Spatial Database Management
GEP 664 / GEP 380
Class #11: Database Management

Frank Donnelly

Dept of EEGS, Lehman College CUNY

Spring 2019

Today's Topics

Transactions

- Represents one logical unit of work
- Allows you to collect a series of SQL statements into one single unit
- Allows you to "preview" changes and undo them
- Prevents database corruption when being updated by multiple users
- While running, write access to the database or to specific db elements in the transaction is locked for other users, until the transactions is complete
- Database uses internal transaction log files to track changes and to reset if needed

ACID Model

Four properties a transaction must have:

Atomicity	: must happen exactly once, as a single unit
Consistency	: must keep the database in a clean state
Isolation	: must be independent of all other transactions
Durability	: once completed, it must stay completed

Transaction Components

BEGIN	starts a transaction
SAVEPOINT	asks the server to remember the current state of the transaction
COMMIT	completes the transaction, updates and unlocks the database
ROLLBACK	abandons the transaction and restores the database to it's previous state. Or use ROLLBACK TO SAVEPOINT to roll back to that specific point

If the statement contains or causes an error, the transaction enters an ABORT phase. No other commands are accepted except an explicit COMMIT or ROLLBACK.

Transaction Example

```
 BEGIN;

CREATE TABLE airport_geog (
iata varchar(3) PRIMARY KEY,
state varchar(2),
lon numeric(9,6),
lat numeric(9,6),
geog geography(point,4326));

INSERT INTO airport_geog
VALUES ('JFK','NY',−73.7789,40.6397),
('IAD','VA',−77.4558, 38.9444),
('LAX','CA',−118.4081,33.9425),
('ANC','AK',−149.9983,61.1742),
('HNL','HI',−157.9225,21.3186);

UPDATE airport_geog
SET geog=ST_SetSRID(ST_Point(lon,lat),4326);
COMMIT;
```

Today's Topics

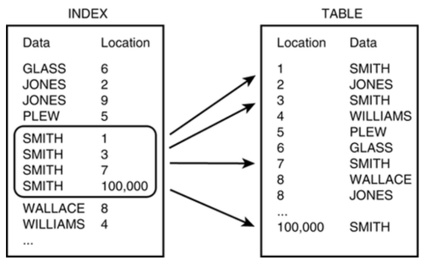
- Transactions
- Efficiencies
- SQL DCL - Users and Roles
- Next Class

Vacuum and Analyze

These commands can be run against individual tables or an entire database

VACUUM	use to reclaim storage space (removes defunct rows from the database) and update database statistics
VACUUM ANALYZE	run this first to determine whether vacuuming is necessary (as it can take a while to perform)

Indexes



Data	Location
GLASS	6
JONES	2
JONES	9
PLEW	5
SMITH	1
SMITH	3
SMITH	7
SMITH	100,000
WALLACE	8
WILLIAMS	4
...	

Location	Data
1	SMITH
2	JONES
3	SMITH
4	WILLIAMS
5	PLEW
6	GLASS
7	SMITH
8	WALLACE
8	JONES
...	
100,000	SMITH

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);
```

Spatial Index

A spatial index on a geometry or geography column speeds up spatial operations and is almost always a net gain. Automatic when loading shapefiles, but must be done manually after building geometry from coordinates or when inserting existing geometry into a new table.

```
CREATE INDEX indexname
ON tablename
USING gist (column with geometry or geography);
```

Clustering

CLUSTER is used on an index or a spatial index to reorganize records on the physical disk space so that they are stored adjacently. Speeds up processing time. Example below is for a regular (non-spatial) index.

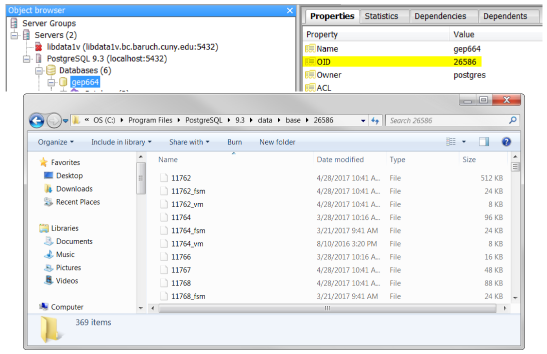
```
CREATE INDEX nyc.wstation_idx
ON nyc.weather_daily (station_id);
```

