Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _Інформаційних радіотехнологій і технічного захисту інформації_____
(повна назва)

Кафедра _____Радіотехнологій інформаційно-комунікаційних систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалавр)_____

Проектування та оптимізація бази даних для ефективного зберігання та управління інформацією про студентів з використанням PostgreSQL
(тема)

Виконав:
студент _IV_ курсу, групи ITIP-20-1_____
_____ Перцевий М. А._____
(прізвище, ініціали)

Спеціальність 126 Інформаційні системи та технології
(код і повна назва спеціальності)

Освітня програма Інформаційні технології інтернету речей
(повна назва освітньої програми)

Керівник _____ст.викл. Ганшин Д.Г._____
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри _____ ___Зрудний О.А._____
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет_Інформаційних радіо технологій і технічного захисту інформації
Кафедра_Радіотехнологій інформаційно-комунікаційних систем_____
Рівень вищої освіти_____перший (бакалавр)_____
Спеціальність_____126 Інформаційні системи та технології_____
(код і повна назва)

Освітня програма Інформаційні технології інтернету речей _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____ Перцевий Максим Андрійович_____
(прізвище, ім'я, по батькові)

1. Тема роботи Проектування та оптимізація бази даних для ефективного зберігання та управління інформацією про студентів з використанням PostgreSQL_____
_____
_____
затверджена наказом університету від 27.05.24 р. №500 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 2024 р.

3. Вихідні дані до роботи:
літературні джерела та електронні ресурси за темою кваліфікаційної роботи
_____
_____
_____
_____

4. Перелік питань, що потрібно опрацювати в роботі:
перелічити назви всіх розділів роботи від вступу до додатків (див. зміст)
Вступ. 1 ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ БАЗ ДАНИХ 2 КОНЦЕПЦІЇ ПРОЕКТУВАННЯ БД 3. РОЗРОБКА БД.Висновки. Перелік джерел посилання. Додатки

_____
_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Комп'ютерна презентація_____
_____
_____
_____

_____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	Ганшин Д.Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Види та сучасні системи керування базами даних	06.05.24- 15.05.2024	вик.
2	Характеристика реляційних СУБД	16.05.2024-20.05.2024	вик.
3	Створення БД в PostgreSQL	20.05.2024-23.05.2024	вик.
4	Висновки	24.05.2024-30.05.2024	вик.
5	Оформлення пояснювальної записки	01.06.2024-09.06.2024	вик.
6	Представлення роботи на кафедру	10.06.2024	вик.

Дата видачі завдання __06__ ___травня__2024 р.

Студент _____ _____ Перцевий М. А._____
(підпис) (прізвище, ініціали)

Керівник роботи _____ _____Ганшин Д.Г._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи магістра містить 51 сторінки тексту, 13 рисунків, 12 джерел посилання, 2 додатки.

БАЗА. ДАНІ. ІНФОРМАЦІЯ. ТАБЛИЦЯ.

Предметом дослідження є реляційна база даних.

Мета роботи – розробка бази даних відомостей про студентів.

У наслідок виконаної роботи побудовано учбову БД в PostgreSQL розглянуто системи СУБД.

Результати дослідження можуть бути використані для впровадження на кафедрі для обліку студентів та для учбових матеріалів.

ABSTRACT

The explanatory note of the master's qualification work contains 51 hundred pages of text, 13 figures, 12 references, 2 appendices.

BASIS. DATA. INFORMATION. TABLES.

The subject of research is a relational database.

The purpose of the work is to develop a database of information about students.

As a result of the work done, a training database was built in PostgreSQL and DBMS systems were considered.

The results of the study can be used for implementation at the department for student registration and for educational materials.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ, ОДИНИЦЬ І СКОРОЧЕНЬ

СУБД – системи управління базами даних;

SQL – Structured Query Language;

БД – база даних;

ACID - atomicity, consistency, isolation, durability;

MVCC – MultiVersionConcurrencyControl;

IBM – International Business Machines;

ВСТУП

Сучасний світ вимагає від інформаційних систем великої ефективності та швидкості обробки даних. Особливо важливою є ця проблематика у сферах, де великі обсяги даних обробляються одночасно та потрібно забезпечити їхню надійність та доступність. Одним з ключових інструментів для розв'язання цих завдань є системи керування базами даних (СКБД).

