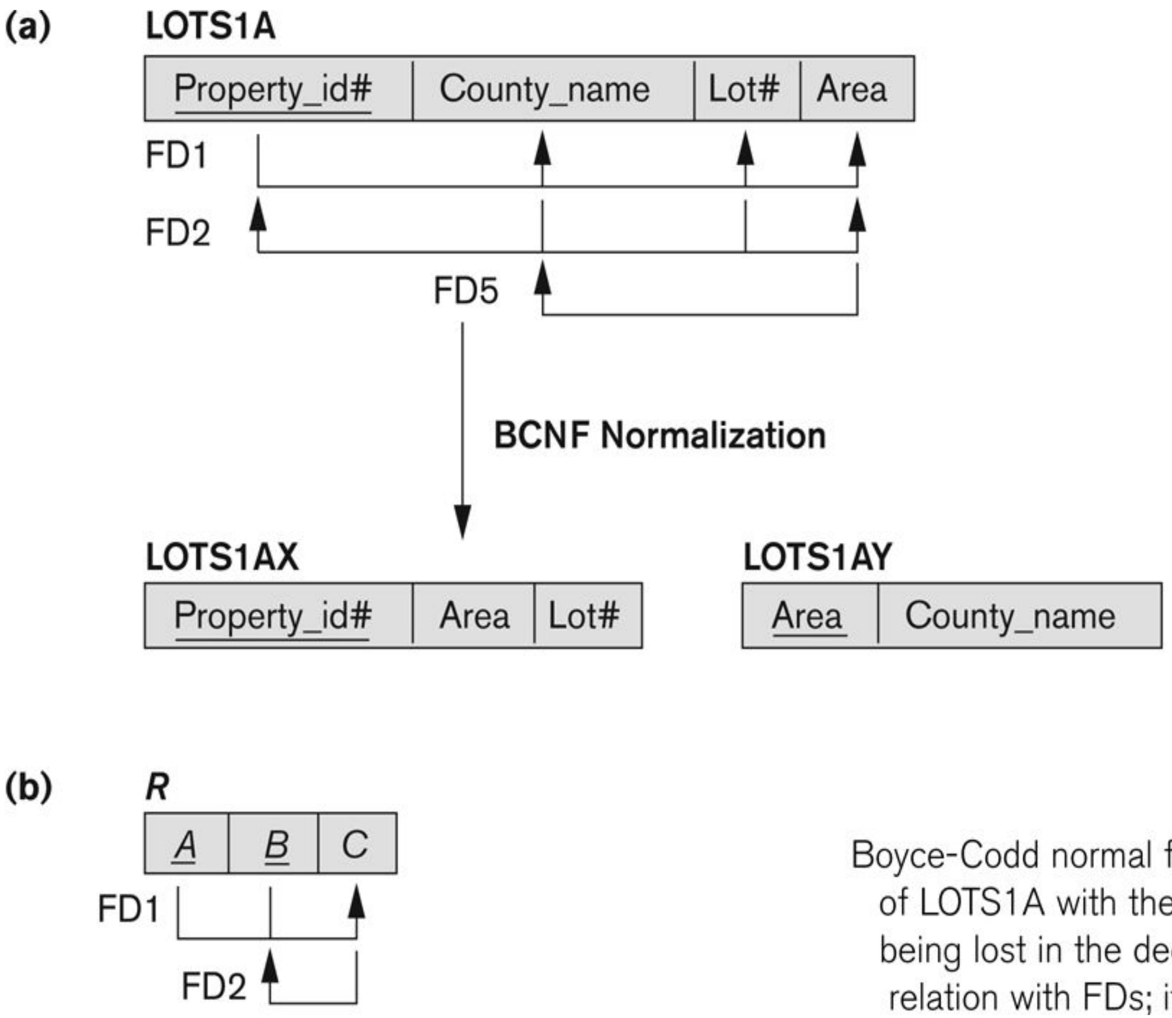


Figure 10.12
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Multivalued Dependency and Fourth Normal form

- **Multivalued dependency** occurs when two attributes in a table are independent of each other but, both depend on a third attribute.

- A **multivalued dependency** consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

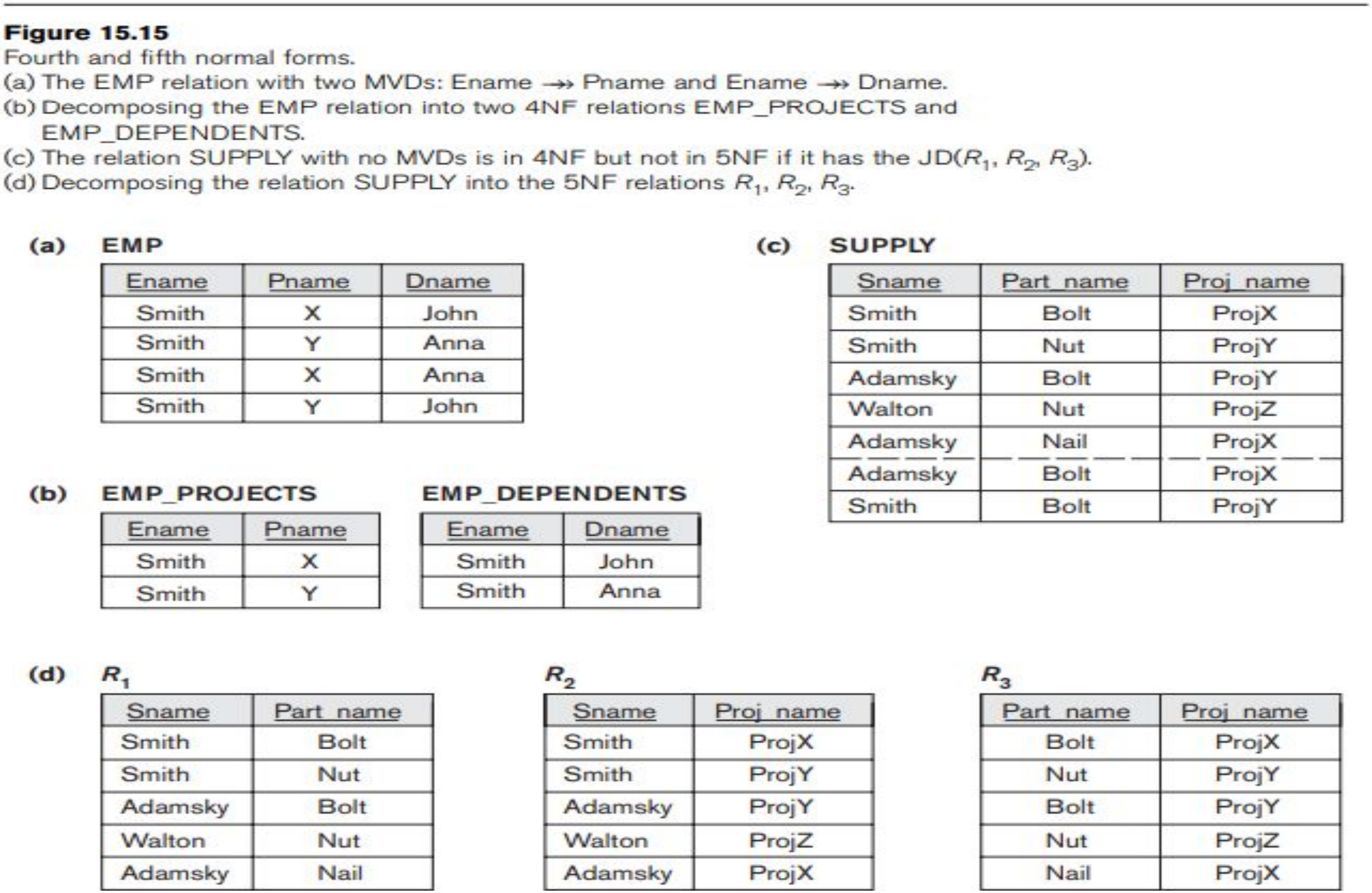


Figure 15.15
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename $\twoheadrightarrow$ Pname and Ename $\twoheadrightarrow$ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.
(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

(a) EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) R_1

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Join dependencies and fifth normal form.

