Spatial Database Management
GEP 664 / GEP 380
Class #3: Database Fundamentals and SQL DDL

Frank Donnelly

Dept of EEGS, Lehman College CUNY

Spring 2019

Today's Topics

Updating Database Records

Database Structure, Tables and Constraints

Views, Importing Data

Next Class

Subsets of the SQL Language

- Data Manipulation Language
  - SELECT... FROM... WHERE
  - INSERT... INTO... VALUES
  - DELETE FROM... WHERE
  - UPDATE... SET... WHERE
- Data Definition Language
  - CREATE...
  - DROP...
  - ALTER...
  - RENAME...
- Data Control Language
  - GRANT...
  - REVOKE...

Updating Database Records

Add new rows

```
INSERT INTO nyc.weather_stations
VALUES ('WBAN:04781', 'ISLIP AIRPORT NY US', 25.6,
   40.7939, −73.1017);
```

Update existing rows

```
UPDATE nyc.weather_stations
SET station_name='ISLIP LI MACARTHUR AIRPORT NY US'
WHERE station_id = 'WBAN:04781';
```

Delete rows

```
DELETE FROM nyc.weather_stations
WHERE station_id= 'WBAN:04781';
```

Today's Topics

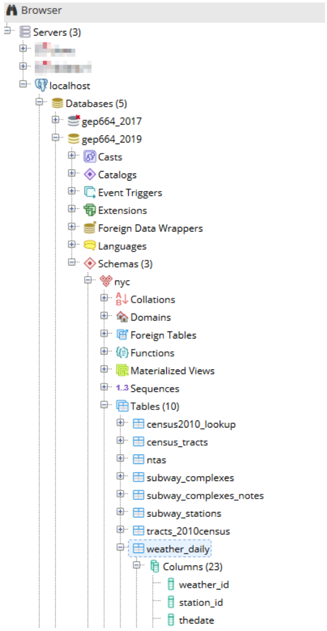
Updating Database Records

Database Structure, Tables and Constraints

Views, Importing Data

Next Class

Database Objects



Database
- Schema
  - Table (or object)
    - Attribute / Column

Schemas

The level between a database and individual objects, a schema is a grouping of closely related objects.

- Helpful for managing user access
- Group tables thematically or by project
- Default schema is called 'public',
- PostgreSQL assumes you're using public unless you say otherwise
- By default all users have access to read and write to public

```
CREATE SCHEMA schema_name
AUTHORIZATION postgres;
```

Creating Tables

```
 CREATE TABLE nyc.weather_station (
 station_id varchar(20) PRIMARY KEY,
 station_name text  NOT NULL,
 elevation numeric(6,1),
 lat numeric(9,6),
 lon numeric(9,6)
);
```

Identifiers - Naming Things

For naming all objects, you can use:

- upper-case letters A to Z
- lower-case letters a to z
- digits 0 to 9
- underscore _ character

Restrictions:

- no longer than 128 characters (shorter is better)
- must start with a letter (not numbers)
- must not contain spaces (use underscore)
- must not use reserved keywords

Consider specificity vs. length, singular vs. plural, mixing cases, and context. Stay consistent.

Data Types

Most basic restriction assigned to columns to insure data integrity and to define permissible functions.

- numbers
- text
- dates
- others

Numeric Types

Whole numbers

integer most common, range -2147483648 to +2147483647

smallint for limited cases, range -32768 to +32767

bigint for limited cases, insanely large

serial same as integer, automatically generated

Decimals

numeric (p,s) specify precision (number of digits) and scale (number of decimal places)

real use for big numbers (6 decimal places)

double for limited cases, insanely large

Numeric Types

For decimals it's usually best to use numeric and provide specs, or use real if 6 decimals is enough. For whole numbers it's best to use integer

Scale and precision examples:

Numeric(5,0) stores number up to 99999

Numeric(5,1) stores number up to 9999.9

Numeric(5,2) stores number up to 999.99

Character Types

For storing text, aka strings

char	store a single character
char(n)	avoid - stores characters of n length padded with spaces
varchar(n)	stores variable number of characters up to n length
text	store unlimited characters, good when length is large or unknown

Numbers can be stored as text, common for ID numbers that don't represent quantities. Text cannot be stored as numbers.

