Chapter 6: Basic SQL

CS-6360 Database Design

Instructor: Chris Irwin Davis

chrisirwindavis@utdallas.edu

- 6.1 – SQL Data Definition and Data Types
- 6.2 – Specifying Constraints in SQL
- 6.3 – Basic Retrieval Queries in SQL
- 6.4 – `INSERT`, `DELETE`, and `UPDATE` Statements in SQL
- 6.5 – Additional Features of SQL

6.1 – SQL Data Definition and Data Types

Basic SQL

- SQL (Structured Query Language)
  - Considered one of the major reasons for the commercial success of relational databases
  - Appeared 1974
  - Last stable release SQL:2011
  - Core specification - Standards
  - ANSI (since 1986)
  - ISO/IEC 9075 (since 1987)

Basic SQL

- **Statements for data definitions, queries, and updates**
  - DDL, DML, and VDL
  - Plus specialized **extensions** (which may be implementation specific)

Basic SQL

- SQL *language* is case insensitive
  - keywords
  - namespaces
- SQL *data values* are case sensitive
- For readability, some case *style* conventions may be used
- Each statement in SQL ends with a semicolon
  - with some exemptions (e.g. `USE`)

Schema and Catalog Concepts in SQL

- **SQL schema (In most systems, a Database)**
  - Identified by a **schema name**
  - Includes an **authorization identifier** and **descriptors** for each element
- Schema **elements** include
  - Tables
  - Constraints
  - Views
  - Domains
  - and other constructs

SQL Terminology

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- `CREATE` statement
  - Main SQL command for data definition
- Much of what you’ll see in the Data Definition Language is normally done with user-friendly tools like SQL Server Management Studio, etc.

Schema and Catalog Concepts in SQL

- **`CREATE SCHEMA`** statement
  - `CREATE SCHEMA` *`schema_name [auth]`*`;`
  - e.g.
    `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

The CREATE TABLE Command in SQL

- Specify a new relation (table)
  - Provide name
  - Specify attributes and initial constraints
- Can optionally specify schema:
  - `CREATE TABLE COMPANY.EMPLOYEE ...`

    or
  - `CREATE TABLE EMPLOYEE ...`

The CREATE TABLE Command in SQL

- **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

CREATE TABLES for Company Data

```
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 TABLES for Company Data

```
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.

CREATE TABLE Command

- Does the order of table creation matter?
- Some foreign keys may cause errors
- Specified either via:
  - Circular references
  - Or because they refer to a table that has not yet been created

Attribute Data Types and Domains in SQL

- Different dialects of SQL may have different types
  - Microsoft SQL Server
  - Oracle 11g, 10g, XE, etc.
  - MySQL
  - IBM DB2
  - PostgreSQL
  - SQLite

Attribute Data Types and Domains in SQL

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

Attribute Data Types and Domains in SQL

- **Bit-string** data types
  - Fixed length: `BIT(`*n*`)`
  - Varying length: `BIT VARYING(`*n*`)`
- **Boolean** data type
  - Values of `TRUE` or `FALSE` or `NULL`
- **DATE** data type (Use DateTime instead)
  - Ten positions
  - Components are `YEAR`, `MONTH`, and `DAY` in the form YYYY-MM-DD

Attribute Data Types and Domains in SQL

- Additional data types
  - **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 value that can be used to increment or decrement an absolute value of a date, time, or timestamp

Attribute Data Types and Domains in SQL

- Custom Domain
  - Name used with the attribute specification
  - Makes it easier to change the data type for a domain that is used by numerous attributes
  - Improves schema readability
  - Example:
    - `CREATE DOMAIN SSN_TYPE AS CHAR(9);`

6.2 – Specifying Constraints in SQL

Specifying Constraints in SQL

- Basic constraints:
  - Key and referential integrity constraints
  - Restrictions on attribute domains and NULLs
  - Constraints on individual tuples within a relation

Giving Names to Constraints

- Keyword **`CONSTRAINT`**
  - Explicitly name a constraint
  - Useful for later altering

