ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №3

Обмеження цілісності даних та індекси в SQL

Виконав:
Студент Процьків Назарій
Група <u>ПМі-21</u>
Оцінка
Перевірила:
доц. Малець Р.Б.

Тема: Обмеження цілісності даних та індекси в SQL.

Мета роботи: Ознайомлення з поняттями обмеження цілісності даних та індексами в SQL, їх створенням і використанням.

Теоретичний матеріал

Типи даних - це спосіб обмеження виду даних, які можуть зберігатися в таблиці. Однак для багатьох застосувань такі обмеження занадто грубі. Наприклад, стовпець, що містить ціну товару, повинен, мабуть, приймати лише позитивні значення. Але не існує стандартного типу даних, який приймає лише додатні значення. Інша проблема полягає в тому, що ви можете обмежити дані стовпців стосовно інших стовпців або рядків. Наприклад, у таблиці, що містить інформацію про товар, має бути лише один рядок для кожного кода продукту.

3 цією метою SQL дозволяє визначати обмеження для стовпців та таблиць. Обмеження дають вам стільки контролю над даними в таблицях, скільки ви хочете. Якщо користувач намагається зберігати дані у стовпці, що порушує обмеження, виникає помилка. Це застосовується, навіть якщо це значення за замовчуванням.

Перелік розділів та понять, з якими необхідно самостійно ознайомитись для виконання завдання лабораторної роботи:

- 1. Обмеження цілісності даних в SQL
 - 1.1. Обмеження-перевірки
 - 1.2. Обмеження NOT NULL
 - 1.3. Обмеження унікальності
- 1.4. Первинні ключі
- 1.5. Зовнішні ключі
- 1.6. Обмеження-виключення
- 2. Індекси
 - 2.1. Типи індексів
 - 2.2. Складені (багатостовпчикові) індекси
 - 2.3. Інлекси і ORDER BY
 - 2.4. Об'єднання декількох індексів
 - 2.5. Унікальні індекси
 - 2.6. Індекси за виразами
 - 2.7. Часткові індекси
 - 2.8. Сканування тільки індексу і покриття індексів
 - 2.9. Сімейства і класи операторів
 - 2.10. Індекси і правила сортування
 - 2.11. Контроль використання індексів

Хід роботи

1. Обмеження.

У моїй базі даних ϵ чотири таблиці з датою початку і датою кінця певної

діяльності, відповідно для цього можна створити обмеження.

```
Query Query History
                                Query Query History
                                1 ALTER TABLE studyplaces
1 ALTER TABLE beingplaces
2 ADD CONSTRAINT date_constraint 2 ADD CONSTRAINT date_constraint
3 CHECK (enddate > startdate) 3 CHECK (enddate > startdate)
                               Query Query History
Query Query History
1 ALTER TABLE workplaces
                                1 ALTER TABLE serviceplaces
2 ADD CONSTRAINT date_constraint 2 ADD CONSTRAINT date_constraint
3 CHECK (enddate > startdate)
3 CHECK (enddate > startdate)
CREATE TABLE IF NOT EXISTS public.beingplaces
    beingid being_check COLLATE pg_catalog."default",
    city text COLLATE pg_catalog."default",
    startdate date,
    enddate date,
    CONSTRAINT date_constraint CHECK (enddate > startdate)
CREATE TABLE IF NOT EXISTS public.serviceplaces
   serviceid service_check COLLATE pg_catalog."default",
   service text COLLATE pg_catalog."default",
   startdate date,
   enddate date,
   CONSTRAINT date constraint CHECK (enddate > startdate)
CREATE TABLE IF NOT EXISTS public.studyplaces
    studyid study_check COLLATE pg_catalog."default",
    study text COLLATE pg_catalog."default",
    startdate date,
    enddate date,
    CONSTRAINT date_constraint CHECK (enddate > startdate)
CREATE TABLE IF NOT EXISTS public.workplaces
    workid work_check COLLATE pg_catalog."default",
    company text COLLATE pg_catalog."default",
    startdate date,
    enddate date,
    CONSTRAINT date_constraint CHECK (enddate > startdate)
```

