Lesson 7.1: Structured Query Language (SQL)

CSC430/530 – DATABASE MANAGEMENT SYSTEMS

OUTLINE

- Introduction.
- Data Definition Language (DDL) commands.
  - CREATE, ALTER, DROP, TRUNCATE.
  - Domains & data types.
  - Constraints specification.
- Data Manipulation Language (DML) commands.
  - INSERT, DELETE, UPDATE.
  - SELECT-FROM-WHERE.

INTRODUCTION

- **Structured Query Language (SQL).**
  - Most widely used relational query language.
  - Serves as a standard language for storing, manipulating, and retrieving data in relational databases.
- **Data Definition Language (DDL).**
  - Commands used to define and modify database schema.
  - *CREATE, ALTER, DROP, TRUNCATE.*
- **Data Manipulation Language (DML).**
  - Commands used to retrieve and manipulate data in a database.
  - *INSERT, DELETE, UPDATE, SELECT.*

INTRODUCTION

- **MySQL.**
  - Open-source relational database management system.
  - Software
    - MySQL Server – service running on a server side of client-server database management system architecture.
    - MySQL Workbench – IDE used for database design, development, and maintenance.
    - WAMP – Windows Apache MySQL PHP
    - phpMyAdmin – Comes with WAMP and similar to MySQL Workbench

DDL CREATE

- **CREATE** statement allows us to create a **database** (*schema*) or a **table** (*relation*).
  - CREATE DATABASE **company;**
    - USE **company;**
  - CREATE TABLE **employee(…);** or CREATE TABLE **company.employee(…);**
- When creating a table:
  - Provide a **name**;
  - Specify **attributes**, their **data types,** and **constraints**;
  - Specify **table constraints** (*optionally*).
    - Giving each constraint a name is a good database implementation practice.

DDL CREATE: DATA TYPES & DOMAIN

- Basic data types of attributes:
  - <u>**Numeric**</u>
    - **TINY INT** – 8 bits
    - **SMALL INT** – 16 bits
    - **MEDIUM INT** – 24 bits
    - **INT** – 32 bits
    - **BIG INT** – 64 bits
    - **FLOAT**
    - **DOUBLE**
  - <u>**Character strings**</u>
    - **CHAR(n)** – fixed length string up to n characters (padded with space if shorter)
    - **VARCHAR(n)** – variable length string up to n characters (only stores required characters)

DDL CREATE: DATA TYPES & DOMAIN

- Basic **data types** of attributes:
  - **Bit strings**
    - **BINARY(n)** – fixed length binary string of n bits
    - **VARBINARY(n)** – variable length binary string
  - **Boolean** (can be true, false, or null)
  - **Date and Time**
    - **Date** – format is YYYY-MM-DD
    - **Time** – format is HH:MM:SS
    - **DateTime** – format is YYYY-MM-DD HH:MM:SS)
    - **Timestamp** – stored as the number of seconds since epoch (i.e. January 1, 1970)
    - **Year** – format is YYYY or YY
- **Domain** can be explicitly created and used for multiple attributes.
  - CREATE DOMAIN **ssn_type** AS VARCHAR(9);
  - Unfortunately, this is not supported in MySQL ☹

DDL CREATE: CONSTRAINTS (1)

- **Attribute constraints**:
  - **NOT NULL**
    - Can be applied to any regular attribute and is automatically applied to primary key attribute(s).
  - **AUTO_INCREMENT**
    - Useful for surrogate keys. Automatically increments value for each row inserted.
    - Deleted rows do NOT have their value reused.
  - **DEFAULT** *<value>*
    - Value used if the value for an attribute is not specified.
  - **CHECK**
    - Specify a certain condition.
    - CHECK **(salary > 0);**

- **Table constraints**:
  - **Key constraint**.
    - PRIMARY KEY **(ssn);**
  - **Unique constraint** (can be used for candidate keys)
    - UNIQUE **(dname);**
  - **Referential integrity constraint**.
    - FOREIGN KEY **(dno)** REFERENCES **DEPARTMENT (dnumber);**

- **Tuples constraints**:
  - **CHECK** at the end of **CREATE TABLE**
    - Applied to each tuple individually.
    - CHECK **(dept_create_date <= mgr_start_date);**

