

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ  
СІКОРСЬКОГО"

Кафедра програмного забезпечення комп'ютерних систем

Лабораторна робота №3  
із дисципліни «Бази даних»  
на тему «**Засоби оптимізації роботи СУБД PostgreSQL**»

Виконав:  
студент 2 курсу ФПМ групи КП-92  
Остапенко Іван Петрович

Прийняв:

	Бали
Якість виконання	
Термін здачі	
Сумарний бал	

КИЇВ — 2020

Метою роботи є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

Завдання роботи полягає у наступному:

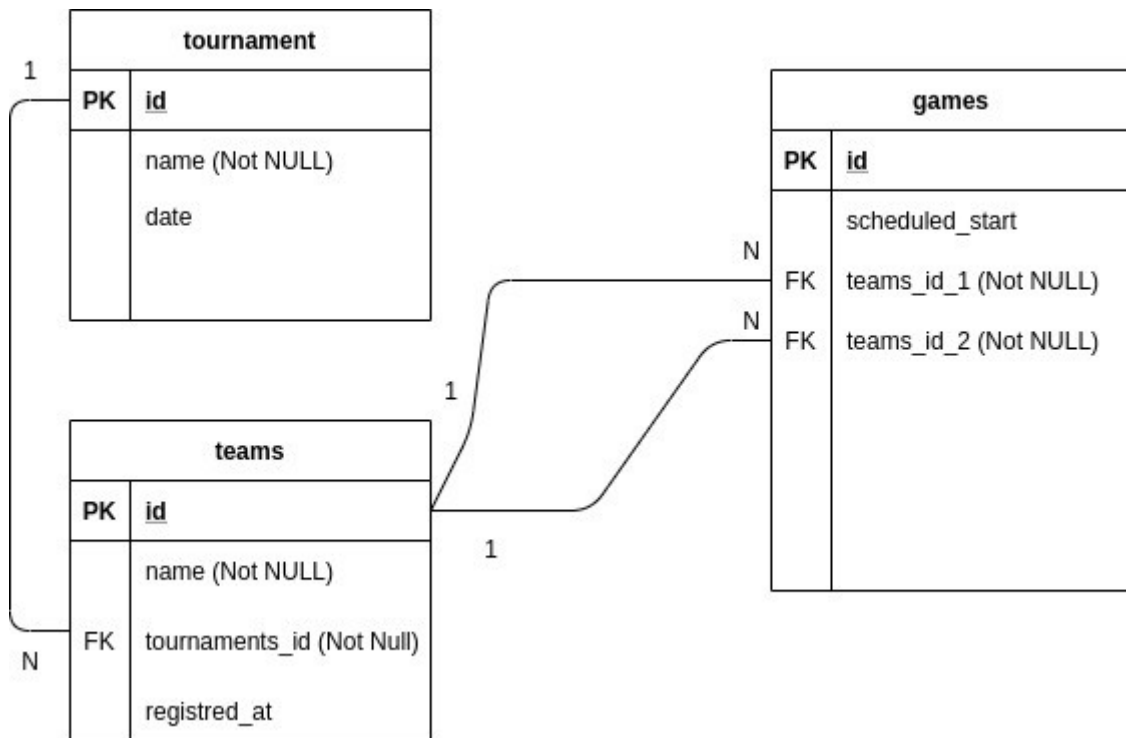
1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

Посилання на репозиторій:

<https://github.com/ostapenko-study/database-lab/tree/master/lab3>

## Завдання 1

Схема даних:



Класи ORM:

```
class Tournament(Base):
    __tablename__ = 'tournaments'

    id = Column(Integer, Sequence('tournaments_id_seq'), primary_key=True)
    name = Column(String)
    date = Column(Date)
```

```
class Team(Base):
    __tablename__ = 'teams'

    id = Column(Integer, Sequence('teams_id_seq'), primary_key=True)
    name = Column(String)
    tournaments_id = Column(Integer, ForeignKey('tournaments.id'))
    registered_at = Column(DateTime, default=datetime.datetime.utcnow)
```

```
class Game(Base):
    __tablename__ = 'games'

    id = Column(Integer, Sequence('games_id_seq'), primary_key=True)
    teams_id_1 = Column(Integer, ForeignKey('teams.id'))
    teams_id_2 = Column(Integer, ForeignKey('teams.id'))
    scheduled_start = Column(DateTime)
```

