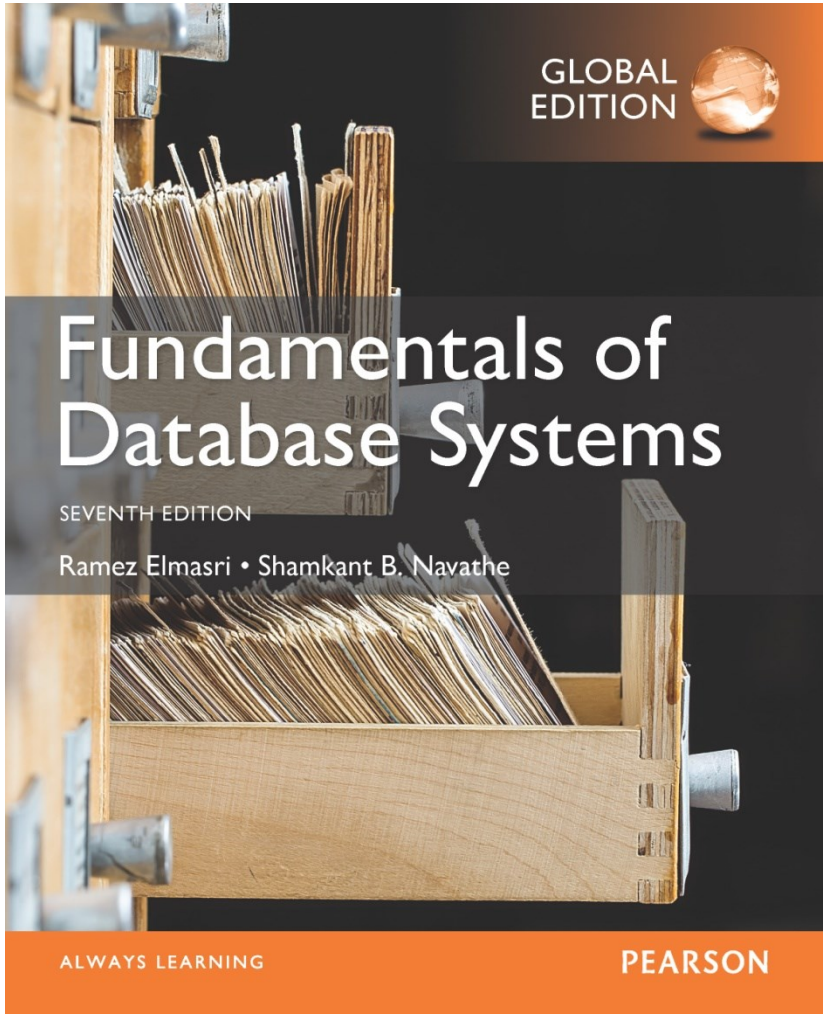


CHAPTER 6

Basic SQL

Chapter 6 Outline

- 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

Basic SQL

- SQL language
  - Considered one of the major reasons for the commercial success of relational databases
  - SQL1 (SQL-86), SQL2 (SQL-92), SQL3 (SQL-99), SQL2003, SQL2006, SQL2008, SQL2011
- SQL
  - The origin of SQL is relational predicate calculus called tuple calculus (see Ch.8) which was proposed initially as the language SQUARE ("Specifying queries as relational expressions").
  - SQL Actually comes from the word "SEQUEL" **(Structured English QUEry Language)** which was the original term used in the paper: "SEQUEL TO SQUARE" by Chamberlin and Boyce. IBM could not copyright that term, so they abbreviated to SQL and copyrighted the term SQL.
  - Now popularly known as "Structured Query language".
  - SQL is an informal or practical rendering of the relational data model with syntax

SQL Standards

- SQL has gone through many standards: starting with SQL-86 or SQL 1. A revised and expanded standard, SQL-92, is referred to as SQL-2.
- Later standards (from SQL-1999) are divided into **core** specification and specialized **extensions**. The extensions are implemented for different applications – such as data mining, data warehousing, multimedia etc.
- SQL-2006 added XML features (Ch. 13); In 2008 they added Object-oriented features (Ch. 12). Further update is SQL-2011.
- SQL-3 is the current standard which started with SQL-1999. It is not fully implemented in any RDBMS.

