МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота №3
з дисципліни “Бази даних”
тема ««Засоби оптимізації роботи СУБД PostgreSQL»»

Виконав
студент ІІІ курсу
групи КП-82
Анікєєв Ігор Анатолійович

Посилання на репозиторій:

https://github.com/flain1/DB_Labs

Мета роботи

Метою роботи є здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL. Оптимізація запитів за допомогою індексів, робота з ORM SQLAlchemy, тригери та функції в Postgres.

Постановка завдання

Завдання роботи полягає у наступному:
1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи No2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

Приклади коду

db_labs/model/developer.py

```
from jetkit.db.model import TSTZ
from sqlalchemy import Integer, ForeignKey, Text, Index
from db_labs.db import db
from db_labs.model.constant import RecruitmentStatus
from db_labs.model.trgm_extension import TrgmExtension


class Developer(db.Model, TrgmExtension):
   first_name = db.Column(Text)
   last_name = db.Column(Text)
   email = db.Column(Text)
   birthdate = db.Column(TSTZ)

   vacancy_id = db.Column(Integer,
ForeignKey("vacancy.id", ondelete="SET NULL"))
   vacancy = db.relationship("Vacancy",
```

```
back_populates="developers")

   skills = db.relationship("Skill",
secondary="developer_skill")

   recruitment_status =
db.Column(db.Enum(RecruitmentStatus))

   developer_first_name_trgm_idx = Index(
       "developer_first_name_trgm_idx",
       first_name,
       postgresql_using="gin",
       postgresql_ops={"first_name": "gin_trgm_ops",},
   )

   developer_last_name_trgm_idx = Index(
       "developer_last_name_trgm_idx",
       last_name,
       postgresql_using="gin",
       postgresql_ops={"last_name": "gin_trgm_ops",},
   )


Developer.add_create_trgm_extension_trigger()
```

db_labs/api/developer/__init__.py

```
from typing import Dict, Union
from jetkit.api import CursorPage
from flask_smorest import Blueprint, abort
from sqlalchemy.orm import joinedload
from db_labs.api.developer.decorators import
searchable_by_skills
```

```
from db_labs.api.developer.schema import DeveloperSchema
from db_labs.db import db
from db_labs.domain.util.search import combined_search_by
from db_labs.model import Developer, Skill

blp = Blueprint("Developer", __name__,
url_prefix=f"/api/developer")


@blp.route("", methods=["GET"])
@blp.response(DeveloperSchema(many=True))
@blp.paginate(CursorPage)  # - it's slow here
@combined_search_by(
   Developer.first_name, Developer.last_name
)
def get_developers():
   """Get a paginated list of devs or search for specific
developers by first_name,last_name or skill_name."""

   return Developer.query


@blp.route("", methods=["POST"])
@blp.response(DeveloperSchema)
@blp.arguments(DeveloperSchema)
def create_developer(args: Dict[str, str]):
   """Create a developer entry."""

   #  All CRUD methods can be generalized and use a
generic controller method here. In this small lab it'd be
redundant.
   developer = Developer(**args)

   db.session.add(developer)

   db.session.commit()
```

```
    return developer


@blp.route("/<int:developer_id>", methods=["PATCH"])
@blp.response(DeveloperSchema)
@blp.arguments(DeveloperSchema)
def update_developer(args: Dict[str, Union[str, int]],
developer_id: int):
    """Check if developer with given id exists, then
update the entry."""

    #  All CRUD methods can be generalized and use a
generic controller method here. In this small lab it'd be
redundant.

    # remove None values so they do not override existing
data.
    values = {key: value for key, value in args.items() if
value is not None}

    developer = Developer.query.filter(Developer.id ==
developer_id).one_or_none()

    if not developer:
        abort(404, message="Couldn't find developer to
update.")

    for attr, value in values.items():
        if hasattr(developer, attr):
            setattr(developer, attr, value)

    db.session.commit()

    return developer
```

```python
@blp.route("/<int:developer_id>", methods=["DELETE"])
@blp.response()
def delete_developer(developer_id: int):
    """Check if developer with given id exists, then
update the entry."""

    #  All CRUD methods can be generalized and use a
generic controller method here. In this small lab it'd be
redundant.
    developer = Developer.query.filter(Developer.id ==
developer_id).one_or_none()

    if not developer:
        abort(404, message="Couldn't find developer to
update.")

    db.session.delete(developer)

    db.session.commit()
```

тригер BEFORE UPDATE та ON DELETE

```sql
DROP FUNCTION IF EXISTS
update_developers_on_vacancy_update_or_delete() CASCADE;
CREATE FUNCTION
update_developers_on_vacancy_update_or_delete() RETURNS
trigger AS $$
BEGIN
    IF NEW.id THEN
```

```
        UPDATE developer set recruitment_status =
'in_progress' where vacancy_id = NEW.id;
    ELSE
        UPDATE developer set recruitment_status =
'rejected' where vacancy_id = OLD.id;
        DELETE FROM vacancy where id = OLD.id;
    END IF;
    RETURN NEW;
END;
  $$ LANGUAGE 'plpgsql';
CREATE TRIGGER update_latest_vacancy_stage_trigger BEFORE
UPDATE OR DELETE ON vacancy
FOR EACH ROW WHEN (pg_trigger_depth() = 0) EXECUTE
PROCEDURE
update_developers_on_vacancy_update_or_delete();
```

GIN та Hash індекси

```
-- Hash index on the recruitment_status column on the
developer table. Hash indexes ought to be used for
comparison with "=" operator
EXPLAIN ANALYSE select * from developer where
recruitment_status='rejected';

CREATE INDEX developer_status_idx on developer USING hash
(recruitment_status);

DROP index developer_status_idx;
--

-- GIN trgm indexes on developer fist- lastname.
Primarily for use with ILIKE, full text search etc.
```

```sql
EXPLAIN ANALYSE select * from developer where first_name
ilike '%Dizzzmas%';

CREATE INDEX developer_first_name_trgm_idx ON developer
USING GIN (first_name gin_trgm_ops);
CREATE INDEX developer_last_name_trgm_idx ON developer
USING GIN (last_name gin_trgm_ops);


DROP INDEX developer_first_name_trgm_idx;
DROP INDEX developer_last_name_trgm_idx;
--
```

Висновок

Виконавши дану лабораторну роботу було відпрацьовано навички проектування та нормалізації схем БД, використання БД клієнту DataGrip. Було здобуто досвід роботи з SQLAlchemy ORM, тригерами та функціями в Postgres, GIN та Hash індексами.