МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп’ютерних систем

Лабораторна робота № 3

з дисципліни “Бази даних”
тема “Засоби оптимізації роботи СУБД PostgreSQL”

Виконала
студентка II курсу
групи КП-02
Красношапка Анастасія Андріївна

Перевірив
“____” “_____” 20___ р.
Викладач
Радченко Костянтин Олександрович

варіант №5

Київ 2021

Мета роботи

Здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

Постановка завдання

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.

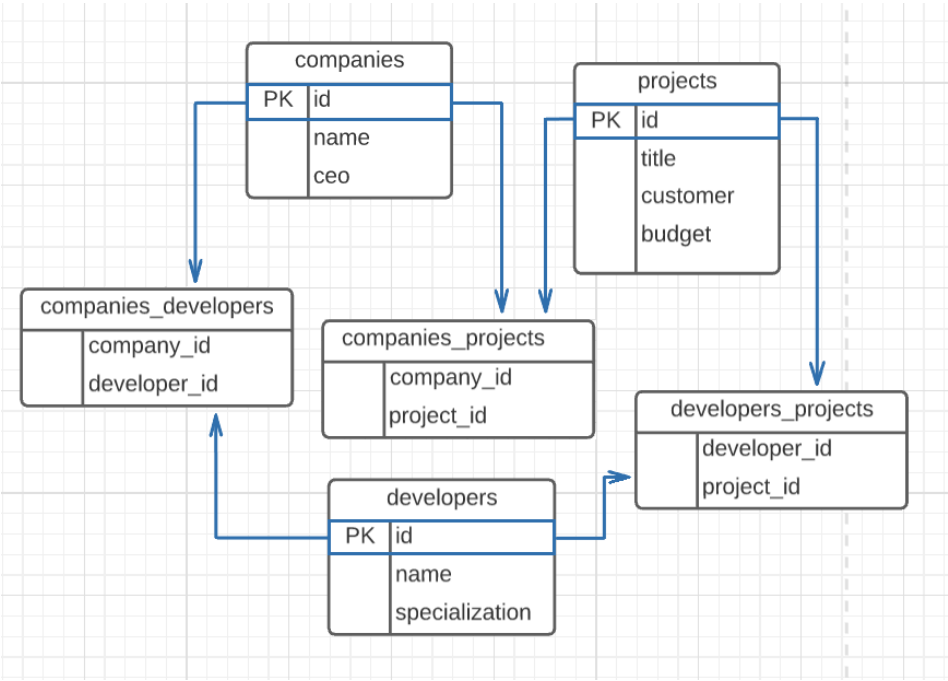
3. Розробити тригер бази даних PostgreSQL.

Виконання завдання

Вимоги до пункту №1 деталізованого завдання:

- Схему бази даних у вигляді таблиць і зв’язків між ними, а також класи ORM, що відповідають таблицям бази даних. Навести приклади запитів у вигляді ORM.

Схема бази даних:



Класи ORM:

```python
class Company(Base):
    __tablename__ = 'companies'
    id = Column(Integer, primary_key=True)
    name = Column('name', String)
    ceo = Column('ceo', String)

    Developers = relationship("Developer", secondary=companies_developers)
    Projects = relationship("Project", secondary=companies_projects)

    def __init__(self, name: str, ceo: str):
        self.name = name
        self.ceo = ceo

    def __repr__(self):
        return "<Company(id='%s', name='%s', birthday='%s')>\n" % (
            self.id, self.name, self.ceo)
```

```python
class Developer(Base):
    __tablename__ = 'developers'
    id = Column(Integer, primary_key=True)
    name = Column('name', String)
    specialization = Column('specialization', String)

    Companies = relationship("Company", secondary=companies_developers)
    Projects = relationship("Project", secondary=developers_projects)

    def __init__(self, name: str, specialization: str):
        self.name = name
        self.specialization = specialization

    def __repr__(self):
        return "<Developer(id='%s', name='%s', specialization='%s')>\n" % (
            self.id, self.name, self.specialization)
```

```python
class Project(Base):
    __tablename__ = 'projects'
    id = Column(Integer, primary_key=True)
    title = Column('title', String)
    customer = Column('customer', String)
    budget = Column('pages', Integer)

    Companies = relationship("Company", secondary=companies_projects)
    Developers = relationship("Developer", secondary=developers_projects)

    def __init__(self, title: str, customer: str, budget: int):
        self.title = title
        self.customer = customer
        self.budget = budget

    def __repr__(self):
        return "<Project(id='%s', title='%s', customer='%s', budget='%s')>\n" % (
            self.id, self.title, self.customer, self.budget)
```

```python
companies_developers = Table(
    'companies_developers', Base.metadata,
    Column('company_id', Integer, ForeignKey('companies.id')),
    Column('developers_id', Integer, ForeignKey('developers.id'))
)

developers_projects = Table(
    'developers_projects', Base.metadata,
    Column('developer_id', Integer, ForeignKey('developers.id')),
    Column('project_id', Integer, ForeignKey('projects.id'))
)

companies_projects = Table(
    'companies_projects', Base.metadata,
    Column('company_id', Integer, ForeignKey('companies.id')),
    Column('project_id', Integer, ForeignKey('projects.id'))
)
```