Specifying Attribute Constraints and Attribute

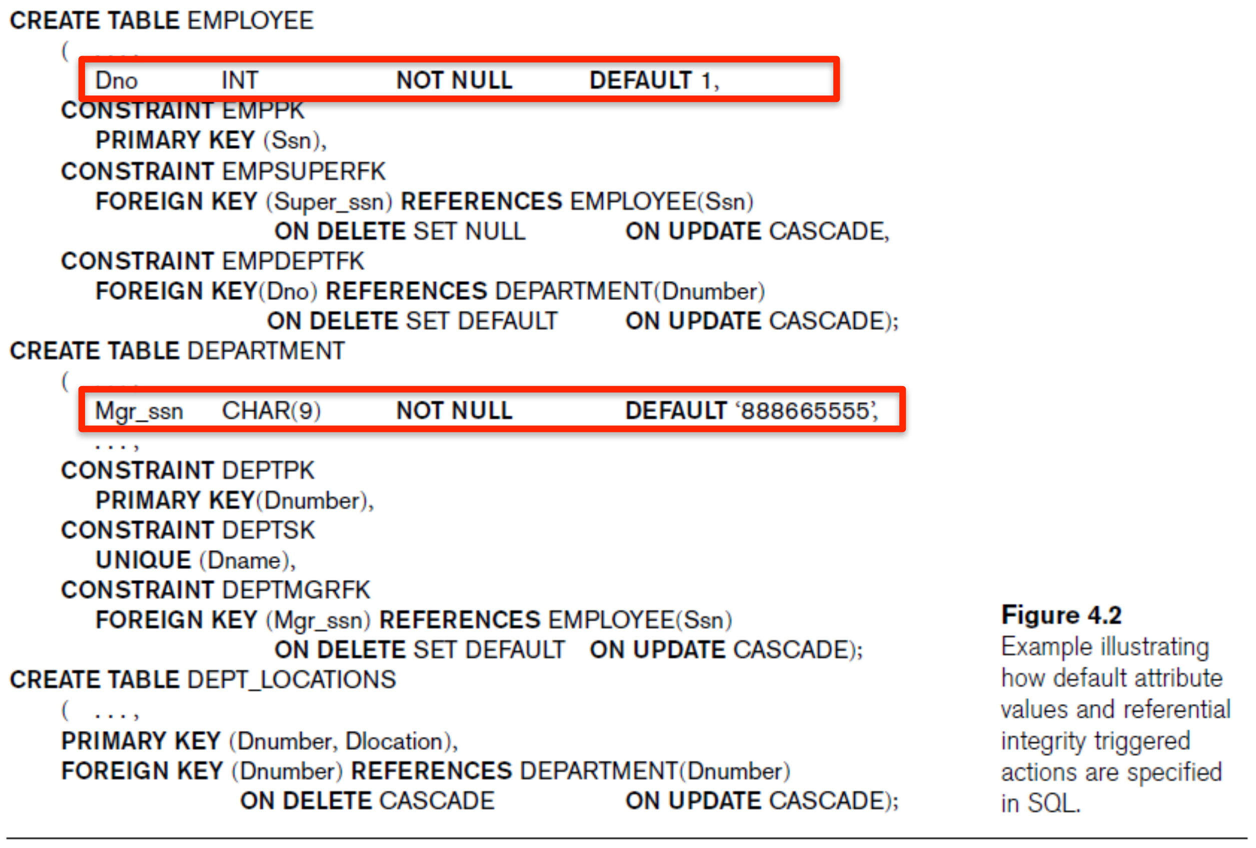
- **`NOT NULL`**
  - **`NULL`** is not permitted for a particular attribute
- Default value
  - **`DEFAULT`** `<value>`
- **`CHECK`** clause

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

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);`**

DEFAULT Clause



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

Specifying Key and Referential Integrity Constraints

- **`FOREIGN KEY`** clause
  - Default operation: reject update on violation
  - 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`**
    - **`CASCADE`** option suitable for "relationship" relations

