Linux Users Group
at NC State University

Introduction to PostgreSQL

Bennett Petzold

LUG @ NC State

January 28, 2022

NC STATE UNIVERSITY

Outline

NC STATE UNIVERSITY

- Database Choices
  - Why Use a Database?
  - Database Dangers
  - Why use PostgreSQL?

NC STATE UNIVERSITY

Why Use a Database? (Database Choices)

- Simultaneous access
  - Processes can write and read without communicating
  - Multiple users can safely act simultaneously
- Speed
  - Indexes
  - Sharding
- Management
  - Modular control of permissions
  - Table formatting rules
  - Script hooks
  - Default entries

NC STATE UNIVERSITY

Database Dangers (Database Choices)

- Code injection
- Centrality
- Incompetence

NC STATE UNIVERSITY

Why use PostgreSQL? (Database Choices)

- Free
  - FOSS, even
- SQL
  - Any standard SQL syntax works on any SQL database
  - Relational tables and queries
  - ACID (Atomicity, Consistency, Isolation, Durability)
- Easy to install
- Nice syntax and defaults
- Easy and effective recovery

NC STATE UNIVERSITY

Installing

- Get and Start
- Minimal Database
- Authentication

Get and Start (Installing)

- Get the postgres package
  - This typically adds the server, psql, and the postgres user
- Switch to the postgres user
  - sudo -iu postgres
  - su -l postgres
- (Some distributions) initialize the cluster as postgres
  - ArchWiki recommends “initdb -D /var/lib/postgres/data”

NC STATE UNIVERSITY

Minimal Database (Installing)

- Add a user: “createuser”
  - “createuser –interactive” is the most useful form.
  - Avoid specifying logins by matching database username with OS username
  - Giving your default shell database superuser allows you to manage outside of the postgresql user
- Make a database: “createdb [name]”
  - Database name is your username, if not specified
- Drop a database: “dropdb [name]”

Authentication (Installing)

- Managed in pg_hba.conf
  - Located in the postgres directory. Check your distribution's package.
  - Postgres' directory is usually the postgres user's home.
- All local users are trust (can log in as any user) by default
  - Replace "local all all trust" with "local all all peer" to allow local connections to matching usernames
- Remote from any machine with password configuration
  - Set "listen_addresses = '*' " in postgresql.conf
  - Set "password_encryption = scram-sha-256" in postgresql.conf
  - Add "host all all 0.0.0.0/0 scram-sha-256" in pg_hba.conf
  - Add "host all all :/0 scram-sha-256" in pg_hba.conf
  - Open port 5432
  - Restart postgresql.service

NC STATE UNIVERSITY

Syntax

- Create Table (1)
- Create Table (2)
- Add Data to Table
- Select and View
- Manipulate Tables
- Joins
- Outer Joins
- Bonus Round

NC STATE UNIVERSITY

Create Table (1) (Syntax)

- Syntax capitalization is optional, but is the convention
- CREATE works for anything
- Functions work like any other language

```
CREATE TABLE name (column type [modifier], ...);
CREATE TEMPORARY TABLE name (column type [modifier], ...);
CREATE TABLE name as {query};

CREATE TABLE swipes (id INT NOT NULL, time TIMESTAMPTZ NOT NULL DEFAULT NOW());
CREATE TABLE employees
(id SERIAL, name TEXT NOT NULL, title TEXT, annual_salary_dollars INT);
```

Create Table (2) (Syntax)

- Types are all the basics you know (int, char, etc.) and some bonuses
  - Text is the string equivalent
  - timestamp is without timezone, timestamptz is with timezone
  - Range types and arrays
  - User defined types, functions, and casts
- Modifiers like NOT NULL and DEFAULT give the database rules to enforce
- Temporary gets automatically dropped when connection ends
- Example: create a table to track swipes and a table to track employees, with a shared ID field
- Serial is equivalent to "INTEGER NOT NULL DEFAULT NEXTVAL('employees_id_seq')"

NC STATE UNIVERSITY

Add Data to Table (Syntax)

- INSERT INTO writes in a single row at a time.
- COPY writes in as many rows as specified (default = all) from an input file.
- Both can address some or all columns.
  - Must respect NOT NULL columns to succeed

```
INSERT INTO name (column, ...) values (value, ...);
COPY name FROM filename;

INSERT INTO employees (name, title, annual_salary_dollars)
values ("Bennett", "Gofer", 20);
INSERT INTO swipes (id) values (1);
INSERT INTO swipes (id) values (1);
```

NC STATE UNIVERSITY

Select and View (Syntax)

