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

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

КУРСОВА РОБОТА

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: “Програмний додаток користувача для керування електронною базою
анімаційних персонажів”

Студент
групи КП-91 **Маховой Олександр Вікторович** _____ (підпис)

Викладач
к.т.н, доцент кафедри СПіСКС **Петрашенко А.В.** _____ (підпис)

Захищено з оцінкою _____

Київ – 2020

АНОТАЦІЯ

Дана курсова робота включала в себе здобуття практичних навичок у створенні прикладних програмних додатків, які взаємодіють із базою даних PostgreSQL. Було виокремлено наступні етапи розробки додатку:

- Створення системи отримання/генерації та фільтрації даних
- Створення системи реплікації даних
- Створення системи аналізу даних предметної галузі
- Створення системи резервування/відновлення даних

Результатом виконання курсової роботи стала реалізація усіх пунктів, описаних вище, та отримання кінцевої інформаційно-аналітичної системи, яка виконує аналіз даних, отриманих із зовнішніх ресурсів, та взаємодіє із реляційною СУБД PostgreSQL

Зміст

Зміст

Вступ

На сьогоднішній день, аніме – це один з найбільш популярних на стрімко розвиваючихся видів мистецтва. Тому було вирішено присвятити цю курсову роботу розробці бази даних анімаційних персонажів. На відміну від існуючих в інтернеті подібних баз, в цій наявно більше можливостей для аналізу даних та їх візуалізації.

У сервісі, що використовує цю базу даних можна буде створювати такі основні сутності як:

- Персонажі
- Аніме
- Продюсери

1. Аналіз інструментарію для виконання курсової роботи

Для виконання даної роботи у якості системи керування базами даних було обрано PostgreSQl. Такий вибір був зроблений у зв'язку з такими факторами:

- Відкрите ПЗ відповідає стандарту SQL - PostgreSQL - безкоштовне ПЗ з відкритим вихідним кодом. Ця СУБД є дуже потужною системою.
- Підтримка великої кількості типів даних, включно з власними
- Цілісність даних з усіма необхідними обмеженнями
- Надійність, безпека
- PostgreSQL не просто реляційна, а об'єктно-реляційна СУБД, що надає певні переваги
- Працює з багатьма типами мереж
- Велика місткість
- Велика спільнота – просто знайти вирішення потенційних проблем при розробці
- Це повністю опен-сорсний проект
- Розширення - існує можливість розширення функціоналу за рахунок своїх процедур

Для взаємодії з базою даних було обрано бібліотеку Npgsql, оскільки:

- Добре підходить для зручного використання у мові програмування C#
- Розроблена спеціально для PostgreSQL
- Найпопулярніша для взаємодії з PostgreSQL у мові програмування C#
- Має чітку, зрозумілу та вичерпну документацію з хорошими прикладами

Для візуалізації результатів аналізу даних було обрано API QuickChart, оскільки:

- Вона надає зручний інтерфейс для автоматичного будування графічних об'єктів
- Для графічних об'єктів наявна можливість дуже гнучкого налаштування з великою кількістю опцій для вигляду

- Наявна можливість будувати надзвичайно різноманітні графічні об’єкти
- Наявна чітка, зрозуміла та вичерпна документація з хорошими прикладами побудови різних графічних об’єктів з різними налаштуваннями
- Присутні недоліки у вигляді відсутності програмного налаштування масштабу графіків.

2. Структура бази даних

База даних має такі таблиці з полями:

- Сутність girl, що містить поля id_girl – ідентифікатор сутності, fullname – ім’я сутності, age – вік дівчини, hair – колір волосся, eyes – колір очей.
- Сутність anime, що містить поля id_anime ідентифікатор сутності, title – назва аніме, year - рік, в якому почав виходити серіал/фільм, series – кількість серій, rating – рейтинг аніме.
- Сутність producer, що містить поля id_producer - ідентифікатор сутності, name – ім’я режисера, studio – назва студії, в якій він працював над картиною, number_of_works – загальна кількість робіт режисера.
- Таблиця links_anime_producers для зв’язу аніме та продюсерів.
- Таблиця links_girls_anime для зв’язку дівчин та аніме.
- Таблиця producers_log яка містить інформацію про дату додавання або зміну сутності producer.
- Таблиця rating_changes з інформацією про оновлення рейтингу аніме.

3. Опис програмного забезпечення

3.1. Загальна структура програмного забезпечення

Розроблене програмне забезпечення містить такі компоненти:

1. База даних, що зберігає інформацію про сутності
2. Засоби псевдовипадкової генерації даних
3. Засоби пошуку та валідації
4. Засоби реплікації
5. Засоби резервного копіювання з можливістю вибору версії
6. Засоби аналізу даних

3.2. Опис модулів програмного забезпечення

Розроблене програмне забезпечення було розбите на такі модулі:

Модуль model:

Цей модуль власне взаємодіє із базою даних. У цьому модулі містяться усі запити для отримання, вилучення, вставки або редагування даних.

Модуль view:

Цей модуль потрібен для взємодії із користувачем цього програмного забезпечення. Він містить у собі консольний інтерфейс та засоби валідації вхідних даних.

У цьому модулі користувач може обрати дію, яку він хоче виконати та відправити цей запит у контролер.

Модуль controller:

Цей модуль допомагає взаємодіяти модулю model та модулю view між собою.

При відповіді модуля model він форматує дані для зручного їх відображення у view.

3.3. Опис основних алгоритмів роботи

Через те, що знайти датасет із потрібними даними знайти не вдалося, дані для таблиць прийшлося генерувати псевдовипадково.

При генерації даних для кожної з таблиць процес був побудований так, аби генерувалися більш-менш адекватні дані для обраної предметної галузі та для обраної структури бази даних.