```
CLUSTER nyc.wstation_idx ON nyc.weather_daily;
```

How is Data Stored on the Physical Disk?

Default data directory: PostgreSQL - (version #) - data - base

Each database is assigned a unique OID number which represents a folder. Every database object also has a unique OID number and is stored in a file with this number.



Property	Value
Name	gep664
OID	26586
Owner	postgres
ACL	

Name	Date modified	Type	Size
11762	4/28/2017 10:41 A...	File	512 KB
11762_fsm	4/28/2017 10:41 A...	File	24 KB
11762_vm	4/28/2017 10:41 A...	File	8 KB
11764	3/28/2017 10:16 A...	File	96 KB
11764_fsm	3/21/2017 9:41 AM	File	24 KB
11764_vm	8/10/2016 3:20 PM	File	8 KB
11766	3/28/2017 10:16 A...	File	16 KB
11767	4/28/2017 10:41 A...	File	48 KB
11768	4/28/2017 10:41 A...	File	88 KB
11768_fsm	3/21/2017 9:41 AM	File	24 KB

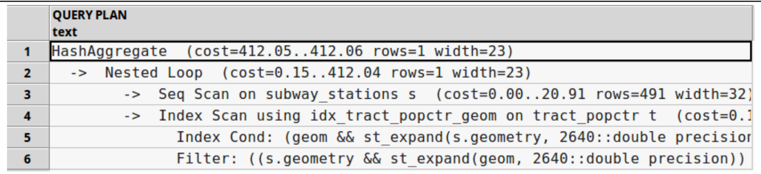
Don't EVER mess with these files. You can specify the default data directory during installation.

EXPLAIN

Use EXPLAIN prior to executing a statement to display the database's plan for executing the statement, including an estimate of how long it will take. EXPLAIN ANALYZE will show you the plan and will actually execute the statement, while EXPLAIN VERBOSE will provide you with additional output.

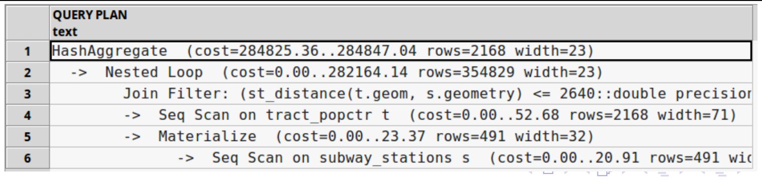
Explain Example

```
EXPLAIN SELECT DISTINCT t.geoid, t.tract, t.pop2010
FROM nyc.tract_popctr t, nyc.subway_stations s
WHERE ST_DWithin(t.geom, s.geometry, 2640);
```



```
QUERY PLAN
text
HashAggregate  (cost=412.05..412.06 rows=1 width=23)
  ->  Nested Loop  (cost=0.15..412.04 rows=1 width=23)
        ->  Seq Scan on subway_stations s  (cost=0.00..20.91 rows=491 width=32)
        ->  Index Scan using idx_tract_popctr_geom on tract_popctr t  (cost=0.1
              Index Cond: (geom && st_expand(s.geometry, 2640::double precisior
              Filter: ((s.geometry && st_expand(geom, 2640::double precision))
```

```
EXPLAIN SELECT DISTINCT t.geoid, t.tract, t.pop2010
FROM nyc.tract_popctr t, nyc.subway_stations s
WHERE ST_Distance(t.geom, s.geometry) <= 2640;
```