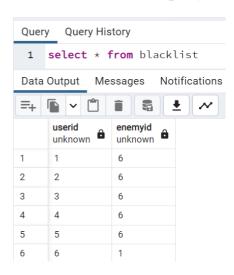
Також додав обмеження на id користувача, щоб воно було більшим за нуль.

```
Query Query History

1 ALTER TABLE users
2 ADD CONSTRAINT id_constraint
3 CHECK (id > 0)
```

```
CREATE TABLE IF NOT EXISTS public.users
(
   id user_id NOT NULL,
   firstname text COLLATE pg_catalog."default",
   lastname text COLLATE pg_catalog."default",
   birthdate birthdate_check,
   phonenumber text COLLATE pg_catalog."default",
   password text COLLATE pg_catalog."default",
   address text COLLATE pg_catalog."default",
   "workPlace" work_check COLLATE pg_catalog."default",
   "studyPlace" study_check COLLATE pg_catalog."default",
   "servicePlace" service_check COLLATE pg_catalog."default",
   "beingPlace" being_check COLLATE pg_catalog."default",
   CONSTRAINT users_pkey PRIMARY KEY (id),
   CONSTRAINT id_constraint CHECK (id::bigint > 0)
)
```

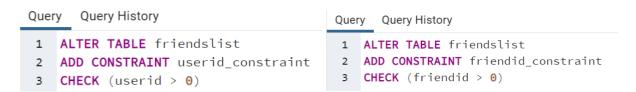
У моїй базі даних присутня таблиця blacklist, яка показує чорний список користувача.



Наприклад, користувач, який має id 1 заблокував користувача, який має id 6. Користувач, який має id 2 заблокував користувача, який має id 6. І так далі. Ці числа є користувацькими id, тому потрібно додати обмеження на них.



Для таблиці друзів зробив так само.



Для обох цих таблиць потрібно додати ще одне обмеження: користувач не може сам себе заблокувати чи сам себе додати у друзі.

```
Query Query History

1 ALTER TABLE blacklist
2 ADD CONSTRAINT self_c
3 CHECK (userid != enemyid)

Query Query History

1 ALTER TABLE friendslist
2 ADD CONSTRAINT self_c
3 CHECK (userid != friendid)
```

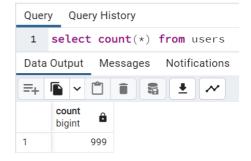
Остаточні таблиці з обмеженнями:

```
CREATE TABLE IF NOT EXISTS public.friendslist
(
   userid user_id,
    friendid user_id,
   CONSTRAINT friendslist_userid_fkey FOREIGN KEY (userid)
        REFERENCES public.users (id) MATCH SIMPLE
       ON UPDATE NO ACTION
       ON DELETE NO ACTION,
   CONSTRAINT friendid_constraint CHECK (friendid::bigint > 0),
   CONSTRAINT self_c CHECK (userid::bigint <> friendid::bigint),
   CONSTRAINT userid_constraint CHECK (userid::bigint > 0)
CREATE TABLE IF NOT EXISTS public.blacklist
(
    userid user_id NOT NULL,
    enemyid user_id,
    CONSTRAINT blacklist_userid_fkey FOREIGN KEY (userid)
        REFERENCES public.users (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT enemyid_constraint CHECK (enemyid::bigint > 0),
    CONSTRAINT self_c CHECK (userid::bigint <> enemyid::bigint),
    CONSTRAINT userid_constraint CHECK (userid::bigint > 0)
```

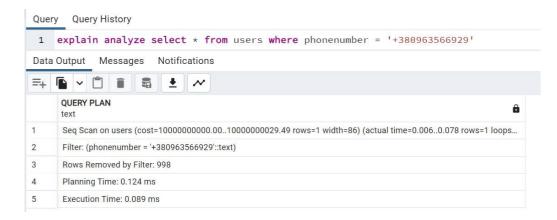
2. Індекси.

Для виконання завдання створення індексів я обрав таблицю users.

Спочатку заповнив її багатьма кортежами.



Перед тим як створювати індекси написав простий запит аби порівняти роботу таблиці з індексами та без.



Запит повернув час 0.089 сек.

Створив індекс на номер телефону.



Написав цей же запит.



Бачимо, що запит виконується швидше — за $0.037~{\rm cek}$.

Висновок: під час виконання лабораторної роботи я ознайомився з поняттями обмеження цілісності даних та індексами в SQL, їх створенням і використанням.