CHAPTER - 6

Basic SQL

Note: Slides, content, web links and end chapter questions are prepared from Pearson textbook (Elmasri & Navathe), and other Internet resources.

Topics of Discussion

A. Introduction to Structure Query Language.
B. SQL Data Definition and Data Types
    A. CREATE SCHEMA
    B. CREATE TABLE
C. Specifying Constraints in SQL
D. INSERT, DELETE, and UPDATE Statements in SQL
E. Basic Retrieval Queries in SQL
F. Additional Features of SQL

Introduction - Basic SQL

- Structure Query Language (SQL)
  - Considered one of the major reasons for the commercial success of relational databases.
- SQL
  - The origin of SQL is relational predicate calculus called tuple calculus, which was proposed initially as the language SQUARE.
  - (SQL Actually comes from the word “SEQUEL” which was the original term used in the research article: “SEQUEL TO SQUARE” by Chamberlin and Boyce. IBM could not copyright that term, so they abbreviated it 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.

Introduction - Basic SQL

- SQL has gone through many standards: starting with SQL-86 or SQL1. SQL-92 is referred to as SQL-2.
- Later standards (from SQL-1999) are divided into **core** specifications and specialized **extensions**. The extensions are implemented for different applications – such as data mining, data warehousing, multimedia, etc.
- SQL-2006 added XML features; In 2008, added Object-Oriented features.
- SQL-3 is the current standard which started with SQL-1999 and latest within SQL-3 is SQL-2022.
- It is not fully implemented in any (mostly) RDBMS.

Introduction - Basic SQL

- We will discuss the basic standard SQL syntax – there are variations in existing RDBMS systems.
- For example, there may little variation of SQL used for Oracle and for MS SQL Server.
- **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

SQL Data Definition, Data Types

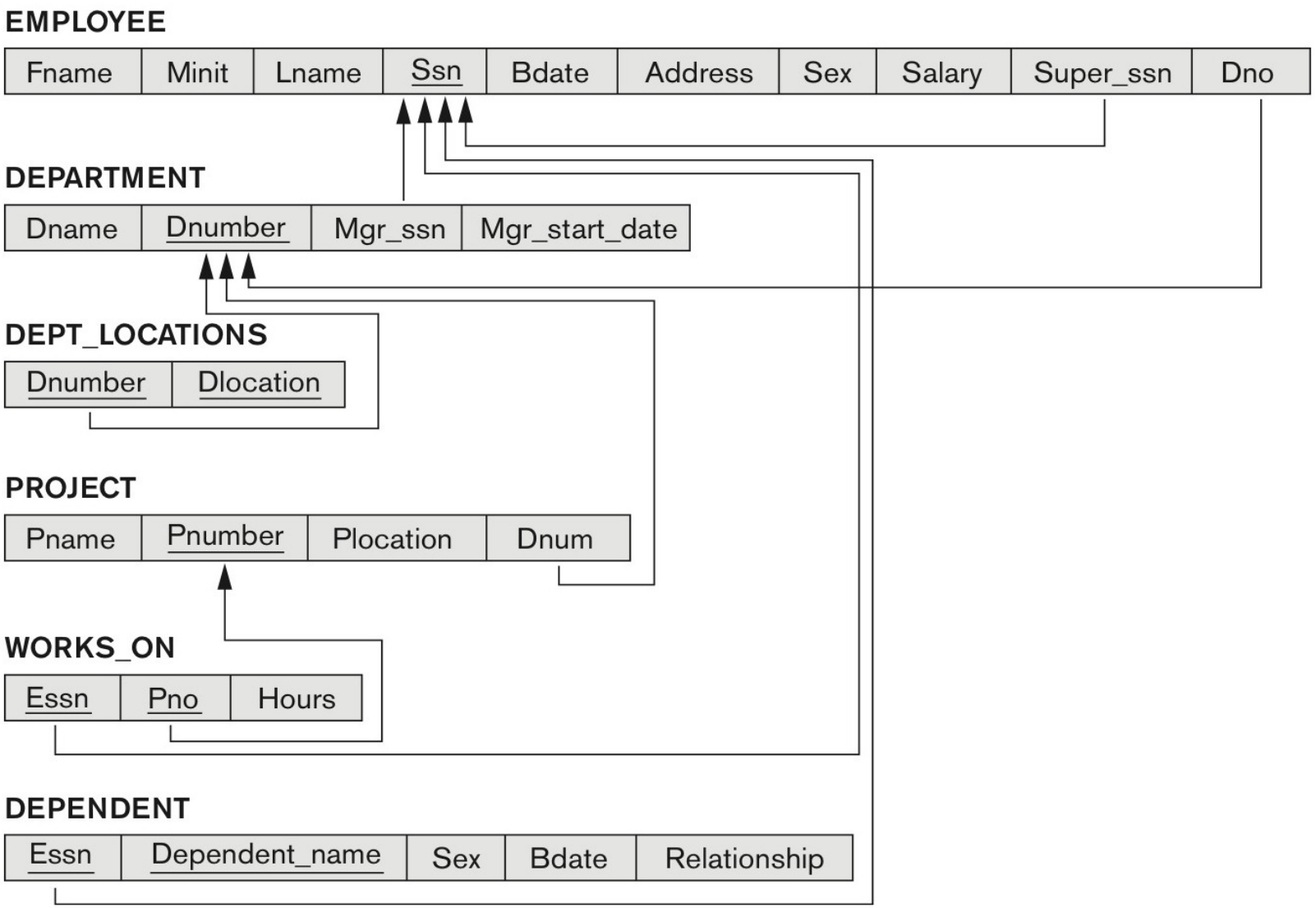
- Terminology:
  - **Table**, **row**, and **column** used for relational model terms **relation, tuple**, and **attribute.**
- `CREATE` statement
  - Main SQL command for data definition
- `CREATE SCHEMA` statement
  - `CREATE SCHEMA` **`COMPANY`** `AUTHORIZATION` ‘apatel38’;
- Catalog
  - Named collection of schemas in an SQL environment
- SQL also has the concept of a cluster of catalogs.
- Each statement in SQL ends with a **semicolon**