SQL Data Definition, Data Types, Standards

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms relation, tuple, and attribute
- `CREATE` statement
  - Main SQL command for data definition
- The language has features for : Data definition, Data Manipulation, Transaction control (Transact-SQL, Ch. 20), Indexing (Ch.17), Security specification (Grant and Revoke- see Ch.30), Active databases (Ch.26), Multi-media (Ch.26), Distributed databases (Ch.23) etc.
  - SQL is both a DDL and a DML

Schema and Catalog Concepts in SQL

- We cover the basic standard SQL syntax – there are variations in existing RDBMS systems
- **SQL schema**
  - Identified by a **schema name**
  - Includes an **authorization identifier** and **descriptors** for each element
- **Schema elements** include
  - Tables, constraints, views, domains, and other constructs
- Each statement in SQL ends with a **semicolon**

Schema and Catalog Concepts in SQL (cont'd.)

- `CREATE SCHEMA` statement
  - **`CREATE SCHEMA`** `COMPANY` **`AUTHORIZATION`** `'Jsmith';`
- **Catalog**
  - Named collection of schemas in an SQL environment
- SQL also has the concept of a cluster of catalogs.
- SQL **environment**
  - Installation of an SQL-compliant RDBMS on a computer system

The CREATE TABLE Command in SQL

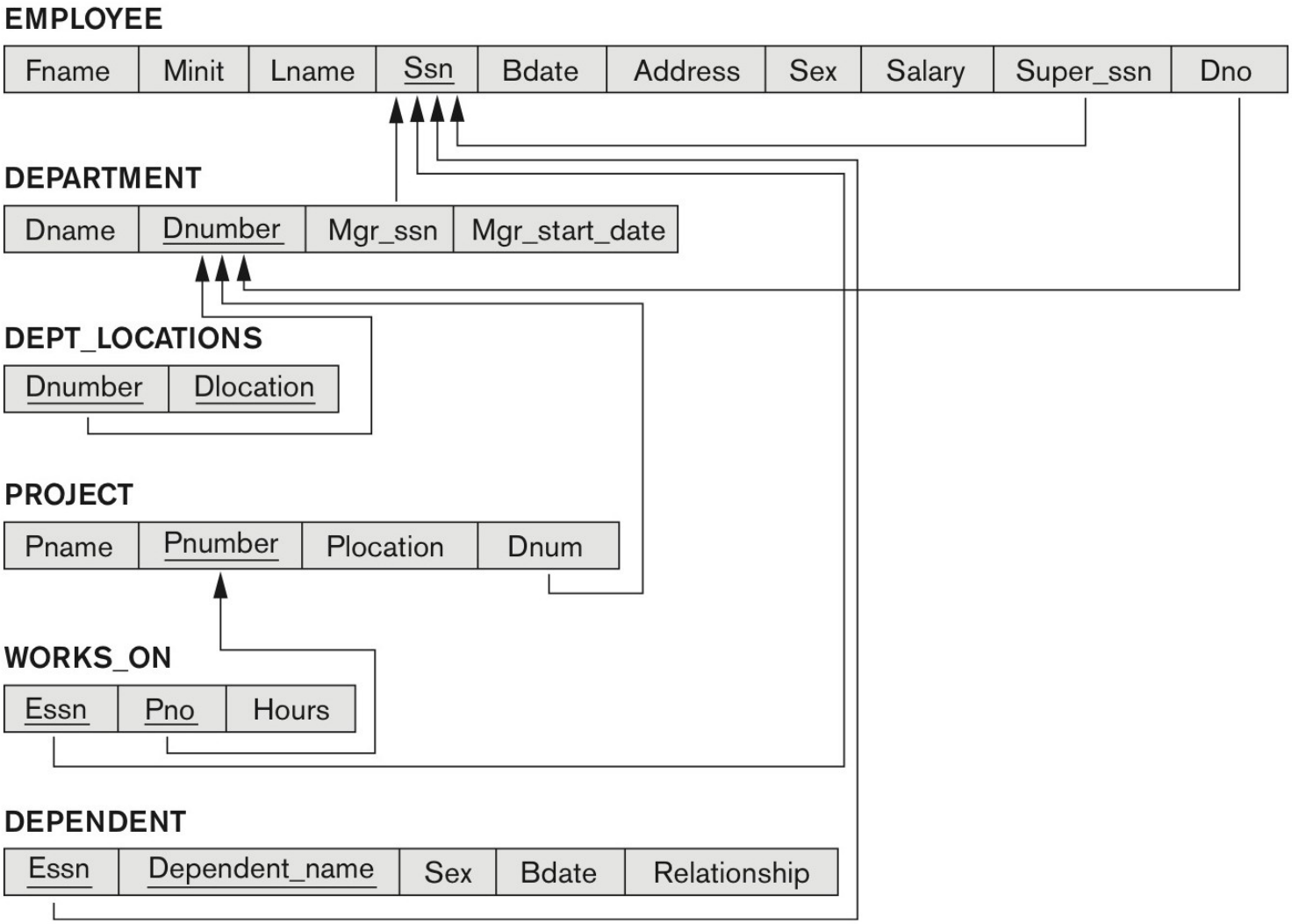
- Specifying a new relation
  - Provide name of table
  - Specify attributes, their types and initial constraints
- Can optionally specify schema:
  - **CREATE TABLE** COMPANY.EMPLOYEE ...

