



CSIE30600/CSIEB0290
Database Systems

Lecture 4:
Basic SQL

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

- Textbook Chapter 6

Basic SQL

- SQL language
 - Considered one of the major reasons for the commercial success of relational databases
- **SQL**
 - **Structured Query Language**
 - Statements for data definitions, queries, and updates (both DDL and DML)
 - **Core specification**
 - Plus specialized **extensions**

What is SQL?

- **Data manipulation**: ad-hoc queries and updates

```
SELECT *
FROM Account
WHERE Type = "checking";
```
- **Data definition**: creation of tables and views

```
CREATE TABLE Account
(Number integer NOT NULL,
Owner character,
Balance currency,
Type character,
PRIMARY KEY (Number));
```
- **Control**: assertions to protect data integrity

```
CHECK (Owner IS NOT NULL)
```

History of SQL

- IBM **Sequel** (Structured English Query Language) developed as part of System R at the IBM San Jose Research Lab
- Renamed **Structured Query Language (SQL)**
- ANSI and ISO standard SQL:
 - SQL-86, SQL-89, SQL-92 (SQL2)
 - SQL:1999 (SQL3, language name became Y2K compliant!)
 - SQL:2003, 2006 (add XML)
 - SQL:2008 (new statements and better integration with XML)
 - SQL:2011 (add temporal database features)
 - SQL:2016 (add row pattern matching, JSON, etc.)
- Commercial systems may not offer the complete set of features of the standard. May also provide proprietary functions.
- Correctly pronounced “**es cue ell**”, not “sequel”!

CSIE30600/CSIEB0290 Database Systems

Basic SQL 5

SQL is a Declarative Language

- A **procedural (imperative)** language describes **how** to perform some task:
 - relational algebra
 - “project(lname, join(EMPLOYEE, DEP, ssn == essn)))”
- A **declarative** language describes **what** the results are like not how to create it
 - HTML, latex, SQL, tuple relational calculus
 - “The set of all last names of employees such that the SSN of that employee is the ESSN of at least one member of the dependent relation”

CSIE30600/CSIEB0290 Database Systems

Basic SQL 6

SQL has multiple roles

- Data definition language (DDL)
 - Eg, define relation schemas
- Data control language (DCL)
 - Eg, security and authorization controls
- Data manipulation language (DML)
 - Query for tuples
 - Insert, delete and modify tuples
- SQL supports constraints, transactions, views & triggers
- New SQL even supports XML, temporal DB and JSON

Data Definition Language

- Allows the specification of not only a set of relations but also information about each relation, including:
 - The **schema** for each relation.
 - The **domain** of values associated with each attribute.
 - **Integrity constraints**
 - The set of **indices** to be maintained for each relations.
 - **Security** and **authorization** information for each relation.
 - The **physical storage structure** of each relation on disk.
- SQL is case insensitive.

SQL Data Definition and 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

- **CREATE SCHEMA** `<db_name>`
 - creates a DB with the given name
- called **CREATE DATABASE** in MySQL
- example:
 - **CREATE SCHEMA testDB;**
 - **CREATE SCHEMA Company AUTHORIZATION 'Jsmith' ;**
- Each statement in SQL ends with a **semicolon**

Schema and Catalog

- **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
- **Catalog**
 - Named collection of schemas in an SQL environment
- **SQL environment**
 - Installation of an SQL-compliant RDBMS on a computer system

Create Table Construct

- An SQL relation is defined using the **CREATE TABLE** command:

```
CREATE TABLE r (A1 D1, A2 D2, ..., An Dn,
                (integrity-constraint1),
                ...,
                (integrity-constraintk));
```

- *r* is the name of the relation
- each *A*_{*i*} is an attribute name in the schema of relation *r*
- *D*_{*i*} is the data type of values in the domain of attribute *A*_{*i*}
- Example:

```
CREATE TABLE branch
  (branch_name    char(15) not null,
   branch_city   char(30),
   assets         integer);
```

Domain Types in SQL

- **char(n)**. Fixed length (n) character string.
- **varchar(n)**. Variable length character strings, with user-specified maximum length *n*.
- **int**. Integer (a finite subset of the integers that is machine-dependent).
- **smallint**. Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,s)**. Fixed point number, with user-specified precision of *p* digits, with *s* digits to the right of decimal point.
- **real, double precision**. Floating point and double-precision floating point numbers, with machine-dependent precision.

CSIE30600/CSIEB0290 Database Systems

Basic SQL 13

Domain Types in SQL (cont.)