Метою даної дипломної роботи є дослідження та розробка бази даних для ведення обліку студентів, груп та дисциплін на кожному курсі університету. Враховуючи потреби сучасної освітньої системи, де необхідно забезпечувати ефективне управління та аналіз даних про навчальний процес, ця робота є актуальною та важливою.

Розробка бази даних для вищої школи дозволить ефективно вести облік студентів, контролювати їхній успіх та вибірку дисциплін. Крім того, така база даних забезпечить зручний доступ до інформації для викладачів та адміністрації навчального закладу, що покращить організацію навчального процесу та підвищить його якість.

Дослідження у галузі баз даних має велике значення для покращення ефективності та якості управління різноманітною інформацією. Тому, вирішення задачі створення бази даних для управління навчальним процесом є актуальним та перспективним напрямом в області інформаційних технологій.

1 ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ БАЗ ДАНИХ

1.1 Види та сучасні системи керування базами даних

Сьогодні встановлення та адміністрування баз даних - це набагато менш складний процес, ніж ще кілька років тому. Проектування та управління базою даних великою мірою автоматизовано. Програмне забезпечення, що дає змогу розв'язати це завдання - створювати базу даних, оновлювати інформацію, яка в ній зберігається, - і забезпечує швидкий і зручний доступ до неї з метою перегляду та пошуку інформації, називається системою управління базами даних.

Система керування базами даних створює на екрані комп'ютера певне середовище для роботи користувача (користувацький інтерфейс). Крім того, СУБД має певні режими роботи та систему команд. На основі СУБД створюються і функціонують інформаційні системи. Так само, системи управління базами даних - це одна з найуспішніших технологій у всій комп'ютерній галузі. Системи керування базами даних відіграють виняткову роль в організації сучасних промислових, інструментальних і дослідницьких інформаційних систем.

Характерними режимами роботи з СУБД є створення, редагування, управління та пошук у базі даних. Для роботи в кожному режимі існує своя система команд СУБД. Будь-яка робота користувача з базою даних будується у формі алгоритму, складеного з цих команд. Такі алгоритми можуть виконуватися в режимі прямого виконання (віддана команда відразу виконується) і в режимі автоматичного виконання, тобто в програмному режимі [1].

На сьогодні існує досить багато СУБД різного призначення. У сучасних інформаційних системах стандартом дефакто є реляційні СУБД, дані в яких зберігаються в таблицях. Усі ці системи мають свої відмінності. Перелічимо ключові параметри, які є важливими як для користувачів, так і для розробників

системи.

Спосіб доступу до БД:

- файл-серверні СУБД;

- клієнт-серверні СУБД;

- вбудовувані СУБД.

У файл-серверних СУБД (MicrosoftAccess, Paradox, FoxPro, dBase тощо) для застосунків відкрито спільний доступ до всіх файлів бази даних (які зазвичай зберігаються в якомусь файловому сховищі, що розділяється), і вони можуть спільно обробляти ці дані. Кожен додаток обробляє дані самостійно. На сьогодні файлсерверна технологія вважається застарілою, і її використання у великих інформаційних системах є недоліком. Суть проблеми полягає в тому, що у файл-серверних СУБД відсутні багато переваг клієнт-серверних, таких як: паралелізм запитів, кешування даних, висока продуктивність, і водночас їм притаманна низка недоліків (складнощі з відновленням, блокуванням, з підтримкою цілісності бази тощо), що зі свого боку призводить до зниження продуктивності та надійності. У файлових СУБД необхідно постійно відстежувати стан бази і час від часу здійснювати її "лікування" за допомогою вбудованих або зовнішніх утиліт [2].