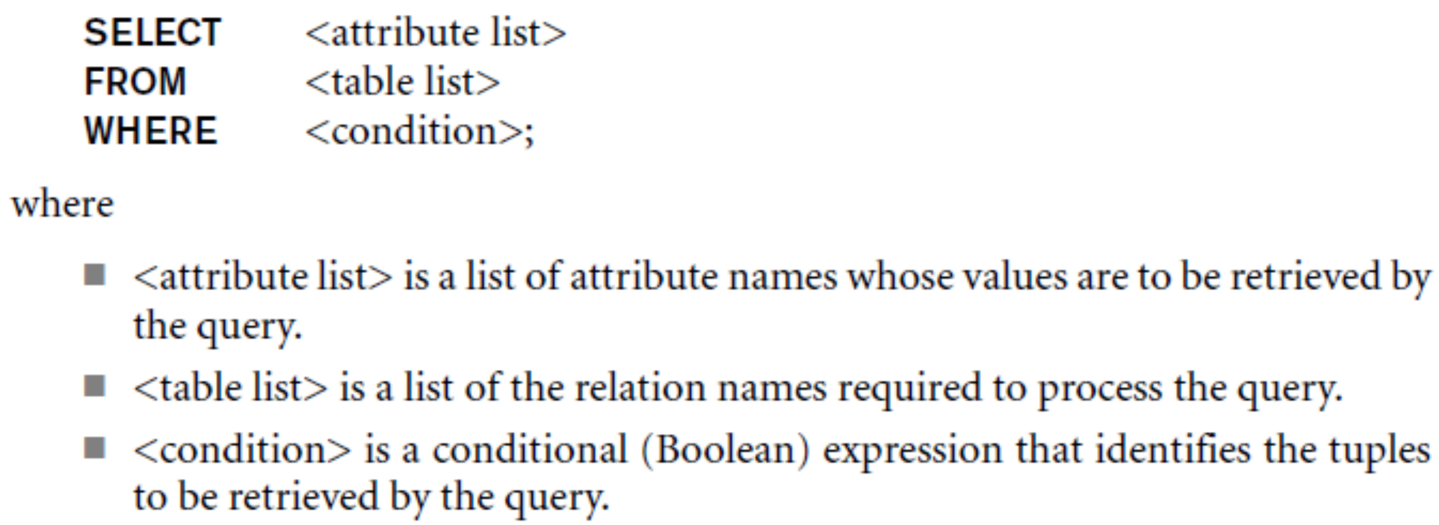
6.3 – Basic Retrieval Queries in SQL

Basic Retrieval Queries in SQL

- **`SELECT`** statement
  - One basic statement for retrieving information from a database
- SQL allows a table to have two or more tuples that are identical in all their attribute values
  - Unlike relational model
  - Multi-set or bag behavior

The 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.