Для спрощення обробки всіх вийняткових ситуацій, усі операції із базою даних виконуються через одну точку, що знаходиться в блоці try...except.

4. Аналіз функціонування засобів реплікації

Для реплікації була створена друга інстанція серверу PSQL на цій же самій системі. Реалізована реплікація за принципом master-slave, що є єдиним вбудованим способом реалізації реплікації у даній СУБД. На master сервері присутня спеціальна роль для виконання реплікації.

Також, через те що робота виконується на системі Microsoft Windows 10 мені не вдалось знайти адекватний метод для забезпечення автоматичного переходу на slave-сервер, бо усі знайдені інструменти для цього були розроблені виключно для Linux систем. Отже, у випадку виходу з ладу головного сервера умовний системний адміністратор має виконати наступні кроки:

- Змінити порт slave-сервера на 5432 у postgresql.conf
- Написати у cmd:
    - pg_ctl promote -D SLAVE_SERVER_PATH
    - sc stop SLAVE_SERVER_SERVICE
    - sc start SLAVE_SERVER_SERVICE

Ці дії переведуть сервер на основний адрес та підвищать його до рівня master-сервера. Далі, вийшовший з ладу сервер можна перевести у режим slave після виконання його обслуговування.

Приклад у додатку.

5. Аналіз функціонування засобів резервування/відновлення бази даних

Вбудований механізм резервного копіювання має ручний режим роботи та автоматичний, наприклад раз у день чи тиждень. Для цього потрібно використовувати скрипти. Це можна реалізувати так:

- Отримати двійкові файли pg_dump, pg_dumpall, потрібно їх витягти з сервера PostgreSQL Server, встановити, скомпілювати або завантажити двійкові файли з EDB. Немає пакету для отримання лише цих файлів.
- Перейдіть на сервер резервного копіювання / розташування, створіть каталог, який називається Drive:\PostgresqlBack, а потім створіть підкаталог під назвою "bin" на Drive:\PostgresqlBack і розмістіть у цьому каталозі наступні файли:

```
libeay32.dll
libiconv-2.dll
libintl-8.dll
libintl-9.dll
libpg.dll
libwinpthread-1.dll
msvr120.dll
pg_dump.exe
pg_dumpall.exe
ssleay32.dll
zlib1.dll
```

- Створіть пакетний файл, який називається приблизно як postgresqlBackup.bat. Файл повинен знаходитись у каталозі PostgresqlBack, а не в папці bin.
- Відкрийте файл, а потім скопіюйте наступне:

```
@echo off
   for /f "tokens=1-4 delims=/ " %%i in ("%date%") do (
     set dow=%%i
     set month=%%j
     set day=%%k
     set year=%%l
   )
   set datestr=%month%_%day%_%year%
   echo datestr is %datestr%

   set BACKUP_FILE=<NameOfTheFile>_%datestr%.backup
   echo backup file name is %BACKUP_FILE%
   SET PGPASSWORD=<PassWord>
   echo on
   bin\pg_dump -h <HostName> -p 5432 -U <UserName> -F c -b -v -
f %BACKUP_FILE% <DATABASENAME>
```

- Змініть <NameOfTheFile> на щось. Одна ідея полягає у використанні імені бази даних. (Переконайтеся, що після слова BACKUP_FILE відсутні пробіли, оскільки будь-які пробіли спричинять непрацювання цього параметра.) Параметр - це перша частина імені файлу, а потім дата створення файлу з розширенням .backup
- Змініть наведений вище параметр <PassWord> на правильний пароль для користувачів, які резервно копіюють. (переконайтеся, що після слова PGPASSWORD немає пробілів, будь-які пробіли призведуть до того, що це налаштування не буде працювати. Опис pgPassword
- Змініть <HostName> на ip-адресу або dns-ім'я сервера, що розміщує Postgresql.
- Змініть <UserName> на користувача резервного копіювання, переконайтесь, що цей користувач має доступ до бази даних для резервного копіювання
- Змініть <DATABASENAME> на ім'я бази даних, для якої створюється резервна копія.
- Збережіть файл
- Створити завдання для планувальника завдань MS
- Після того, як ви вибрали контекст безпеки, в якому буде виконуватися Завдання, рекомендується змінити захист каталогу, де виконується резервне копіювання та зберігаються файли, оскільки ім’я користувача та пароль високого рівня зберігаються у простому тексті.
- Інший варіант - змінити файл pg_hba.conf, додавши сервер резервної копії як надійне з'єднання

Але, я обрав вже готове рішення для резервного копіювання SQLBackupAndFTP. У цій утиліті можна налаштувати все це за допомогою лише декількох кліків.

Зображення програми та часу резервування та відновлення у додатку.

6. Аналіз результатів підвищення швидкодії виконання запитів

З метою підвищення швидкодії запитів для отримання деяких даних було використано індексування полів, які будуть використовуватися найчастіше. Серед них поля, що входять до складу запитів пошуку, такі як anime.series та producers.number_of_work. В останньому присутні найменші за діапазоном числа, тому B-Tree індексація найбільш доцільна саме на цьому полі

Також для текстового пошуку використовується індекс типу GIN, але через псевдовипадкову генерацію він є майже не є ефективним.

У випадку коли даних у таблиці багато (наприклад, 50 тисяч та більше) лінійний пошук стає заповільним, у зв’язку з чим для великих баз даних і потрібні індекси. Однак у разі малої бази даних індекси є неефективними, їхні алгоритми складніші і довші ніж просто лінійний пошук коли даних мало. У зв’язку з цим у таблицях, що не є основними та де зберігається мало елементів, індекси не використовувались.