У клієнт-серверних СУБД (InterBase, MicrosoftSQLServer, Firebird, Oracle, MySQL, PostgreSQL та ін.) уся обробка даних виконується в одному місці - на сервері, там же, де зазвичай зберігаються дані, при цьому доступ до файлів даних має тільки один сервер, одна система - сама система управління базами даних. При цьому додатки-клієнти лише надсилають запити до СУБД на обробку та отримання даних і отримують відповіді; безпосереднього доступу до файлів даних додатки-клієнти не мають.

Вбудовувані СУБД (MicrosoftSQLServerCompact, FirebirdEmbedded, SQLite, та ін.) постачаються у складі якого-небудь готового програмного продукту, для них потрібне самостійне встановлення. Завданням СУБД цього типу є локальне зберігання даних додатка. На колективне використання в мережі вони не розраховані. Наприклад, одна з вбудованих безкоштовних

СУБД SQLite широко використовується в мобільній ОС Android і в багатьох мобільних додатках [3].

За схемою ліцензування СУБД бувають:

- безкоштовні;

- комерційні промислові (у більшості випадків з наявністю безкоштовної обмеженої версії).

Практично всі файл-серверні та вбудовувані СУБД є безкоштовними, найбільш відомі з безкоштовних клієнт-серверних СУБД: PostgreSQL, Firebird і MySQL.

Багато виробників промислових СУБД надають можливість користуватися безкоштовними редакціями своїх продуктів, які мають обмеження за продуктивністю і за функціями на відміну від повнофункціональної версії СУБД.

До плюсів вільних СУБД можна віднести:

- вони безкоштовні;

- не вимогливі до ресурсів персонального комп'ютера;

- при правильному налаштуванні володіють широкими функціями і великою продуктивністю;

- надійні.

Мінуси: немає гарантії, що через певний час проєкт не припиниться, оскільки його підтримують у спільноті аматорів, також буде непросто знайти обізнаного фахівця для роботи з СУБД на кшталт PostgreSQL або Firebird [4].

Перевагами комерційних СУБД є:

- висока продуктивність;

- надійність;

- масштабність;

- підтримуваність;

- хороша задокументованість;

- наявність вбудованих інструментів для адміністрування та розробки.

І такі компанії як, Microsoft, IBM або Oracle, не припинять підтримку

своїх системи.

Мінусами є: вища вимогливість до ресурсів, ніж у безкоштовних аналогів і висока вартість.

В основі класифікації СУБД лежить використовувана модель баз даних, це дає змогу виділити кілька класів СУБД: мережеві, ієрархічні, реляційні, об'єктні тощо. Є й такі СУБД, які підтримують кілька моделей даних одночасно. Більш ранні СУБД, такі як мережеві та ієрархічні, мають деревоподібну структуру і принцип побудови "Предок - нащадок". Але такі системи вважаються застарілими і ними майже перестали користуватися. Їм на зміну прийшли реляційні СУБД [5].

1.2 Характеристика реляційних СУБД

Ще в 1970-х роках було отримано перші теоретичні розробки в галузі реляційних СУБД і водночас з'явилися перші прототипи реляційних СУБД. Тривалий час неможливо було домогтися ефективної реалізації таких систем. Але поступове нагромадження методів і алгоритмів організації реляційних баз даних і управління ними призвели до того, що вже в середині 80-х років ранні СУБД були витіснені зі світового ринку реляційними системами.

У реляційному підході організації СУБД передбачається наявність набору відношень (двовимірних таблиць), які є пов'язаними між собою. Зв'язок у цьому випадку - це асоціювання двох або більше відносин (таблиць). База даних, що має обмежену структуру і не має зв'язків між таблицями, не може називатися реляційною. Запити до таких баз даних повертають таблицю, яка повторно може брати участь у наступному запиті. Дані в одних таблицях пов'язані з даними інших таблиць, звідки й походить назва "реляційні" [6].