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

Some Queries

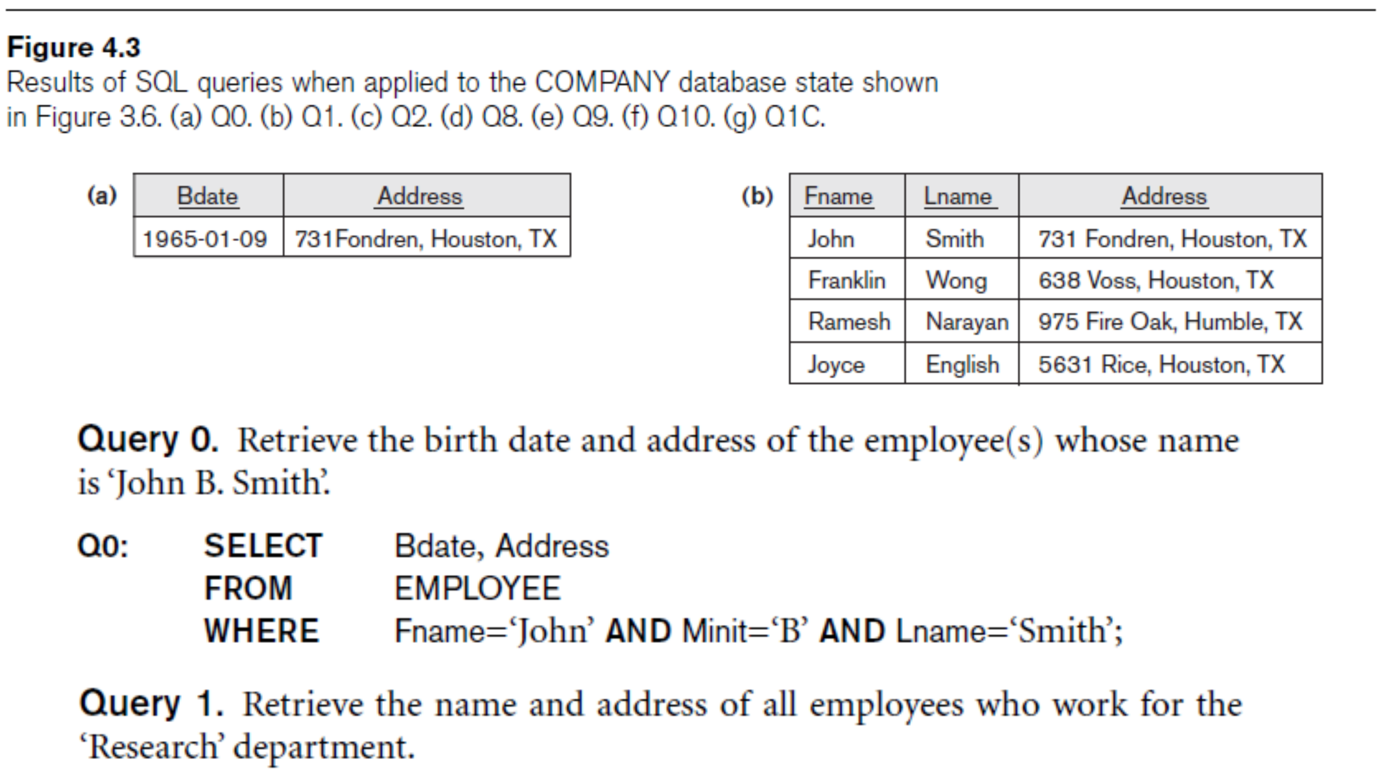


Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(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.