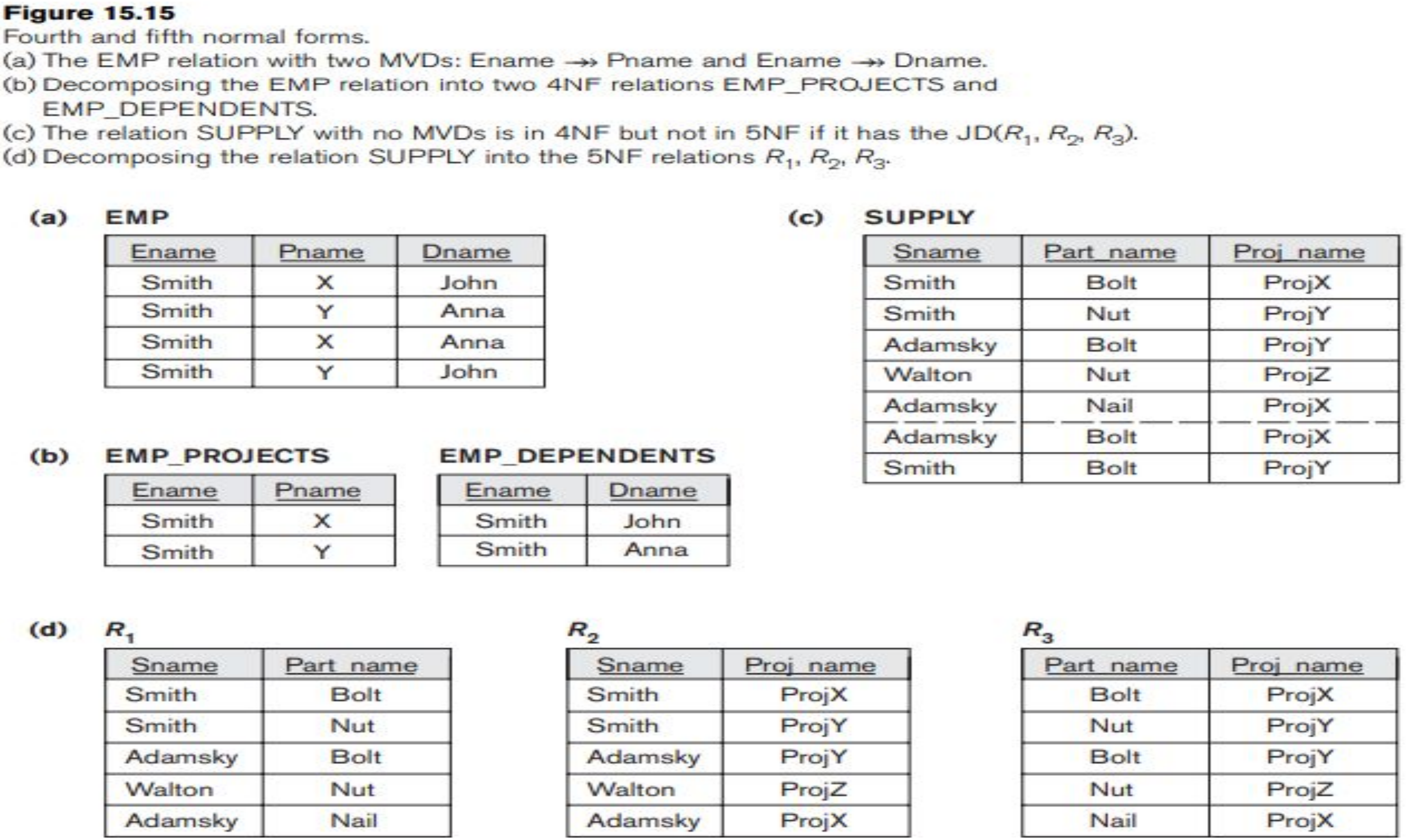


Figure 15.15
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename →→ Pname and Ename →→ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.
(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

(a) EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) R_1

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

SESSION-II: TOPICS TO BE COVERED

Basic SQL

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

SQL Data Definition and Data Types

Basic SQL

- **SQL language**
  - Considered one of the major reasons for the commercial **success of relational databases**

- **SQL Features**
  - Structured Query Language
  - Required Statements available for data definitions, queries, and updates (ie. both DDL and DML)
  - Core specification with respect to all DDL and DML
  - Plus specialized extensions also available.

SQL Data Definition and Data Types

SQL Data Definition

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms **relation**, **tuple**, and **attribute**
    - Table -> Relation
    - Row -> Tuple
    - Column -> Attribute

- CREATE statement
  - Main SQL command statement for data definition

Schema and Catalog Concepts in SQL

- **SQL schema**
  - Identified by a **schema name**
  - Includes an **authorization identifier (**the user or account who owns the **schema)** and **descriptors (**Characteristics or information**)** for each element
- Schema **elements** include
  - Tables, constraints, views, domains, and other constructs
- Each statement in SQL ends with a semicolon
- **CREATE SCHEMA statement**
  - For example, the following statement creates a schema called COMPANY, owned by the user with authorization identifier ‘Jsmith’. Note that each statement in SQL ends with a semicolon.
    - CREATE SCHEMA COMPANY AUTHORIZATION ‘Jsmith’;

- **Catalog**
  - Named collection of schemas in an SQL environment

- **SQL environment**
  - Installation of an **SQL-compliant** RDBMS on a computer system

ACID **compliance** is an attribute of relational technology which is required to satisfied by the SQL software while installation.

What is **ACID Compliance**?

The presence of four components — atomicity, consistency, isolation and durability — can ensure that a database transaction is completed in a timely manner.

When **databases** possess these components, they are said to be **ACID- compliant**

The CREATE TABLE Command in SQL

- To Specify a new relation
  - Provide name
  - Specify attributes and initial constraints