Реляційний підхід у побудові СУБД має низку переваг:

- невеликий набір абстракцій, які дають змогу легко моделювати більшу частину поширених предметних областей і допускають точні формальні визначення, залишаючись зрозумілими;

- наявність простого і водночас потужного математичного апарату, що

спирається головним чином на теорію множин і математичну логіку і забезпечує теоретичний базис реляційного підходу до організації баз даних реляційного підходу до організації баз даних;

- можливість ненавігаційного маніпулювання даними без необхідності знання конкретної фізичної організації баз даних у зовнішній пам'яті.

Реляційна модель має суворе теоретичне обґрунтування. Для визначення та управління реляційними базами даних було створено декларативну мову SQL. Інші сильні сторони реляційної моделі - простота, придатність для систем інтерактивної обробки транзакцій, забезпечення незалежності від даних. Однак реляційна модель даних і реляційна СУБД, зокрема, мають і деякі мінуси [7].

Головним мінусом реляційних СУБД вважається обмеженість використання в галузях, де досить складні структури даних. Єдність і неподільність даних - це основні аспекти традиційної реляційної моделі даних, які зберігаються на перетині рядків і стовпців таблиці. Таке правило було закладено в основу реляційної алгебри під час її розроблення як математичної моделі даних. Крім того, специфіка реалізації реляційної моделі не дає змоги адекватно відображати реальні зв'язки між об'єктами в описуваній предметній області. Ці обмеження суттєво заважають ефективній реалізації сучасних застосунків, які вимагають уже дещо інших підходів до організації даних.

Головний принцип реляційної моделі - видаляти повторювані поля і групи за допомогою процесу, який називається нормалізацією. Плоскі нормалізовані таблиці універсальні, прості в розумінні й теоретично достатні для відображення даних будь-якої предметної області. Вони добре підходять для додатків, пов'язаних зі зберіганням і відображенням даних у традиційних галузях, як-от банківські або облікові системи, але їхнє використання в системах, що ґрунтуються на складніших структурах даних, буває складним. Переважно це пов'язано з простотою механізмів зберігання даних, що лежать в основі реляційної моделі [6].

ORACLE, Informix, IBM (DB2), Sybase, Microsoft, Progress та інші - це відомі нині фірми виробники реляційних СУБД. Програми виробників СУБД розраховані на роботу на різних типах комп'ютерів (від майнфреймів до портативних) і на різних операційних системах. Також виробники СУБД

звернули увагу на програми, що працюють на настільних комп'ютерах, як-от dBase, FoxPro, Access і їм подібні. Ці СУБД призначені для роботи на PC і вирішують локальні завдання на одному PC або невеликій групі PC. Часто ці СУБД використовують, як дзеркальне відображення невеликої частини загальної корпоративної СУБД, для мінімізації необхідних апаратних і ресурсних витрат для розв'язання невеликих завдань.

Різні СУБД працюють під управлінням різних операційних систем і апаратної частини. UNIX, VAX, Solaris, Windows - ці відомі операційні системи. Залежно від обсягу зберігання даних, кількості користувачів, які здійснюють одночасний доступ до даних, складності завдань - використовуються різні СУБД на різних платформах. Для розв'язання завдань із забезпечення даними сотні тисяч користувачів, підходить СУБД Oracle на Unix - багатопроцесорний сервер. Нині найбільший інтерес становлять СУБД, спрямовані на операційну систему Windows, які використовують платформу Intel.

Поява об'єктних баз даних була визначена необхідністю розв'язувати задачі, пов'язані з обробкою і зберіганням складних багатопов'язних даних, а також слабоструктурованої та неструктурованої інформації: текстом, зображеннями, музикою і т. д. Об'єктна СУБД ідеально підходить для опису такого роду даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки застосунків.

1.3 Характеристика об'єктних СУБД

Поява об'єктних баз даних була визначена необхідністю розв'язувати задачі, пов'язані з обробкою і зберіганням складних багатопов'язних даних, а також слабоструктурованої та неструктурованої інформації: текстом, зображеннями, музикою і т. д. Об'єктна СУБД ідеально підходить для опису такого роду даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого

збільшення термінів і вартості розробки застосунків.