Інтерфейс модуля “Модель” не змінився, бо реалізований клас Storage не змінив свій інтерфейс. Замінені виклики SQL-запитів на відповідні запити засобами SQLAlchemy по роботі з об’єктами:

```
class Storage:

    def __init__(self, instance):
        self.instance = instance

    @benchmark
    def get_by_id(self, id: int):
        return session.\
            query(self.instance).\
            get(id)

    @benchmark
    def get_all(self):
        return session.\
            query(self.instance).\
            all()

    @benchmark
    def delete(self, id: int):
        result = session.\
            query(self.instance).\
            filter(self.instance.id == id).\
            delete()
        session.commit()
        return result

    @benchmark
    def insert(self, entity):
        entity.id = None
        session.add(entity)
        session.commit()
        return entity.id

    @benchmark
    def update(self, entity):
        result = session.\
            query(self.instance).\
            filter(self.instance.id == entity.id).\
            update(entity)
        session.commit()
        return result
```

Рис. 1 - Частина файлу models.storage

## Завдання 2

Індекси:

```
create index btree_teams_name on teams using btree(name);  
create index hash_teams_name on teams using hash(name);
```

Запит 1:

```
explain analyse  
select id, name from teams where name like 'X%'
```

без використання hash та btree індексів:

1	Seq Scan on teams (cost=0.00..311.70 rows=542 width=68) (actual time=...
2	Filter: (name ~~ 'X%':text)
3	Rows Removed by Filter: 11561
4	Planning Time: 0.137 ms
5	Execution Time: 1.905 ms

btree:

1	Bitmap Heap Scan on teams (cost=13.31..180.42 rows=542 width=68) (actual time=0.159..0....
2	Filter: (name ~~ 'X%':text)
3	Heap Blocks: exact=154
4	-> Bitmap Index Scan on btree_teams_name_inx (cost=0.00..13.18 rows=489 width=0) (actu...
5	Index Cond: ((name >= 'X':text) AND (name < 'Y':text))
6	Planning Time: 0.307 ms
7	Execution Time: 0.822 ms

hash (опція SET enable\_seqscan TO off):

Seq Scan on teams (cost=10000000000.00..100000000311.70 rows=542 width=68) (actual ti...
Filter: (name ~~ 'X%':text)
Rows Removed by Filter: 11561
Planning Time: 0.124 ms
JIT:
Functions: 4
Options: Inlining true, Optimization true, Expressions true, Deforming true
Timing: Generation 0.343 ms, Inlining 79.321 ms, Optimization 9.506 ms, Emission 7.110 ms, ...
Execution Time: 145.653 ms

Висновок: при фільтруванні рядків за іменем, що починається на певну послідовність символів, ефективніше використовувати btree, а hash – не використовувати (даний hash-індекс не використовується для виконання даного запиту).

Запит 2:

```
EXPLAIN ANALYSE
select * from teams where name like '%S%'
```

без використання hash та btree індексів:

1	Seq Scan on teams (cost=0.00..1450.70 rows=5024 width=80) (ac...
2	Filter: (name ~~ '%S%':text)
3	Rows Removed by Filter: 51600
4	Planning Time: 0.072 ms
5	Execution Time: 10.909 ms

btree/hash:

1	SET enable_seqscan TO off;
2	EXPLAIN ANALYSE
3	select * from teams where name like '%S%'
<div><div>Data Output</div><div>Explain</div><div>Messages</div><div>Notifications</div></div>	
	<div>QUERY PLAN</div> <div>text</div> <div>🔒</div>
1	Seq Scan on teams (cost=100000000000.00..10000001450.70 rows=...
2	Filter: (name ~~ '%S%':text)
3	Rows Removed by Filter: 51600
4	Planning Time: 0.100 ms
5	JIT:
6	Functions: 4
7	Options: Inlining true, Optimization true, Expressions true, Deforming...
8	Timing: Generation 0.467 ms, Inlining 77.585 ms, Optimization 10.5...
9	Execution Time: 139.137 ms

Висновок: при фільтруванні запропоновані індекси btree та hash – не використовуються.

