Indexing

StudentID :

StudentName:

```
-- account table
CREATE TABLE account(
    account_id serial PRIMARY KEY,
    name text NOT NULL,
    dob date
);
```

```
-- thread table
CREATE TABLE thread(
    thread_id serial PRIMARY KEY,
    account_id integer NOT NULL REFERENCES account(account_id),
    title text NOT NULL
);
```

~~~
-- post table
CREATE TABLE post(
    post_id serial PRIMARY KEY,
    thread_id integer NOT NULL REFERENCES thread(thread_id),
    account_id integer NOT NULL REFERENCES account(account_id),
    created timestamp with time zone NOT NULL DEFAULT now(),
    visible boolean NOT NULL DEFAULT TRUE,
    comment text NOT NULL
);```


```sql
-- word table create with word in linux file
CREATE TABLE words (word TEXT) ;
\copy words (word) FROM '/data/words';
~~~

```
-- create account data
INSERT INTO account (name, dob)
SELECT
    substring('AEIOU', (random()*4)::int + 1, 1) ||
    substring('ctdrdwftmkndnfnjnknsntnyprpsrdrgrkrmrnzslstwl',
(random()*22*2 + 1)::int, 2) ||
    substring('aeiou', (random()*4 + 1)::int, 1) ||
    substring('ctdrdwftmkndnfnjnknsntnyprpsrdrgrkrmrnzslstwl',
(random()*22*2 + 1)::int, 2) ||
    substring('aeiou', (random()*4 + 1):: int, 1),
    Now() + ('1 days':: interval * random() * 365)
FROM generate_series (1, 100)
;
```

```
-- create thread data
INSERT INTO thread (account_id, title)
SELECT
    RANDOM () * 99 + 1,
    (
        SELECT initcap(string_agg (word, ' '))
        FROM (TABLE words ORDER BY random() * n LIMIT 5) AS words (word)
    )
FROM generate_series (1, 1000) AS s(n)
;
```

```
-- create post data
INSERT INTO post (thread_id, account_id, created, visible, comment)
WITH random_comments AS (
    -- สร้างประโยคสุ่มเตรียมไว้ 1,000 แบบ (ปรับจำนวนได้)
    -- เพื่อลดภาระการ Sort ตาราง words
    SELECT row_number() OVER () as id, sentence
    FROM (
        SELECT (SELECT string_agg(word, ' ') FROM (SELECT word FROM words
ORDER BY random() LIMIT 20) AS w) as sentence
        FROM generate_series(1, 1000)
    ) s
)
SELECT
    (RANDOM() * 999 + 1)::int,
    (RANDOM() * 99 + 1)::int,
    NOW() - ('1 days'::interval * random() * 1000),
    (RANDOM() > 0.1),
    -- สุ่มหยิบประโยคจากที่เราสร้างไว้ 1,000 แบบมาใช้
    (SELECT sentence FROM random_comments WHERE id = floor(random() * 1000
+ 1)::int)
FROM generate_series(1, 100000)
;
```

WITHOUT INDEXING

```
-- table and index data
SELECT
    t.table_name,
    pg_size_pretty(pg_total_relation_size('public.' || t.table_name)) AS
total_size,
    pg_size_pretty(pg_indexes_size('public.' || t.table_name)) AS
index_size,
    pg_size_pretty(pg_relation_size('public.' || t.table_name)) AS
table_size,
    COALESCE(pg_class.reltuples::bigint, 0) AS num_rows
FROM
    information_schema.tables t
LEFT JOIN
    pg_class ON pg_class.relname = t.table_name
LEFT JOIN
    pg_namespace ON pg_namespace.oid = pg_class.relnamespace
WHERE
    t.table_schema = 'public'
    AND pg_namespace.nspname = 'public'
ORDER BY
    t.table_name ASC;

-- Output
 table_name | total_size | index_size | table_size | num_rows
------------+------------+------------+------------+----------
 account    | 32 kB      | 16 kB      | 8192 bytes |      100
 post       | 28 MB      | 2208 kB    | 26 MB      |       -1
 thread     | 168 kB     | 40 kB      | 96 kB      |       -1
 words      | 10024 kB   | 0 bytes    | 9984 kB    |   235976
(4 rows)
```

Exercise 2 See all my posts

```
-- Query 1: See all my posts
EXPLAIN ANALYZE
SELECT * FROM post
WHERE account_id = 1
;

-- Output
                                              QUERY PLAN
---------------------------------------------------------------------------
-----------------------------
 Seq Scan on post  (cost=0.00..4584.00 rows=497 width=227) (actual
time=0.099..21.629 rows=486 loops=1)
   Filter: (account_id = 1)
   Rows Removed by Filter: 99514
 Planning Time: 0.472 ms
 Execution Time: 21.697 ms
(5 rows)
```

Exercise 3 How many post have i made?