Приклади запитів у вигляді ORM:

```
self.session.add(company)
self.session.commit()
self.session.refresh(company)
return company.id
```

```
item = self.session.query(models.Company).filter_by(id=company.id).update(
    {models.Company.name: company.name, models.Company.ceo: company.ceo})
self.session.commit()
```

```
c = self.session.delete(item)
self.session.commit()
```

```
return self.session.query(models.Company).get(i)
```

Вимоги до пункту №2 деталізованого завдання:

- Команди створення індексів, тексти, результати і час виконання запитів SQL, пояснити чому індекси прискорюють (або не прискорюють) швидкість виконання запитів.

Створення індексів:

```
create index btree_budget on projects using btree(budget)
```

```
CREATE INDEX gn_idx
   ON projects USING gin (title gin_trgm_ops, customer gin_trgm_ops);
```

Демострація збільшення швидкодії запитів при використанні індексів:

1.
```
explain analyse select title from projects where customer like '%dfh%'
group by title
order by title
```

```
explain analyse select title from projects where customer like '%dfh%'
group by title
order by title
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Group (cost=3.83..3.84 rows...
2	[...] Group Key: title
3	[...] -> Sort (cost=3.83..3.84 r...
4	[...] Sort Key: title
5	[...] Sort Method: quicksort M...
6	[...] -> Seq Scan on projects (...
7	[...] Filter: (customer ~~ '%dfh...
8	[...] Rows Removed by Filter: ...
9	Planning Time: 0.122 ms
10	Execution Time: 2.385 ms

Без індексу – 2.385 ms

```
explain analyse select title from projects where customer like '%dfh%'
group by title
order by title
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Group (cost=3.83..3.84 rows...
2	[...] Group Key: title
3	[...] -> Sort (cost=3.83..3.84 r...
4	[...] Sort Key: title
5	[...] Sort Method: quicksort M...
6	[...] -> Seq Scan on projects (...
7	[...] Filter: (customer ~~ '%dfh...
8	[...] Rows Removed by Filter: ...
9	Planning Time: 0.167 ms
10	Execution Time: 0.074 ms

З індексом – 0.074 ms

2. `explain analyse select * from projects where budget > 20000;`

```
explain analyse select * from projects where budget > 20000;
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: (budget > 20000)
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.180 ms
5	Execution Time: 0.130 ms

Без індексу – 0.130 ms

```
explain analyse select * from projects where budget > 20000;
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: (budget > 20000)
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.076 ms
5	Execution Time: 0.035 ms

З індексом – 0.035 ms

3. `explain analyse select * from projects where title like '%se%' and customer like '%a%';`

```
explain analyse select * from projects where title like '%se%' and customer like '%a%';
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: ((title ~~ '%se%'::te...
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.133 ms
5	Execution Time: 0.124 ms

Без індексу – 0.124 ms

```
explain analyse select * from projects where title like '%se%' and customer like '%a%';
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: ((title ~~ '%se%'::te...
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.100 ms
5	Execution Time: 0.036 ms

З індексом – 0.036 ms

4. `explain analyse select * from projects where title = 'lwhdkndjscccqkg';`

```
7  explain analyse select * from projects where title = 'lwhdkndjscccqkg';
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: (title = 'lwhdkndjscc...
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.331 ms
5	Execution Time: 0.155 ms

Без індексу – 0.155 ms