- **float(n)**. Floating point number, with user-specified precision of at least *n* digits.
- **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
 - Ten positions
 - Components are `YEAR`, `MONTH`, and `DAY` in the form `YYYY-MM-DD`

CSIE30600/CSIEB0290 Database Systems

Basic SQL 14

Domain Types in SQL (cont.)

- 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
- More are covered in the book.

Time Related Data Types

- Has `DATE`, `TIME`, and `TIMESTAMP` data types
 - **DATE**:
 - Made up of year-month-day in the format `yyyy-mm-dd`
 - **TIME**:
 - Made up of hour:minute:second in the format `hh:mm:ss`
 - **TIME(i)**:
 - Made up of hour:minute:second plus `i` additional digits specifying fractions of a second
 - format is `hh:mm:ss:ii...i`

Date and Time

- Special data types for dates and times
- Date constant represented by keyword **DATE** followed by a quoted string
 - E.g., **DATE '1971-03-04'**
 - **SELECT * FROM Students**
WHERE birth_date < DATE '1971-03-04'
- Time constant represented by keyword **TIME** followed by a quoted string
 - E.g., **TIME '17:00:02.5'**

Timestamp and Interval

- **TIMESTAMP:**
 - Has both DATE and TIME components
- **INTERVAL:**
 - Specifies a relative value rather than an absolute value
 - Can be DAY/TIME intervals or YEAR/MONTH intervals
 - Can be positive or negative when added to or subtracted from an absolute value, the result is an absolute value

Time Related Examples

Type	Stores	Literal
DATE	year, month, day	DATE 'YYYY-MM-DD'
TIME	hour, minute, and second	TIME 'HH:MM:SS'
TIMESTAMP	year, month, day, hour, minute, and second	TIMESTAMP 'YYYY-MM-DD HH:MM:SS'

Type	Example Literal	Description
Year-Month	INTERVAL '5' YEAR	5 years
	INTERVAL '2' MONTH	2 months
	INTERVAL '3-1' YEAR TO MONTH	3 years and 1 month
Day-Time	INTERVAL '5 10:30:22.5' DAY TO SECOND	5 days, 10 hours, 30 minutes, and 22.5 seconds
	INTERVAL '-5' DAY	5 days ago
	INTERVAL '2 18:00' DAY TO MINUTE	2 days and 18 minutes

CSIE30600/CSIEB0290 Database Systems

Introduction 19

CREATE TABLE

- Specifies a **new base relation** by giving it a **name**, and specifying each of its **attributes** and their **data types** (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint **NOT NULL** may be specified on an attribute

```
CREATE TABLE DEPARTMENT (
    DNAME      VARCHAR(10)    NOT NULL,
    DNUMBER    INTEGER        NOT NULL,
    MGRSSN     CHAR(9) ,
    MGRSTARTDATE CHAR(9) ) ;
```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 20

CREATE TABLE

- We can use the CREATE TABLE command for specifying the **primary key** attributes, **secondary keys**, and referential integrity constraints (**foreign keys**).
- Key attributes can be specified via the **PRIMARY KEY** and **UNIQUE** phrases

```
CREATE TABLE DEPT (
  DNAME          VARCHAR(10)      NOT NULL,
  DNUMBER        INTEGER          NOT NULL,
  MGRSSN         CHAR(9) ,
  MGRSTARTDATE  CHAR(9) ,
  PRIMARY KEY (DNUMBER) ,
  UNIQUE (DNAME) ,
  FOREIGN KEY (MGRSSN) REFERENCES EMP );
```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 21

A more complete 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),
  )
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) );
```

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

CSIE30600/CSIEB0290 Database Systems

Basic SQL 22

```

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) );

```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 23

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);
```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 24

Specifying Constraints in SQL

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

Specifying Attribute Constraints and Attribute Defaults

- **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);`

```

CREATE TABLE EMPLOYEE
(
  ... ,
  Dno          INT          NOT NULL          DEFAULT 1,
  CONSTRAINT EMPFK
  PRIMARY KEY (Ssn),
  CONSTRAINT EMPSUPERFK
  FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
  ON DELETE SET NULL          ON UPDATE CASCADE,
  CONSTRAINT EMPDEPTFK
  FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
  ON DELETE SET DEFAULT      ON UPDATE CASCADE);
CREATE TABLE DEPARTMENT
(
  ... ,
  Mgr_ssn CHAR(9)          NOT NULL          DEFAULT '888665555',
  ... ,
  CONSTRAINT DEPTPK
  PRIMARY KEY(Dnumber),
  CONSTRAINT DEPTSK
  UNIQUE (Dname),
  CONSTRAINT DEPTMGRFK
  FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
  ON DELETE SET DEFAULT      ON UPDATE CASCADE);
CREATE TABLE DEPT_LOCATIONS
(
  ... ,
  PRIMARY KEY (Dnumber, Dlocation),
  FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)
  ON DELETE CASCADE          ON UPDATE CASCADE);