    or
  - **CREATE TABLE** EMPLOYEE ...

The CREATE TABLE Command in SQL (cont'd.)

- **Base tables (base relations)**
  - Relation and its tuples are actually created and stored as a file by the DBMS
- **Virtual relations (views)**
  - Created through the `CREATE VIEW` statement. Do not correspond to any physical file.

COMPANY relational database schema (Fig. 5.7)



One possible database state for the COMPANY relational database schema (Fig. 5.6)

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

One possible database state for the COMPANY relational database schema – continued (Fig. 5.6)

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

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

```
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) );
```

continued on next slide

SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7 (Fig. 6.1)-continued

```
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) );
```

The CREATE TABLE Command in SQL (cont'd.)

- Some foreign keys may cause errors
  - Specified either via:
    - Circular references
      - Ex: EMPLOYEE.Super_ssn refers EMPLOYEE itself
    - Or because they refer to a table that has not yet been created
      - EMPLOYEE.Dno refers to DEPARTMENT
  - DBA's have ways to stop referential integrity enforcement to get around this problem.
    - The constraints are left out of the initial CREATE TABLE statement and then added later using the ALTER TABLE statement

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`
    - **Format is DECIMAL(*i*, *j*) or DEC(*i*, *j*) or NUMERIC(*i*, *j*) where *i* is the total number of decimal digits (the *precision*) and *j* is the number of digits after the decimal point (the *scale*)**
      - **The default for *scale* is zero, and the default for *precision* is implementation-defined**

Attribute Data Types and Domains in SQL (cont'd.)

- **Character-string** data types
  - Fixed length: `CHAR(n)`, `CHARACTER(n)`
    - A shorter string is padded with blank characters to the right
  - Varying length: **`VARCHAR(n)`**, `CHAR VARYING(n)`, `CHARACTER VARYING(n)` where *n* is the maximum number of characters
  - The literal string value is placed between single quotation and is case sensitive. E.g., 'yourString'
  - Concatenation operator denoted by || that can concatenate two string
    - E.g., 'abc'||'XYZ' results in a single string 'abcXYZ'
  - Large text values, such as document: CHARACTER LARGE OBJECT or CLOB
    - CLOB (20M)

Attribute Data Types and Domains in SQL (cont'd.)