7. Опис результатів аналізу предметної галузі

У розробленому консольному додатку наявний такий аналіз даних, що містяться у базі:

- Рейтинг аніме. Додається графічне представлення у вигляді стовпчастих діаграм, де на вертикальних осях вказані назви картин, а по горизонталі відкладений їх рейтинг.
- Динаміка зміни рейтингу обраного аніме. Додається графічне представлення у вигляді лінійної діаграми.
- Рейтинг студій за кількістю картин за рік. Додається графічне представлення у вигляді кругової діаграми.

Висновки

Під час виконання даної курсової роботи виконано таку роботу та отримано такі результати:

- Було розроблено базу даних, яка відповідає 3-ій нормальній формі та організована максимально зручно та просто
- Засоби реплікації були реалізовані до рівня мануального переведення додаткового сервера у режим основного в разі виходу того з ладу.
- Резервне копіювання було реалізовано повне, що дає можливість швидкого відновлення
- Була розроблена псевдовипадкова генерація для всіх таблиць, яка генерує реалістичні значення
- Була підвищена швидкодія запитів до бази даних шляхом індексування деяких полів деяких таблиць
- Були розроблені засоби для аналізу даних із бази, які також надають можливість виводити графічне представлення його результату для наочності висновків
- Був розроблений консольний інтерфейс який також обробляє всі помилки та валідує дані

У результаті виконання даної курсової роботи було набуто практичні навички розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуто навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації.

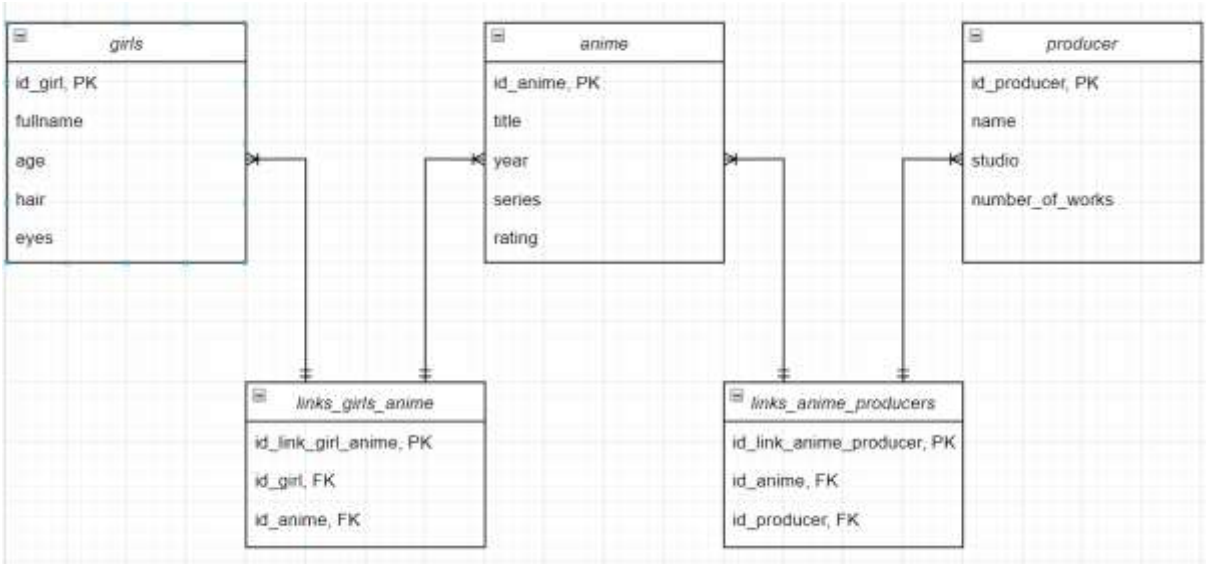
Завдяки виконанню даної роботи було здобуто вміння розробляти програмне забезпечення для реляційних баз даних, відбулося оволодіння основами використання СУБД, а також інструментальними засобами підтримки розробки додатків для подібних баз даних.

Література

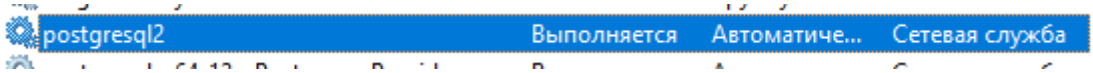
1. PostgreSQL 12.5 Documentation [Електронний ресурс] / The PostgreSQL Global Development Group // PostgreSQL: The World's Most Advanced Open Source Relational Database веб-сайт. URL: https://www.postgresql.org/docs/12/index.html. –
2. Advantages of PostgreSQL. bitnine: веб-сайт. URL: https://bitnine.net/blog-postgresql/advantages-of-postgresql/?ckattempt=1
3. Advantages of PostgreSQL. Cybertec PostgreSQL: веб-сайт. URL: https://www.cybertec-postgresql.com/en/postgresql-overview/advantages-of-postgresql/
4. Automated Backup on Windows. Wiki PostgreSQL: веб-сайт. URL: https://wiki.postgresql.org/wiki/Automated_Backup_on_Windows
5. QuickChart Documentation веб-сайт. URL: https://quickchart.io/documentation/
6. SQL Tutorials. URL: https://www.w3schools.com/sql/default.asp
7. URL Processing. URL: https://stackoverflow.com/questions/4580263/how-to-open-in-default-browser-in-c-sharp
8. Replication. URL: https://info.crunchydata.com/blog/postgres-streaming-replication-on-windows-a-quick-guide
9. Npgsql: URL: https://www.npgsql.org/doc/basic-usage.html
10. Entity Framework: https://metanit.com/sharp/entityframeworkcore/
11. MVC: URL: https://www.geeksforgeeks.org/basic-crud-create-read-update-delete-in-asp-net-mvc-using-c-sharp-and-entity-framework/