```

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

CSIE30600/CSIEB0290 Database Systems

Basic SQL 27

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;

CSIE30600/CSIEB0290 Database Systems

Basic SQL 28

Primary Key can have more than one attributes

- **primary key** (A_1, \dots, A_n)

Example: Declare *branch_name* as the primary key for *branch*.

```
CREATE TABLE branch
  (branch_name char(15),
   branch_city char(30),
   assets integer,
   primary key (branch_name )
```

primary key declaration on an attribute automatically ensures **not null** in SQL-92 onwards, needs to be explicitly stated in SQL-89

Specifying 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 (next slide)
 - 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

Referential Integrity Options

- We can specify **CASCADE**, **SET NULL** or **SET DEFAULT** on referential integrity constraints (foreign keys)

```
CREATE TABLE DEPT (
    DNAME          VARCHAR(10)    NOT NULL,
    DNUMBER        INTEGER        NOT NULL,
    MGRSSN         CHAR(9),
    MGRSTARTDATE   CHAR(9),
    PRIMARY KEY (DNUMBER),
    UNIQUE (DNAME),
    FOREIGN KEY (MGRSSN) REFERENCES EMP (ESSN)
    ON DELETE SET DEFAULT
    ON UPDATE CASCADE);
```

Referential Integrity Options (cont.)

```
CREATE TABLE EMP (
    ENAME          VARCHAR(30)    NOT NULL,
    ESSN           CHAR(9),
    BDATE          DATE,
    DNO            INTEGER DEFAULT 1,
    SUPERSSN       CHAR(9),
    PRIMARY KEY (ESSN),
    FOREIGN KEY (DNO) REFERENCES DEPT
    ON DELETE SET DEFAULT
    ON UPDATE CASCADE,
    FOREIGN KEY (SUPERSSN) REFERENCES EMP
    ON DELETE SET NULL
    ON UPDATE CASCADE);
```


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);`

CHECK

- **CHECK (*P*)**, where *P* is a predicate
 - *P* must be satisfied by all tuples
- Example: Declare branch-name as the primary key for branch and ensure that the values of assets are nonnegative.

CREATE TABLE branch

```
(branch-name char(15),
branch-city char(30)
assets integer,
PRIMARY KEY (branch-name),
CHECK (assets >= 0))
```

Drop and Alter Table Constructs

- The **DROP TABLE** command deletes all information about the dropped relation from the database.
- The **ALTER TABLE** command is used to **add attributes** to an existing relation:

ALTER TABLE *r* add *A D*

where *A* (attribute name) and *D* is the domain of *A*.

- All tuples in the relation are assigned *null* as the value for the new attribute.
- The **ALTER TABLE** command can also be used to **drop attributes** of a relation:

ALTER TABLE *r* drop *A*

where *A* is the name of an attribute of relation *r*

- Dropping of attributes is not supported by many databases

DROP TABLE

- Used to remove a relation (base table) and its definition
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

```
DROP TABLE DEPENDENT ;
```

ALTER TABLE

- Used to add an attribute to one of the base relations
 - The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is not allowed for such an attribute
 - Example:
- ```
ALTER TABLE EMPLOYEE ADD JOB VARCHAR(12) ;
```
- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple.
    - This can be done using the UPDATE command.

## Retrieval Queries

- SQL has one basic statement for retrieving information from a database: **SELECT** statement
  - This is *not the same as* the  $\sigma$  of the relational algebra
- Important distinction between SQL and the formal relational model:
  - SQL allows a table (relation) to have **two or more** tuples that are identical in all their attribute values
  - Hence, an SQL relation (table) is a **multi-set** (a **bag**) of tuples; it is *not* a set of tuples
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

CSIE30600/CSIEB0290 Database Systems

Basic SQL 39

## Bag

- 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 is also 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

CSIE30600/CSIEB0290 Database Systems

Basic SQL 40

## SELECT Statement

- Basic form of the SQL SELECT statement is called a *mapping* or a SELECT-FROM-WHERE *block*