Об'єктна технологія - це досвід відображення того, як у дійсності людина думає про інформацію та використовує її. Теоретично, сутності реального світу описуються як об'єкти, які мають стан (що представляється поточними значеннями даних об'єктів) і поведінку (яку можна спостерігати і на яку можна впливати через програмний код об'єктів) [8].

Об'єкти, на відміну від реляційних таблиць, тісно пов'язують дані та програмний код. Об'єкт являє собою пакет, що містить значення всіх даних цього об'єкта ("властивості") і копію всіх його кодів ("методи"). Методи об'єкта надсилають повідомлення для взаємодії з іншими методами цього ж або інших об'єктів.

В об'єктній технології властивості даних не зводяться до простих "комп'ютерним" типам даних. Побудова точних і зручних моделей даних може здійснюватися, коли об'єкт містить у собі інші об'єкти або посилання на них.

Об'єктні СУБД реалізують весь набір функцій, характерних для системам керування базами даних плюс можливості об'єктного програмування. Таким чином, ми отримуємо всі переваги СУБД поряд із потужною об'єктною мовою програмування (серед них C++, Java) об'єктів бази [9].

В об'єктній технології складність структур даних знаходяться всередині самих об'єктів, а доступ до інформації здійснюється через простий різноманітний інтерфейс. Об'єктне програмування підходить для розробки складних додатків тому, що об'єкти дають змогу легко і просто моделювати комплексні дані. Аналогічно відбувається поповнення і зміна бази даних через об'єктний доступ.

Доступ до різних джерел даних, зокрема й до даних реляційних СУБД, можна здійснити за допомогою об'єктної бази даних реляційних СУБД можна здійснити за допомогою об'єктної бази даних. Інтерфейсами СУБД з об'єктними мовами програмування є C++, Java і набір ActiveX-елементів (модулів, що сприймають високорівневі команди від застосунків VisualBasic, Delphi і т. ін.), які розробник може використовувати у своїй програмі для роботи з СУБД.

Основними поняттями, з якими працює ця модель, є такі:

1) Спадкування - породжує один клас об'єктів з іншого. Новий клас (підклас) зберігає всі властивості та методи свого "батька", може мати додаткові властивості та методи, характерні тільки для нього. Множинне успадкування - підклас може мати більше одного "батька".

2) Інкапсуляція дає можливість описувати об'єкт як своєрідну "чорну скриньку". Незалежно від рівня складності або класу, може мати певну кількість загальнодоступних властивостей і методів. Додатку не обов'язково знати, як об'єкт влаштований і діє зсередини. Він взаємодіє тільки з властивостями і методами об'єкта.

3) Поліморфізм означає, що методи, які належать різним класам, можуть використовувати один і той самий інтерфейс незалежно від конкретної реалізації цих методів. При цьому кожен об'єкт здійснює метод так, як це визначено для даного класу об'єктів.

Додатково існує ще дві особливості об'єктного підходу - типізація та збереженість. Типізація захищає розробника від некоректного використання в прикладних програмах об'єктів одного класу замість іншого. Збереженість (або збереженість) дає змогу об'єкту існувати в системі після завершення процесу, це є найважливішим для концепції об'єктних СУБД [10].

Порівнюючи реляційний підхід з об'єктним, можна з'ясувати, що в реляційних БД існують тільки два принципово різних класи об'єктів:

- реляційна таблиця з кінцевим набором операцій, які допустимі для відносин (маються на увазі операції над множинами);

- вбудовані процедури, що працюють із відношеннями.

Але з цих двох класів об'єктів не можна створювати абсолютно нові типи з огляду на те, що в реляційних базах даних відсутні повноцінні механізми, характерні для об'єктного підходу.

Таким чином, можна виділити переваги об'єктних СУБД. Виділимо кілька з них. Об'єктні СУБД забезпечують інкапсуляцію логіки і даних в одному об'єкті; підтримують складні типи даних і роботу на вищому рівні абстракції,

що дає змогу, з одного боку, створювати складні структури даних, зокрема мультимедійні, а з іншого - забезпечити простоту їхнього розвитку.