Додатки

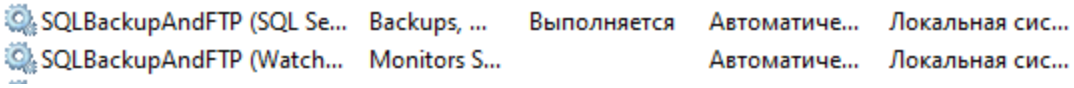
А. Графічні матеріали



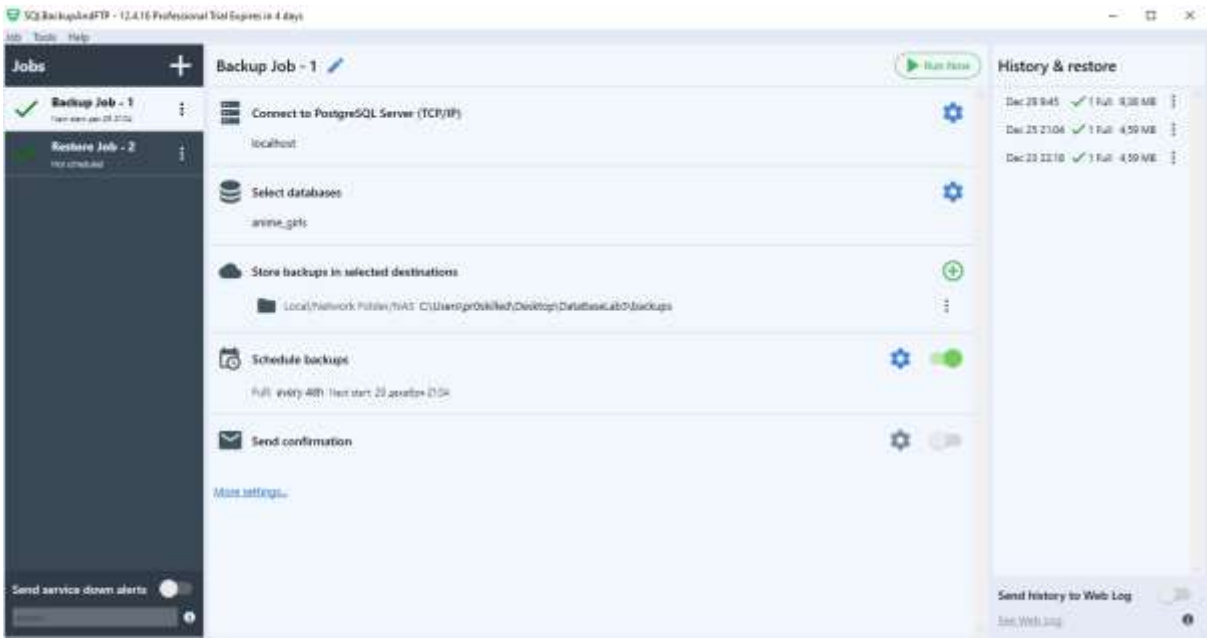
Структура бази даних



