Lesson 7.1: Structured Query Language (SQL)

CSC430/530 – DATABASE MANAGEMENT SYSTEMS

DR. ANDREY TIMOFEYEV

OUTLINE

- Introduction.
- Data definition language (DDL) commands.
  - CREATE, ALTER, DROP.
  - 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, DROP, ALTER.*
- **Data Manipulation Language (DML).**
  - Commands used to retrieve and manipulate data in a database.
  - *INSERT, DELETE, UPDATE, SELECT.*
- **MySQL.**
  - Open-source relational database management system.
  - MySQL Workbench – IDE used for database design, development, and maintenance.
  - MySQL Server – service running on a server side of client-server database management system architecture.

DDL CREATE

- **CREATE** statement allows to create a **schema** (*database*) or a **relation** (*table*).
  - CREATE DATABASE **company;** CREATE SCHEMA is basically the same
    - USE **company;** ← allows you to refer to tables directly w/o dot operator
  - CREATE TABLE **employee(…);** or CREATE TABLE **company.employee(…);**
    attributes here
- When **creating** a relation (*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:
  - **Numeric** (INT, SMALLINT, FLOAT, REAL, etc.)
  - **Character-string** (CHAR, VARCHAR(n))
  - **Bit-string** (BIT(n), BIT VARYING(n))
  - **Boolean** (TRUE, FALSE, NULL)
  - **Date** (YYYY-MM-DD)
    - **Time** (HH:MM:SS)
    - **Timestamp** (DATE & TIME)
    - **Interval** (YEAR/MONTH, DAY/TIME)
- **Domain** can be explicitly created and used for multiple attributes.
  - CREATE DOMAIN **ssn_type** AS VARCHAR(9);

DDL CREATE: CONSTRAINTS (1)

1. - **Attribute constraints**:
  - **NOT NULL**
    - On primary key attribute(s) (entity integrity) or any regular attribute.
  - **DEFAULT** *<value>*
    - Value used if the value for an attribute is not specified.
  - **CHECK**
    - Specify a certain condition.
    - CHECK **(salary > 0);**

3. - **Table constraints**:
  - **Key constraint**.
    - PRIMARY KEY **(ssn);**
  - **Unique constraint**.
    - UNIQUE **(dname);**
  - **Referential integrity constraint**.
    - FOREIGN KEY **(dno)** REFERENCES **DEPARTMENT (dnumber);**

2. - **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 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 **drop** named schema elements.
  - Tables, domains, constraints, or schema itself.
- **Drop behavior options**:
  - CASCADE.
  - RESTRICT.
- DROP SCHEMA **company** CASCADE;
  - This removes the schema and all its elements including tables, views, constraints, etc.

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, '653298653');
    ```

It wont insert if there is a violation.

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.
    - Missing WHERE deletes **all** tuples.

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

```
DELETE FROM     employee
WHERE           ssn = '123456789';
```

```
DELETE FROM     employee
WHERE           dno = 5;
```

```
DELETE FROM     employee;
```

$\longleftarrow$ deletes every Tuple in employee (doesn't delete employee table)

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. ← with ON UPDATE
    - 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;
```

} semantically, everyon in dept 5 is getting a raise.

DML SELECT (1)

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

  -
    ```
    SELECT    <attribute list>
    FROM      <table list>
    WHERE     <condition>;
    ```
    - <attribute list> - **attribute names** whose 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.

  - 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 e, employee s
    WHERE     e.super_ssn = s.ssn;
    ```
- **Aliasing of attributes** can be done in SELECT part of the query.

  -
    ```
    SELECT    fname AS fn, lname AS ln, bdate AS bd
    ```
- **Missing WHERE** selects all tuples (single relation) or does CROSS PRODUCT (multiple relations).
- **Asterisk *** used to select all the attributes (no projection / projection on all attributes).