The CREATE TABLE Command in SQL

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

    or
  - `CREATE TABLE EMPLOYEE`
- Base tables (base relations)
  - Relation and its tuples are actually created and stored as a file by the DBMS. (Internally when it stored at physical storage like hard disk, it is known as flat file.)
- Virtual relations (views)
  - Created through the `CREATE VIEW` statement.
  - Do not correspond to any physical file.

Example : database schema (Fig. 5.7)



Example : database state (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

continued on next slide

Example : database state (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

CREATE TABLE data definition statements for defining the COMPANY schema (Figure 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 next slide

CREATE TABLE data definition statements for defining the COMPANY schema (Figure 6.1)

```
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) );
```

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

Attribute Data Types and Domains in SQL

- **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
- **Timestamp** data type 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

- **DATE, TIME, Timestamp, INTERVAL** data types can be **cast** or converted to string formats for comparison **DATE** data type.

- **Bit-string** data types
  - Fixed length: `BIT(n)` Varying length: `BIT VARYING(n)`.
  - `Literal bit strings example B'10101'.`
- **BLOB (Binary Large Object)**
  - For large binary value such as images.

Attribute Data Types and Domains in SQL

- **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.
  - 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
  - `Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 10);`

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

- **`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
- Giving Name to Constraint
  - Using the Keyword **`CONSTRAINT`**
  - Name a constraint. Useful for later altering

Default attribute values and referential integrity triggered action specification in SQL (Figure 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

```
CONSTRAINT EMPSUPERFK
  FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
                ON DELETE SET NULL          ON UPDATE CASCADE,
```

- This means that if the row for supervising employee is deleted, the value of Super_ssn is automatically set to NULL for all employee rows that were referencing the deleted employee row.
- On the other hand, if the Ssn value for a supervising employee is updated, (in case of incorrect entry of Ssn) the new value is cascaded to super_ssn for all employee rows referencing the updated employee row.