**SELECT**     <attribute list>  
**FROM**       <table list>  
**WHERE**      <condition>

- <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

## SELECT Statement

- 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

## Basic Query Structure

- A typical SQL query has the form:

**SELECT**  $A_1, A_2, \dots, A_n$   
**FROM**  $r_1, r_2, \dots, r_m$   
**WHERE**  $P$

$A_i$  represents an attribute

- $r_i$  represents a relation
- $P$  is a predicate.
- This query is equivalent to the relational algebra expression (more about this in later chapter)

$$\prod_{A_1, A_2, \dots, A_n} (\sigma_P (r_1 \times r_2 \times \dots \times r_m))$$

- The result of an SQL query is a relation.

## COMPANY Database Schema

### EMPLOYEE

|       |       |       |            |       |         |     |        |          |     |
|-------|-------|-------|------------|-------|---------|-----|--------|----------|-----|
| FNAME | MINIT | LNAME | <u>SSN</u> | BDATE | ADDRESS | SEX | SALARY | SUPERSSN | DNO |
|-------|-------|-------|------------|-------|---------|-----|--------|----------|-----|

### DEPARTMENT

|       |                |        |              |
|-------|----------------|--------|--------------|
| DNAME | <u>DNUMBER</u> | MGRSSN | MGRSTARTDATE |
|-------|----------------|--------|--------------|

### DEPT\_LOCATIONS

|                |                  |
|----------------|------------------|
| <u>DNUMBER</u> | <u>DLOCATION</u> |
|----------------|------------------|

### PROJECT

|       |                |           |      |
|-------|----------------|-----------|------|
| PNAME | <u>PNUMBER</u> | PLOCATION | DNUM |
|-------|----------------|-----------|------|

### WORKS\_ON

|             |            |       |
|-------------|------------|-------|
| <u>ESSN</u> | <u>PNO</u> | HOURS |
|-------------|------------|-------|

### DEPENDENT

|             |                       |     |       |              |
|-------------|-----------------------|-----|-------|--------------|
| <u>ESSN</u> | <u>DEPENDENT_NAME</u> | SEX | BDATE | RELATIONSHIP |
|-------------|-----------------------|-----|-------|--------------|

# COMPANY Database

| EMPLOYEE | FNAME | MINIT | LNAME   | SSN       | BDATE      | ADDRESS                   | SEX | SALARY | SUPERSSN  | DNO |
|----------|-------|-------|---------|-----------|------------|---------------------------|-----|--------|-----------|-----|
| John     | B     |       | Smith   | 123456789 | 1965-01-09 | 731 Foulton, Houston, TX  | M   | 3000   | 333445555 | 5   |
| Franklin | T     |       | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX     | M   | 4000   | 888665555 | 5   |
| Alice    | J     |       | Zeltya  | 999887777 | 1968-07-19 | 3321 Castle, Spring, TX   | F   | 2500   | 987654321 | 4   |
| Jennifer | S     |       | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellare, TX    | F   | 4300   | 888665555 | 4   |
| Blanche  | K     |       | Nanyan  | 666884444 | 1965-09-15 | 975 First Oak, Humble, TX | M   | 3000   | 333445555 | 5   |
| Joyce    | A     |       | Eynghn  | 453403453 | 1972-07-31 | 5631 Rice, Houston, TX    | F   | 2500   | 333445555 | 5   |
| Ahmad    | V     |       | Jahbar  | 987867867 | 1969-03-29 | 980 Dallas, Houston, TX   | M   | 2500   | 987654321 | 4   |
| James    | E     |       | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston, TX    | M   | 5000   | null      | 1   |

  

| DEPARTMENT     | DNAME | DNUMBER | MGRSSN    | MGRSTARTDATE |
|----------------|-------|---------|-----------|--------------|
| Research       |       | 5       | 333445555 | 1988-05-22   |
| Administration |       | 4       | 987654321 | 1985-01-01   |
| Headquarters   |       | 1       | 888665555 | 1981-06-19   |

  

| DEPT_LOCATIONS | DNUMBER | LOCATION  |
|----------------|---------|-----------|
|                | 1       | Houston   |
|                | 4       | Stafford  |
|                | 5       | Bellare   |
|                | 5       | Sugarland |
|                | 5       | Houston   |