Але об'єктні СУБД також мають низку недоліків і обмежень, серед яких насамперед слід виокремити відсутність розвинених засобів вибірки та аналізу даних, і єдиної методології проектування об'єктної БД.

Серед сучасних програмних продуктів-лідерів об'єктних СУБД, можна відзначити: VERSANT (Versant, Inc), ObjectStore (ObjectDesign, Inc), POET (POET Software, Inc), Jasmine (ComputerAssociates, Inc).

Велике значення для створення корпоративних інформаційних систем і різних прикладних програм, має об'єктна мультимедійна СУБД Jasmine (компанія ComputerAssociatesInternatonalInc, спільно з Fujitsu).

Популярність об'єктних СУБД нині сильно зростає, що пов'язано з широкими можливостями щодо їх застосування для побудови інформаційних систем корпоративного рівня.

Наприкінці розглянемо ще один вид СУБД. Цей вид почав з'являтися на початку 1990-х років. У той час ринок об'єктних СУБД почав швидко набирати обертів. Через що доходи компаній від продажів реляційних СУБД почали падати. Тоді ними була зроблена спроба включити деякі особливості об'єктної моделі в реляційні СУБД. Так і з'явилися гібридні реляційно-об'єктні СУБД [10].

Кілька дослідників доводили, що реляційно-об'єктне представлення даних є наступним кроком у розвитку об'єктної моделі. Під час детального знайомства з такими продуктами було з'ясовано, що такий підхід буде неповноцінним. Практика показала, що на базі реляційно-об'єктних СУБД не можна створити ефективні прикладні системи. Під час порівняння реляційного та об'єктного підходів стає зрозуміло, у чому причина провалу такого роду систем [11].

1.4 Характеристика розподілених СУБД

Ще одна класифікація СУБД будується на методах організації зберігання

та обробки даних, тому їх поділяють на централізовані та розподілені. Централізована СУБД зберігається на одному комп'ютері, але користувач працює з базою даних через віддалений доступ (у режимі клієнт-сервер). Більшість централізованих СУБД виконуючи тільки свої стандартні функції, які ускладнюються за рахунок одночасності доступу багатьох користувачів до даних, а організацію віддаленого доступу до даних залишає на мережеве забезпечення.

Розподілена СУБД - комплекс програм, призначений для управління розподіленою базою даних, що дає змогу зробити розподіленість інформації "прозорою" для кінцевого користувача. Термін "прозорість" означає те, що для кінцевого користувача має бути повністю прихований той факт, що розподілена база даних складається з декількох фрагментів, які можуть розміщуватися на декількох комп'ютерах, розташованих у мережі, і до неї можливий паралельний доступ декількох користувачів [12].

Основне призначення розподіленої СУБД полягає в забезпеченні засобів інтеграції локальних баз даних, розташованих у деяких вузлах комп'ютерної мережі, для того, щоб користувач, який працює в будь-якому вузлі мережі, мав доступ до всіх цих баз даних як до єдиної. Розподілені СУБД мають абсолютні переваги перед централізованими, а саме:

- відображають структуру організації;
- мають роздільність і локальну автономність;
- забезпечують високу доступність даних;
- мають високу надійність і підвищену продуктивність.

Розподілені СУБД мають і недоліки:

- розподілені СУБД є більш складними програмними комплексами, ніж централізовані СУБД, що зумовлено розподіленою природою використовуваних ними даних, а також реплікацією даних;
- збільшення складності означає і збільшення витрат на придбання і супровід розподілених СУБД;
- у розподілених системах потрібно організувати контроль доступу не

тільки до даних, що реплікуються на кілька різних вузлів, а і захист мережевих з'єднань самих по собі;

- у розподілених СУБД підвищена вартість передавання й оброблення даних може заважати організації ефективного захисту від порушень цілісності даних;

- відсутні стандарти на канали зв'язку та протоколи доступу до даних, а також відсутні інструментальні засоби та методології, здатні допомогти користувачам у перетворенні централізованих систем на розподілені;

- ще не накопичено необхідного досвіду промислової експлуатації розподілених систем, який можна порівняти з досвідом експлуатації централізованих систем;

- розподілені СУБД складні в управлінні, що визначає потенційну небезпеку втрати цілісності даних.

