

Spatial Database Management

GEP 664 / EES 79903

Class #3: Database Fundamentals and SQL DDL

Frank Donnelly

Dept of EEGS, Lehman College CUNY

Spring 2017

In Review

Database Structure, Tables and Constraints

Other Objects, Importing Data

Next Class



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';
```

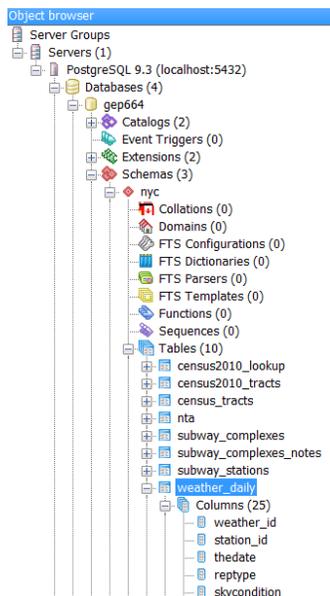
Delete rows

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



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...



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

In Review

Database Structure, Tables and Constraints

Other Objects, Importing Data

Next Class

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;
```

```
CREATE TABLE nyc.weather_station (  
  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)  
);
```

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

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

- ▶ numbers
- ▶ text
- ▶ dates
- ▶ others

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

double for limited cases, insanely large

It's usually best to use numeric and provide specs, rather than using real or double. 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

`bita` 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



In the CREATE statement, add constraints to specific columns

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

```
CREATE TABLE nyc.weather_station (
  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_station (
  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) );
```

- ▶ Generally refers to an attributes allowable set of values
- ▶ Specifically, another way of stating constraints that can be applied to multiple objects

```
CREATE DOMAIN nyc.wcolor AS VARCHAR(20)
DEFAULT 'BLUE'
CHECK (VALUE IN ('BLUE', 'RED', 'YELLOW', 'GREEN'));
```

```
CREATE TABLE nyc.weather_alert (
  aid serial PRIMARY KEY,
  aname nyc.wcolor NOT NULL,
  conditions text
);
```

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

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

```
CREATE TABLE nyc.weather_daily...
```

```
weather_id PRIMARY KEY
station_id REFERENCES nyc.weather_station (station_id);
```

```
CREATE TABLE nyc.weather_daily...
```

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

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

In Review

Database Structure, Tables and Constraints

Other Objects, Importing Data

Next Class

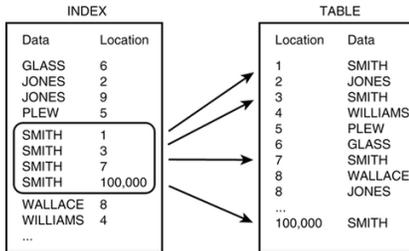


Image source:
https://en.wikipedia.org/wiki/Database_index

- ▶ Speeds up access based on a specific column
- ▶ Keys are indexed by default
- ▶ Values can be unique or not (unique is best)
- ▶ Bad for small tables or small set of values

```
CREATE INDEX indexname
ON tablename (column);
```

Objects that are virtual tables - good way to save a complex query. Saves the statement - not the data.

```
CREATE VIEW nyc.summary2015 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=2015
GROUP BY year, month
ORDER BY month;
```

Insert Individual Records

Method 1

1. Create the empty target table
2. Insert records via the statement

Caveat: values must be listed in order of columns

```
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);
```

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

Example 1: Path on MS Windows, comma-delimited with no header row

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

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

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

Make sure to move data files to directory BEFORE launching psql; it won't detect files moved there after it is launched.

<https://www.postgresql.org/docs/9.4/static/sql-copy.html>

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%';
```

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;
```

Today's Topics

In Review

Database Structure, Tables and Constraints

Other Objects, 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