```
QUERY PLAN
text
HashAggregate  (cost=284825.36..284847.04 rows=2168 width=23)
  ->  Nested Loop  (cost=0.00..282164.14 rows=354829 width=23)
        Join Filter: (st_distance(t.geom, s.geometry) <= 2640::double precisior
        ->  Seq Scan on tract_popctr t  (cost=0.00..52.68 rows=2168 width=71)
        ->  Materialize  (cost=0.00..23.37 rows=491 width=32)
              ->  Seq Scan on subway_stations s  (cost=0.00..20.91 rows=491 wic
```

Triggers

- Triggers define an action the database should take when a specific event occurs
- Another method for helping to insure entity and referential integrity
- Typically performed BEFORE or AFTER an INSERT, UPDATE, or DELETE
- Provides a level of automation
- Triggers are often tied to user-created functions
- See the textbook for examples

CASE

For conditional expressions. Avoid division by zero:

```
SELECT val1, val2,
  CASE WHEN val2=0 THEN NULL
  ELSE val1/val2
  END
FROM test;
```

Assign additional description:

```
SELECT val1, val2,
  CASE WHEN val2=1 THEN 'one'
  WHEN val2=2 THEN 'two'
  ELSE 'other'
  END
FROM test;
```

Today's Topics

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...

Multiple Users

Many databases are multi-user environments where many people can access, read, and write to the database at once.

- The superuser account postgres is fine for single-user databases
- In a multi-user environment you would rarely if ever use this account, and would NOT share it with others. You would also choose a real password for it
- Each user should be assigned an individual account with different privileges

Authorization and Privileges

Individual users can be assigned their own credentials and given certain rights. These can exist on a column, table, schema, or database level. Common privileges include:

- SELECT
- INSERT
- UPDATE
- DELETE
- CREATE
- USAGE

By default, all users have CREATE and USAGE privileges in the public schema

Privileges and New Tables

When a user creates a table, they become the owner and have full privileges for the table. Others won't have rights unless they are granted. The same doesn't apply to views - the creator is the owner but doesn't necessarily have full privileges.

Users

Create new users with CREATE ROLE. You can also ALTER and DROP roles.

Create a user with a password.

```
CREATE ROLE user1
WITH LOGIN PASSWORD '12345';
```

Create a user with a password that expires.

```
CREATE ROLE user2
WITH LOGIN PASSWORD 'abcde' VALID UNTIL '
   2017-12-31';
```

PostgreSQL also allows you to use the term USER instead of ROLE.

GRANT

Give user1 and user2 certain privileges on certain tables.

```
GRANT SELECT, UPDATE
ON nyc.weather_daily, nyc.weather_station
TO user1, user2;
```

Give user admin1 access to all privileges on all tables in nyc schema, with the authority to grant others privileges.

```
GRANT ALL
ON ALL TABLES IN SCHEMA nyc
TO admin1 WITH GRANT OPTION;
```

Give all users the ability to SELECT a certain table.

```
GRANT SELECT
ON nyc.weather_daily
TO PUBLIC;
```

REVOKE

Remove all privileges for user1 and user2 on certain tables.

```
REVOKE ALL
ON nyc.weather_daily, nyc.weather_station
FROM user1, user2;
```

Remove user admin1's ability to grant privileges to others on the tables in the nyc schema.

```
REVOKE GRANT OPTION
ON ALL TABLES IN SCHEMA nyc
FROM admin1;
```

Remove all users' ability to SELECT a certain table.

```
REVOKE SELECT
ON nyc.weather_daily,
FROM PUBLIC;
```

Group Roles

It's often better to create group roles that have specific privileges, and then assign users to that group. For example, you can have different groups for viewers, workers, and admins.

Create a group role.

```
CREATE ROLE viewer;
```

Assign privileges the group role.