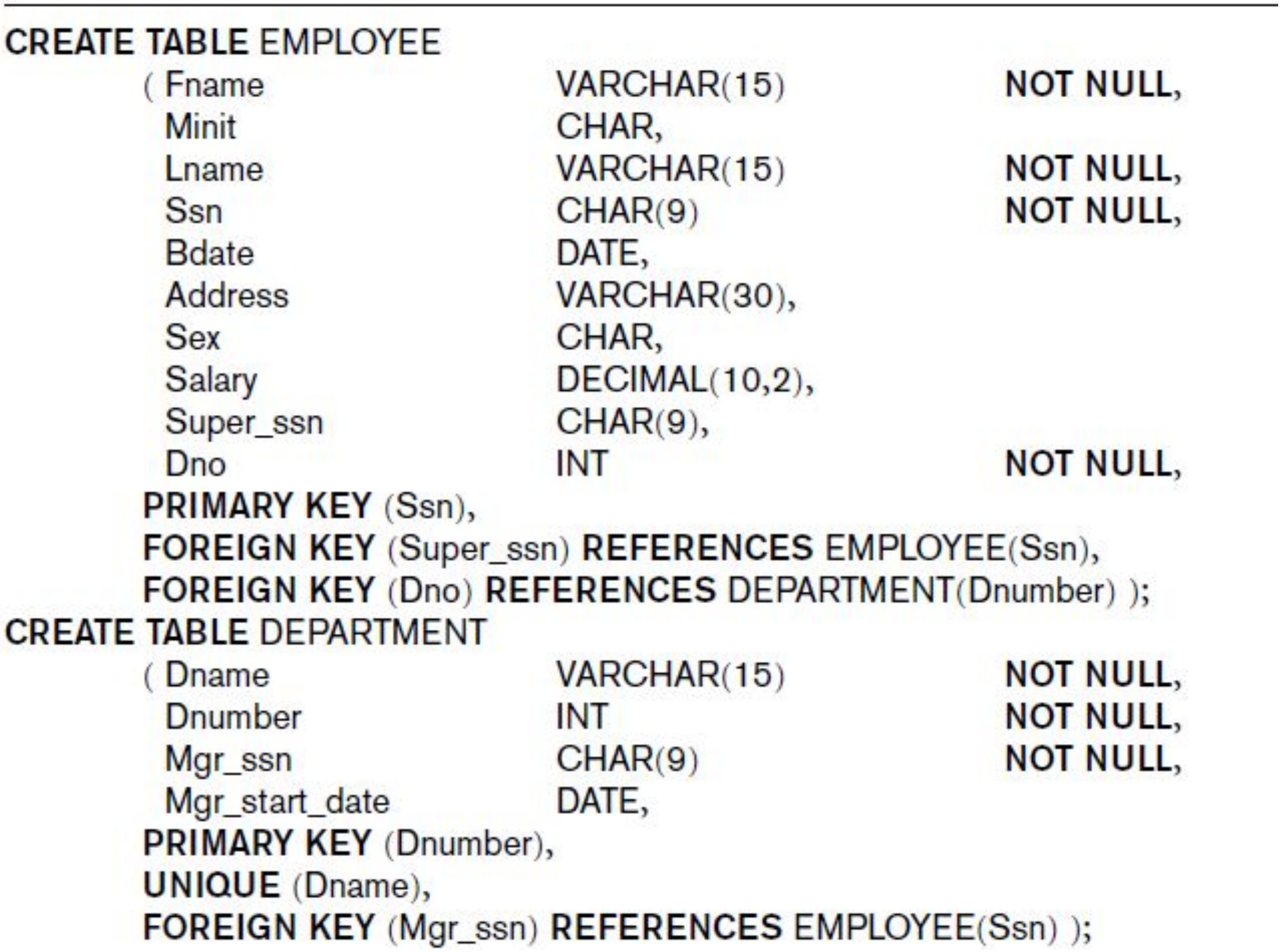
- Can optionally specify schema:
  - 1) CREATE TABLE COMPANY.EMPLOYEE ...

    or
  - 2) CREATE TABLE EMPLOYEE ...
  - Where in 1 - COMPANY – Schema name & EMPLOYEE – Relation name
    - We can explicitly attach the schema name to the relation name, separated by a period
  - In 2 – Without using Schema name COMPANY (Implicitly used)

- **Base tables (base relations)**
  - Relation and its tuples are actually created and stored as a file by the DBMS

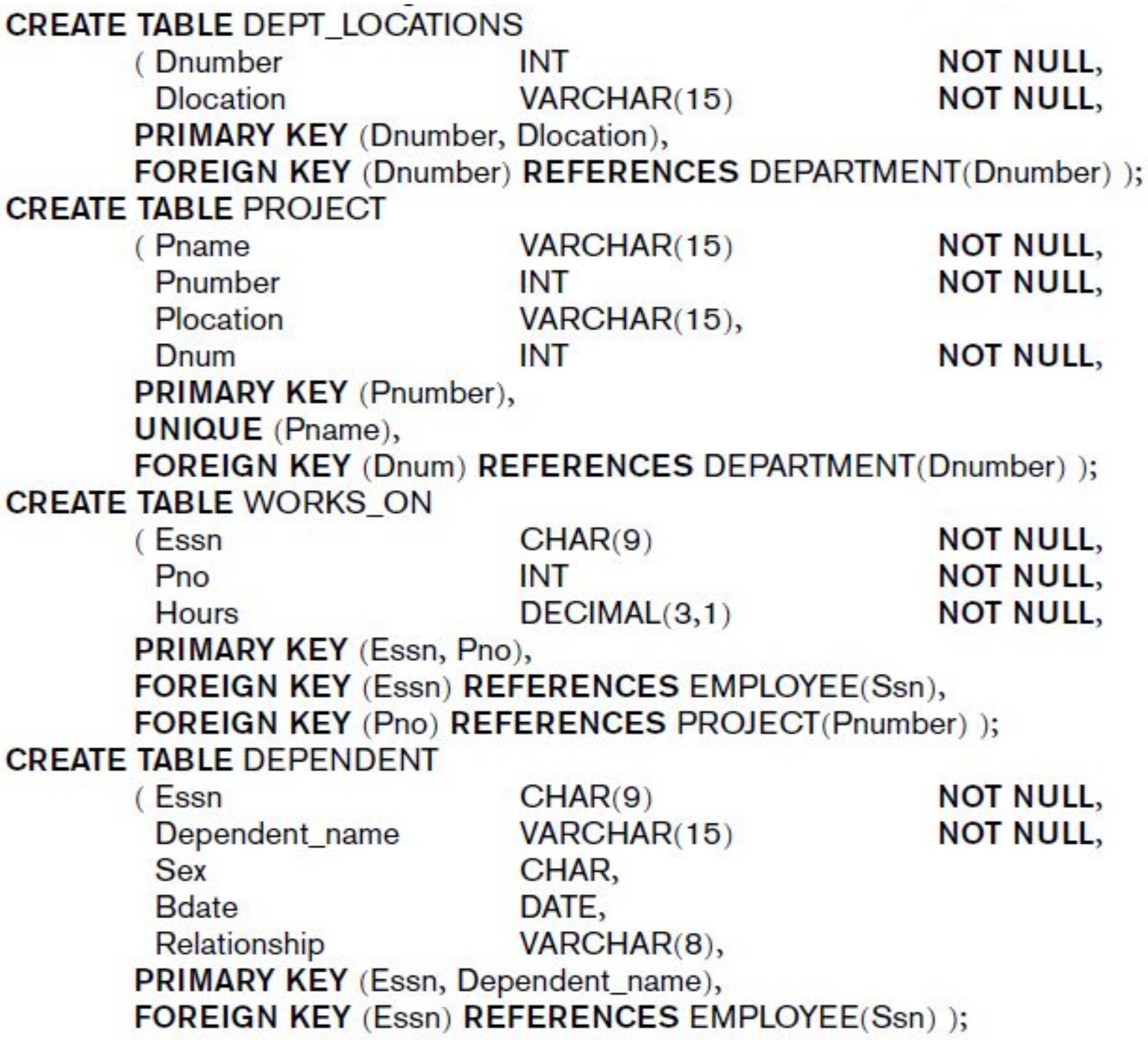
- **Virtual relations**
  - Created through the CREATE VIEW statement
  - From the base relation derived relation has been derived in order to view in different ways of looking at base relation. It is not physically existed one.

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),
        FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber) );
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) );
```

Figure 4.1
SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 3.7.



```
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 4.1
SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 3.7.

- Some foreign keys may cause errors
  - When it specified either via:
    - Circular references
    - Or because they refer to a table that has not yet been created

SQL Data Types

The following are the attribute data types in SQL.