Some Queries

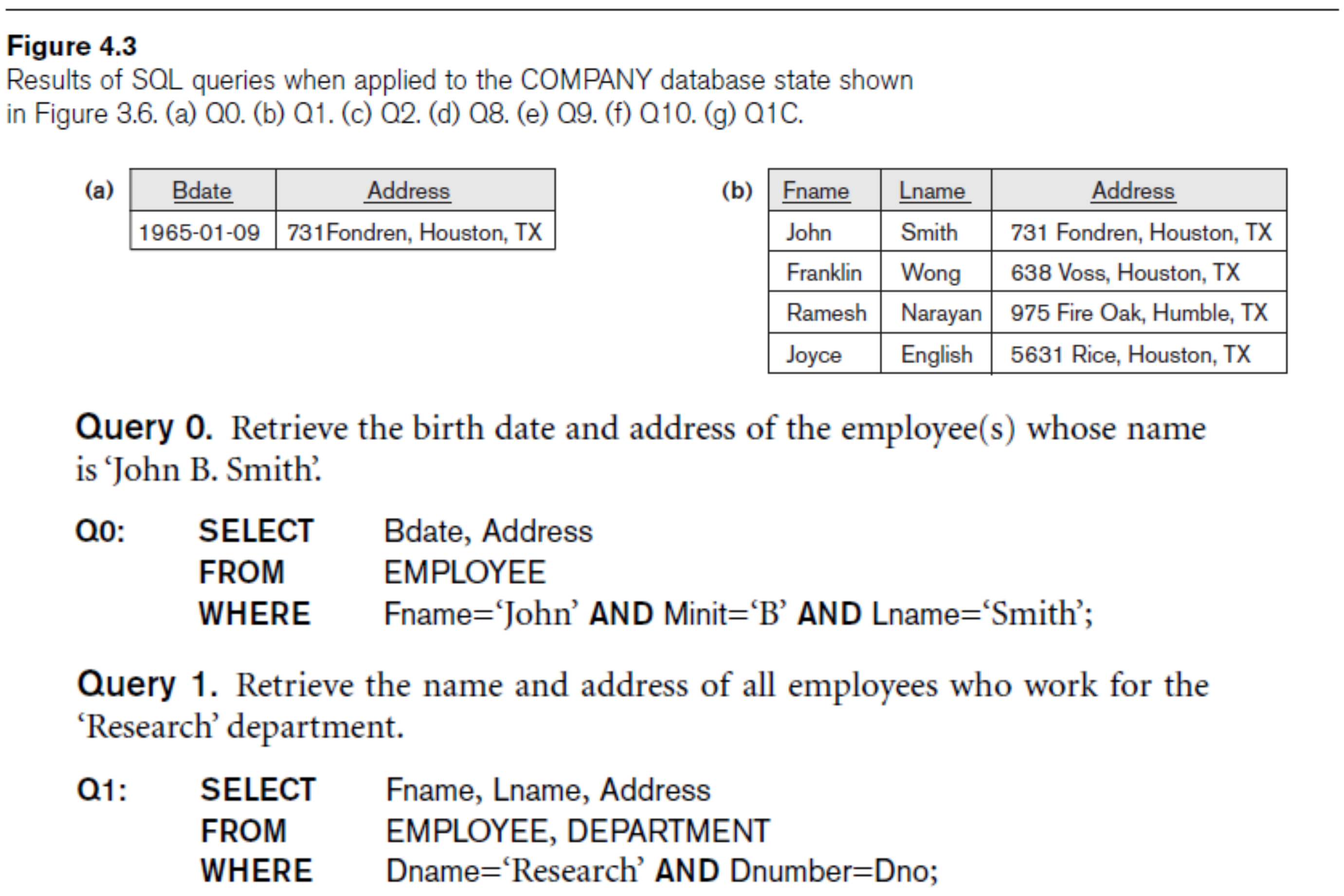


Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(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';
```

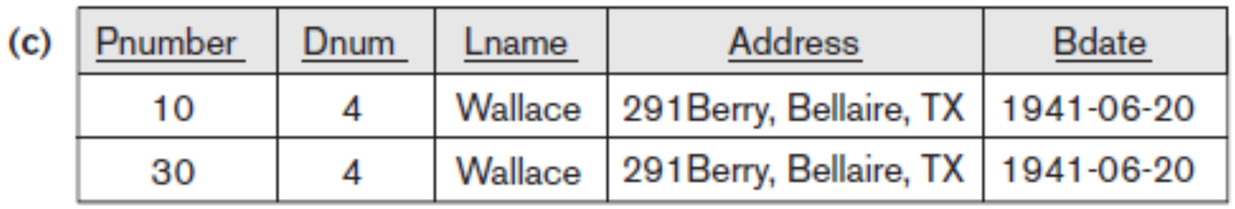
Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1:    SELECT    Fname, Lname, Address
       FROM      EMPLOYEE, DEPARTMENT
       WHERE     Dname='Research' AND Dnumber=Dno;
```

- Query 1
  - 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.

Some Queries

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.