### Запит 3:

```
explain analyse
select * from teams
where name = 'SQ'
order by id DESC
```


без використання hash та btree індексів:

1	Index Scan Backward using teams_pkey on teams (cost=10000000000.29..10000002698.67 rows=88 width=80) (actual time...
2	Filter: (name = 'SQ'::name)
3	Rows Removed by Filter: 56054
4	Planning Time: 0.139 ms
5	JIT:
6	Functions: 4
7	Options: Inlining true, Optimization true, Expressions true, Deforming true
8	Timing: Generation 0.841 ms, Inlining 2.055 ms, Optimization 17.209 ms, Emission 10.399 ms, Total 30.504 ms
9	Execution Time: 42.030 ms

btree:

1	Sort (cost=260.52..260.74 rows=88 width=80) (actual time=0.221..0.233 rows=82 loops=1)
2	Sort Key: id DESC
3	Sort Method: quicksort Memory: 36kB
4	-> Bitmap Heap Scan on teams (cost=4.97..257.68 rows=88 width=80) (actual time=0.037..0.178 rows=82 loops=1)
5	Recheck Cond: (name = 'SQ'::name)
6	Heap Blocks: exact=80
7	-> Bitmap Index Scan on btree_teams_name (cost=0.00..4.95 rows=88 width=0) (actual time=0.020..0.020 rows=82 loop...
8	Index Cond: (name = 'SQ'::name)
9	Planning Time: 0.207 ms
10	Execution Time: 0.267 ms

hash:

	QUERY PLAN	
	text	
1	Sort (cost=260.23..260.45 rows=88 width=80) (actual time=0.224..0.234 rows=82 loops=1)	
2	Sort Key: id DESC	
3	Sort Method: quicksort Memory: 36kB	
4	-> Bitmap Heap Scan on teams (cost=4.68..257.39 rows=88 width=80) (actual time=0.033..0.184 rows=82 loops=1)	
5	Recheck Cond: (name = 'SQ'::name)	
6	Heap Blocks: exact=80	
7	-> Bitmap Index Scan on hash_teams_name (cost=0.00..4.66 rows=88 width=0) (actual time=0.014..0.014 rows=82 loop...	
8	Index Cond: (name = 'SQ'::name)	
9	Planning Time: 0.144 ms	
10	Execution Time: 0.267 ms	

Висновки: індекси значно прискорюються виконання запиту з 42 ms до 0.267 ms


Запит 4:

```
EXPLAIN ANALYSE
select * from teams
where name > 'SQ'
order by name DESC
```

без використання hash та btree індексів:

1	Sort (cost=10000002440.49..10000002476.33 rows=14337 width=80) (actual time=42.736..44...
2	Sort Key: name DESC
3	Sort Method: quicksort Memory: 2390kB
4	-> Seq Scan on teams (cost=100000000000.00..10000001450.70 rows=14337 width=80) (act...
5	Filter: (name > 'SQ'::name)
6	Rows Removed by Filter: 41878
7	Planning Time: 0.118 ms
8	JIT:
9	Functions: 4
10	Options: Inlining true, Optimization true, Expressions true, Deforming true
11	Timing: Generation 0.679 ms, Inlining 1.194 ms, Optimization 20.325 ms, Emission 8.146 ms, ...
12	Execution Time: 46.085 ms

btree:

	QUERY PLAN	
	text	
1	Sort (cost=2193.40..2229.24 rows=14337 width=80) (actual time=14.464..16.250 rows=14258 ...	
2	Sort Key: name DESC	
3	Sort Method: quicksort Memory: 2390kB	
4	-> Bitmap Heap Scan on teams (cost=275.40..1203.61 rows=14337 width=80) (actual time=3...	
5	Recheck Cond: (name > 'SQ'::name)	
6	Heap Blocks: exact=749	
7	-> Bitmap Index Scan on btree_teams_name (cost=0.00..271.82 rows=14337 width=0) (ac...	
8	Index Cond: (name > 'SQ'::name)	
9	Planning Time: 0.285 ms	
10	Execution Time: 16.763 ms	



hash: не використовується

Висновок: btree прискорює виконання вибірки в заданому діапазоні (в даному запиті від 'SQ' до найбільшого). Послідовний пошук (без використання індексу) має приблизно однаковий час виконання для запиту 3 та 4.