- **Bit-string** data types
  - Fixed length: `BIT(n)`
    - The literal bit strings are placed between single but preceded by a B to distinguish them from character string. E.g., B'10101'
    - Large binary values, such as image: **BINARY LARGE OBJECT (or BLOB)**
      - **BLOB(30G)**
  - 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
  - Multiple mapping functions available in RDBMSs to change date formats

Attribute Data Types and Domains in SQL (cont'd.)

- **TIME:**
  - Made up of hour:minute:second in the format hh:mm:ss
- **TIME(i):** time fractional seconds precision
  - Made up of hour:minute:second plus i additional digits specifying fractions of a second
  - format is hh:mm:ss:ii...i
- **TIME WITH TIME ZONE:**
  - includes an additional six positions for specifying the displacement from the standard universal time zone, which is in the range +13:00 ---- -12:59

Attribute Data Types and Domains in SQL (cont'd.)

- Additional data types
  - **Timestamp** data type
    - Includes the `DATE` and `TIME` fields
    - Plus a minimum of six positions for decimal fractions of seconds
    - E.g., TIMESTAMP '2008-09-27 09:12:47.648302'
    - 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
    - Intervals are qualified to be either YEAR-MONTH intervals or DAY-TIME intervals
  - **DATE, TIME, Timestamp, INTERVAL** data types can be **cast** or converted to string formats for comparison.

INTERVAL Data Type

Type identifier	Class	Description
MONTH	Year-Month	Number of months between two dates.
YEAR	Year-Month	Number of years between two dates.
YEAR_TO_MONTH	Year-Month	Number of years and months between two dates.
DAY	Day-Time	Number of days between two dates.
HOUR	Day-Time	Number of hours between two date/times.
MINUTE	Day-Time	Number of minutes between two date/times.
SECOND	Day-Time	Number of seconds between two date/times.
DAY_TO_HOUR	Day-Time	Number of days/hours between two date/times.
DAY_TO_MINUTE	Day-Time	Number of days/hours/minutes between two date/times.
DAY_TO_SECOND	Day-Time	Number of days/hours/minutes/seconds between two date/times.
HOUR_TO_MINUTE	Day-Time	Number of hours/minutes between two date/times.
HOUR_TO_SECOND	Day-Time	Number of hours/minutes/seconds between two date/times.
MINUTE_TO_SECOND	Day-Time	Number of minutes/seconds between two date/times.

```
CREATE TABLE test (
id    DECIMAL PRIMARY KEY,
col1 INTERVAL YEAR TO MONTH,
col2 INTERVAL DAY TO SECOND(6)
);
```

Attribute Data Types and Domains in SQL (cont'd.)

- **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);`
- **TYPE**
  - User Defined Types (UDTs) are supported for object-oriented applications. (See Ch.12) Uses the command: `CREATE TYPE`

Specifying Constraints in SQL

Basic constraints:

- Relational Model has 3 basic constraint types that are supported in SQL:
  - **Key** constraint: A primary key value cannot be duplicated
  - **Entity Integrity** Constraint: A primary key value cannot be null
  - **Referential integrity** constraints : The "foreign key" must have a value that is already present as a primary key, or may be null.

Specifying Attribute Constraints

Other Restrictions on attribute domains:

- Default value of an attribute
  - **DEFAULT <value>**
  - NULL is not permitted for a particular attribute (**NOT NULL**)
- **CHECK** clause (restrictions on attribute domains or individual tuples)
  - CREATE DOMAIN D_NUM AS INTEGER **CHECK (D_NUM > 0 AND D_NUM < 21);**
  - Dnumber INT NOT NULL **CHECK (Dnumber > 0 AND Dnumber < 21);**

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 (called CANDIDATE keys in the relational model).
  - `Dname VARCHAR(15) UNIQUE;`

Specifying Key and Referential Integrity Constraints (cont'd.)