Date Types

Can hold dates and times in many formats

date	just the date
time	just the time
timestamp	full date and time
timestampz	full date and time with timezone

Suggested format is ISO 8601 standard, Y-M-D-H-M-S: TIMESTAMP '2016-09-08 10:30:00'

Other Types

boolean	true/false, yes/no, on/off, 1/0
bitea	for storing binary strings, sequences of octets or bytes
xml and json	particular to these data formats
enum	enumerated - make up your own types
money	avoid - use numeric instead
geometric	avoid - use PostGIS instead

Functions

Are particular to specific types of data

- Numbers: mathematical functions, arithmetic, algebraic, trigonometric, rounding
- Text: searching and pattern matching, string modification and substitution
- Comparison operations can be performed on all types
- Formatting functions can temporarily cast from one type to another

CAST

Use for converting data types on the fly to achieve a desired result, such as a decimal after dividing two integers (below)

```
SELECT weather_id, station_id,
CAST(windspeed_mph AS numeric) / CAST(windgust_mph AS
   numeric) * 100 AS pct_gust
FROM nyc.weather_daily
WHERE year = 2017 AND windgust_mph IS NOT NULL;
```

In PostgreSQL :: is a shortcut for CAST:

```
(windspeed_mph::numeric / windgust_mph::numeric) * 100 AS
     pct_gust
```

Basic Constraints

In the CREATE statement, add constraints to specific columns

- NOT NULL
- UNIQUE
- PRIMARY KEY
- DEFAULT (default value)
- CHECK (condition)
- REFERENCES

Column and Table Constraints

```
CREATE TABLE nyc.weather_stations (
station_id varchar(20) PRIMARY KEY,
station_name text  NOT NULL,
elevation numeric(6,1),
lat numeric(9,6) CHECK(lat>0),
lon numeric(9,6) CHECK(lon<0) );
```

```
CREATE TABLE nyc.weather_stations (
station_id varchar(20),
station_name text  NOT NULL,
elevation numeric(6,1),
lat numeric(9,6),
lon numeric(9,6),
CONSTRAINT pksid PRIMARY KEY (station_id),
CONSTRAINT lat_pos CHECK (lat>0),
CONSTRAINT lon_neg CHECK (lon<0) );
```

Entity Integrity

Insured with Primary Keys

- Unique, not null id column for each row
- Can also be a composite key (combo of several columns)
- Insures no duplication of records
- Automatically indexed
- Natural key is derived outside the database, has meaning
- Surrogate key is artificial, made in the database
- Use SERIAL type to create auto-sequential integer key

Referential Integrity

Insured with Foreign Keys

- Links row in child table to a parent table
- If foreign key contains a value, it must refer to existing value in the parent table
- Insures there are no mismatches between tables
- As column constraint: use REFERENCES followed by foreign table and column names
- As table constraint: name the constraint and explicitly state FOREIGN KEY followed by REFERENCES clause

Keys as Column or Table Constraints

```
CREATE TABLE nyc.weather_daily (
weather_id integer PRIMARY KEY,
station_id varchar(20) REFERENCES nyc.weather_stations (
 station_id),
...);
```

```
CREATE TABLE nyc.weather_daily (
weather_id integer,
station_id varchar(20), ...

CONSTRAINT pkwid PRIMARY KEY (weather_id),
CONSTRAINT fksid FOREIGN KEY (station_id)
REFERENCES nyc.weather_stations (station_id));
```

Modifying and Deleting

Add or remove any database objects. For tables:

- ALTER TABLE table name
  - ADD COLUMN name datatype...
  - RENAME COLUMN name TO newname
  - DROP COLUMN name datatype...
  - Can do the same with checks and constraints
- DROP TABLE table name

Be careful when dropping!

- RESTRICT stops DROP operations if there are table dependence issues
- CASCADE proceeds with DROP and deletes all table dependencies too

ALTER with UPDATE - SET

These commands are often used to create a new column and populate it with data from another column based on some condition. Executed in two separate statements:

```
ALTER TABLE nyc.weather_stations
ADD COLUMN airport varchar(3);
```

```
UPDATE nyc.weather_stations
SET airport='yes'
WHERE station_name LIKE '%AIRPORT%';
```

(Alternatively, for this particular example you could assign the airport column a boolean type and SET values = True)

Comment

COMMENT is a command unique to PostgreSQL for adding brief descriptions for objects.

```
COMMENT ON TABLE nyc.weather_station
 IS 'Selection of NOAA weather stations in the NYC metro area';
```

```
SELECT description
FROM   pg_description
WHERE  objoid = 'nyc.weather_station'::regclass;
```

Alternative: use pgadmin to view and edit comments (under Properties tab)

Today's Topics

Updating Database Records

Database Structure, Tables and Constraints

Views, Importing Data

Next Class

Views

Objects that are virtual tables - good way to save a complex query. Saves the statement - not the data. Views are tied to the underlying table(s).

```
CREATE VIEW nyc.summary2017 AS
SELECT year, month,
COUNT(weather_id) AS records,
MIN(drybulb_temp_f) AS mintemp,
MAX(drybulb_temp_f) AS maxtemp,
AVG(drybulb_temp_f) AS avgtemp
FROM nyc.weather_daily
WHERE year=2017
GROUP BY year, month
ORDER BY month;
```