```
7  explain analyse select * from projects where title = 'lwhdkndjscccqkg';
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: (title = 'lwhdkndjscc...
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.471 ms
5	Execution Time: 0.029 ms

З індексом – 0.029 ms

5. `explain analyse select * from projects where customer like '%dfh%';`

```
7  explain analyse select * from projects where customer like '%dfh%';
8
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: (customer ~~ '%dfh...
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.109 ms
5	Execution Time: 0.093 ms

Без індексу – 0.093 ms

```
7  explain analyse select * from projects where customer like '%dfh%';
8
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	Seq Scan on projects (cost=...
2	[...] Filter: (customer ~~ '%dfh...
3	[...] Rows Removed by Filter: ...
4	Planning Time: 0.088 ms
5	Execution Time: 0.034 ms

З індексом – 0.034 ms

Індекси збільшують швидкодію виконання запитів за рахунок використання більш оптимальних алгоритмів пошуку рядків.

Вимоги до пункту №3 деталізованого завдання:

- Команди, що ініціюють виконання тригера, текст тригера та скріншоти зі змінами у таблицях бази даних.

Створення тригера before update:

```
CREATE FUNCTION trigger_company_name_function()
   RETURNS TRIGGER
   LANGUAGE PLPGSQL
AS $$
BEGIN
   IF NEW.name <> OLD.name THEN
         INSERT INTO company_name_changes(old_name,new_name)
         VALUES(OLD.name, NEW.name);
    END IF;

    RETURN NEW;
END;
$$
```

```
CREATE TRIGGER company_name_changed
  BEFORE UPDATE
  ON companies
  FOR EACH ROW
  EXECUTE PROCEDURE trigger_company_name_function();
```

Команди, що ініціюють виконання тригера:

```
UPDATE public.companies
    SET name='kendrick'
    WHERE id=6;
```

Зміни у таблицях:

До виконання команди:

	id [PK] integer	name text	ceo text
1	1	aaa	bbb
2	5	Play with us.Co	WWR
3	6	tsttwlklgyvjnkg	ngipnbkipowfwdr

	old_name text	new_name text

Після виконання команди:

	id [PK] integer	name text	ceo text
1	1	aaa	bbb
2	5	Play with us.Co	WWR
3	6	kendrick	ngipnbkipowfwdr

	old_name text	new_name text
1	tsttwlklgyvjnkg	kendrick

Створення тригера before delete:

```
CREATE OR REPLACE FUNCTION trigger_project_deleted_function()
   RETURNS TRIGGER
   LANGUAGE PLPGSQL
AS $$
BEGIN
   DELETE FROM developers_projects WHERE project_id = OLD.id;
   DELETE FROM companies_projects WHERE project_id = OLD.id;
   RETURN NEW;
END;
$$
```

```
CREATE TRIGGER project_deleted
  BEFORE DELETE
  ON projects
  FOR EACH ROW
  EXECUTE PROCEDURE trigger_project_deleted_function();
```

Команди, що ініціюють виконання тригера:

```
DELETE FROM public.projects
    WHERE id=1;
```

Зміни у таблицях:

До виконання команди:

	id [PK] integer	title text	customer text	budget integer
1	1	gvffuyqhcxiqaxd	lasanftcsvuhhvr	56467

	company_id integer	project_id integer
1	1	1

	developer_id integer	project_id integer
92	7	112
93	48	90
94	39	33
95	47	49
96	40	45
97	15	65
98	25	117
99	1	1

Після виконання команди:

id [PK] integer	title text	customer text	budget integer
2	wgnlhrejlpcnkox	igjnsxnpuhocscx	26688

	company_id integer	project_id integer

	developer_id integer	project_id integer
91	28	112
92	7	112
93	48	90
94	39	33
95	47	49
96	40	45
97	15	65
98	25	117

Висновки

Під час виконання лабораторної роботи я навчилася користуватися засобами оптимізації СУБД PostgreSQL, такими як індекси і тригери. Навчилася перетворювати модель з шаблону MVC у вигляд об’єктно-реляційної проекції. А також краще розібралася у користуванні бібліотекою sqlAlchemy.