  

| WORKS_ON | ESSN      | DNO | HOURS |
|----------|-----------|-----|-------|
|          | 123456789 | 1   | 32.5  |
|          | 123456789 | 2   | 7.5   |
|          | 666884444 | 3   | 40.0  |
|          | 453403453 | 1   | 20.0  |
|          | 453403453 | 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  |
|          | 98767867  | 10  | 15.0  |
|          | 98767867  | 30  | 5.0   |
|          | 987654321 | 30  | 20.0  |
|          | 987654321 | 20  | 15.0  |
|          | 888665555 | 20  | null  |

  

| PROJECT | PNAME           | PNUMBER | LOCATION  | DNUM |
|---------|-----------------|---------|-----------|------|
|         | ProductX        | 1       | Bellare   | 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   | 1988-04-05 | DAUGHTER     |
|           | 333445555 | Theodore       | M   | 1983-10-25 | SON          |
|           | 333445555 | Joy            | F   | 1958-05-03 | SPOUSE       |
|           | 987654321 | Almer          | M   | 1948-03-28 | SPOUSE       |
|           | 123456789 | Michael        | M   | 1988-01-04 | SON          |
|           | 123456789 | Alice          | F   | 1988-12-30 | DAUGHTER     |
|           | 123456789 | Elizabeth      | F   | 1987-05-05 | SPOUSE       |

CSIE30600/CSIEB0290 Database Systems Basic SQL 45

# Simple SQL Queries

- Basic SQL queries correspond to using the following operations of the relational algebra:
  - SELECT
  - PROJECT
  - JOIN
- All subsequent examples use the COMPANY database

CSIE30600/CSIEB0290 Database Systems Basic SQL 46

## Simple SQL Queries (contd.)

- Example of a simple query on one relation

**Figure 6.3**

Results of SQL queries when applied to the COMPANY database state shown in Figure 5.6. (a) Q0. (b) Q1. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(a)

| Bdate      | Address                  |
|------------|--------------------------|
| 1965-01-09 | 731 Fondren, 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;
```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 47

## Simple SQL Queries (contd.)

**Figure 6.3**

Results of SQL queries when applied to the COMPANY database state shown in Figure 5.6. (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';
```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 48



## Simple SQL Queries (contd.)

- In Q<sub>2</sub>, there are **two** join conditions
- The join condition **Dnum=Dnumber** relates a project to its controlling department
- The join condition **Mgrssn=Ssn** relates the controlling department to the employee who manages that department

## Ambiguous Attribute Names

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in *different relations*
- A query that refers to two or more attributes with the same name must **qualify** the attribute name with the relation name by **prefixing** the relation name to the attribute name
- Example:

```

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

```

## Aliases and Tuple Variables

- Some queries need to refer to the same relation **more than once**
  - In this case, **aliases** are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.  
 Q8: 

```
SELECT E.Fname, E.Lname, S.Fname, S.Lname
 FROM EMPLOYEE E S
 WHERE E.Superssn=S.ssn
```

  - In Q8, 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*

CSIE30600/CSIEB0290 Database Systems

Basic SQL 51

## Aliases and Tuple Variables (contd.)

- Aliasing can also be used in any SQL query for convenience
- Can also use the **AS** keyword to specify aliases  
 Q8: 

```
SELECT E.Fname, E.Lname,
 S.Fname, S.Lname
 FROM EMPLOYEE AS E,
 EMPLOYEE AS S
 WHERE E.Superssn=S.Ssn
```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 52

## Unspecified WHERE-clause

- A *missing WHERE-clause* indicates no condition; hence, all tuples of the relations in the FROM-clause are selected
  - This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.
  - Q9: 

```
SELECT Ssn
FROM EMPLOYEE
```
- If more than one relation is specified in the FROM-clause *and* there is no join condition, then the *CARTESIAN PRODUCT* of tuples is selected

## Unspecified WHERE-clause (contd.)

- Example:
  - Q10: 

```
SELECT Ssn, Dname
FROM EMPLOYEE, DEPARTMENT
```
- It is extremely important not to overlook specifying any selection and join conditions in the WHERE-clause; otherwise, incorrect and very large relations may result

## Use of \*

- To retrieve all the attribute values of the selected tuples, a \* is used, which stands for *all the attributes*
- Examples:

```

Q1C: SELECT *
 FROM EMPLOYEE
 WHERE Dno=5;

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

Q10A: SELECT *
 FROM EMPLOYEE, DEPARTMENT;

```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 55

## Use of DISTINCT

- SQL does not treat a relation as a set; duplicate tuples can appear
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

