Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Кафедра програмної інженерії

Практична робота № 1
з дисципліни «Аналіз та рефакторинг коду»
з теми: « Основні рекомендації написання коду на мові SQL»

Виконав
Ст. гр. ПЗПІ-22-10
Клецов Микита Денисович

Перевірив
Ст. викл.
Сокорчук І. П.

2024

1. Основні положення мови SQL. Вступ.SQL (Structured Query Language) - це мова програмування, що використовується для управління та обробки даних у реляційних базах даних. SQL є стандартом для роботи з базами даних і підтримується більшістю систем керування базами даних (СКБД), такими як MySQL, PostgreSQL, Microsoft SQL Server та Oracle. Його основне призначення - маніпуляція даними (вставка, оновлення, видалення) та їх запит (вибірка).
2. **Рекомендація 1: Використовуйте зрозумілі та логічні назви таблиць і стовпців**

   Опис: Імена таблиць і полів повинні бути змістовними та відповідати даним, які вони містять.
   Переваги:

- Полегшує розуміння структури бази.
- Робить код більш читабельним і зручним для інших розробників.
- Запобігає плутанині та спрощує подальшу підтримку.

- **Наслідки неправильного підходу:**
- Ускладнене читання та розуміння коду.
- Підвищена ймовірність помилок у запитах.

Приклад:

```
//Поганий приклад
SELECT * FROM tbl1;
//Добрий приклад
SELECT * FROM Customers;
```

3.Рекомендація 2: Уникайте використання SELECT *****

Опис: Завжди вказуйте конкретні стовпці при виконанні запитів, щоб покращити їхню продуктивність і зрозумілість.

Переваги:

- Зменшує кількість переданих даних.
- Оптимізує швидкість виконання запитів.
- Захищає код від можливих змін у структурі таблиці.

- **Наслідки неправильного підходу:**
- Надмірне навантаження на систему.
- Складність у підтримці коду при змінах у базі.

Приклад:

Поганий приклад

SELECT * FROM Orders;

Добрий приклад

SELECT OrderID, CustomerName, OrderDate FROM Orders;

4. Рекомендація 3: Використовуйте індекси для оптимізації запитів

Опис:

Індекси значно пришвидшують пошук даних у таблицях, оскільки дозволяють швидше знаходити потрібні записи без перегляду всіх рядків.

Чому це важливо:

- **Збільшує швидкість виконання запитів**. Завдяки індексам база даних може знаходити необхідні дані значно швидше, ніж при повному перегляді таблиці.
- **Зменшує використання ресурсів сервера**. Оптимізація вибірки дозволяє зменшити навантаження на сервер, що особливо важливо при роботі з великими обсягами даних.

- **Підвищує масштабованість системи**. Використання індексів робить базу даних більш ефективною, що важливо для зростання кількості користувачів і запитів.

Наслідки недотримання:

- Значне уповільнення роботи бази даних.
- Високе навантаження на сервер при виконанні складних запитів.
- Погіршення продуктивності системи при зростанні кількості даних.

Приклад:

```
-- Створення індексу
CREATE INDEX idx_customer_name ON Customers (CustomerName);
```

5. Рекомендація 4: **Використовуйте транзакції для забезпечення цілісності даних**

Опис:
Транзакції дозволяють виконувати групу SQL-операцій як єдине ціле, забезпечуючи узгодженість даних навіть у разі помилки або збою.

Чому це важливо:

- **Запобігає частково виконаним змінам**. Якщо одна частина операції виконується успішно, а інша – ні, база даних може залишитися в некоректному стані.
- **Гарантує цілісність даних**. Використання транзакцій дозволяє зберігати дані в узгодженому вигляді, що особливо важливо для фінансових та банківських систем.
- **Захищає від некоректних змін**. У разі помилки зміни можуть бути скасовані без шкоди для бази даних.

Наслідки недотримання:

- Втрата або пошкодження даних через незавершені зміни.
- Неможливість відновлення попереднього стану бази у разі збою.
- Підвищений ризик логічних помилок при внесенні змін до даних.

Приклад:

```
BEGIN TRANSACTION;
UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 1;
UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 2;
COMMIT;
```

6. Рекомендація 5 **Дотримуйтеся нормалізації бази даних**

Опис:

Нормалізація – це процес організації даних у базі з метою усунення надмірного дублювання та забезпечення логічної структури.

Чому це важливо:

- **Запобігає дублюванню даних**. Дані зберігаються централізовано, що дозволяє зменшити зайве використання пам’яті.
- **Зменшує ризик логічних помилок**. Якщо інформація зберігається в одному місці, її зміна не призводить до невідповідностей у різних таблицях.
- **Полегшує оновлення даних**. Оскільки дані розподілені за логічною структурою, оновлення відбуваються швидше і без зайвих змін у пов’язаних таблицях.

Наслідки недотримання:

- Надмірне дублювання записів, що призводить до перевитрати ресурсів.
- Ускладнене управління базою через залежності між дубльованими даними.
- Підвищена складність оновлення та підтримки бази.