- **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, for relations that represent multivalued attributes, and for relations that represent weak entity types

Giving Names to Constraints

- Using the Keyword **CONSTRAINT**
  - Name a constraint
  - Useful for later altering
    - Use to identify a particular constraint to be dropped later and replaced with another constraint

Default attribute values and referential integrity triggered action specification (Fig. 6.2)

```
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 on Tuples Using CHECK

- Additional Constraints on individual **tuples** within a relation are also possible using CHECK
- `CHECK` clauses at the end of a `CREATE TABLE` statement
  - Apply to each tuple individually and check whenever a tuple is inserted or modified
  - Example: at the end of the CREATE TABLE statement for the DEPARTMENT table to make sure …
    - `CHECK (Dept_create_date <= Mgr_start_date);`
  - CHECK can be used to specify more general constraints using the CREATE ASSERTION statement of SQL

Examples of CHECK Constraints

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

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 (relational model is strictly set-theory based)
  - Multiset or bag behavior
  - Tuple-id may be used as a key
  - SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

Retrieval Queries in SQL (contd.)

- A **bag** or **multi-set** is like a set, but an element may appear more than once.
  - Example: {A, B, C, A} is a bag. {A, B, C} is also a bag that also is a set.
  - Bags also resemble lists, but the order is irrelevant in a bag.
- Example:
  - {A, B, A} = {B, A, A} as bags
  - However, [A, B, A] is not equal to [B, A, A] as lists

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.

The SELECT-FROM-WHERE Structure of Basic SQL Queries (cont'd.)

- 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. Selection conditions include join conditions (see Ch.8) when multiple relations are involved.

Basic Retrieval Queries

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

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';
```

selection condition

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;
```

join condition

selection condition

(Dname='Research') is a selection condition (corresponds to a SELECT operation in relational algebra)
(Dnumber=Dno) is a join condition (corresponds to a JOIN operation in relational algebra)
Such queries are often called **select-project-join** queries

Basic Retrieval Queries (Contd.)

(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';
```

- In Q2, there are two join conditions
- The join condition Dnum=Dnumber relates a project to its controlling department
- The join condition Mgr_ssn=Ssn relates the controlling department to the employee who manages that department

Ambiguous Attribute Names

- Same name can be used for two (or more) attributes in different relations
  - 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, and Renaming

- **Aliases** or **tuple variables**
  - Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:
    - `EMPLOYEE AS E, EMPLOYEE AS S`
- **Query 8.** For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.
  - **SELECT** E.Fname, E.Lname, S.Fname, S.Lname
    FROM EMPLOYEE **AS** E, EMPLOYEE **AS** S
    WHERE E.Super_ssn=S.Ssn;
  - Recommended practice to abbreviate names and to prefix same or similar attribute from multiple tables.

Aliasing,Renaming and Tuple Variables (contd.)

- The attribute names can also be renamed
  - `EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)`
- Note that the relation EMPLOYEE now has a variable name E which corresponds to a tuple variable
  - The alternate relation names E and S are called aliases or **tuple variables** for the EMPLOYEE relation
  - We can think of E and S as two ***different*** **copies** of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*
- The keyword "AS" may be dropped in most SQL implementations (e.g., EMPLOYEE E, EMPLOYEE S)

Unspecified WHERE Clause and Use of the Asterisk

- Missing `WHERE` clause
  - Indicates no condition on tuple selection
  - This is equivalent to the condition WHERE TRUE
- Effect is a **`CROSS PRODUCT`**
  - no join condition
  - Result is all possible tuple combinations (or the Algebra operation of Cartesian Product– see Ch.8)

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 (cont'd.)