- 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
  - `CHECK (Dept_create_date <= Mgr_start_date);`

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 rows to a table.
- Attribute values should be listed in the same order as the attributes 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;
```

BULK LOADING OF TABLES

- Most DBMSs have bulk loading tools that allow a user to load formatted data from a file into a table without having to write a large number of INSERT commands.
- One variation of **INSERT** is used for bulk-loading of several tuples into tables is as follow.
  - Example: A new table D5EMPS can be created with the same attributes as EMPLOYEE and using LIKE and DATA in the syntax, it can be loaded with entire data. (or load it with rows of employees who work in department 5.
  - Can be written as below.

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

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
- Ex1: 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

- Ex2: Employee whose ssn is ‘333445555’ is now transferred to department 5.
  - **U1**: UPDATE Employee
    SET dno = 5 WHERE ssn = '333445555’;
- Ex3: 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

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).
  - 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 rows from a table
  - 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;
```

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

```
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

- 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, when multiple relations are involved.

Basic Retrieval Queries in SQL

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';
```

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;
```

Basic Retrieval Queries in SQL

(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 in different relations (tables)
  - As long as the attributes are in different tables.
  - Must **qualify** the attribute name with the table name to prevent ambiguity.
  - Ex: Dno attribute of the EMPLOYEE table was called Dnumber and Dname attribute of DEPARTMENT was also called Name; to prevent ambiguity, query Q1 would be rephrased as shown in Q1A, as below.

```
Q1A: select fname, Lname, address
     from dbo.employee, dbo.department
     where Department.Dname = 'Research' and Department.Dnumber = Employee.Dno;
```

Aliasing, and Renaming

- **Aliases** or **tuple variables**
  - Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:

  Q8. 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

- The attribute names can also be renamed

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

- Note that the table EMPLOYEE now has a variable name E which corresponds to a tuple variable
- The “AS” may be dropped in most SQL implementations

Unspecified WHERE Clause and Use of *

- Missing `WHERE` clause
  - Indicates no condition on tuple selection
- Effect is a `CROSS PRODUCT`
  - Result is all possible tuple combinations (or the Algebra operation of Cartesian Product) result

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 *

- Specify an asterisk (*)
  - Retrieve all the attribute values of the selected tuples
  - The * can be prefixed by the relation name; e.g., EMPLOYEE *
  - Query Q1C retrieves all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5 (Figure 6.3(g))

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

Figure 6.3g Results of SQL queries when applied to the COMPANY database state shown in Figure 5.6. Q1C.

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

Unspecified WHERE Clause and Use of *

- Specify an asterisk (*)
  - 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.
  - Query Q10A specifies the CROSS PRODUCT of the EMPLOYEE and DEPARTMENT tables.

```
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
- For aggregate operations (will see sec 7.1.7) duplicates must be accounted for
- Use the keyword **`DISTINCT`** in the `SELECT` clause
  - Only distinct tuples –rows - 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**, **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' );
```

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%';
- Query 12. Retrieve all employees whose address is in Houston, Texas.

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

Results | Messages

	Fname	Lname
1	John	Smith
2	Franklin	Wong
3	Joyce	English
4	James	Borg
5	Ahmad	Jabbar

Substring Pattern Matching and Arithmetic Operators

- **BETWEEN** comparison operator
- Ex: Q14 : WHERE (Salary **BETWEEN** 30000 **AND** 40000) **AND** Dno = 5;
- Query 14. Retrieve all employees 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;

- Alternate condition: The condition (Salary **BETWEEN** 30000 **AND** 40000) in Q14 is equivalent to the condition ((Salary >= 30000) **AND** (Salary <= 40000)).

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.

Q13: **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';

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
  - Typically placed at the end of the query

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

- **Q15**

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

Basic SQL Retrieval Query Block

- A simple retrieval query in SQL can consist of up to four clauses, but only first two—SELECT and FROM—are mandatory.
- The SELECT clause lists the attributes to be retrieved, and the FROM clause specifies all tables needed in the query. The WHERE clause identifies the conditions for selecting the rows from these tables, including join conditions if needed. ORDER BY specifies an order for displaying the results of a query.

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

Additional Features of SQL

- Techniques for specifying complex retrieval queries. (will 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). (will 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

- Transaction control commands.
- Specifying the granting and revoking of privileges to users (will see at Ch.30)
- Constructs for creating triggers (will see at Ch.10 and 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

End Chapter Questions

- 6.1. How do the relations (tables) in SQL differ from the relations defined formally in Chapter 3? Discuss the other differences in terminology.

- 6.4. Describe the four clauses in the syntax of a simple SQL retrieval query. Show what type of constructs can be specified in each of the clauses. Which are required and which are optional?