Приклади:

```
//Погана структура
CREATE TABLE Orders (
```

```
    OrderID INT PRIMARY KEY,
    CustomerName VARCHAR(255),
    ProductName VARCHAR(255)
);
//Оптимізована структура
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(255)
);
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(255)
);
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    ProductID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

Висновки: Використання правильних підходів до написання SQL-коду значно покращує ефективність роботи бази даних, підвищує швидкість виконання запитів і забезпечує зручність у підтримці системи. Дотримання рекомендацій допомагає уникнути помилок та зробити код більш читабельним і масштабованим.

Додаток А:

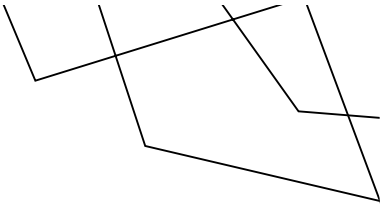

РЕКОМЕНДАЦІЇ
НАПИСАННЯ
КОДУ НА МОВІ
SQL
Клецов Микита ПЗПІ-22-10

ОСНОВНІ ПОЛОЖЕННЯ МОВИ SQL. ВСТУП.

- SQL або Structured Query Language – це мова програмування, яка використовується для управління та обробки даних у реляційних базах даних. Вона є стандартною для більшості систем керування базами даних і дозволяє виконувати операції вибірки, оновлення, вставки та видалення даних.

РЕКОМЕНДАЦІЯ 1: ВИКОРИСТОВУЙТЕ ОСМИСЛЕНІ НАЗВИ ТАБЛИЦЬ І КОЛОНОК

1. Опис: Імена таблиць і колонок повинні бути зрозумілими та відображати їх призначення. Використовуйте англійську мову та уникайте скорочень.
   Чому це важливо:
- Читабельність: Легке розуміння структури бази даних.
- Зручність підтримки: Полегшує оновлення та рефакторинг.
  Наслідки недотримання:
- Ускладнена підтримка та інтеграція з іншими системами.



ПРИКЛАД:

- Поганий приклад

```
SELECT * FROM tbl1;
```

- Добрий приклад

```
SELECT * FROM Customers;
```

РЕКОМЕНДАЦІЯ 2: УНИКАЙТЕ ВИКОРИСТАННЯ SELECT *

Опис: Завжди вказуйте конкретні стовпці при вибірці даних. Це покращує продуктивність запитів і зменшує навантаження на сервер.
Чому це важливо:

- Оптимізація продуктивності: Менше навантаження на базу даних.
- Запобігання неочікуваним змінам: Захист від змін у структурі таблиці.
  Наслідки недотримання:
- Низька продуктивність та ускладнена підтримка коду.

ПРИКЛАД:

Поганий приклад

- SELECT * FROM Orders;

Добрий приклад

- SELECT OrderID, CustomerName, OrderDate FROM Orders;

РЕКОМЕНДАЦІЯ 3: ВИКОРИСТОВУЙТЕ ІНДЕКСИ

Опис: Для пришвидшення пошуку даних у великих таблицях використовуйте індекси.
Чому це важливо:

- Збільшення швидкості запитів.
- Оптимізація роботи з великими обсягами даних.
  Наслідки недотримання:
- Повільна робота запитів.

ПРИКЛАД:

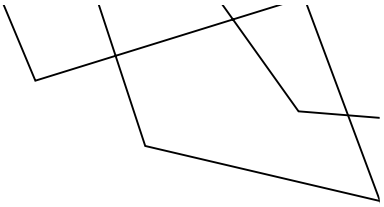
Створення індексу

- CREATE INDEX idx_customer_name ON Customers (CustomerName);

РЕКОМЕНДАЦІЯ 4: ВИКОРИСТОВУЙТЕ ТРАНЗАКЦІЇ

Опис: Транзакції забезпечують цілісність даних і дозволяють відновити базу до попереднього стану в разі помилки.
Чому це важливо:

- Захист від втрати даних.
- Контроль виконання групи операцій.
  Наслідки недотримання:
- Пошкодження даних у разі збою.



ПРИКЛАД:

- BEGIN TRANSACTION;
- UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 1;
  UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 2;
  COMMIT;

РЕКОМЕНДАЦІЯ 5: ВИКОРИСТОВУЙТЕ НОРМАЛІЗАЦІЮ БАЗИ ДАНИХ

Опис: Нормалізація допомагає уникнути дублювання даних та покращує логіку зберігання інформації.
Чому це важливо:

- Оптимізація використання пам'яті.
- Запобігання аномаліям при оновленні даних.
  Наслідки недотримання:
- Зайві витрати на зберігання та потенційні помилки при оновленні.

ПРИКЛАД:

Погана структура

- CREATE TABLE Orders (OrderID INT PRIMARY KEY, CustomerName VARCHAR(255), ProductName VARCHAR(255));

Правильна нормалізація CREATE TABLE Customers (CustomerID INT PRIMARY KEY, CustomerName VARCHAR(255));

- CREATE TABLE Products (ProductID INT PRIMARY KEY, ProductName VARCHAR(255));
- CREATE TABLE Orders (OrderID INT PRIMARY KEY, CustomerID INT, ProductID INT, FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID), FOREIGN KEY (ProductID) REFERENCES Products(ProductID));