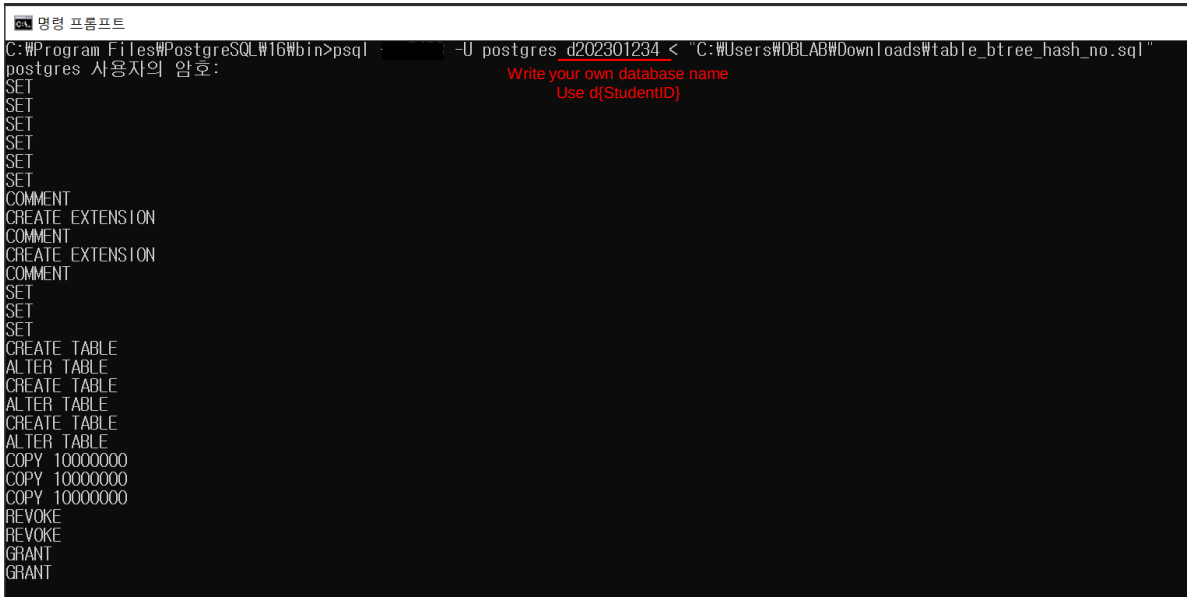
Chapter 14 - Lab

Indexing 2

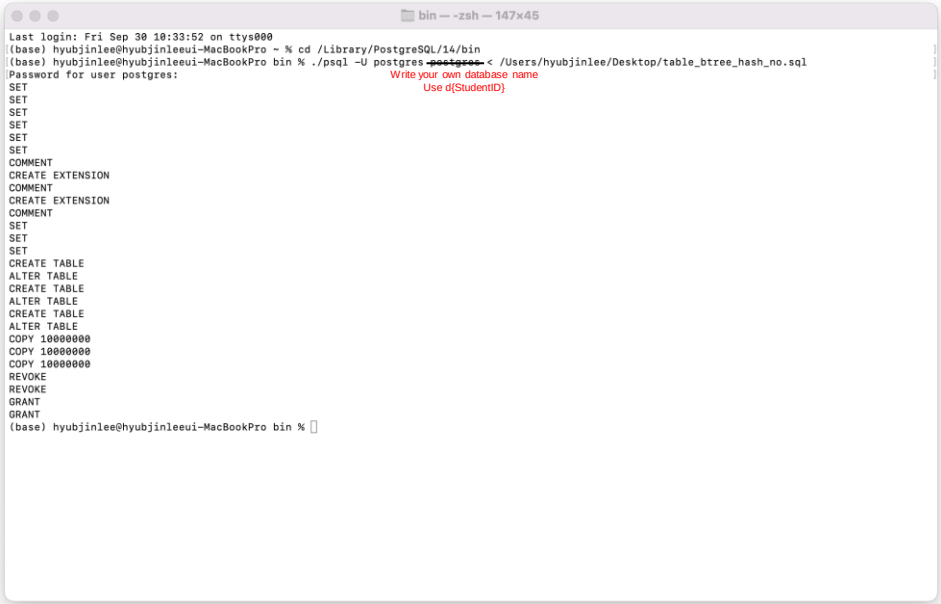
Lab Setup (Windows)