DDL CREATE: CONSTRAINTS (2)

- **Violation** of **referential integrity constraint** is **rejected** by default.
- Alternatively, **referential triggered action** can be specified:
  - ON DELETE

    SET NULL
    SET DEFAULT
    CASCADE

  - ON UPDATE

    SET NULL
    SET DEFAULT
    CASCADE

DDL CREATE: EXAMPLES

```
CREATE DATABASE university;

USE university;  /* or click on university database */

CREATE TABLE student (
    cwid CHAR(8)  PRIMARY KEY  CHECK(length(cwid) = 8),
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    balance_owed FLOAT  DEFAULT 0.0  CHECK(balance_owed >= 0)
);

DESCRIBE student; /* or click on student table then on "Structure" */
```

DDL CREATE: EXAMPLES

```
CREATE TABLE student (
    cwid CHAR(8),
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    balance_owed FLOAT  DEFAULT 0.0,
    CONSTRAINT student_pk PRIMARY KEY (cwid),
    CONSTRAINT cwid_len CHECK(length(cwid) = 8),
    CONSTRAINT no_neg_bal CHECK(balance_owed >= 0)
);
```

DDL CREATE: EXAMPLES

```
CREATE TABLE staff_member (
    id INT PRIMARY KEY AUTO_INCREMENT,
    firstname VARCHAR(30) NOT NULL  CHECK(firstname <> ""),
    lastname VARCHAR(30) NOT NULL
);
```

DDL ALTER

- **ALTER** used for several table **modifications**:
  - Adding or dropping a **column** (*attribute*).
  - Changing a **column definition**.
  - Adding or dropping **table constraints**.

```
ALTER TABLE employee
ADD COLUMN job VARCHAR(12);


ALTER TABLE department
ADD CONSTRAINT dept_mgr_fk
FOREIGN KEY (mgr_ssn) REFERENCES employee (ssn)
ON DELETE SET NULL     ON UPDATE CASCADE;
```

DDL DROP

- **DROP** used to **delete** entire named schema elements.
  - Tables, domains, constraints, or database itself.

- Examples:
  - DROP TABLE **employee**;
    - This removes the employee table and all its data.
  - DROP DATABASE **company**;
    - This removes the company database and all its elements including tables, views, constraints, etc.

DDL TRUNCATE

- **TRUNCATE** used to remove all data from a table but keep the table schema.
- Essentially it drops the table, then recreates it back, thereby clearing all rows.

- Examples:
  - TRUNCATE TABLE **employee**;
    - This removes all data in the employee table but the table itself remains.

DDL DROP AND TRUNCATE – BE CAREFUL!!

- These commands will remove data, that is their purpose.
- This cannot be undone (hopefully you have a backup!).
- Always take a second look before committing DROP or TRUNCATE queries!

DDL Hands-On

Hands-on

Lab 0: Parts A & B

DML INSERT

- **INSERT** is used to **add** one or more **row** (*tuple*) into **relation** (*table*).
  - Attribute values listed in the **same order** as specified in CREATE TABLE.
  - Rejected if any of defined **constraints** are violated.

```
INSERT INTO employee
VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '123456789', 4);
```

- In addition, INSERT allows to assign values only for a **subset of attributes**.

```
INSERT INTO employee (fname, lname, dno, ssn)
VALUES ('Richard', 'Marini', 4, '653298654');
```

DML DELETE

- **DELETE** is used to **remove** one or more **row** (*tuple*) from **relation** (*table*).
  - **Propagates** to other tuple(s) if **referential trigger actions** are specified.
  - Uses **WHERE** as a **condition** to select tuples to delete.

```
DELETE FROM employee WHERE lname = 'Marini';

DELETE FROM employee WHERE ssn = '653298653';

DELETE FROM employee WHERE dno = 5;
```

DML DELETE – BE CAREFUL!!

- Missing WHERE clause deletes **ALL** tuples.

```
DELETE FROM employee;
```

- This cannot be undone (hopefully you have a backup!).
- Always take a second look before committing DELETE queries!