**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;

```

CSIE30600/CSIEB0290 Database Systems

Basic SQL 56

## Use of DISTINCT

### Figure 6.4

Results of additional SQL queries when applied to the COMPANY database state shown in Figure 5.6. (a) Q11. (b) Q11A. (c) Q16. (d) Q18.

| (a) | Salary |
|-----|--------|
|     | 30000  |
|     | 40000  |
|     | 25000  |
|     | 43000  |
|     | 38000  |
|     | 25000  |
|     | 25000  |
|     | 55000  |

| (b) | Salary |
|-----|--------|
|     | 30000  |
|     | 40000  |
|     | 25000  |
|     | 43000  |
|     | 38000  |
|     | 55000  |

## Challenge Question

- Under what conditions are the following two queries equivalent?

```
SELECT DISTINCT A
FROM Table1;
```

```
SELECT A
FROM Table1;
```

- Note: Two queries are equivalent only if they produce identical results for *all instances*

## DISTINCT: A Word of Caution

- In theory, placing a DISTINCT after select is harmless
- In practice, it is **very expensive**
  - **The time it takes to sort a relation so that duplicates are eliminated can be greater than the time to execute the query itself!**
- *Use DISTINCT only when you really need it*

## Set Operations

- SQL has directly incorporated some set operations
- There is a union operation (**UNION**), and in *some versions* of SQL there are set difference (**MINUS, EXCEPT**) and intersection (**INTERSECT**) 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

## Set Operations (cont.)

- Query 4: Make a list of all project names for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.

**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 (for range)

## Substring Comparison(cont.)

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

```
Q: SELECT FNAME, LNAME
 FROM EMPLOYEE
 WHERE ADDRESS LIKE '%Houston,TX%'
```

## Substring Comparison (cont.)

- Query: Retrieve all employees who were born during the 1950s.
  - Here, '195' must be the prefix of the string (according to the format for date), so the BDATE value is '195\_\_\_\_\_', with each underscore as a place holder for a single arbitrary character.

```
Q: SELECT FNAME, LNAME
 FROM EMPLOYEE
 WHERE BDATE LIKE '195_____'
```

- The LIKE operator allows us to get around the fact that each value is considered atomic and indivisible
  - Hence, in SQL, character string attribute values are not atomic



## String Operations

- Match the name “Main%”  
**LIKE 'Main\%' escape '\'**
- SQL supports a variety of string operations such as
  - concatenation (using “||”)
    - ... **WHERE name = 'Juliana' || 'Freire'**
  - converting from upper to lower case (and vice versa)
    - **UPPER(name); LOWER(name)**
  - finding string length, extracting substrings, etc.

CSIE30600/CSIEB0290 Database Systems

Basic SQL 65

## Arithmetic Operations

- The standard arithmetic operators '+', '-', '\*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result
- Query: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

Q: **SELECT FNAME, LNAME, 1.1\*SALARY**  
**FROM EMPLOYEE, WORKS\_ON, PROJECT**  
**WHERE SSN=ESSN AND PNO=PNUMBER**  
**AND PNAME='ProductX'**

CSIE30600/CSIEB0290 Database Systems

Basic SQL 66

## 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`

## ORDER BY

- Query: 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.

```
Q: SELECT DNAME, LNAME, FNAME, PNAME
FROM DEPARTMENT, EMPLOYEE,
 WORKS_ON, PROJECT
WHERE DNUMBER=DNO AND SSN=ESSN
 AND PNO=PNUMBER
ORDER BY DNAME, LNAME
```

## ORDER BY (cont.)

- The default order is in ascending order of values
- We can specify the keyword **DESC** if we want a descending order; the keyword **ASC** can be used to explicitly specify ascending order, even though it is the default

## 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
 FROM PROJECT P, WORKS_ON W, EMPLOYEE E
 WHERE P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

## INSERT (cont.)

- 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, and SSN attributes.

```
INSERT INTO EMPLOYEE(FNAME, LNAME, SSN)
VALUES ('Richard', 'Marini', '653298653')
```

## INSERT (cont.)

- Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database

## 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;

```

## DELETE (cont.)

- 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 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 (cont.)

- 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**
- **Temporal database** features
- Working with **JSON** data

CSIE30600/CSIEB0290 Database Systems

Basic SQL 79

## 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 update commands

CSIE30600/CSIEB0290 Database Systems

Basic SQL 80



## Assignment 2

- Textbook exercises:
  - Chapter 5: 11, 13, 16
  - Chapter 6: 7, 8, 10, 13
- Due date: **Nov 7, 2017**