- Download the “table_btree_hash_no.sql” file from blackboard
- Open Command Prompt (cmd.exe) and type the following commands:
  1. cd C:\Program Files\PostgreSQL\16\bin
     - This is the default path. If you installed it somewhere else, go to that path.
  2. psql -U postgres d{StudentID} < [filepath]\table_btree_hash_no.sql
     - For [filepath], type the path where you downloaded “table_btree_hash_no.sql”.
  3. Type your own PostgreSQL password



Lab Setup (Max OS X)

- Download the “table_btree_hash_no.sql” file from blackboard
- Open Terminal and type the following commands:
  1. cd /Library/PostgreSQL/16/bin
     - This is the default path. If you installed it somewhere else, go to that path.
  2. ./psql -U postgres d{StudentID} < [filepath]/table_btree_hash_no.sql
     - For [filepath], type the path where you downloaded “table_btree_hash_no.sql”.
  3. Type your own PostgreSQL password



Lab Setup

- Execute PostgreSQL SQL Shell (psql) and login your database
  - Server [localhost]: Press the enter key
  - Database [postgres]: Press the enter key
  - Port [5432]: Press the enter key
  - Username [postgres]: Press the enter key
  - Password for user postgres: Type your own password

- **Type on psql command line**
  - SET enable_bitmapscan=false;
  - SET max_parallel_workers_per_gather=0;

Table Information

- 3 tables (“table_btree”, “table_hash” and “table_noindex”) have the exactly same records
  - Number of records for each table: 10,000,000
- Table schema:

Attribute	Data Type	Data Range
recordid	integer	0 ~ 10,000,000
rndm	integer	0 ~ 100,000
dummy	character(40)	

Exercise 1

- Create two indexes
  - Create indexes on attribute “recordid” in “table_btree” and “table_hash”
    - Create b-tree in “table_btree.recordid”
    - Create hash index in “table_hash.recordid”
  - Type “\h CREATE INDEX” for detailed index creation syntax
    - Use a method name “btree” for creating b-tree and “hash” for creating hash index

Exercise 2

a. Run two queries and compare the query execution plan and execution time
   - SELECT * FROM table_btree WHERE recordid=10001;
   - SELECT * FROM table_hash WHERE recordid=10001;

b. Run two queries and compare the query execution plan and execution time
   - SELECT * FROM table_btree WHERE recordid>250 AND recordid<550;
   - SELECT * FROM table_hash WHERE recordid>250 AND recordid<550;

Exercise 3

a. Update a single “recordid” field in “table_btree”. And update a single “recordid” field in “table_noindex”. Then find a difference
   - Update “recordid” from 9,999,997 to 9,999,998

b. Update 2,000,000 “recordid” fields in “table_btree”. And update 2,000,000 “recordid” fields in “table_noindex”. Then find a difference
   - Increase “recordid” fields by 100% whose value is greater than 8,000,000

c. Update all “recordid” fields in “table_btree”. And update all “recordid” fields in “table_noindex”. Then find a difference
   - Increase all “recordid” fields by 10%

PostgreSQL Geometric Types

Name	Storage Size	Representation	Description
point	16 bytes	Point on a plane	(x,y)
line	32 bytes	Infinite line (not fully implemented)	((x1,y1),(x2,y2))
lseg	32 bytes	Finite line segment	((x1,y1),(x2,y2))
box	32 bytes	Rectangular box	((x1,y1),(x2,y2))
path	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
path	16+16n bytes	Open path	[(x1,y1),...]
polygon	40+16n bytes	Polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	Circle	<(x,y),r> (center point and radius)