```
GRANT SELECT ON ALL TABLES IN SCHEMA nyc
TO viewer;
GRANT USAGE ON SCHEMA nyc TO viewer;
```

Assign a user to this group

```
GRANT user2 TO viewer;
```

Privilege Hierarchy

A user granted top-level privileges will inherit lower-level privileges by default

- A user granted full rights on a schema will have rights to the schema and everything under it (tables, views, etc)
- A user granted full rights on all tables in a schema will have all rights to those tables, but will not have the right to modify or alter the schema itself (unless granted separately)

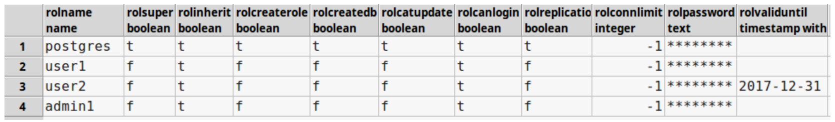
User and Roles Views

List all users in the database and see basic privileges at the database level in the pg_user view:

```
SELECT *
FROM pg_user;
```

More detailed information is available in pg_roles:

```
SELECT *
FROM pg_roles;
```



	rolname name	rolsuper boolean	rolinherit boolean	rolcreaterole boolean	rolcreatedb boolean	rolcatupdate boolean	rolcanlogin boolean	rolreplicatio boolean	rolconnlimit integer	rolpassword text	rolvaliduntil timestamp with
1	postgres	t	t	t	t	t	t	t	-1	********	
2	user1	f	t	f	f	f	t	f	-1	********	
3	user2	f	t	f	f	f	t	f	-1	********	2017-12-31
4	admin1	f	t	f	f	f	t	f	-1	********	

Viewing Privileges

Users and privileges can be managed at an object-level using pgAdmin (under properties menu for an object) or via slash commands in psql.

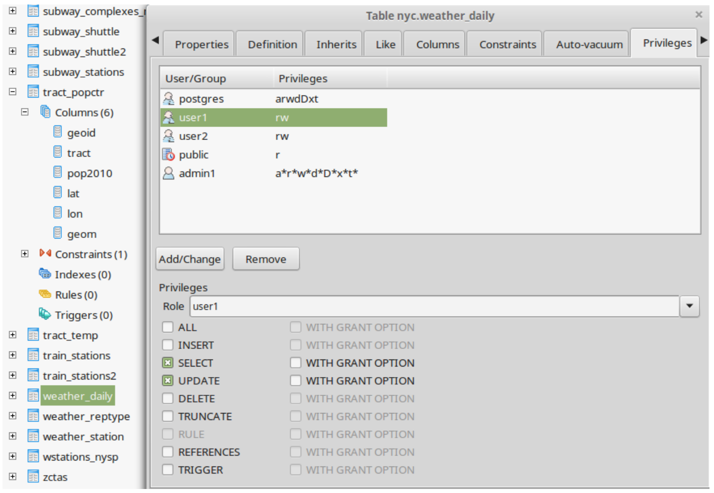


Table nyc.weather_daily

User/Group	Privileges
postgres	arwdDxt
user1	rw
user2	rw
public	r
admin1	a*r*w*d*D*x*t*

Privileges

Role: user1

☐ ALL	☐ WITH GRANT OPTION
☐ INSERT	☐ WITH GRANT OPTION
☒ SELECT	☐ WITH GRANT OPTION
☒ UPDATE	☐ WITH GRANT OPTION
☐ DELETE	☐ WITH GRANT OPTION
☐ TRUNCATE	☐ WITH GRANT OPTION
☐ RULE	☐ WITH GRANT OPTION
☐ REFERENCES	☐ WITH GRANT OPTION
☐ TRIGGER	☐ WITH GRANT OPTION

For Reference

Roles:
`https://www.postgresql.org/docs/10/sql-createrole.html`

Grant:
`https://www.postgresql.org/docs/10/sql-grant.html`

Revoke:
`https://www.postgresql.org/docs/10/sql-revoke.html`

Today's Topics

Transactions

Efficiencies

SQL DCL - Users and Roles

Next Class

Due Next Class

SPRING BREAK
There is no class on Apr 23rd. Our next class is on Apr 30th.

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

Extra Credit
Optional, posted on the course website

Readings for Class #12
Listed in the syllabus, in the *PostGIS In Action* book

PostGIS in Action Readings

Review Section 2.4 in Chapter 2
Review Section 4.5 in Chapter 4

READ Chapter 7