- Specify an asterisk (*)
  - Retrieve all the attribute values of the selected tuples
  - The * can be prefixed by the relation name; e.g., EMPLOYEE.* refers to all attributes of the EMPLOYEE table

```
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
  - Duplicate elimination is an expensive operation (sort and eliminate)
  - User may want to see duplicate tuples in the query results
  - For aggregate operations (See sec 7.1.7) duplicates must be accounted for
- 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 (cont'd.)

- Set operations
  - **UNION**, **EXCEPT** (difference), **INTERSECT**
  - Corresponding multiset operations: UNION ALL, EXCEPT ALL, INTERSECT ALL)
  - Type compatibility is needed for these operations to be valid

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' );
```

Set Operations

- The resulting relations of these set operations are sets of tuples
  - *duplicate tuples* are *eliminated* from the result
- The set operations apply only to *union compatible relations*
  - The two relations must have the same attributes and the attributes must appear in the same order
- SQL also has corresponding multiset operations using keyword ALL
  - UNION ALL, EXCEPT ALL, INTERSECT ALL

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
  - Examples:
    - **WHERE** Address **LIKE** '%Houston,TX%';
    - **WHERE** Ssn **LIKE** '_ _ 1_ _ 8901';

- If '_' or '%' is needed as a literal character in the string, the character should be proceeded by an escape character, which is specified after the string using the keyword ESCAPE
  - **'AB_CD\%EF' ESCAPE'\'** represents the string 'AB_CD%EF' because \ is specified as the escape character

Substring Pattern Matching and Arithmetic Operators (cont'd.)

- Query 12: Retrieve all employees whose address is in Houston, Texas.
  - Here, the value of the ADDRESS attribute must contain the substring 'Houston,TX‘ in it.

```
Q12:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Address LIKE '%Houston,TX%';
```

Substring Pattern Matching and Arithmetic Operators (cont'd.)

- Query 12A: Retrieve all employees who were born during the 1950s.
  - Here, '5' must be the third character of the string (according to our format for date), so the BDATE value is '_ _ 5 _ _ _ _ _ _ _ _', with each underscore as a place holder for a single arbitrary character.

```
Q12A:   SELECT   Fname, Lname
        FROM     EMPLOYEE
        WHERE    Bdate LIKE   '_ _ 5 _ _ _ _ _ _ _ _';
```

Arithmetic Operations

- Standard arithmetic operators:
  - Addition (+), subtraction (–), multiplication (*), and division (/) may be included as a part of **SELECT**

- **Query 13.** Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent 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';
```

Substring Pattern Matching and Arithmetic Operators (cont'd.)

- **BETWEEN** comparison operator
- Query 14: Retrieve all employee in department 5 whose salary is between $30,000 and $40,000

```
Q14:    SELECT    *
        FROM      EMPLOYEE
        WHERE     (Salary BETWEEN 30000 AND 40000)
                  AND Dno=5;
```

- The condition (Salary BETWEEN 30000 AND 40000) is equivalent to ((Salary>=30000) AND (Salary<=40000))

Ordering of Query Results

- Use **ORDER BY** clause
  - Used to sort the tuples in a query result based on the values of some attribute(s)
  - The default order is in ascending order of values
  - Keyword **DESC** to see result in a descending order of values
  - Keyword **ASC** to specify ascending order explicitly
  - Typically placed at the end of the query
  - Ex: if we want descending order on Dname and ascending order on Lname, Fname
    - `ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC`

Ordering of Query Results (cont'd.)

- Query 15: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name, first name.

```
Q15:    SELECT    Dname, Lname, Fname, Pname
        FROM      DEPARTMENT, EMPLOYEE,
                  WORKS_ON, PROJECT
        WHERE     Dnumber=Dno AND Ssn=Essn
                  AND Pno=Pnumber
        ORDER BY  Dname, Lname, Fname;
```

Basic SQL Retrieval Query Block

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

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the simple query
- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- ORDER BY specifies an order for displaying the result of a query

INSERT, DELETE, and UPDATE Statements in SQL

- Three commands used to modify the database:
  - INSERT, DELETE, and UPDATE
- INSERT typically inserts a tuple (row) in a relation (table)
- UPDATE may update a number of tuples (rows) in a relation (table) that satisfy the condition
- DELETE may also update a number of tuples (rows) in a relation (table) that satisfy the condition

INSERT

- In its simplest form, it is used to add one or more tuples to a relation
- Attribute values should be listed in the same order as the attributes were specified in the **CREATE TABLE** command
- Constraints on data types are observed automatically
- Any integrity constraints as a part of the DDL specification are enforced

The INSERT Command

- Specify the relation name and a list of values for the tuple. All values including nulls are supplied.

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

- The variation below inserts multiple tuples where a new table is loaded values from the result of a query.

