



WEEK 04 INTRO TO SQL

Instructor: Yanan Wu
TA: Vanchy Li

Spring 2025



WEEK 03

LECTURE SESSION

Instructor: Yanan Wu

TA: Vanchy Li

Spring 2025



4.1

INTRO

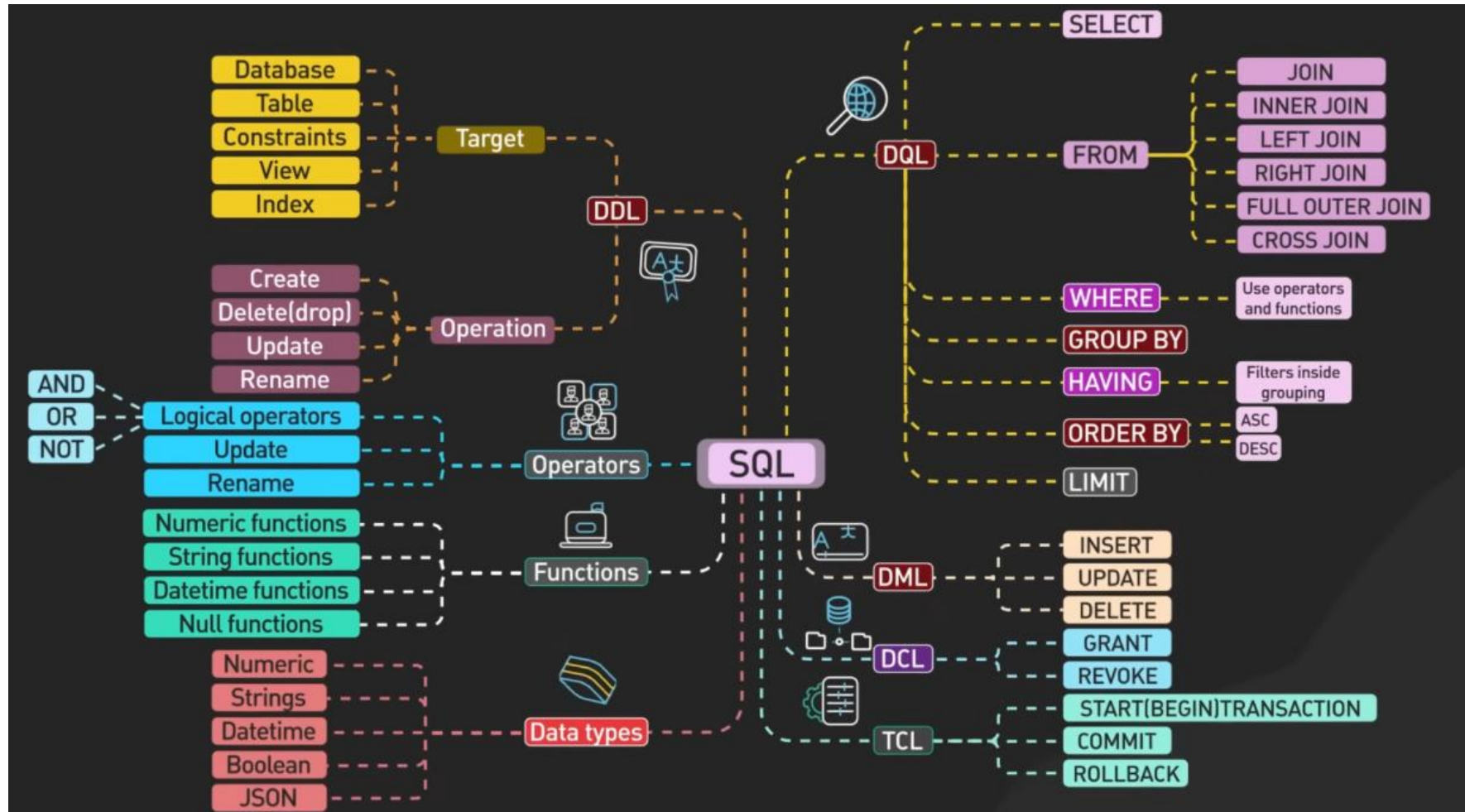


STRUCTURED QUERY LANGUAGE

- SQL (Structured Query Language):

SQL is a standard language for managing relational databases. It provides a set of commands for performing various operations such as querying data, updating data, creating and modifying database schema, and managing access controls.