DML UPDATE

- **UPDATE** is used to **modify attribute** values of one or more selected **tuples**.
  - Uses **WHERE** as a **condition** to select tuples to update.
  - Uses **SET** to **specify** the attributes to be modified and their values.
  - Can cause **referential triggered action** if specified.
    - Updating value of **primary key** attribute will **propagate** an update in respective **foreign keys**.

```
UPDATE project
SET plocation = 'Bellaire', dnum = 5
WHERE pnumber = 10;

UPDATE employee
SET salary = salary * 1.1
WHERE dno = 5;
```

DML UPDATE

- **UPDATE** is used to **modify attribute** values of one or more selected **tuples**.
  - Missing **WHERE** updates all rows

```
UPDATE project
SET dnum = 5;
```

- Also, using where clause that matches all rows will do the same

```
UPDATE project
SET dnum = 5;
WHERE 1 = 1;
```

DML Hands-On

Hands-on

Lab 0: Part C

DML SELECT (1)

- **SELECT** is used to **retrieve** specific data from the database.
- Basic form of SELECT statement (*select-from-where*):

  SELECT ***<attribute list>***

  FROM ***<table list>***

  WHERE ***<condition>*****;**

  - <attribute list> - **attribute names** which values are to be retrieved.
  - <table list> - **relation names** required to process the query.
  - <condition> - **Boolean expression** that identifies the tuples to be retrieved by the query.

- Examples: Select birth date and address of employee John B Smith.

  SELECT **bdate, address**

  FROM **employee**

  WHERE **fname = 'John'** AND **minit = 'B'** AND **lname = 'Smith';**

DML SELECT (2)

- **SELECT-PROJECT-JOIN** query:
  - Select first name, last name and address of all employees who work for Research department.

```
SELECT fname, lname, address
FROM   employee, department
WHERE  dname = 'Research' AND dnumber = dno;
```

  - Select last name, address, and birth date of employees who manage departments with projects located in Stafford.

```
SELECT pnumber, dnum, lname, address, bdate
FROM   project, department, employee
WHERE  dnum = dnumber AND mgr_ssn = ssn AND plocation = 'Stafford';
```

DML SELECT (3)

- **Prefixing** is used when referencing two (or more) attributes with the same name in different relations.
  - ***employee.name*** and ***department.name***
- **Aliasing of relations** (*tuple variables*) is used to rename a relation with an abbreviation.
  - Useful when referring to the same relation twice.
  - **Example:** Select first name and last name of employees and their supervisors.

```
SELECT e.fname, e.lname, s.fname, s.lname
FROM employee AS e, employee AS s
WHERE e.super_ssn = s.ssn;
```

- **Aliasing of attributes** can be done in SELECT part of the query.

```
SELECT e.fname AS "Emp Firstname", e.lname AS "Emp Lastname",
       s.fname AS "Super Firstname", s.lname AS "Super Lastname"
FROM employee AS e, employee AS s
WHERE e.super_ssn = s.ssn;
```

DML SELECT (4)

- **Missing WHERE** selects all tuples if using a single relation

```
SELECT fname FROM employee;
```

- **Missing WHERE** does CROSS PRODUCT if using multiple relations

```
SELECT fname, dname FROM employee, department;
```

- **Asterisk *** used to select all the attributes (no projection / projection on all attributes).

```
SELECT * FROM employee;
```

PRACTICE 1

- Write a query that will select ssns of all employees

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

PRACTICE 1

- Write a query that will select ssns of all employees

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

```
SELECT ssn
FROM employee;
```

PRACTICE 2

- Write a query that will produce the CROSS PRODUCT of the employee and department relations.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date

PRACTICE 2

- Write a query that will produce the CROSS PRODUCT of the employee and department relations.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date

```
SELECT *
FROM employee, department;
```

PRACTICE 3

- Write a query that will select all attributes of employees who work in department number 5.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno

PRACTICE 3

- Write a query that will select all attributes of employees who work in department number 5.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

```
SELECT *
FROM employee
WHERE dno = 5;
```

PRACTICE 4

- Write a query that will retrieve the birth date and address of the employee whose name is ‘John B. Smith’.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

PRACTICE 4

- Write a query that will retrieve the birth date and address of the employee whose name is ‘John B. Smith’.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

```
SELECT bdate, address
FROM employee
WHERE fname='John' AND minit='B' AND lname='Smith';
```

DML Hands-On

Hands-on

Lab 1 – Simple Queries

(submit on Canvas)