- **Basic data types**
  - **Numeric** data types
    - Integer numbers: INTEGER, INT, and SMALLINT

      (2 bytes per value for **SMALLINT** and 4 bytes per value for **INTEGER)**
    - Floating-point (real) numbers: FLOAT or REAL, and DOUBLE PRECISION
  - **Character-string** data types
    - Fixed length: CHAR(*n*), CHARACTER(*n*)
    - Varying length: VARCHAR(*n*), CHAR VARYING(*n*), CHARACTER VARYING(*n*) (Variable length with limit)

- **Bit-string** data types (1s or 0s)- Flag variables
  - Fixed length: BIT(*n*)
  - Varying length: BIT VARYING(*n*)

- **Boolean** data type
  - Values of TRUE or FALSE or NULL

- **DATE** data type
  - Ten positions
  - Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD

- **Timestamp** data type (TIMESTAMP)
  - Includes the DATE and TIME fields
  - Plus a minimum of six positions for decimal fractions of seconds
  - Optional WITH TIME ZONE qualifier

- **INTERVAL** data type
  - Specifies a relative (Comparative) value that can be used to increment or decrement an absolute value of a date, time, or timestamp

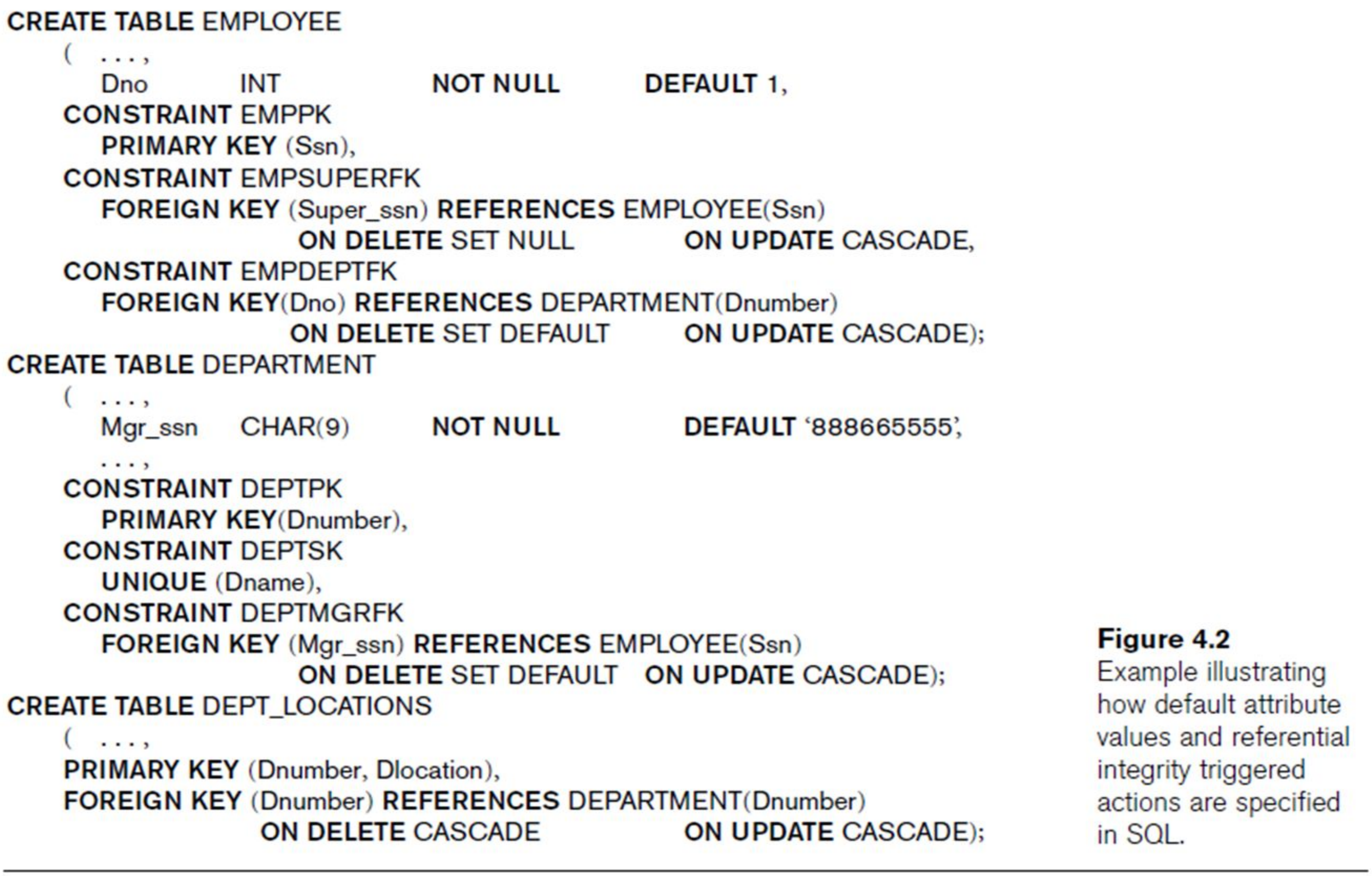
Specifying Constraints in SQL

- Basic constraints:
  - Key and referential integrity constraints
  - Restrictions on attribute domains and NULLs
  - Constraints on individual tuples within a relation (for update and delete process)

- NOT NULL
  - NULL is not permitted for all the attributes (except that attributes have no value and foreign key value)

- Default value
  - **DEFAULT** <value>

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



```
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 4.2
Example illustrating how default attribute values and referential integrity triggered actions are specified 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
  - Dnumber INT PRIMARY KEY;

  (Should have unique value)

- **UNIQUE** clause
  - Specifies alternate (secondary) keys
  - Dname VARCHAR(15) UNIQUE;

- **FOREIGN KEY** clause
  - Default operation: It rejects update operation on violation
  - For update, the following options are given as constraints
  - Attach **referential triggered action** clause
    - Options include SET NULL, CASCADE, and SET DEFAULT
    - Action taken by the DBMS for SET NULL or SET DEFAULT is the same for both ON DELETE and ON UPDATE

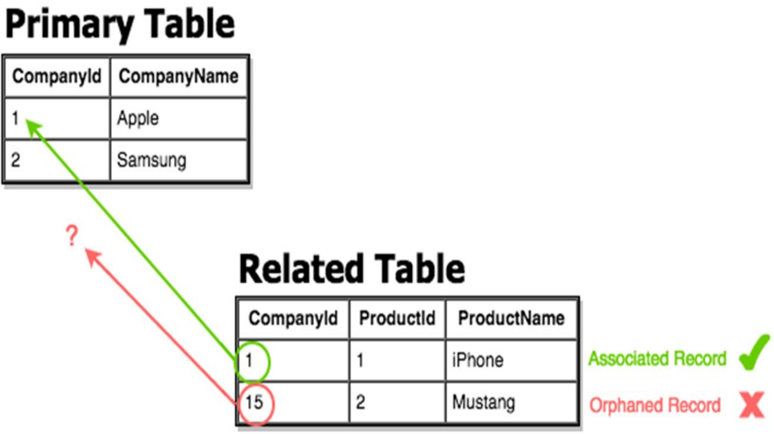
      (By default it will refer primary key reference only)
    - CASCADE option suitable for "relationship" relations

CASCADE

It is used in conjunction with ON DELETE or ON UPDATE

It means that the child data is either deleted or updated when parent data is deleted or updated.

It means that the child data is **set to their default values** when parent data is deleted or updated.



Primary Table

CompanyId	CompanyName
1	Apple
2	Samsung

Related Table

CompanyId	ProductId	ProductName
1	1	iPhone
15	2	Mustang

- **Giving Names to Constraints**
  - Keyword **CONSTRAINT**
    - Name a constraint
    - Useful for later altering

Specifying Constraints on Tuples Using CHECK

- CHECK clauses at the end of a CREATE TABLE statement
  - Apply to each tuple individually
  - CHECK (Dept_create_date <= Mgr_start_date);

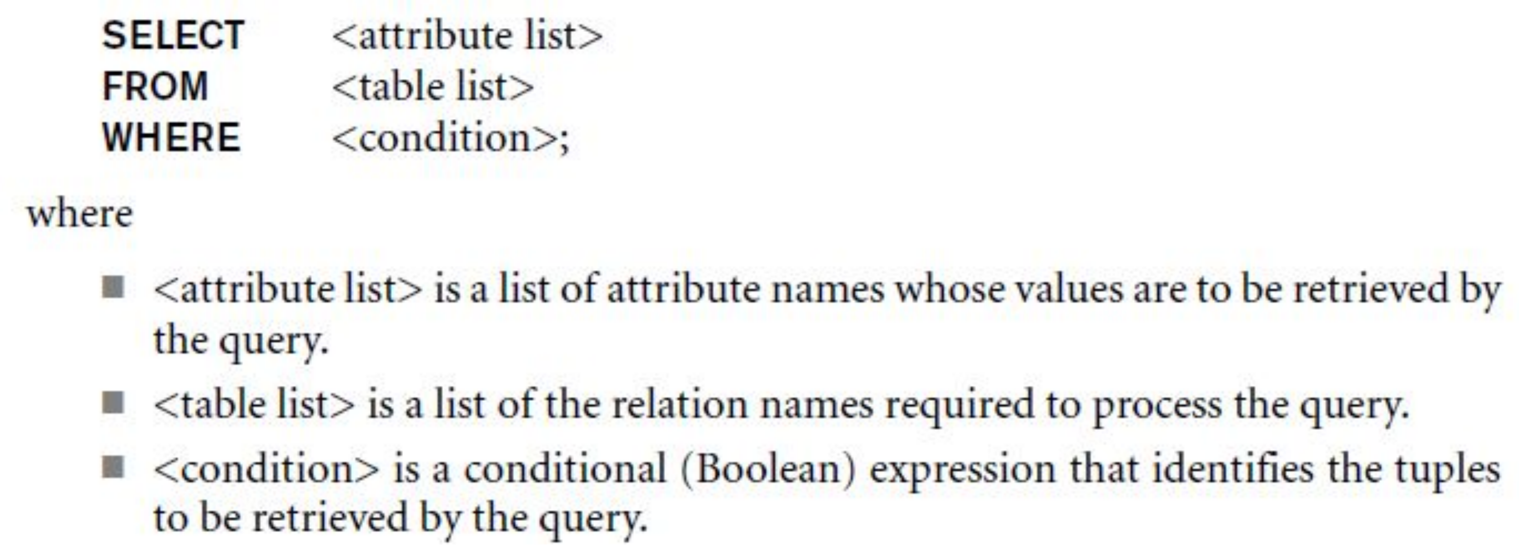
Basic Retrieval Queries in SQL

- SELECT statement
  - One basic statement for retrieving information from a database

- While retrieval SQL allows a table to have two or more tuples that are identical in all their attribute values

The SELECT-FROM-WHERE Structure of Basic SQL Queries

Basic form of the `SELECT` statement:



```
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.

Some of the parts with **SQL SELECT** statements are

- **Logical comparison operators**
  - =, <, <=, >, >=, and <>
- **Projection attributes**
  - Attributes whose values are to be retrieved
- **Selection condition**
  - Boolean condition that must be true for any retrieved tuple

"AND", "**OR**" and "NOT"

- **Example**

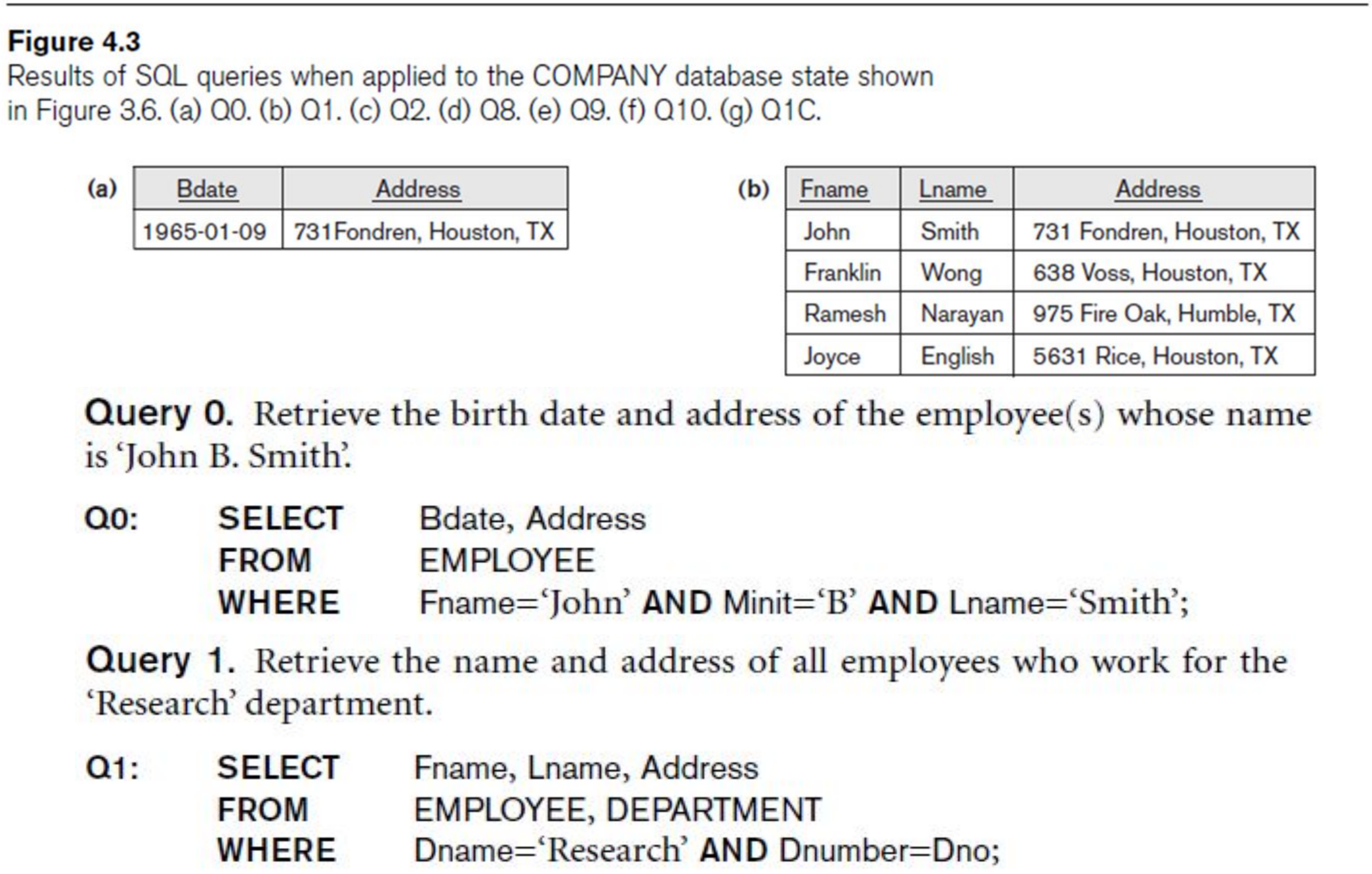


Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(a)

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

(b)

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

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';
```

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;
```