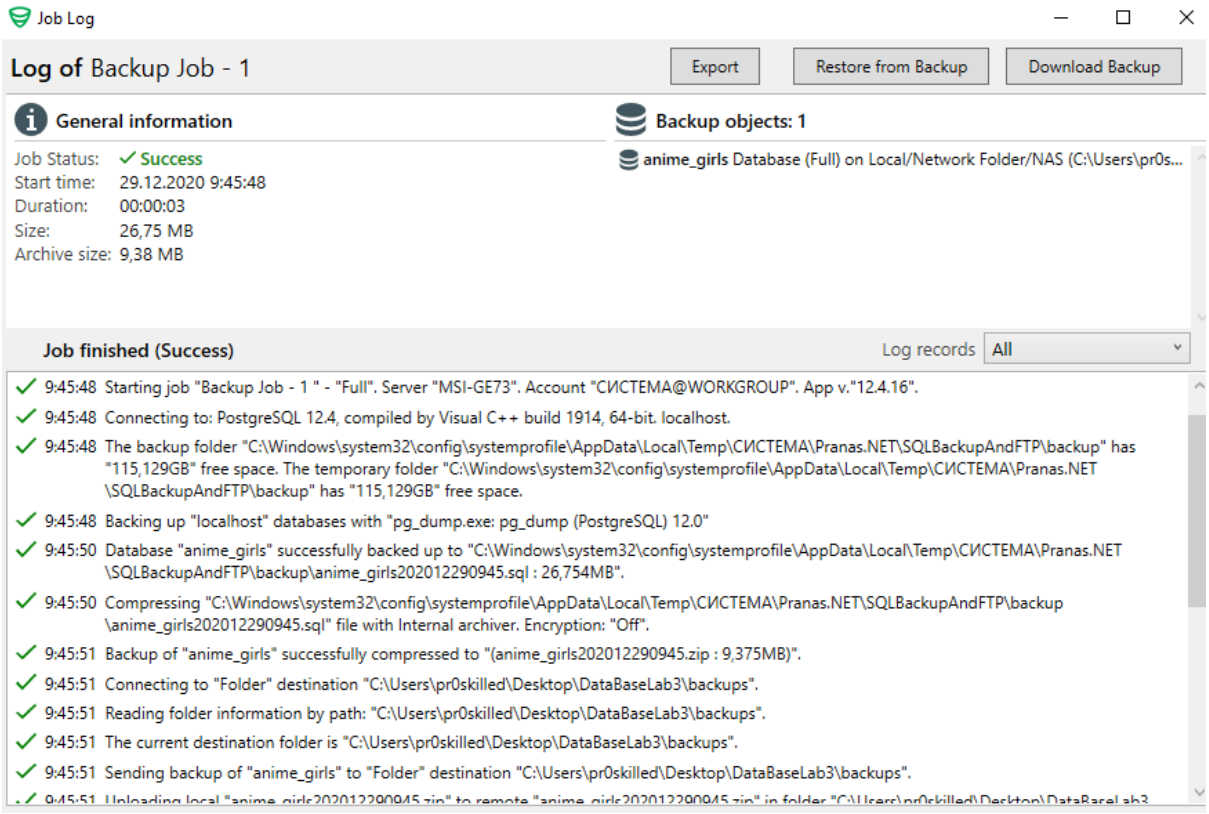
Служба серверу-репліки



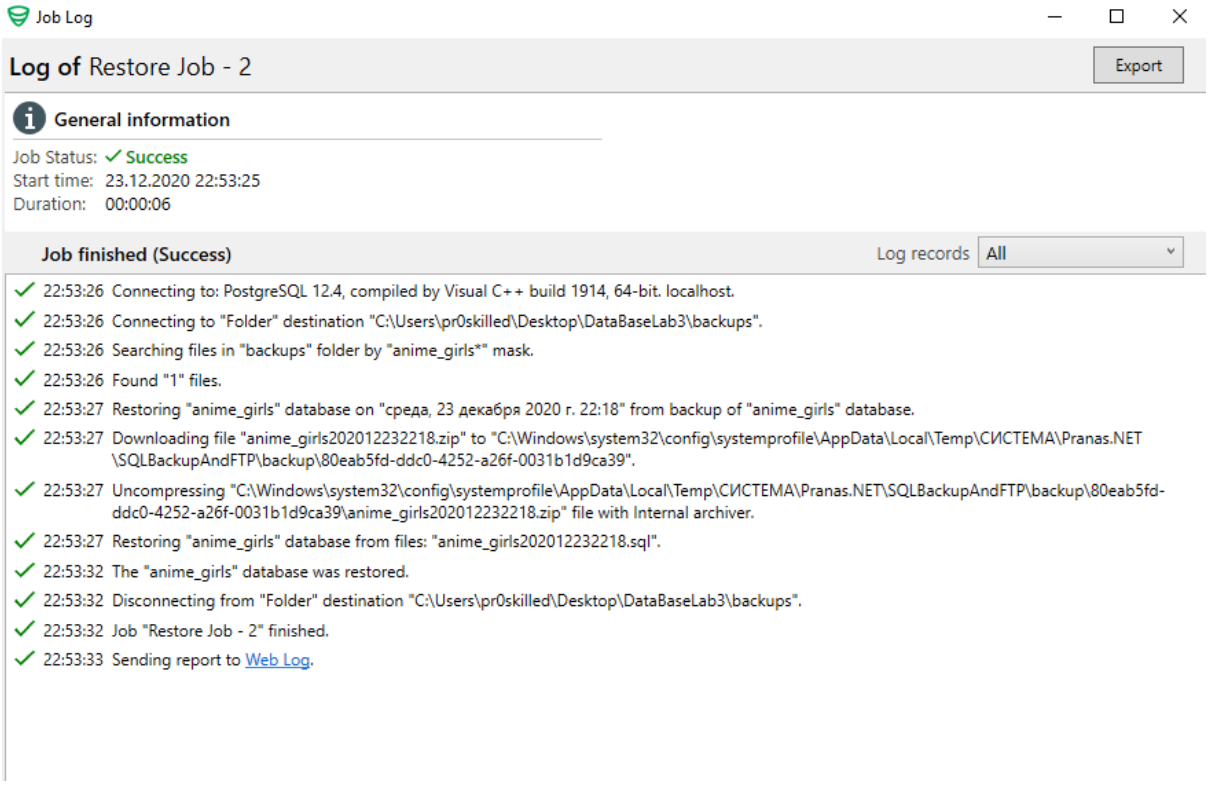
Служба автоматичного резервного копіювання