PostgreSQL Geometric Operators

Operator	Description	Example
+	Translation	box '((0,0),(1,1))' + point '(2.0,0)'
-	Translation	box '((0,0),(1,1))' - point '(2.0,0)'
*	Scaling/rotation	box '((0,0),(1,1))' * point '(2.0,0)'
/	Scaling/rotation	box '((0,0),(2,2))' / point '(2.0,0)'
#	Point or box of intersection	'((1,-1),(-1,1))' # '((1,1),(-1,-1))'
#	Number of points in path or polygon	# '((1,0),(0,1),(-1,0))'
@-@	Length or circumference	@-@ path '((0,0),(1,0))'
@@	Center	@@ circle '((0,0),10)'
##	Closest point to first operand on second operand	point '(0,0)' ## lseg '((2,0),(0,2))'
<->	Distance between	circle '((0,0),1)' <-> circle '((5,0),1)'
&&	Overlaps? (One point in common makes this true.)	box '((0,0),(1,1))' && box '((0,0),(2,2))'
<<	Is strictly left of?	circle '((0,0),1)' << circle '((5,0),1)'
>>	Is strictly right of?	circle '((5,0),1)' >> circle '((0,0),1)'
&<	Does not extend to the right of?	box '((0,0),(1,1))' &< box '((0,0),(2,2))'
&>	Does not extend to the left of?	box '((0,0),(3,3))' &> box '((0,0),(2,2))'
<<|	Is strictly below?	box '((0,0),(3,3))' <<| box '((3,4),(5,5))'
|>>	Is strictly above?	box '((3,4),(5,5))' |>> box '((0,0),(3,3))'
&<|	Does not extend above?	box '((0,0),(1,1))' &<| box '((0,0),(2,2))'
|&>	Does not extend below?	box '((0,0),(3,3))' |&> box '((0,0),(2,2))'
<^	Is below (allows touching)?	circle '((0,0),1)' <^ circle '((0,5),1)'
>^	Is above (allows touching)?	circle '((0,5),1)' >^ circle '((0,0),1)'
?#	Intersects?	lseg '((-1,0),(1,0))' ?# box '((-2,-2),(2,2))'
?-	Is horizontal?	?- lseg '((-1,0),(1,0))'
?-	Are horizontally aligned?	point '(1,0)' ?- point '(0,0)'
?|	Is vertical?	?| lseg '((-1,0),(1,0))'
?|	Are vertically aligned?	point '(0,1)' ?| point '(0,0)'
?-|	Is perpendicular?	lseg '((0,0),(0,1))' ?-| lseg '((0,0),(1,0))'
?||	Are parallel?	lseg '((-1,0),(1,0))' ?|| lseg '((-1,2),(1,2))'
@>	Contains?	circle '((0,0),2)' @> point '(1,1)'
<@	Contained in or on?	point '(1,1)' <@ circle '((0,0),2)'
~=	Same as?	polygon '((0,0),(1,1))' ~= polygon '((1,1),(0,0))'

Lab Setup

- Synthetic data – randomly distributed points
  - CREATE TABLE test0(id serial, x double precision, y double precision);
  - INSERT INTO test0(x, y)
    SELECT tmp.x, tmp.y
    FROM
    (SELECT (0.5-random())*180 as x, random()*360 as y
    FROM (SELECT generate_series(1, 1000000)) as t) as tmp;
  - CREATE INDEX test_idx_x on test0(x);
  - CREATE INDEX test_idx_y on test0(y);
  - SET enable_bitmapscan=false;

Lab Setup

- Synthetic data – randomly distributed points
  - CREATE TABLE test1(id serial, p point);
  - INSERT INTO test1(p)
    SELECT point(tmp.lat, tmp.long)
    FROM
    (SELECT (0.5-random())*180 as lat, random()*360 as long
    FROM (SELECT generate_series(1, 1000000)) as t) as tmp;
  - CREATE INDEX test_rtree_idx on test1 using gist(p);
  - SET enable_bitmapscan=false;

Lab Setup

- Synthetic data – randomly distributed points
  - CREATE TABLE test2(id serial, testbox box);
  - INSERT INTO test2(testbox)
    SELECT box(point(tmp.x1, tmp.y1), point(tmp.x2, tmp.y2))
    FROM
    (SELECT (0.5-random())*180 as x1, random()*360 as y1,
    ((0.5-random())*180 + random()*10) as x2, (random()*360*10) as y2
    FROM (SELECT generate_series(1, 1000000)) as t) as tmp;
  - CREATE INDEX test_box_idx on test2 using gist(testbox);
  - SET enable_bitmapscan=false;

Exercise 4

a. Find all points within a rectangle ((1,1), (10,10)) on the tables “test0” and “test1”
   - Compare an index scan and seq scan
     - SET enable_indexscan=true;
     - SET enable_indexscan=false;

b. Find all boxes overlapped with rectangles ((0,0), (1,1)) and ((9,9), (10,10)) at the same time on the table “test2”
   - Compare an index scan and seq scan
     - SET enable_indexscan=true;
     - SET enable_indexscan=false;

c. Find 10 nearest points to (0,0) on the tables “test0” and “test1”
   - Compare an index scan and seq scan
     - SET enable_indexscan=true;
     - SET enable_indexscan=false;

[Hint]

```
select *
from student
order by score asc limit 10;
```

Homework

- Complete today's practice exercises
- Write your queries and take screenshots of execution results
- Submit your report on blackboard
  - 10:29:59, October 8th, 2024
  - **Only PDF files** are accepted
  - **No late submission**

End of Lab