```
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;
```

INSERT (cont'd.)

- An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple
  - Attributes with NULL values can be left out

- Example: Insert a tuple for a new EMPLOYEE for whom we only know the Fname, Lname, Dno, and Ssn attributes.

```
U1A:  INSERT INTO EMPLOYEE (Fname, Lname, Dno, Ssn)
      VALUES ('Richard', 'Marini', 4, '653298653');
```

INSERT (cont'd.)

- Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database
  - It is the responsibility of user to check that any such constraints whose checks are not implemented by the DBMS are not violated

U2: INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno)
VALUES ('Robert', 'Hatcher', '980760540', 2);

- U2 is rejected if referential integrity checking is provided by DBMS (assume that no DEPARTMENT tuple with Dnumber=2 exists)

U2A: INSERT INTO EMPLOYEE (Fname, Lname, Dno)
VALUES ('Robert', 'Hatcher', 5);

- U2A is rejected if NOT NULL checking is provided by DBMS (assume that there is a NOT NULL constraint on Ssn)

BULK LOADING OF TABLES

- Another variation of **INSERT** is used for bulk-loading of several tuples into tables
- A new table TNEW can be created with the same attributes as T and load some of the data currently in T into TNEW using LIKE and DATA in the syntax, it can be loaded with entire data.
- EXAMPLE:

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

DELETE

- Removes tuples from a relation
  - Includes a WHERE-clause to select the tuples to be deleted
  - Referential integrity should be enforced
  - Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
    - The deletion may propagate to tuples in other relations if *referential triggered actions* are specified in the referential integrity constraints of the DDL
  - A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
  - The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

The DELETE Command

- Removes tuples from a relation
  - Includes a `WHERE` clause to select the tuples to be deleted. The number of tuples deleted will vary.

```
U4A:  DELETE FROM   EMPLOYEE
      WHERE         Lname='Brown';

U4B:  DELETE FROM   EMPLOYEE
      WHERE         Ssn='123456789';

U4C:  DELETE FROM   EMPLOYEE
      WHERE         Dno=5;

U4D:  DELETE FROM   EMPLOYEE;
```

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity specified as part of DDL specification is enforced

UPDATE (contd.)

- Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively

```
U5:     UPDATE    PROJECT
        SET       PLOCATION = 'Bellaire',
                  DNUM = 5
        WHERE     PNUMBER=10
```

UPDATE (contd.)

- Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6: UPDATE    EMPLOYEE
    SET       SALARY = SALARY *1.1
    WHERE     DNO  IN (SELECT    DNUMBER
                       FROM      DEPARTMENT
                       WHERE     DNAME='Research')
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
  - The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
  - The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Additional Features of SQL

- Techniques for specifying complex retrieval queries (see Ch.7)
- Writing programs in various programming languages that include SQL statements: Embedded and dynamic SQL, SQL/CLI (Call Level Interface) and its predecessor ODBC, SQL/PSM (Persistent Stored Module) (See Ch.10)
- Set of commands for specifying physical database design parameters, file structures for relations, and access paths, e.g., CREATE INDEX

Additional Features of SQL (cont'd.)

- Transaction control commands (Ch.20)
- Specifying the granting and revoking of privileges to users (Ch.30)
- Constructs for creating triggers (Ch.26)
- Enhanced relational systems known as object-relational define relations as classes. Abstract data types (called User Defined Types- UDTs) are supported with CREATE TYPE
- New technologies such as XML (Ch.13) and OLAP (Ch.29) are added to versions of SQL

Summary

- SQL
  - A Comprehensive language for relational database management
  - Data definition, queries, updates, constraint specification, and view definition
- Covered :
  - Data definition commands for creating tables
  - Commands for constraint specification
  - Simple retrieval queries
  - Database update commands