

# Data Manipulation

CSC 256, SQL Programming

# Recall: SQL Overview

- DDL data definition language: create, alter, drop
- DML data manipulation language: insert, update, delete, select
- DCL data control language: grant, revoke
- TCL transaction control language: begin, commit, rollback

# SQL Data Manipulation Operations

- The data manipulation operations we will look at here are:
  - Inserting rows into tables
  - Updating existing rows in tables
  - Deleting rows from tables
- We have already discussed selecting information from tables in previous lectures

# INSERT

- INSERT inserts a row (or rows) into a table
- Basic Syntax:

```
INSERT INTO table_name  
VALUES (value1, value2, ...);
```

- Basic Syntax with explicit column names:

```
INSERT INTO table_name (column1, column2, ...)  
VALUES (value1, value2, ...);
```

- Note: in PostgreSQL, the VALUES clause can be substituted with a SELECT statement

# UPDATE

- UPDATE changes the value of a column for a particular row in a table
- Syntax:

```
UPDATE table_name  
SET column1=value1, column2=value2, ...  
WHERE boolean_condition;
```

# DELETE

- DELETE removes a row from a table
- Syntax:

```
DELETE FROM table_name  
WHERE boolean_condition;
```

- Note: on delete we need to consider that various foreign key constraints might prevent orphan data and invalid references

# Transaction Control Language

- A transaction bundles multiple SQL statements into a single unit.
- Transaction properties:
  - Atomicity: all operations within the work unit are either committed or rolled back
  - Consistency: the database properly changes states upon a successfully committed transaction
  - Isolation: transactions operate independently of each other
  - Durability: the result of a committed transaction persists in the event of a system failure

# Transaction Control Language

- SQL Transaction Control Language (TCL):
  - `BEGIN [TRANSACTION]`: start a transaction
  - `COMMIT`: save the changes
  - `ROLLBACK`: rollback the changes



# TCL Syntax Example

- Example:

```
BEGIN;
```

```
INSERT INTO <tablename>(<column 1>, <column 2>)  
VALUES  
    ('...', '...'),  
    ('...', '...'),  
    ('...', '...');
```

```
SELECT * FROM <tablename>;
```

```
ROLLBACK;
```

- Note: the SELECT will return the values, but they will not be saved due to the ROLLBACK.

# PostgreSQL RETURNING

- In PostgreSQL, an INSERT, UPDATE, or DELETE statement can have an optional RETURNING clause.
- The RETURNING clause can contain column names of the associated table, or value expressions using those columns; \* is a shorthand that returns all columns
- The data available to RETURNING depends on the statement:
  - INSERT: the content of the inserted row
  - UPDATE: the new content of the modified row
  - DELETE: the content of the deleted row

# RETURNING Example

- Variation of the previous TCL Example:

```
BEGIN;
```

```
INSERT INTO <tablename>(<column 1>, <column 2>)
```

```
VALUES
```

```
    ('...', '...'),
```

```
    ('...', '...'),
```

```
    ('...', '...')
```

```
RETURNING *;
```

```
ROLLBACK;
```

# PostgreSQL Importing and Exporting Data

- Data is commonly stored in text files or other formats, for example, CSV files
- PostgreSQL has the COPY statement to copy data between a file and a table
- Basic COPY syntax:

```
COPY <table>  
FROM|TO '<path to file>'  
[WITH (<options>)];
```

# Examples: Import and Export CSV Data

- Import CSV data to table:

```
COPY <table>  
FROM <file>  
WITH (FORMAT csv, HEADER TRUE);
```

- Export CSV data to table:

```
COPY <table>  
TO <file>  
WITH (FORMAT csv, HEADER TRUE);
```

- Note: the <table> here can be the result of a subquery.