SQL COMMANDS



SQL TABLE

- A SQL table is a fundamental component of a relational database, organizing data into rows and columns. Database engineers establish relationships between multiple tables to enhance data efficiency and optimize storage.
- For example, a database engineer might create a SQL table to store product details in a retail store:

Product ID Product Name Color ID			Color ID Color Name	
0001	Mattress	Color 1	Color 1	Blue
0002	Pillow	Color 2	Color 2	Red

4.2

SPATIAL DATA

SHAPEFILE

Shapefile: a collection of files with .shp, .shx, .dbf, and other extensions on a common prefix name (e.g., nyc_census_blocks). The actual shapefile relates specifically to files with the .shp extension. However, the .shp file alone is incomplete for distribution without the required supporting files.

Mandatory files:

.shp—shape format; the feature geometry itself

.shx—shape index format; a positional index of the feature geometry

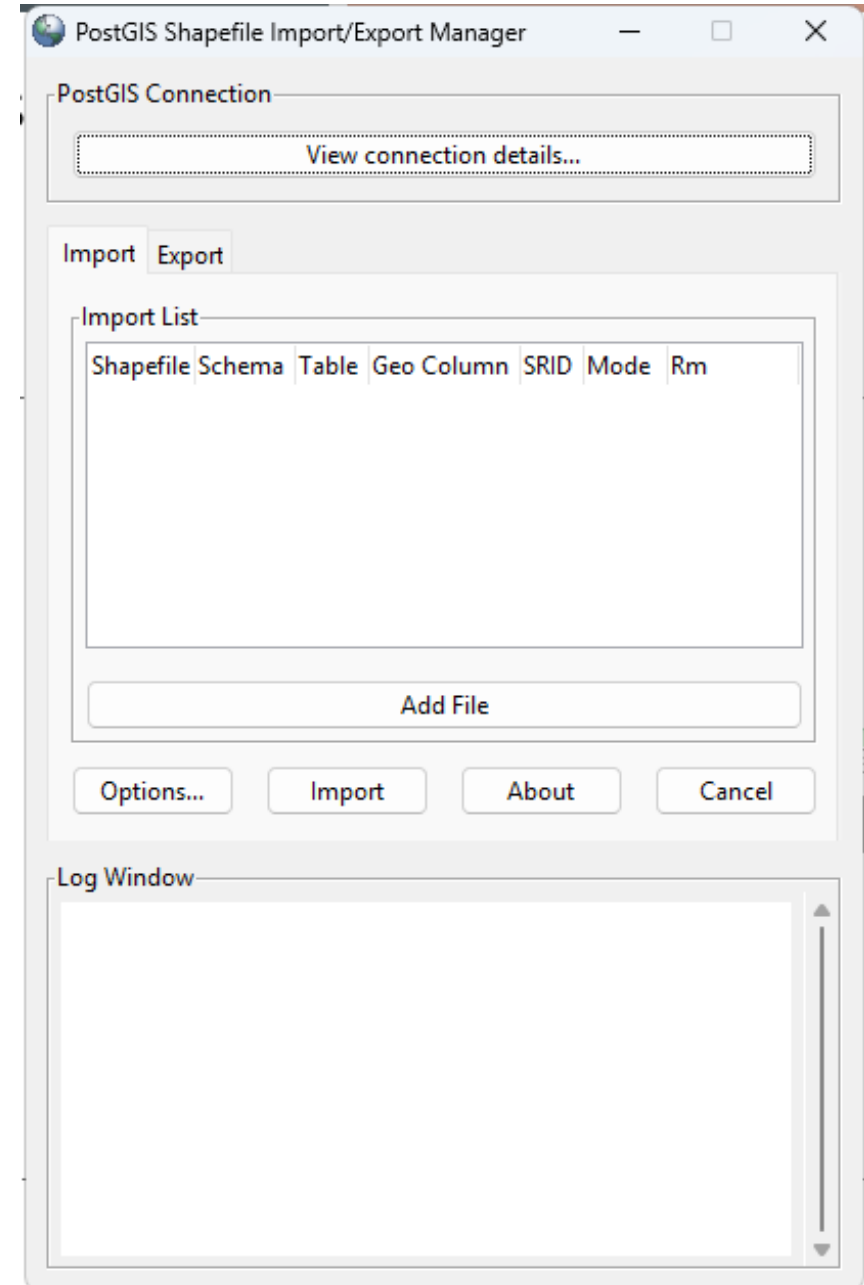
.dbf—attribute format; columnar attributes for each shape, in dBase III

Optional files include:

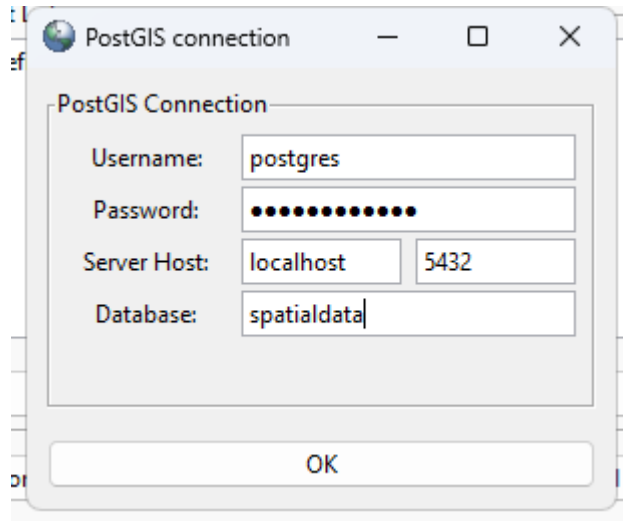
.prj—projection format; the coordinate system and projection information, a plain text file describing the projection using well-known text format

Loading with shp2pgsql -GUI

- 1) Open the Shapefile Import/Export Manager.
- 2) In the Shapefile field, browse and select US_tract_2020.shp.
- 3) Set the SRID to 102003 and Geo Column to geometry (or the SRID and Geo Column match your shapefile).
- 4) Choose a target schema (e.g., postgis) and set a table name (e.g., us_tracts).
- 5) Click Import. The tool will: Create a new table. Insert shapefile data into the table. Create a spatial index for faster querying.