Insert Individual Records

Method 1

1. Prepare target table (create empty table, or use existing well-structured table)
2. Insert records via the statement

```
INSERT INTO nyc.weather_stations
VALUES ('WBAN:04781', 'ISLIP LI MACARTHUR AIRPORT
   NY US', 25.6, 40.7939, −73.1017),
('WBAN:54780', 'MONTAUK AIRPORT, NY US', 2.1,
   41.07306, −71.92333);
```

Caveat: values must be listed in order of table columns, otherwise you must list them before VALUES in ( )

Insert Table Records

Method 2

1. Create the empty target table
2. For internal data: import data from existing table
3. For external data: import into a temporary table, then import to target

Caveat: input and target columns must align

```
INSERT INTO nyc.weather_stations (station_id, station_name,
   elevation, lat, lon)
SELECT sid, sname, elev, latitude, longitude
FROM nyc.temp_stations
WHERE state = 'NY';
```

Import data with COPY

COPY data into temporary staging table then INSERT to target, or COPY data directly to well structured target.
Example 1: Windows, comma-delimited with no header row

```
COPY nyc.weather_staging
FROM 'C:\user\weatherdata\newobservs.csv' WITH
   DELIMITER AS ','
```

Example 2: Mac / Linux, tab-delimited with header row

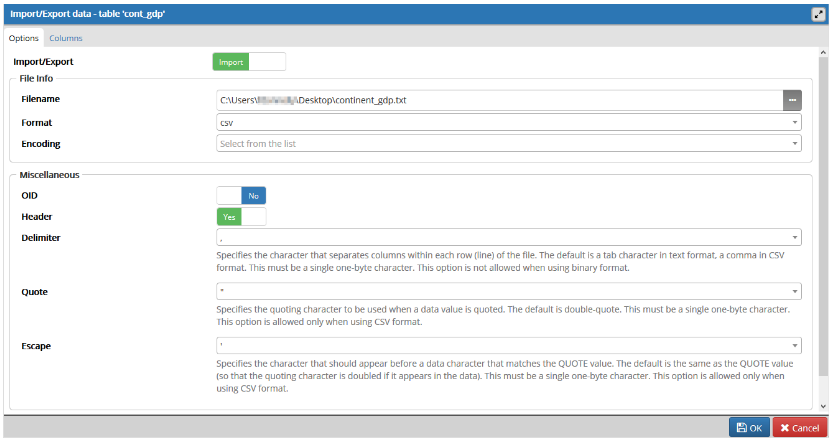
```
COPY nyc.weather_staging
FROM 'user/weatherdata/newobservs.txt' WITH DELIMITER
   AS '\t' CSV HEADER
```

Make sure to move data files to directory BEFORE launching pgadmin; it won't detect files moved there after launch.

https://www.postgresql.org/docs/10/sql-copy.html

Import Data with pgAdmin

Create empty table with structure. Right click on table, choose Import/Export. Must specify: Import, Filename, Format, Header, Delimiter, Quote.



Import/Export data - table 'cont_gdp'

Import/Export: Import

File Info

Filename: C:\Users\[redacted]\Desktop\continent_gdp.txt

Format: csv

Encoding: Select from the list

Miscellaneous

OID: No

Header: Yes

Delimiter: ,

Specifies the character that separates columns within each row (line) of the file. The default is a tab character in text format, a comma in CSV format. This must be a single one-byte character. This option is not allowed when using binary format.

Quote: "

Specifies the quoting character to be used when a data value is quoted. The default is double-quote. This must be a single one-byte character. This option is allowed only when using CSV format.

Escape: '

Specifies the character that should appear before a data character that matches the QUOTE value. The default is the same as the QUOTE value (so that the quoting character is doubled if it appears in the data). This must be a single one-byte character. This option is allowed only when using CSV format.

Create Table As

Method 3

1. Create data from existing table

Caveats: cannot add constraints, cannot assign data types for new fields

```
CREATE TABLE nyc.weather_stations_ny AS
SELECT station_id, station_name, elevation, lat, lon
FROM nyc.weather_stations
WHERE station_name LIKE '%NY%';
```

Create View

Method 4

1. Create view from existing table

If it isn't necessary to save data permanently in a new table:

```
CREATE VIEW nyc.weather_stations_ny AS
SELECT station_id, station_name, elevation, lat, lon
FROM nyc.weather_stations
WHERE station_name LIKE '%NY%';
```

Today's Topics

Updating Database Records

Database Structure, Tables and Constraints

Views, Importing Data

Next Class

Due Next Class

The following are due at the beginning of our next class:

Assignment #3

Posted on the course website

Readings for Class #4

Listed in the syllabus, posted on the library's E-Reserve page