```
-- Query 2: How many post have i made?
EXPLAIN ANALYZE
SELECT COUNT(*) FROM post
WHERE account_id = 1;

-- Output
                                                QUERY PLAN
---------------------------------------------------------------------------
---------------------------------
 Aggregate  (cost=4585.24..4585.25 rows=1 width=8) (actual
time=21.530..21.532 rows=1 loops=1)
   ->  Seq Scan on post  (cost=0.00..4584.00 rows=497 width=0) (actual
time=0.096..21.460 rows=486 loops=1)
         Filter: (account_id = 1)
         Rows Removed by Filter: 99514
 Planning Time: 0.304 ms
 Execution Time: 21.589 ms
(6 rows)
```

Exercise 4 See all current posts for a Thread

```
-- Query 3: See all current posts for a Thread
EXPLAIN ANALYZE
SELECT * FROM post
WHERE thread_id = 1
AND visible = TRUE;

-- Output
                                              QUERY PLAN
---------------------------------------------------------------------------
---------------------------
 Seq Scan on post  (cost=0.00..4584.00 rows=88 width=227) (actual
time=0.223..24.058 rows=52 loops=1)
   Filter: (visible AND (thread_id = 1))
   Rows Removed by Filter: 99948
 Planning Time: 0.212 ms
 Execution Time: 24.097 ms
(5 rows)
```

Exercise 5 How many posts have i made to a Thread?

```
-- Query 4: How many posts have i made to a Thread?
EXPLAIN ANALYZE
SELECT COUNT(*)
FROM post
WHERE thread_id = 1 AND visible = TRUE AND account_id = 1;

-- Output
                                                QUERY PLAN
---------------------------------------------------------------------------
------------------------------
 Aggregate  (cost=4834.00..4834.01 rows=1 width=8) (actual
time=23.328..23.331 rows=1 loops=1)
   ->  Seq Scan on post  (cost=0.00..4834.00 rows=1 width=0) (actual
time=23.320..23.321 rows=0 loops=1)
         Filter: (visible AND (thread_id = 1) AND (account_id = 1))
         Rows Removed by Filter: 100000
 Planning Time: 0.247 ms
 Execution Time: 23.475 ms
(6 rows)
```

Exercise 6 See all current posts for a Thread for this month, in order

```
-- Query 5: See all current posts for a Thread for this month, in order
EXPLAIN ANALYZE
SELECT *
FROM post
WHERE thread_id = 1 AND visible = TRUE AND created > NOW() - '1
month'::interval
ORDER BY created;

-- Output
                                                       QUERY PLAN
----------------------------------------------------------------------------
---------------------------------------------
 Gather Merge  (cost=5167.37..5167.60 rows=2 width=227) (actual
time=37.910..42.901 rows=3 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Sort  (cost=4167.34..4167.35 rows=1 width=227) (actual
time=7.980..7.982 rows=1 loops=3)
         Sort Key: created
         Sort Method: quicksort  Memory: 25kB
         Worker 0:  Sort Method: quicksort  Memory: 25kB
         Worker 1:  Sort Method: quicksort  Memory: 25kB
         ->  Parallel Seq Scan on post  (cost=0.00..4167.33 rows=1
width=227) (actual time=3.284..7.881 rows=1 loops=3)
               Filter: (visible AND (thread_id = 1) AND (created > (now() -
'1 mon'::interval)))
               Rows Removed by Filter: 33332
 Planning Time: 0.396 ms
 Execution Time: 42.978 ms
(13 rows)
```

CREATE INDEXES

```
----- INDEX -----
CREATE INDEX ON post(account_id);

-- Query 1: See all my posts with Index
EXPLAIN ANALYZE
SELECT * FROM post
WHERE account_id = 1
;

-- Output
```

```sql
-- Query 2: How many post have i made? with index
EXPLAIN ANALYZE
SELECT COUNT(*) FROM post
WHERE account_id = 1;

-- Output
```

```sql
-- CREATE another index
CREATE INDEX ON post (thread_id);