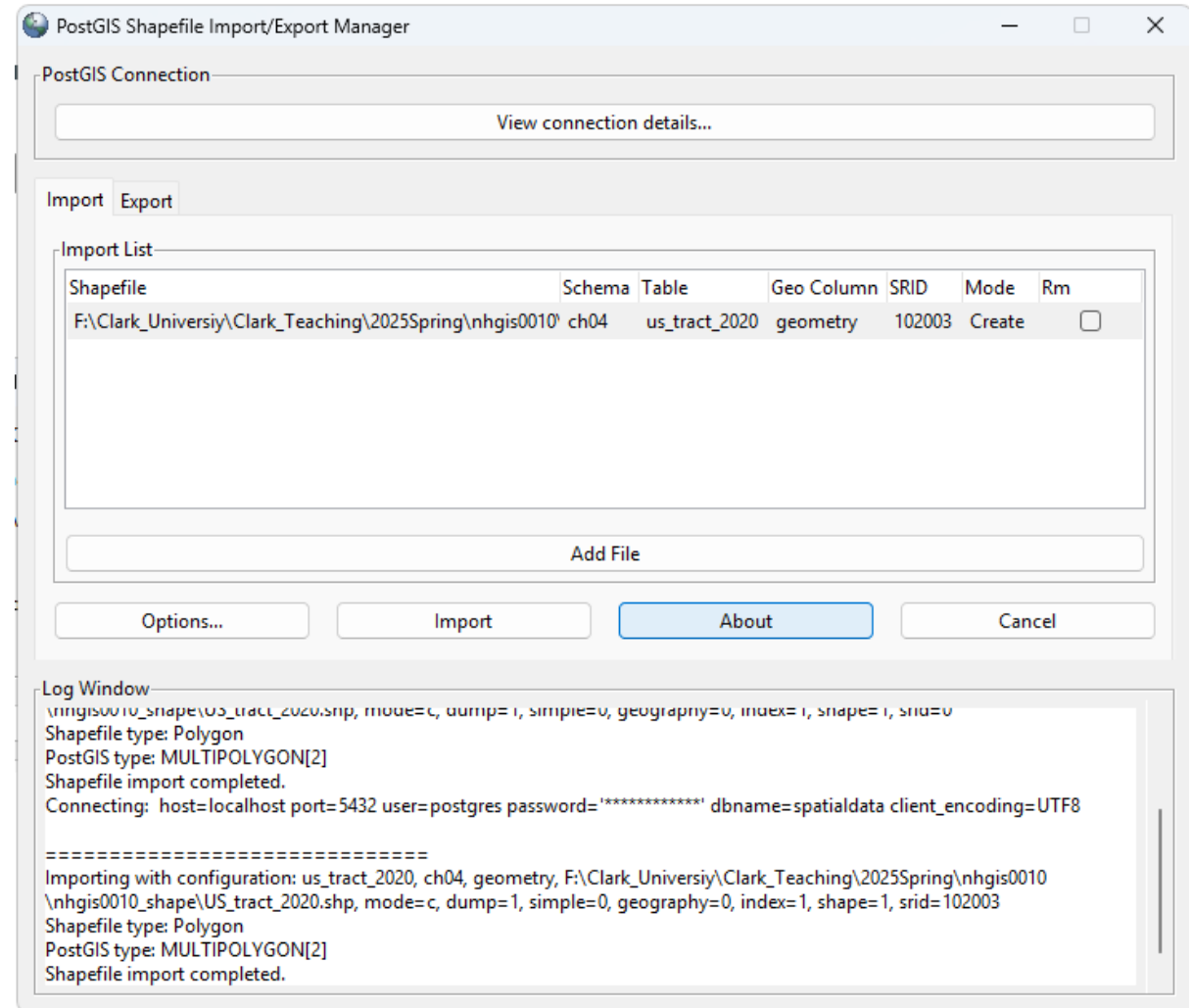


Loading with shp2pgsql -GUI



Geo Column: assign a name to the geometry column in data

SRID: spatial reference system



Loading with shp2pgsql -GUI

Step 3: Verify the Import

Check if the shapefile was imported successfully:

Open the query tool in pgAdmin.

Run the following query:

```
SELECT geometry FROM ch04.us_tract_2020 LIMIT 10;
```

```

CREATE TABLE ch04.us_attribute (
    GEO_ID VARCHAR(20) PRIMARY KEY, -- Census Geographic Area Identifier (Primary Key)
    GISJOIN VARCHAR(20), -- GIS Join Match Code
    STUSAB VARCHAR(20), -- State Abbreviation
    STATE VARCHAR(50), -- State Name
    STATEA VARCHAR(50), -- State Code
    COUNTY VARCHAR(50), -- County Name
    COUNTYA INTEGER, -- County Code
    TRACTA VARCHAR(20), -- Census Tract Code
    TL_GEO_ID VARCHAR(20), -- TIGER/Line Shapefile Geographic Area Identifier
    NAME_E VARCHAR(255), -- Full Geographic Area Name (Estimates)
    -- Population Data
    AQNFE001 INTEGER, -- Total Population
    AQNGE002 INTEGER, -- White Population
    AQNGE003 INTEGER, -- Black or African American Population
    AQNGE004 INTEGER, -- American Indian/Alaska Native Population
    AQNGE005 INTEGER, -- Asian Population
    AQNGE006 INTEGER, -- Native Hawaiian/Other Pacific Islander Population
    AQNGE007 INTEGER, -- Another Race Category
    -- Education Data
    AQPKE017 INTEGER, -- Some Education Field
    AQPKE021 INTEGER, -- Associate's Degree
    AQPKE022 INTEGER, -- Another Degree Field
    AQPKE023 INTEGER, -- Master's Degree
    AQPKE024 INTEGER, -- Professional School Degree
    AQPKE025 INTEGER -- Doctorate Degree);

```

Import csv file

- 1) Create a table with column names, the column names should align with the column names in csv file
- 2) Import Data to table (right click table in server, select Import/Export data)

4.2

DATA TYPES IN SQL



DATA TYPES

Numeric Types

INTEGER (or INT): Represents whole numbers.

NUMERIC (or DECIMAL): Represents fixed-point numbers with exact precision.

String Types

VARCHAR: Represents variable-length character strings with a maximum length specified.

TEXT: Represents variable-length character strings with a maximum length that can be extremely large



DATA TYPES

Date/Time Types

DATE: Represents a date value without time.

TIMESTAMP: Represents date and time values.

Boolean Type

BOOLEAN: Represents a boolean value (TRUE, FALSE, or NULL)

4.2

DATA DEFINITION LANGUAGES (DDL)

DATA DEFINITION LANGUAGE (DDL)

- DDL is used to define the structure and schema of a database. It includes commands for creating, modifying, and deleting database objects such as tables, indexes, views, and schemas.
- It does **not** manipulate data but manages the structure.

Data Definition Language (DDL)

```
CREATE TABLE Students(  
  id int,  
  name varchar(50),  
  address text,  
  grade varchar(50),  
  phone varchar(10)  
);
```

```
ALTER TABLE Customers  
ADD phone varchar(10);
```



KEY DDL COMMANDS

CREATE – To create databases, tables, or indexes.

ALTER – To modify existing structures.

DROP – To delete objects.

TRUNCATE – To remove all data without deleting the structure.

COMMENT – To add metadata.

RENAME – To rename tables or columns.



CREATE STATEMENT

TOPICS ON WEEK 03

CREATE GEOMETRY

CREATE DATABASE ...
CREATE TABLE...