Some Queries

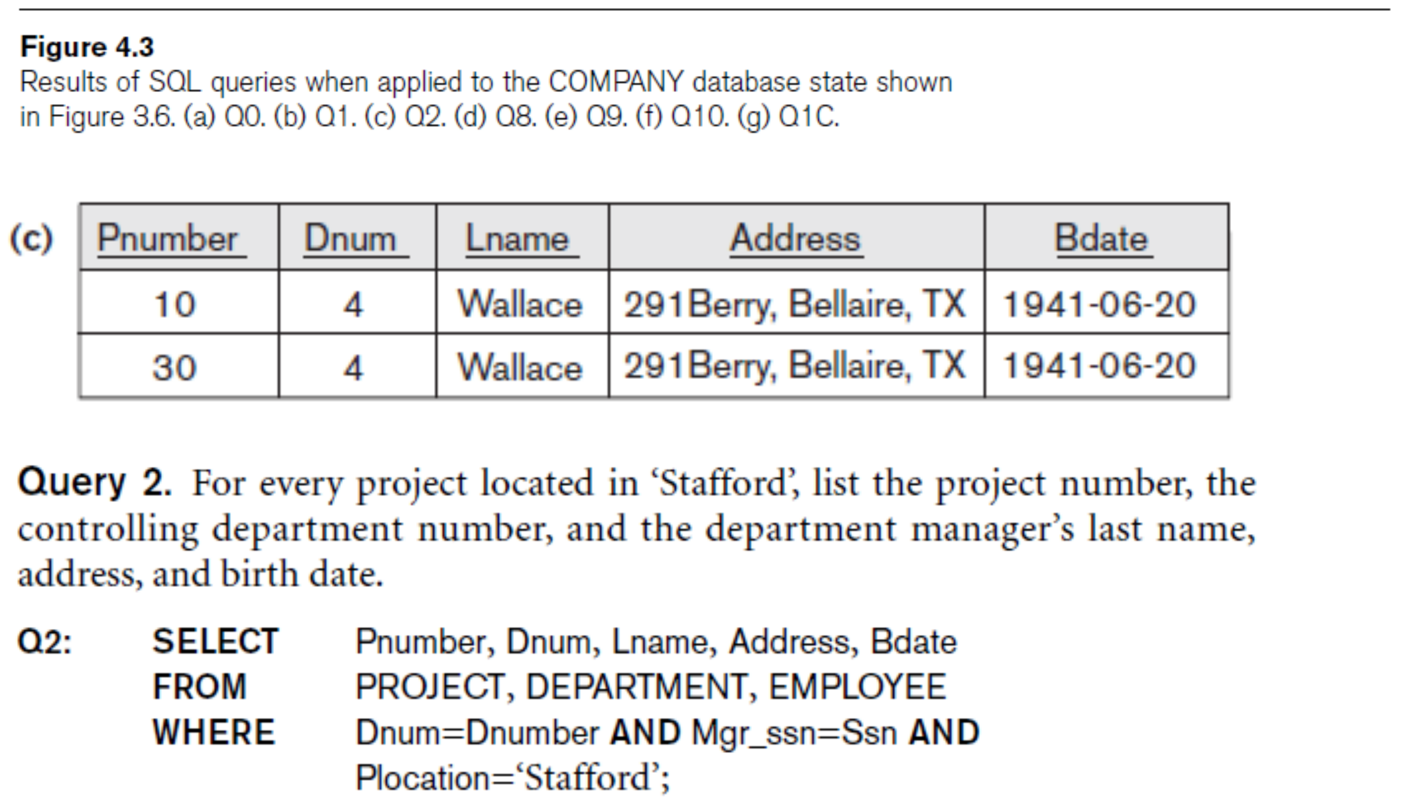


Figure 4.3
Results of SQL queries when applied to the COMPANY database state shown in Figure 3.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(c)

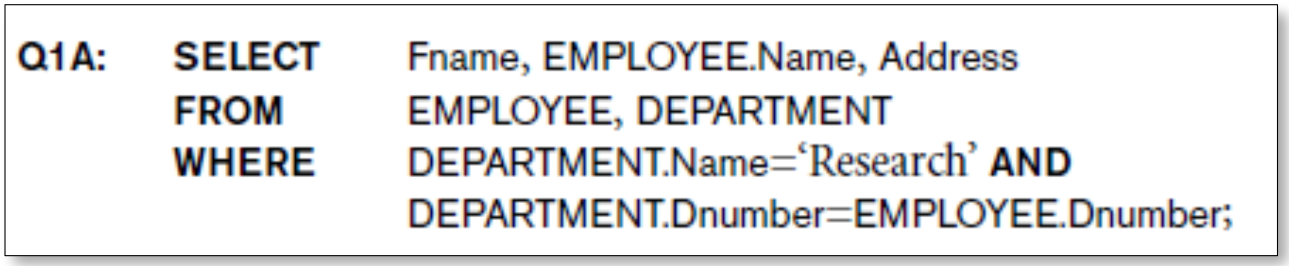
Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