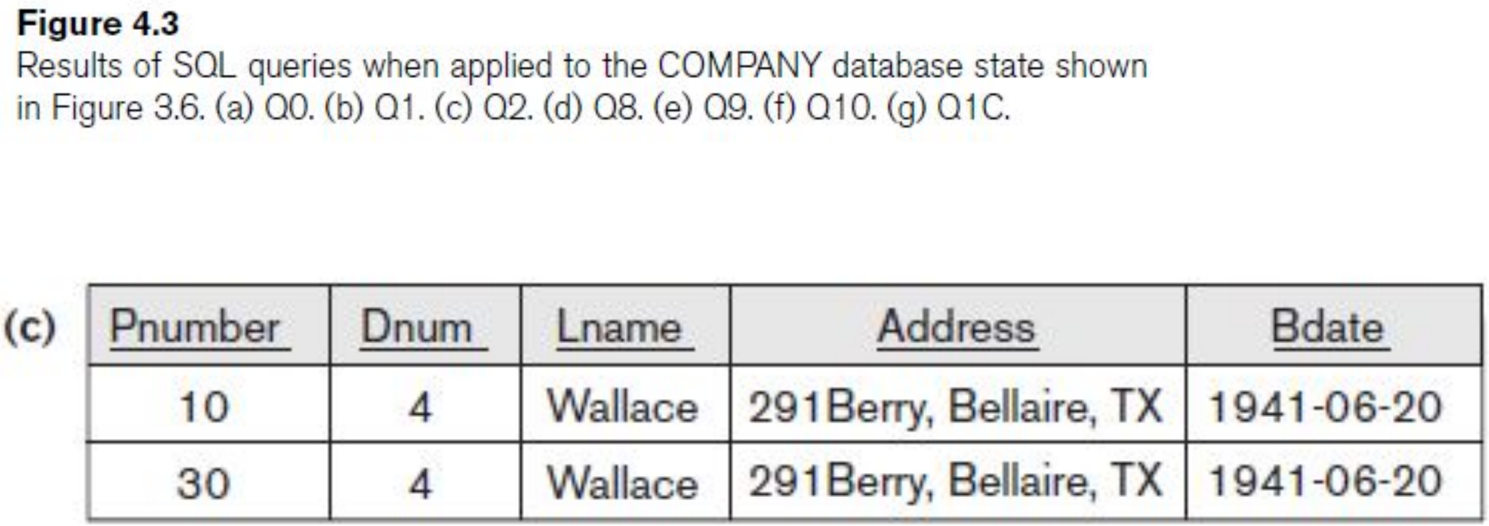
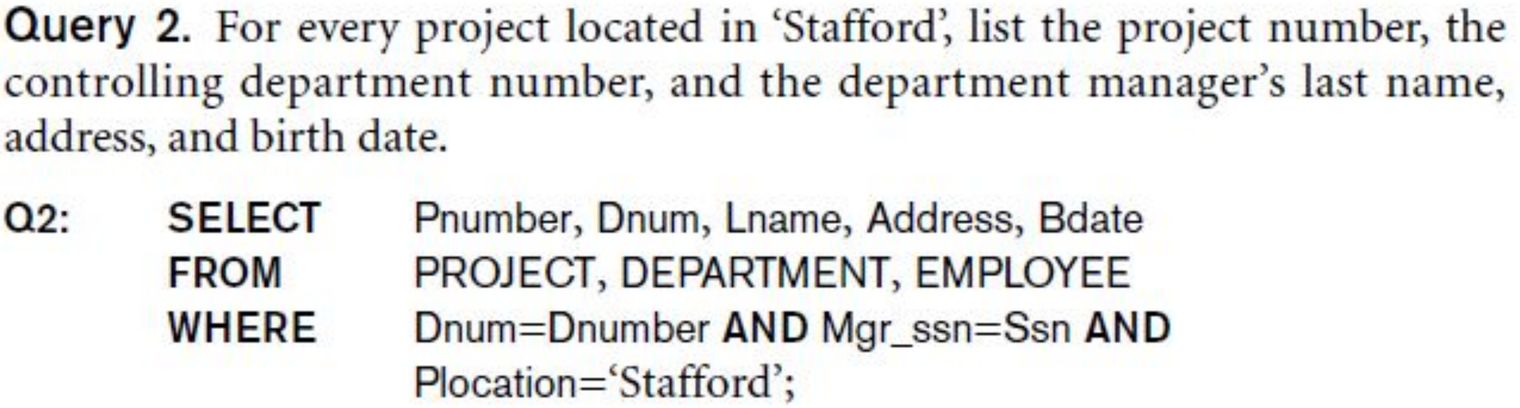


Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(c)

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

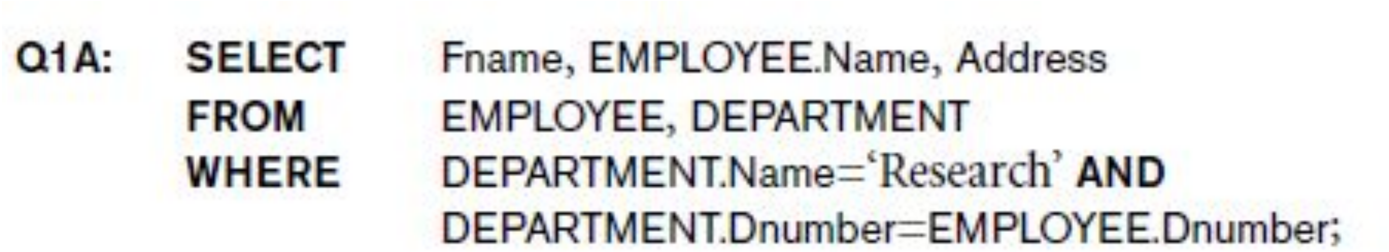


Query 2. For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

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

Ambiguous Attribute Names

- Same name can be used for two (or more) attributes
  - As long as the attributes are in different relations
  - Must **qualify** the attribute name with the relation name to prevent ambiguity (attribute same name but relation name should be different)



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

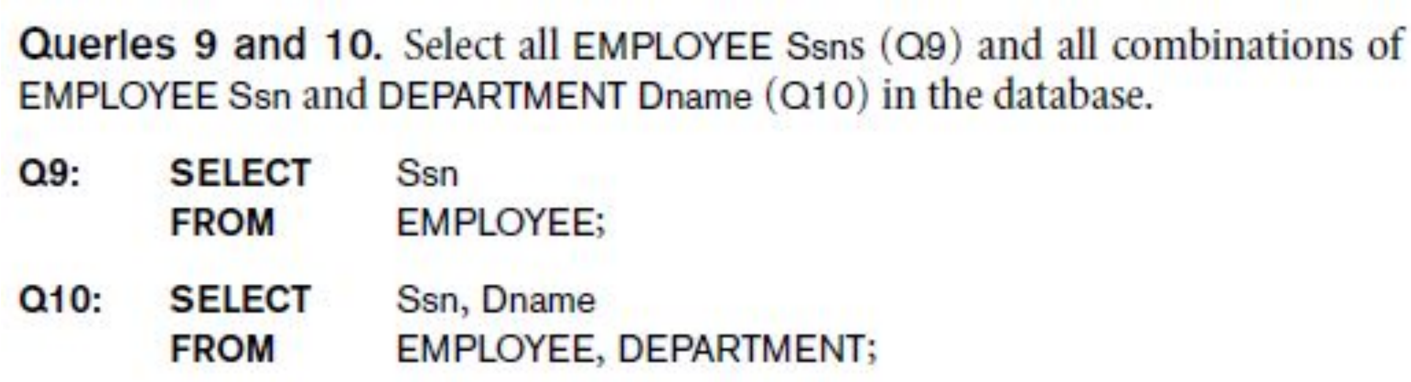
Aliasing, Renaming, and Tuple Variables

- **Aliases** or **tuple variables**
  - Declare alternative relation names E and S
  - EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)