ALTER STATEMENT

- Used to modify an existing table (add, modify, drop columns).

```
ALTER TABLE ch04. us_attribute ADD COLUMN POP VARCHAR(255);
```

```
ALTER TABLE ch04. us_attribute ALTER COLUMN POP TYPE SMALLINT;
```

```
ALTER TABLE ch04. us_attribute RENAME COLUMN POP TO POPULATION;
```

```
ALTER TABLE ch04. us_attribute DROP COLUMN POPULATION;
```

EXERCISE: RENAME TABLE NAME

```
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNFE001 TO total_pop;  
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNGE002 TO white_pop;  
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNGE003 TO black_pop;  
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNGE004 TO native_pop;  
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNGE005 TO asian_pop;  
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNGE006 TO pacific_pop;  
ALTER TABLE ch04.us_attribute RENAME COLUMN AQNGE007 TO other_race;
```

Sample But in MySQL

```
ALTER TABLE table_name  
    RENAME COLUMN AQNFE001 TO total_pop,  
    RENAME COLUMN AQNGE002 TO white_pop,  
    RENAME COLUMN AQNGE003 TO black_pop,  
    RENAME COLUMN AQNGE004 TO native_pop,  
    RENAME COLUMN AQNGE005 TO asian_pop,  
    RENAME COLUMN AQNGE006 TO pacific_pop,  
    RENAME COLUMN AQNGE007 TO other_race;
```

PostgreSQL does not support renaming multiple columns at once, so if you are using PostgreSQL, you must keep one ALTER TABLE statement per line.

However, MySQL 8.0+ allows multiple RENAME COLUMN statements to be combined.



Check Column name and Data Type

```
SELECT column_name, data_type  
FROM information_schema.columns  
WHERE table_schema = 'ch04' AND table_name = 'us_attribute';
```

4.2

DATA QUERY LANGUAGE

DQL

- DQL (Data Query Language) is used to retrieve records from the database.
- SELECT statement: Used to retrieve data from tables.
- Basic Queries, Filtering, and Aggregation.

THE SELECT STATEMENT

- Basic syntax

```
SELECT column1, column2 FROM table_name;
```

- Selecting all columns

```
SELECT * FROM ch04.us_attribute;
```

- Display unique value for column

```
SELECT DISTINCT * FROM ch04.us_attribute;
```

FILTERING DATA WITH WHERE

- Filtering by a single condition:

```
SELECT stusab, statea, total_pop FROM ch04.us_attribute WHERE stusab = 'New Jersey';
```

- Using comparison operators (=, !=, >, <, >=, <=):

```
SELECT stusab, statea, total_pop FROM ch04.us_attribute WHERE total_pop > 10000;
```

- Using logical operators (AND, OR, NOT):

```
SELECT stusab, statea, total_pop FROM ch04.us_attribute WHERE stusab = 'California' AND total_pop > 10000;
```

ORDER DATA WITH ORDER BY

Display the top 10 most populated counties.

```
SELECT stusab, statea, total_pop  
FROM ch04.us_attribute  
ORDER BY total_pop DESC  
LIMIT 10;
```

GROUP DATA BY GROUP BY

Find the number of counties in each state.

```
SELECT stusab, COUNT(COUNTY) AS county_count  
FROM ch04.us_attribute  
GROUP BY stusab  
ORDER BY county_count DESC;
```



For each census tract, what percentage of the population is white?

```
SELECT
    stusab, statea,
    100.0 * Sum(white_pop)/Sum(total_pop) AS white_pct
FROM ch04.us_attribute
GROUP BY statea, stusab
ORDER BY white_pct DESC;
```

CALCULATE THE TOTAL POPULATION BY STATE (SUM)

Find the total population for each state.

```
SELECT stusab, SUM(total_pop ) AS total_population  
FROM ch04.us_attribute  
GROUP BY stusab  
ORDER BY total_population DESC;
```

FIND THE AVERAGE POPULATION OF COUNTIES (AVG)

Calculate the average population of all counties in each state.

```
SELECT stusab, AVG(total_pop) AS avg_population  
FROM ch04.us_attribute  
GROUP BY stusab  
ORDER BY avg_population DESC;
```

PATTERN MATCHING USING LIKE

```
SELECT stusab, statea  
FROM ch04.us_attribute  
WHERE stusab LIKE 'Mass%';
```

PATTERN MATCHING USING IN AND BETWEEN

```
SELECT STATE, COUNTY, AQNFE001  
FROM ch04.us_attribute  
WHERE STATE IN ('Texas', 'California', 'Florida');
```