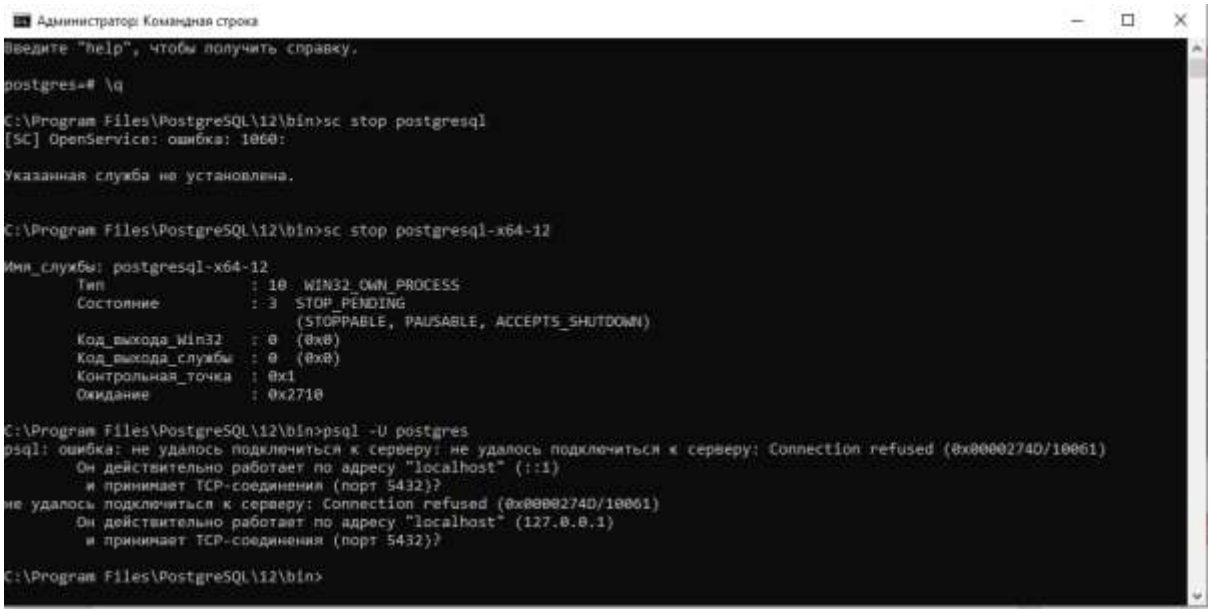
SQLBackupAndFTP



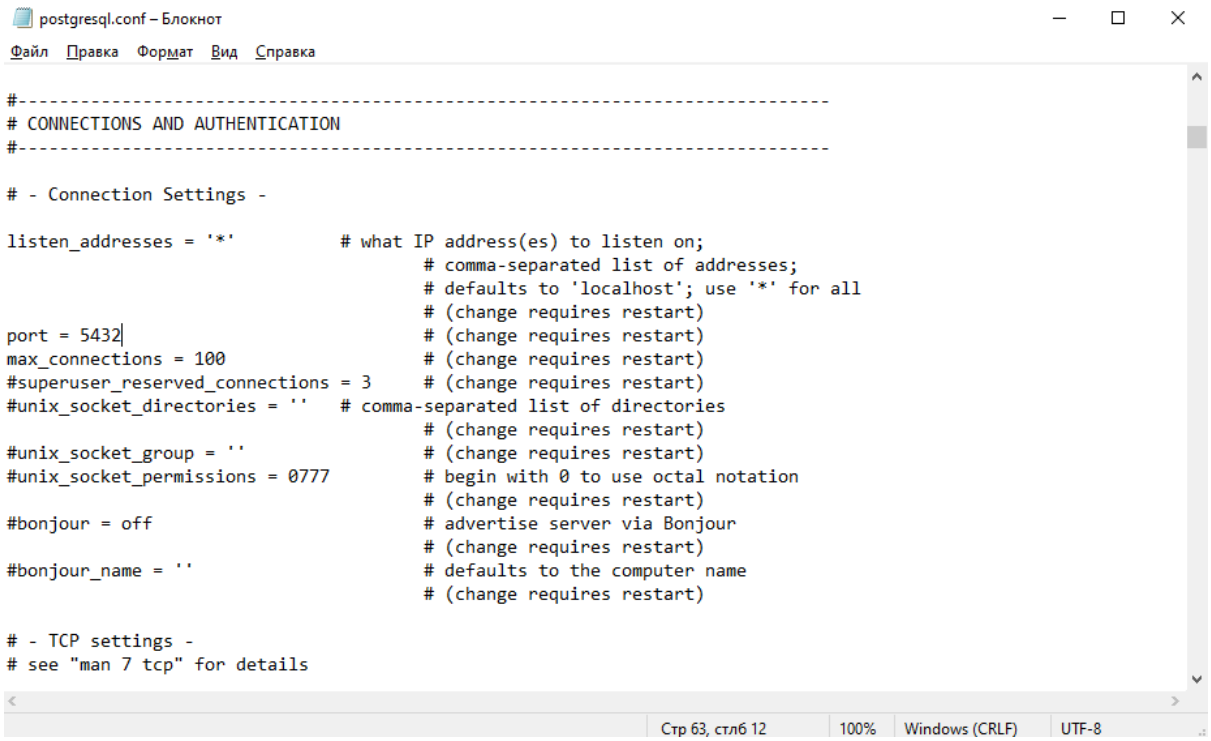
Час резервування



Час відновлення



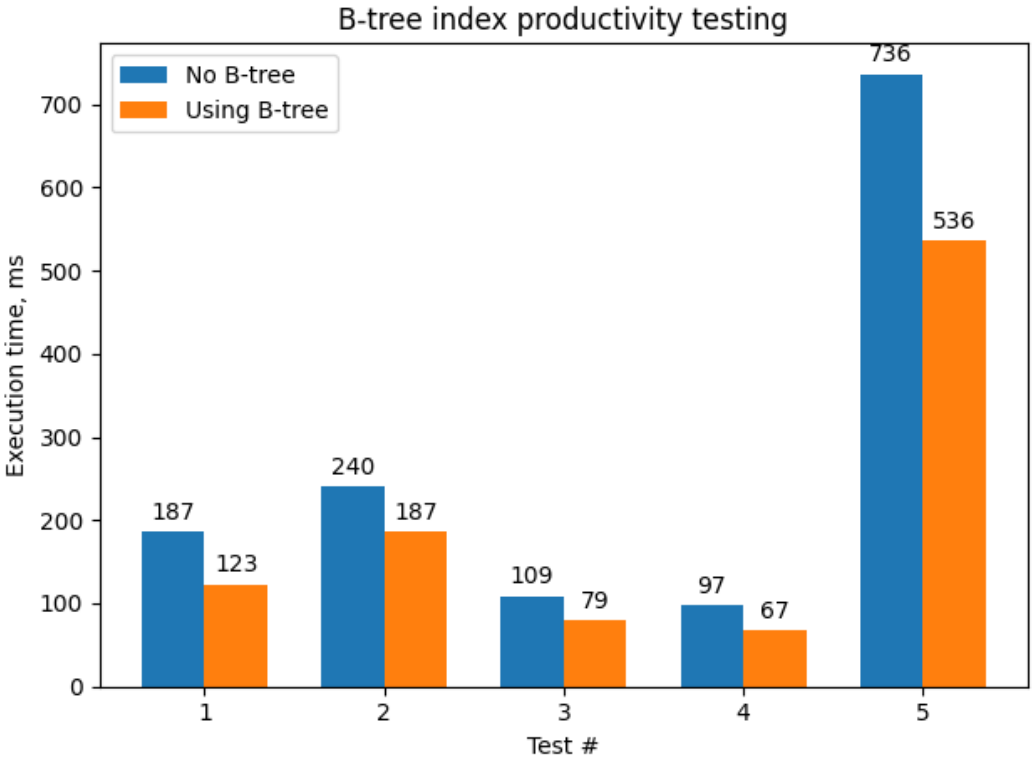
Відключаємо основний сервер



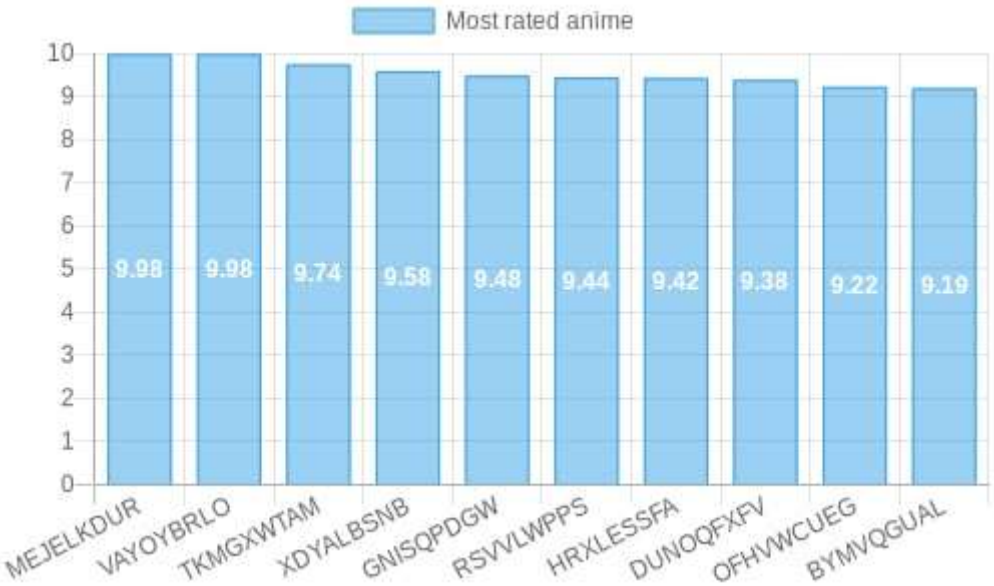
Ставимо порт 5432 на slave-сервері

```
C:\Program Files\PostgreSQL\12\bin>pg_ctl promote -D "C:\Users\pr0skilled\Desktop\A\rep"
ожидание повышения сервера.... готово
сервер повышен

C:\Program Files\PostgreSQL\12\bin>sc stop postgresql2

Имя_службы: postgresql2
        Тип                : 10  WIN32_OWN_PROCESS
        Состояние          : 3  STOP_PENDING
                               (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        Код_выхода_Win32   : 0  (0x0)
        Код_выхода_службы  : 0  (0x0)
        Контрольная_точка  : 0x1
        Ожидание           : 0x2710

C:\Program Files\PostgreSQL\12\bin>sc start postgresql2

Имя_службы: postgresql2
        Тип                : 10  WIN32_OWN_PROCESS
        Состояние          : 2  START_PENDING
                               (STOPPABLE, PAUSABLE, ACCEPTS_SHUTDOWN)
        Код_выхода_Win32   : 0  (0x0)
        Код_выхода_службы  : 0  (0x0)
        Контрольная_точка  : 0x0
        Ожидание           : 0xea60
        ID_процесса        : 4060
        Флаги              :

C:\Program Files\PostgreSQL\12\bin>psql -U postgres
Пароль пользователя postgres:
psql (12.4)
ПРЕДУПРЕЖДЕНИЕ: Кодовая страница консоли (866) отличается от основной
                страницы Windows (1251).
                8-битовые (русские) символы могут отображаться некорректно.
                Подробнее об этом смотрите документацию psql, раздел
                "Notes for Windows users".
Введите "help", чтобы получить справку.

postgres=# _
```