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



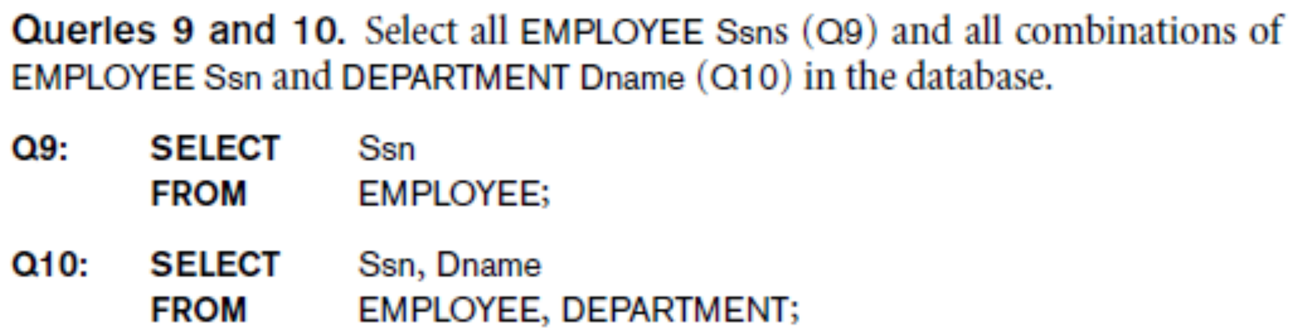
```
Q1A:    SELECT    Fname, EMPLOYEE.Name, Address
        FROM      EMPLOYEE, DEPARTMENT
        WHERE     DEPARTMENT.Name='Research' AND
                  DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Aliasing, Renaming, and Tuple Variables

- The SQL **`AS`** keyword
- Attribute or Table Aliases
- Examples
  - `SELECT` *`attribute`* `AS` *`attr_alias_name`*
  - `FROM` *`table_name`* `AS` *`table_alias_name`*
  - `FROM (`*`subquery`*`) AS` *`virtual_table_name`*
- CAVEAT
  - SQL syntax allows `AS` to be implied
  - A missing comma in a `SELECT` clause can be parsed as an implied `AS`

Unspecified `WHERE` Clause

- 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;
```

Unspecified WHERE Clause and Use of the Asterisk

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

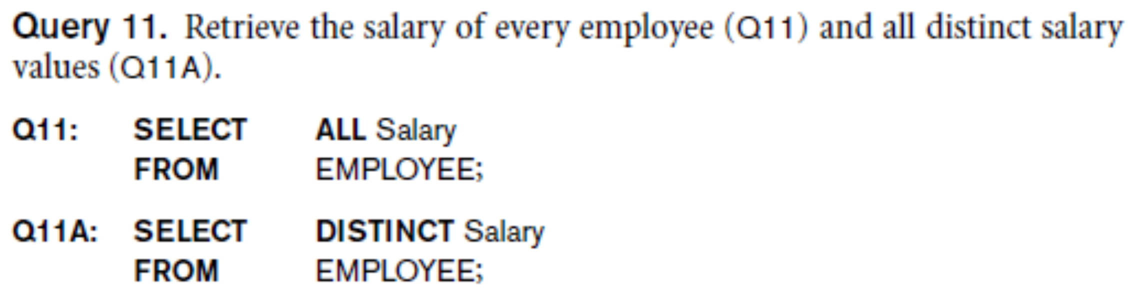
```
Q1C:    SELECT   *
        FROM     EMPLOYEE
        WHERE    Dno=5;

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

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

Tables as Sets in SQL

- SQL does not automatically eliminate duplicate tuples in query results
- Use the keyword **`DISTINCT`** in the **`SELECT`** clause
  - Only distinct tuples should remain in the result



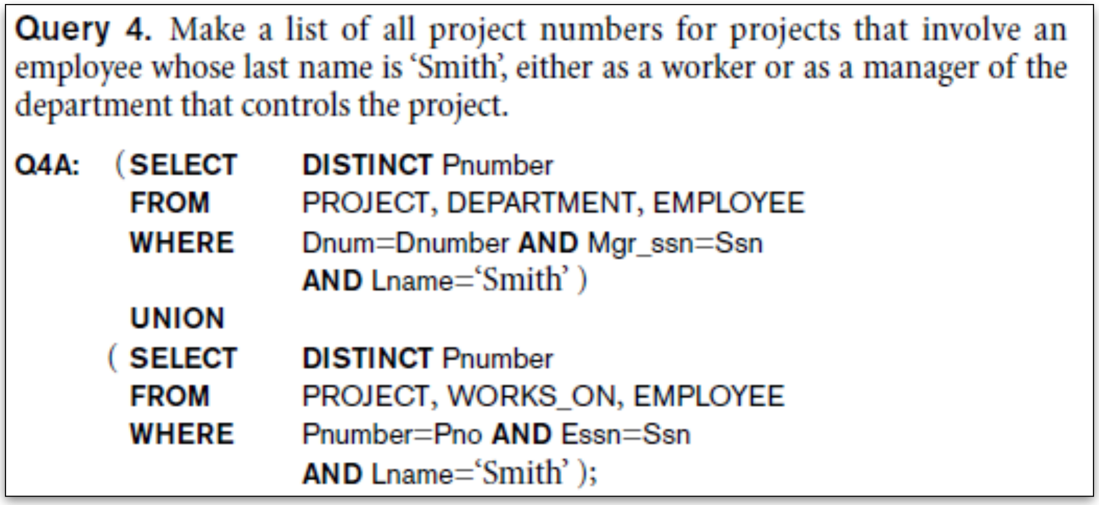
Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).