Unspecified WHERE Clause and Use of the Asterisk

- Missing WHERE clause
  - Indicates no condition on tuple selection

- CROSS PRODUCT
  - All possible tuple combinations

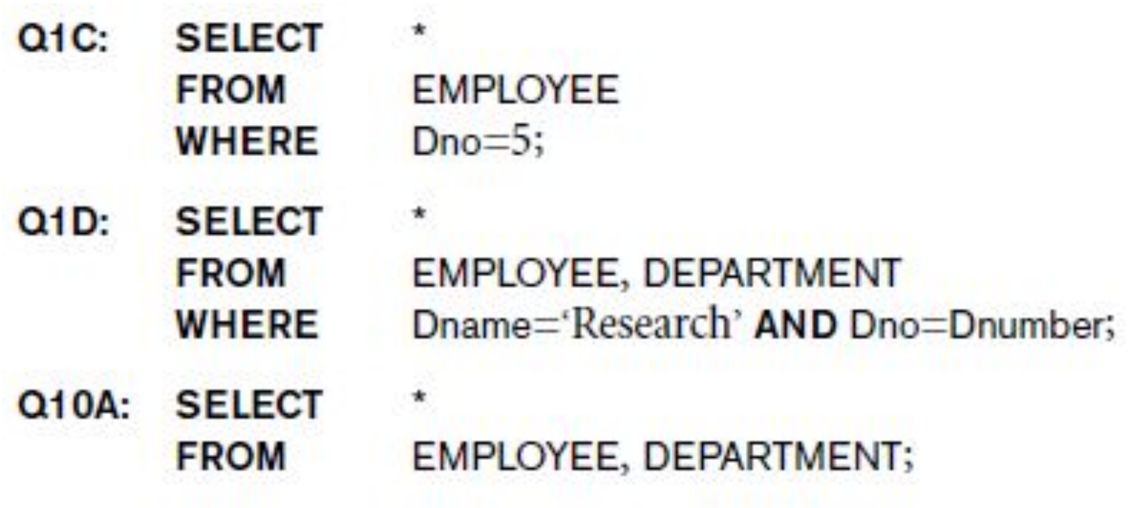


Queries 9 and 10. Select all EMPLOYEE Ssns (Q9) and all combinations of EMPLOYEE Ssn and DEPARTMENT Dname (Q10) in the database.

```
Q9:     SELECT    Ssn
        FROM      EMPLOYEE;

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

Without using WHERE clause (Indicates no condition on tuple selection)

- Specify an asterisk (*)
  - Retrieve all the <u>**attribute values of the selected tuples**</u>

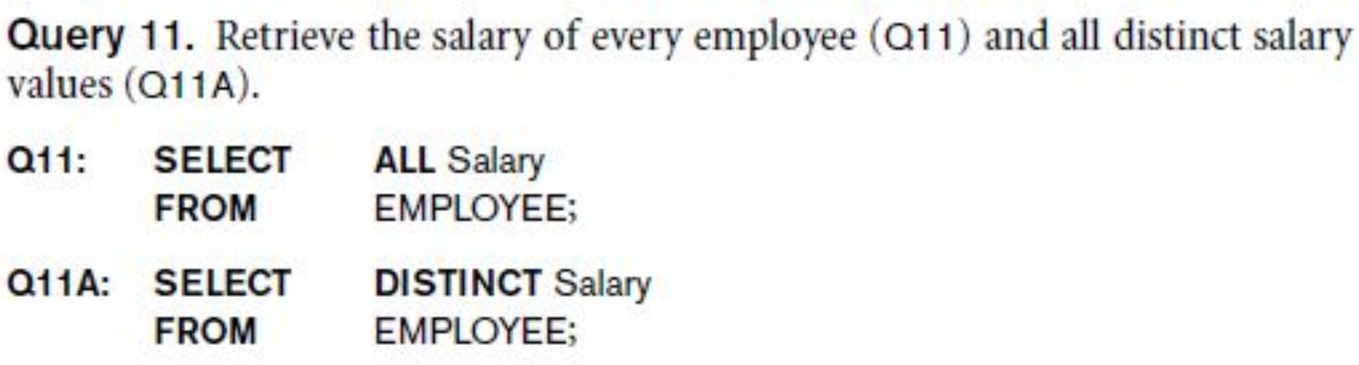


```
Q1C:   SELECT   *
       FROM     EMPLOYEE
       WHERE    Dno=5;

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

Q10A:  SELECT   *
       FROM     EMPLOYEE, DEPARTMENT;
```

Tables as Sets in SQL

- SQL does not **automatically eliminate duplicate tuples** in query results
- So we need to use the keyword **DISTINCT** in the SELECT clause
  - Only **distinct tuples** should remain in the result

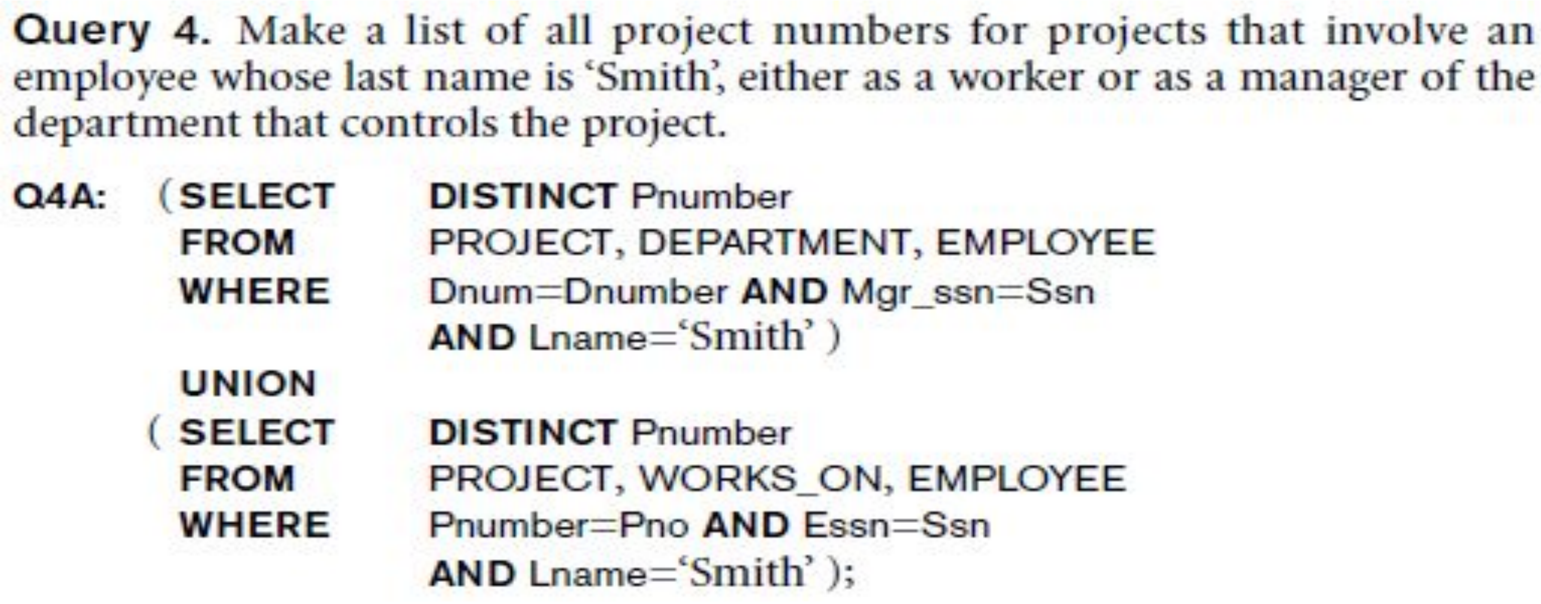
    (It means that duplicates if any that will be avoided)



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;
```

- Set operations
  - UNION, **EXCEPT** (difference), **INTERSECT**
  - Corresponding multiset operations: UNION ALL, EXCEPT ALL, INTERSECT ALL)