-- Query 3: See all current posts for a Thread with index
EXPLAIN ANALYZE
SELECT * FROM post
WHERE thread_id = 1
AND visible = TRUE;

-- Output
```

```sql
-- Query 4: How many posts have i made to a Thread? with index
EXPLAIN ANALYZE
SELECT COUNT(*)
FROM post
WHERE thread_id = 1 AND visible = TRUE AND account_id = 1;

-- Output
```

```
CREATE INDEX ON post (thread_id, visible);

-- Query 3: See all current posts for a Thread with multiple indexes
EXPLAIN ANALYZE
SELECT * FROM post
WHERE thread_id = 1
AND visible = TRUE;

-- Output
```

```
-- Query 4: How many posts have i made to a Thread? with multiple indexes
EXPLAIN ANALYZE
SELECT COUNT(*)
FROM post
WHERE thread_id = 1 AND visible = TRUE AND account_id = 1;

-- Output
```

```
CREATE INDEX ON POST (thread_id, visible, account_id);

-- Query 4: How many posts have i made to a Thread? with multiple 3 indexes
EXPLAIN ANALYZE
SELECT COUNT(*)
FROM post
WHERE thread_id = 1 AND visible = TRUE AND account_id = 1;

-- Output
```

```sql
-- Add indexes name to see detail about tables and indexes
CREATE INDEX ON post(thread_id, account_id)
WHERE visible = TRUE;

SELECT
    t.table_name,
    i.indexname AS index_name,
    COALESCE(pg_class.reltuples::bigint, 0) AS num_rows,
    pg_size_pretty(pg_relation_size('public.' || t.table_name)) AS
table_size,
    pg_size_pretty(pg_relation_size('public.' || i.indexname)) AS
index_size
FROM
    information_schema.tables t
JOIN
    pg_class ON pg_class.relname = t.table_name
JOIN
    pg_namespace ON pg_namespace.oid = pg_class.relnamespace
LEFT JOIN
    pg_indexes i ON i.tablename = t.table_name AND i.schemaname =
t.table_schema
LEFT JOIN
    pg_class ic ON ic.relname = i.indexname
WHERE
    t.table_schema = 'public'
    AND pg_namespace.nspname = 'public'
    AND pg_class.relkind = 'r'  -- 'r' is for regular tables
ORDER BY
    t.table_name ASC, i.indexname;

-- Output
```

```sql
-- Partial Index
-- Query 4: How many posts have i made to a Thread? with partial indexes
EXPLAIN ANALYZE
SELECT COUNT(*)
FROM post
WHERE thread_id = 1 AND visible = TRUE AND account_id = 1;

-- Output
-- Query 3: See all current posts for a Thread with partial indexes
EXPLAIN ANALYZE
SELECT * FROM post
WHERE thread_id = 1
AND visible = TRUE;

-- Output
```

```
-- Query 5: See all current posts for a Thread for this month, in order all
indexes
EXPLAIN ANALYZE
SELECT *
FROM post
WHERE thread_id = 1 AND visible = TRUE AND created > NOW() - '1
month'::interval
ORDER BY created;

-- Output
```

```
-- Add index for Query 5
CREATE INDEX ON post (thread_id, created)
WHERE visible = TRUE;

-- Query 5: See all current posts for a Thread for this month, in order
specic index
EXPLAIN ANALYZE
SELECT *
FROM post
WHERE thread_id = 1 AND visible = TRUE AND created > NOW() - '1
month'::interval
ORDER BY created;

-- Output
```

กรณีการทดสอบ (Indexing Strategy)	Query No.	Execution Time (ms)	Scan Method (จาก Explain Plan)	ข้อสังเกต / การเปลี่ยนแปลง
Case A: No Index (มีเฉพาะ Primary Key)	Q1			จุดเริ่มต้น (Baseline)
	Q2			
	Q3			
	Q5			
Case B: Single Index	Q1			เปรียบเทียบกับ Q1 Case A
CREATE INDEX ON post(account_id);	Q2			
Case C: Composite Index	Q3			เปรียบเทียบกับ Q3 Case A
"CREATE INDEX ON post(thread_id, visible);"	Q4			
Case D: Partial Index	Q3			สังเกตขนาด Index ที่เล็กลง
"CREATE INDEX ON post(thread_id, account_id) WHERE visible = TRUE;"	Q4			
Case E: Index for Sorting	Q5			สังเกตว่ามีขั้นตอน Sort หรือไม่
"CREATE INDEX ON post(thread_id, created) WHERE visible = TRUE;"				