```
Q11:   SELECT   ALL Salary
       FROM     EMPLOYEE;

Q11A:  SELECT   DISTINCT Salary
       FROM     EMPLOYEE;
```

Tables as Sets in SQL

- Set operations
  - **UNION**, **INTERSECT**, **EXCEPT** (Set difference)
  - Corresponding multiset operations: **UNION ALL**, **EXCEPT ALL**, **INTERSECT ALL**
  - Macros around binary set operators



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

- SQL results are inherently not ordered. To change this, 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> ];
```

6.4 – `INSERT`, `DELETE`, and `UPDATE` Statements in SQL

Modifying the Database Data

- Three commands used to modify the database:
  - **`INSERT`**
  - **`DELETE`**
  - **`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 );
```

The INSERT Command

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

```
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 CREATE TABLE Command

- Define auto-increment at table create time

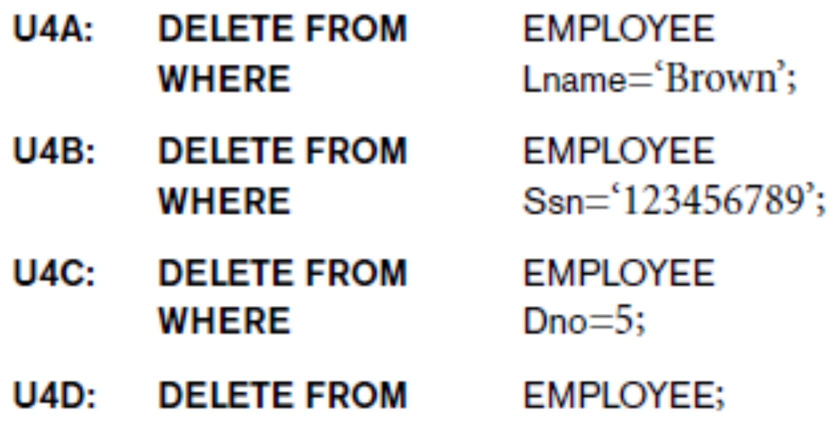
```
CREATE TABLE Persons (
  ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,
  LastName VARCHAR(255) NOT NULL,
  FirstName VARCHAR(255),
  Address VARCHAR(255),
  City VARCHAR(255)
);
```

- Define/Add auto-increment to an existing table

```
ALTER TABLE Persons AUTO_INCREMENT=100
```

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
- Set of commands for specifying physical database design parameters, file structures for relations, and access paths
- Transaction control commands

Additional Features of SQL

- Specifying the granting and revoking of privileges to users
- Constructs for creating triggers
- Enhanced relational systems known as object-relational
- New technologies such as XML and OLAP

Summary

- SQL
  - Comprehensive language
  - Data definition, queries, updates, constraint specification, and view definition
- Covered in Chapter 6:
  - Data definition commands for creating tables
  - Commands for constraint specification
  - Simple retrieval queries
  - Database modification commands