- SELECT outputs fields from a table or query
- VIEWs are basically saved queries
  - Always returns results from current data
  - Creating another table is like a snapshot, instead
  - Temporary views are useful for reducing query complexity
- "SELECT * from table" selects all fields.
  - Only works if there are no name conflicts
- "table.column" can be just "column" if the name doesn't conflict

```
SELECT column, ... FROM table;
SELECT column, ... FROM table WHERE {condition};
CREATE VIEW name AS {query};

SELECT employees.annual_salary_dollars from employees;
CREATE VIEW salaries AS SELECT employees.annual_salary_dollars FROM employees;
CREATE TEMPORARY VIEW best_employee AS SELECT employees.name FROM employees
WHERE employees.name = 'Bennett';
```

NC STATE UNIVERSITY

Manipulate Tables (Syntax)

- ALTER
- RENAME TO
- DROP
  - Get rid of a table/other object
- DELETE
  - Remove rows matching condition

```
ALTER table;
table RENAME TO name;
DROP TABLE table;
DELETE FROM table WHERE {condition};

CREATE TABLE useless (i INT);
INSERT INTO useless VALUES (1);
DELETE FROM uselss WHERE i = 1;
DROP useless;
```

Joins (Syntax)

- Combine two tables. The essential SQL feature.
- CROSS JOIN: combine every row of table 1 with every row of table 2
- INNER JOIN: combine every row meeting condition.
  - "INNER" is optional

```
table1 JOIN table2 ON {condition};
table1 CROSS JOIN table2;

SELECT name, time FROM employees JOIN swipes ON employees.id = swipes.id;
```

Outer Joins (Syntax)

- LEFT OUTER JOIN: all rows on left table, combined with right table where meeting condition.
- RIGHT OUTER JOIN: Same as LEFT OUTER JOIN, with the sides flipped
- FULL OUTER JOIN: all rows, combined when meeting condition.

```
table1 FULL OUTER JOIN table2 ON {condition};

SELECT name, time FROM employees FULL OUTER JOIN swipes
ON employees.id = swipes.id;
```

NC STATE UNIVERSITY

Bonus Round (Syntax)

- AS
  - "SELECT t.column from table as t;"
    - Creates a second name to refer to during the query
  - "SELECT table.column AS c FROM table;"
    - Displays "column" as "c", instead.
- ORDER BY
  - "SELECT * FROM table ORDER BY table.column ASC;"
- GROUP BY
  - "SELECT table.column FROM table GROUP BY table.column;"
    - Groups matching terms together
    - Needed for aggregate functions (sum, count, min, etc)
- DISTINCT
  - "SELECT DISTINCT table.column FROM table"
    - Selects only one of each repeated value
- Subquery
  - "({query}) AS name"

- Privileges
  - Manage
  - List

Manage (Privileges)

- Privileges are controlled by the owner of an object
- Superusers can control all privileges
- "USER" is an alias for "ROLE"
- Group roles simplify management on multiple users/groups
  - "PUBLIC" is a group role of every user

```
GRANT {privilege} ON {object name} TO {PUBLIC, group, or user};
REVOKE {privilege} ON {object name} FROM {PUBLIC, group, or user};
ALTER {object type} {object name} OWNER TO {user};
DROP USER {user};
GRANT group_role to {user};
REVOKE group_role from {user};
```

List (Privileges)

- SELECT
- INSERT
- UPDATE
- DELETE
- TRUNCATE
- REFERENCES
- TRIGGER

- CREATE
- CONNECT
- TEMPORARY
- EXECUTE
  - Call functions
- USAGE
  - Varies according to object

- psql
  - Why?
  - Usage
  - Some Meta Commands

Why? (psql)

- Command line!
  - Tab completion, etc can be configured
  - Can take data in STDIN and write to STDOUT
  - Embedding in scripts
  - Write stuff with your editor
- Visually clear (compared to clutter like pgAdmin)
- Developed alongside PostgreSQL
- Feels cool

NC STATE UNIVERSITY

Usage (psql)

- Can put passwords in “ /.pgpass”
- Tuples-only doesn’t print headers
- Pipe is read by copying from SDTIN
- Copying to STDOUT is easier than playing with tuples
- Single transaction: all commands do nothing if one fails

```
psql
psql {database name}
psql -q // Quiet
psql -t // Tuples-only
psql -c '{command}'
psql -f {command file}
psql -1c // Single transaction
psql -d {database name} -h {hostname} -p {port} -U {username}
process | psql -1tc '{command}' > file
```

Some Meta Commands (psql)

- \help, \?
- \d
  - Information about tables when blank, subjects when specified
- \dt
  - Summarize tables
- \q or CTRL-d
  - Exit
- \d+
  - \d with more information
- \copy
  - Used identically to COPY, but works with the local machine instead of the server
  - Can use local files
- \!
  - Shell command (clear, etc)

NC STATE UNIVERSITY

References

[1] PostgreSQL documentation
https://www.postgresql.org/docs/

[2] Practical SQL
https://nostarch.com/practical-sql-2nd-edition

[3] ArchWiki: PostgreSQL
https://wiki.archlinux.org/title/PostgreSQL