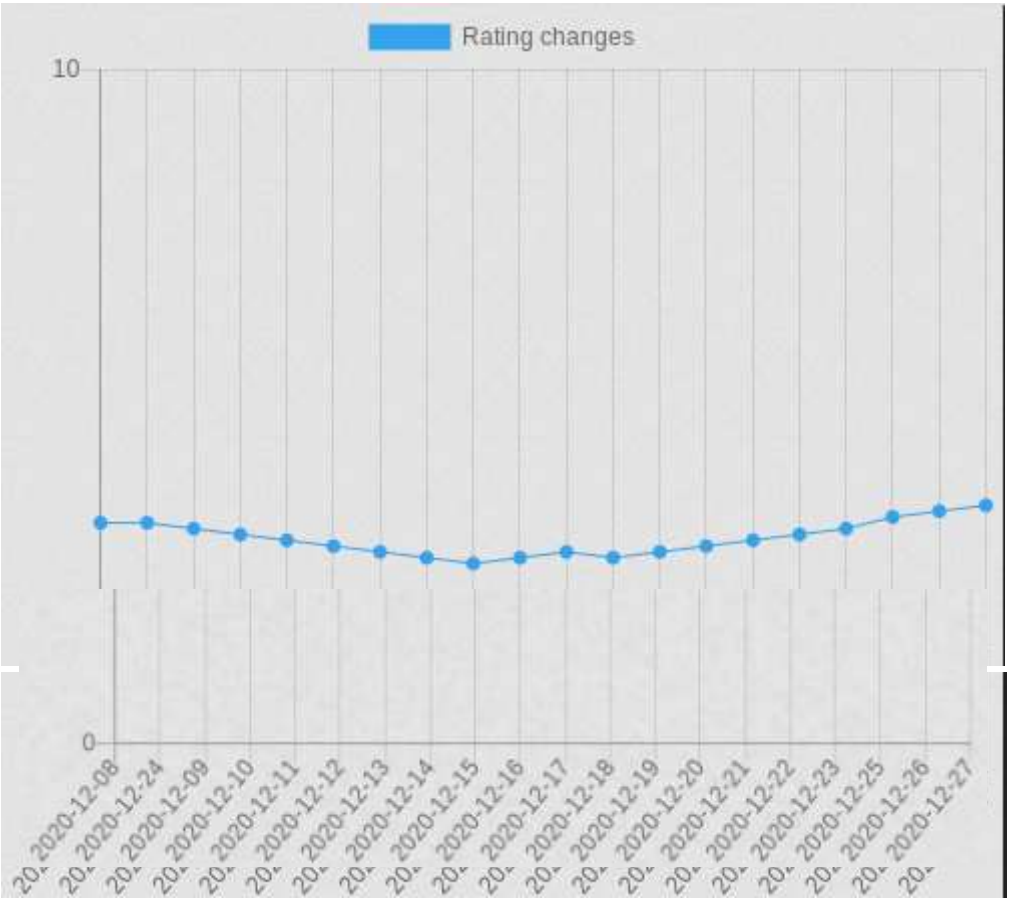
Виконуємо потрібні команди та перевіряємо можливість підключення



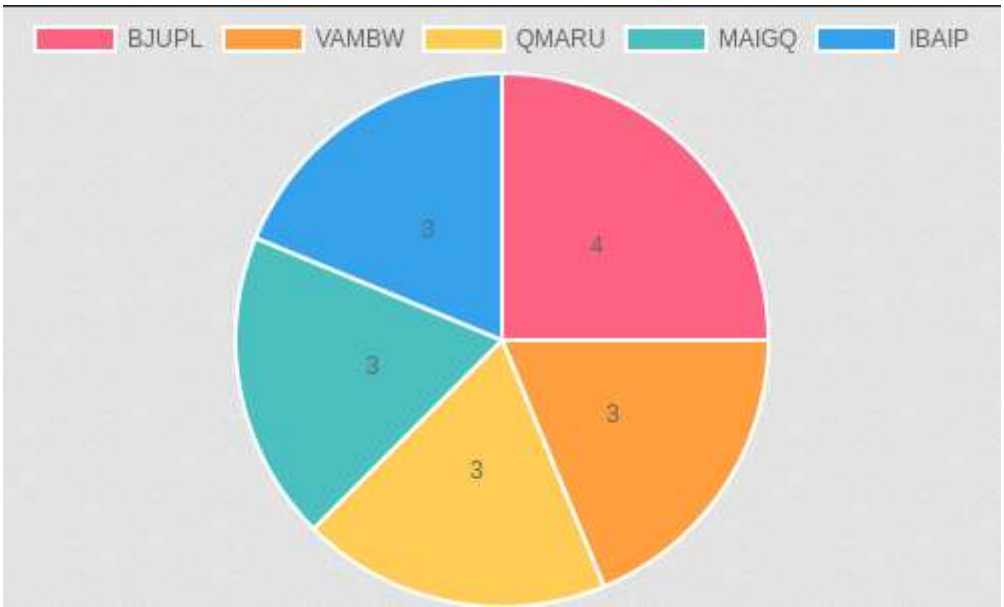
Аналіз впливу індексування на час виконання запитів



Накйращі за рейтингом аніме



Динаміка зміни рейтингу обраного аніме



Top 5 Studios with most titles per year

Б. Фрагменти програмного коду

Псевдовипадкова генерація даних для таблиці аніме

```
CREATE OR REPLACE PROCEDURE public.random_anime(IN n integer)
   LANGUAGE 'plpgsql'

AS $BODY$
declare
  counter integer := 0;
begin
while(counter < n) loop
counter:= counter + 1;
   INSERT INTO anime
      (title, year, series, rating)
   (select(chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65
+ random() * 25)::int)
                                    || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 +
random() * 25)::int)
                                    || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)) as title,
   (trunc(121 * random() + 1900)::int) as year,
            (trunc(1 + random() * 1000)::int) as series,
            (random() * 10) as rating);
            end loop;
end
$BODY$;
```

Запити для аналізу даних

```
SELECT
an.anime_id, an.title, an.series, prod.producer_id, prod.name, prod.number_of_works,
gir.girl_id, gir.fullname
FROM anime AS an
    LEFT JOIN links_anime_producers AS anprod
        ON an.anime_id = anprod.anime_id
    LEFT JOIN producers AS prod
        ON anprod.producer_id = prod.producer_id
    LEFT JOIN links_girls_anime AS ganime
        ON an.anime_id = ganime.anime_id
    LEFT JOIN girls AS gir
        ON ganime.girl_id = gir.girl_id
WHERE an.series > @minSeries
AND prod.number_of_works > @minWorks
AND gir.fullname like '%" + name + "%'

with tab1 as(
  select count(anime_id), producer_id
  from links_anime_producers
  group by producer_id
  order by count DESC LIMIT 5)
select studio, count
from tab1, producers
where tab1.producer_id = producers.producer_id
```