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' );
```

Substring Pattern Matching and Arithmetic Operators

- **LIKE** comparison operator
  - Used for string **pattern matching**
  - % replaces an arbitrary number of zero or more characters
  - underscore (_) replaces a single character

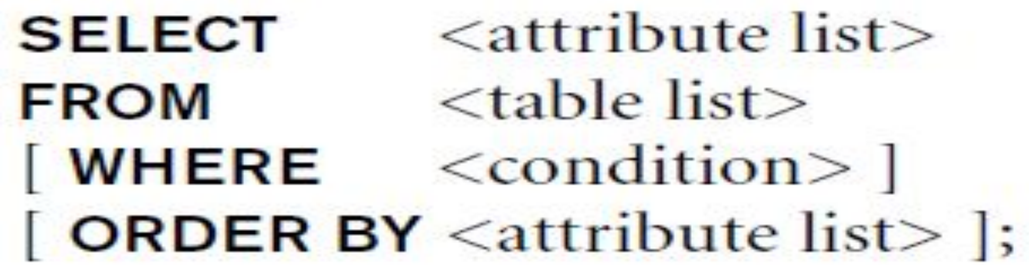
- Standard arithmetic operators:
  - Addition (+), subtraction (–), multiplication (*), and division (/)

- **BETWEEN** comparison operator

Ordering of Query Results

- Use **ORDER BY** clause
  - Keyword **DESC** to see result in a descending order of values
  - Keyword **ASC** to specify ascending order explicitly
  - ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC

Discussion and Summary of Basic SQL Retrieval Queries



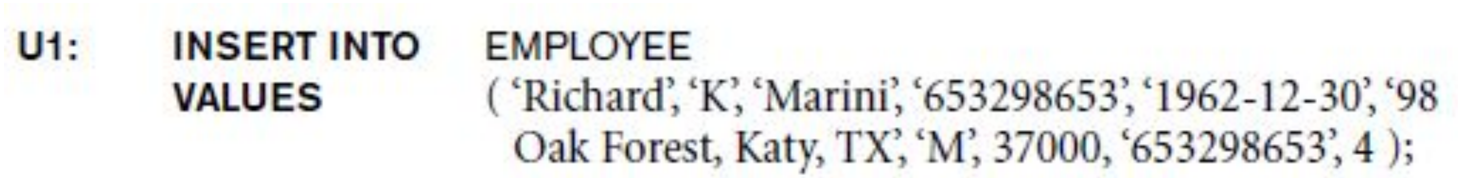
```
SELECT        <attribute list>
FROM          <table list>
[ WHERE       <condition> ]
[ ORDER BY <attribute list> ];
```

INSERT, DELETE, and UPDATE Statements in SQL

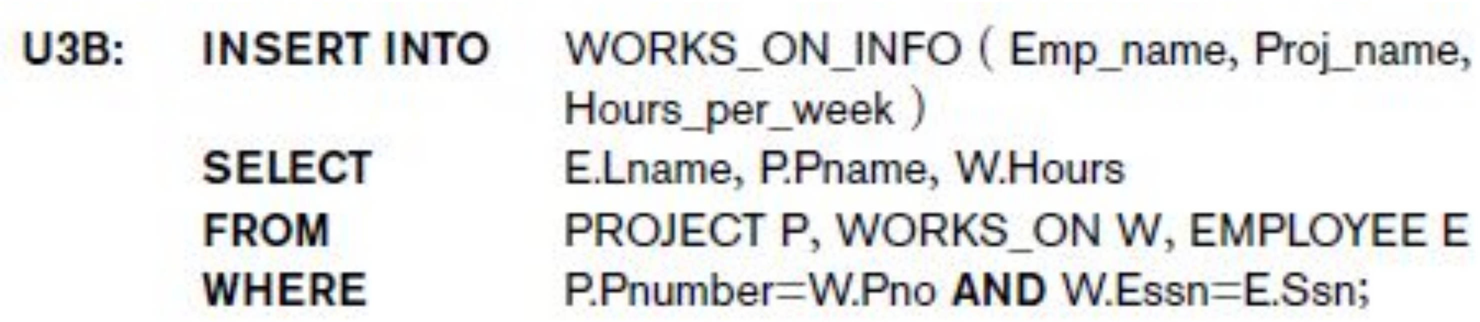
- Three commands used to modify the database:
  - INSERT, DELETE, and UPDATE

The INSERT Command

- Specify the relation name and a list of values for the tuple



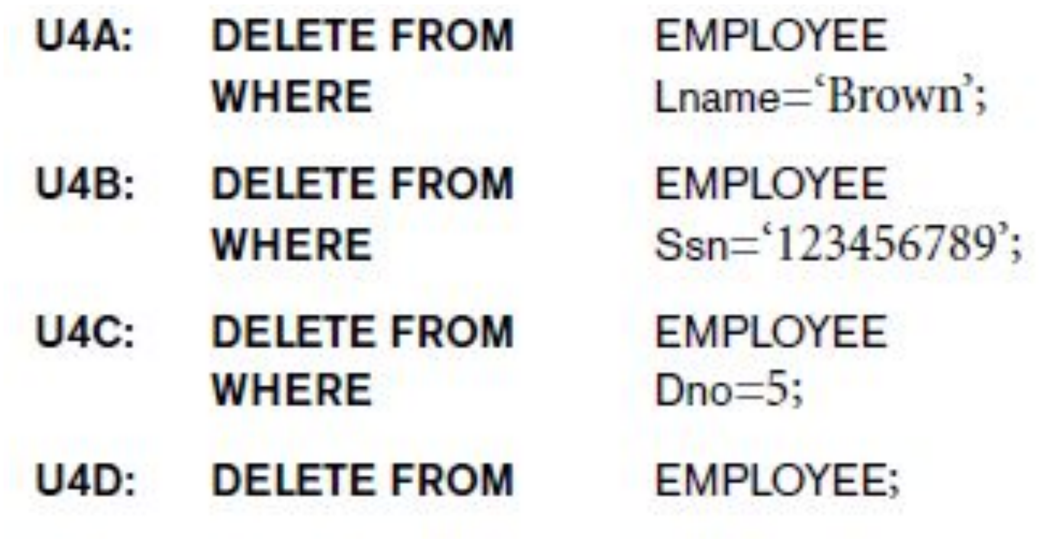
```
U1:    INSERT INTO    EMPLOYEE
       VALUES         ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                        Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```



```
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 DELETE Command

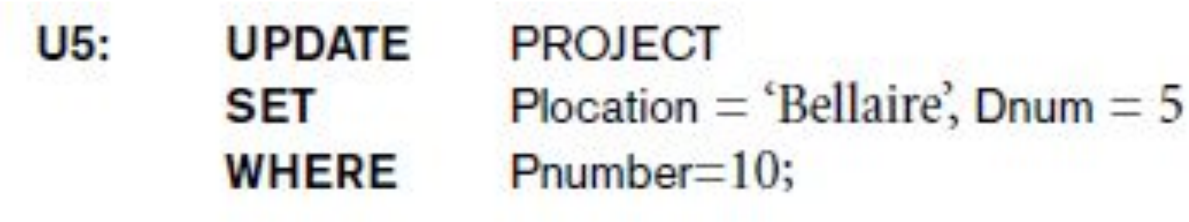
- Removes tuples from a relation
  - Includes a WHERE clause to select the tuples to be deleted



```
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;
```

The UPDATE Command

- 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;
```

Additional features of SQL

- Techniques for specifying complex retrieval queries
- Writing programs in various programming languages that include SQL statements
- Top 5 Programming Languages that include SQL statements

  https://learnsql.com/blog/programming-language-for-sql-developer-and-dba/
- Set of commands for specifying physical database design parameters, file structures for relations, and access paths
- Transaction control commands
- Specifying the granting and revoking of privileges to users

- Constructs for creating triggers
    - A **trigger** is a special type of **stored procedure** that automatically runs when an event occurs in the database server.
    - DML **triggers** run when a user tries to modify data through a data manipulation language (DML) event.
    - DML events are INSERT, UPDATE, or DELETE statements on a table or view.

- Enhanced relational systems known as object-relational

- New technologies such as XML and OLAP (Online Analytical Processing)
    - Stands for "Online Analytical Processing." **OLAP** allows **users to analyse database information from multiple database systems at one time.**
    - While relational databases are considered to be **two-dimensional**, **OLAP** data is **multidimensional** - meaning the information can be compared in many different ways.

Summary

- SQL
  - Comprehensive or Complete language
  - Data definition, queries, updates, constraint specification, and view definition

- Topics Covered in Unit-III
  - Data definition commands for creating tables
  - Commands for constraint specification
  - Simple retrieval queries
  - Database update commands

END OF UNIT - III