Запит 5

```
explain analyse
select name, count(name) from teams
group by name
order by name DESC
```

без використання hash та btree індексів:

GroupAggregate (cost=10000007852.55..10000008279.82 rows=625 width=72) (actual time=122.188..137.948 rows=1205 l...
Group Key: name
-> Sort (cost=10000007852.55..10000007992.89 rows=56136 width=64) (actual time=122.155..131.794 rows=56136 loops...
Sort Key: name DESC
Sort Method: external merge Disk: 4072kB
-> Seq Scan on teams (cost=10000000000.00..10000001310.36 rows=56136 width=64) (actual time=76.043..84.432 ro...
Planning Time: 0.178 ms
JIT:
Functions: 7
Options: Inlining true, Optimization true, Expressions true, Deforming true
Timing: Generation 1.875 ms, Inlining 5.753 ms, Optimization 44.620 ms, Emission 25.470 ms, Total 77.718 ms
Execution Time: 140.699 ms

btree:

Sort (cost=2549.32..2550.88 rows=625 width=72) (actual time=28.594..28.717 rows=1205 loops=1)
Sort Key: name DESC
Sort Method: quicksort Memory: 218kB
-> HashAggregate (cost=2514.04..2520.29 rows=625 width=72) (actual time=27.603..27.967 rows=1205 loops=1)
Group Key: name
-> Bitmap Heap Scan on teams (cost=923.00..2233.36 rows=56136 width=64) (actual time=3.322..13.205 rows=56136 l...
Heap Blocks: exact=749
-> Bitmap Index Scan on btree_teams_name (cost=0.00..908.97 rows=56136 width=0) (actual time=3.225..3.225 row...
Planning Time: 1.410 ms
Execution Time: 28.870 ms

hash: не використовується

Висновок: btree прискорює виконання агрегатних функцій (з 140.7 до 28.9 ms), якщо групувати за полем, що індексується.

Висновок після аналізу результатів запитів:

Hash-індекс ефективний, коли фільтруються дані за допомогою операції рівності (запит 3). Це пов'язано з фізичною реалізацією індекса на основі хеш-таблиці.

Btree-індекс ефективний у більшості випадків. Це пов'язано з фізичною реалізацією індекса на основі збалансованих дерев, що мають багато гілок.

SeqScan ефективніший за запропоновані індекси у випадках специфічних шаблонів.

### Завдання 3

Опис роботи тригерної функції: після операцій (INSERT, UPDATE, DELETE) над рядками таблиці, формується рядок у таблиці logs, в якому описано:

- користувача, що зробив операцію
- час виконання операції
- таблиця над якою виконувалась операція
- тип операції
- id рядка над яким виконувалася операція

```
CREATE FUNCTION public.log_operation_after()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
    type_operation char := '-';
    id_record int := 0;
BEGIN

    IF (TG_OP = 'DELETE') THEN
        type_operation := 'D';
        id_record = OLD.id;
    ELSIF (TG_OP = 'UPDATE') THEN
        type_operation := 'U';
        id_record = OLD.id; -- if change id (PK) OLD.id != NEW.id
    ELSIF (TG_OP = 'INSERT') THEN
        type_operation := 'I';
        id_record = NEW.id;
    END IF;

    INSERT INTO logs (user_name, datetime_event, table_name, operation, id_record)
    VALUES (current_user, now(), TG_TABLE_NAME, type_operation, id_record);

    RETURN NEW;

EXCEPTION
    WHEN others THEN
        RAISE NOTICE 'SQLSTATE: %', SQLSTATE;
        RAISE;
END;
$BODY$;
```

Тригер реалізовано для таблиці tournaments після INSERT, UPDATE, DELETE команд:

```
CREATE TRIGGER log_tournaments_after
AFTER INSERT OR DELETE OR UPDATE
ON public.tournaments
FOR EACH ROW
EXECUTE PROCEDURE public.log_operation_after(\x);
```

Команди, що ініціюють виконання тригера:

```
-- insert
SELECT generate_tournament();

-- update
UPDATE tournaments
SET name = 'updated_name'
WHERE id = 16;

-- delete
DELETE FROM tournaments
WHERE id = 10;
```

Останні рядки в таблиці logs:

```
1 select * from logs
2 limit 3 offset (select count(*) from logs) - 3
```

Data Output Explain Messages Notifications

	id integer	user_name name	datetime_event timestamp without time zone	table_name name	operation "char" (1)	id_record integer
1	12073	postgres	2020-11-29 13:57:49.976523	tournaments	I	20
2	12074	postgres	2020-11-29 13:57:49.976523	tournaments	U	16
3	12075	postgres	2020-11-29 13:57:49.976523	tournaments	D	10