Найповніше функції розподіленої СУБД реалізовано в системах: IN-GRES/STAR, розроблена відділенням IngresDivision фірми The ASK GroupInc.; ORACLE фірми ORACLE Corp.; модулі розподіленої системи DB2 фірми IBM. Найбільш близько підійшли до реалізації функцій розподілених СУБД такі як: InformixOn-line фірми InformixSoftware; SybaseSystem 10 фірми SybaseInc.

1.5 PostgreSQL

PostgreSQL це високопродуктивна об'єктно-реляційна СУБД з відкритими вихідними текстами. Її розробка триває понад 15 років і покращується архітектура, чим вона і завоювала репутацію надійної, масштабної та інгерованої СУБД. Вона може запускатися на всіх основних платформах, включаючи Linux, Windows, MacOSX. Вона має повну відповідність з ACID, повністю підтримує ключі, об'єднання, уявлення, тригери, і збережені процедури (різними мовами). Вона має API для Java, C/C++, Perl, Ruby, Python, ODBC, Tcl, і багато інших [7].

PostgreSQL є СУБД класу підприємства і надає такі особливості, як

відновлення за точкою в часі, MultiVersionConcurrencyControl (MVCC), табличний простір, асинхронну реплікацію, вкладені транзакції (точки збереження), планувальник/оптимізатор запитів, гаряче резервування і попереджувальне журналювання на випадок збою. Він має підтримку міжнародних кодувань, зокрема й багатобайтових, при використанні різних кодувань можна розрізняти регістр, використовувати повнотекстовий пошук і сортування.

Велика кількість працюючих одночасно користувачів і велика кількість підконтрольних даних, проте, не особливо впливають на масштабність системи. Існують діючі PostgreSQL системи, які можуть керувати більш ніж 4 терабайтами даних. У таблиці 1.1 наведено деякі загальні обмеження PostgreSQL [12].

Таблиця 1.1 - Перелік обмежень СУБД PostgreSQL

Обмеження	Значення
Максимальний розмір бази даних	Необмежено
Максимальний розмір таблиці	32 TB
Максимальний розмір рядка	1.6 TB
Максимальний розмір поля	1 GB
Максимальна кількість рядків у таблиці	Необмежено
Максимальна кількість стовпців у таблиці	250 - 1600 залежно від типу стовпців
Максимальна кількість індексів у таблиці	Необмежено

У PostgreSQL можуть виконуватися збережені процедури, які написані на різних мовах програмування, включаючи C/C++, Java, Perl, Ruby і власною PL/pgSQL, яка аналогічна Oracle'sPL/SQL. В стандартній бібліотеці функцій є сотні вбудованих функцій – від базових строкових і математичних операцій до криптографічних функцій і функцій, які забезпечують сумісність з Oracle. Процедури і тригери, що зберігаються, можна писати на Сі і завантажувати в базу даних як бібліотеку, розширюючи тим самим її можливості. Також у PostgreSQL включено засоби розробки, які дають змогу створювати користувацькі типи даних разом з операторами і функціями, що описують їхню

поведінку. Як результат, можуть бути створені та додані до системи різні типи даних - від геометричних і просторових примітивів до спеціальних типів даних, визначених в ISBN/ISSN [12].

Поряд із мовами, які можуть використовуватися під час написання збережених процедур, існує і велика кількість інтерфейсних бібліотек, які дають змогу як мовам, що компілюються, так і мовам, що інтерпретуються, взаємодіяти з PostgreSQL. Це інтерфейси для ODBC, Perl, Java (JDBC), C, C++, Lisp, Scheme, PHP, Qt тощо.

Також, вихідний код PostgreSQL доступний під ліцензією BSD - найбільш ліберальною з відкритих ліцензій. Ця ліцензія дозволяє вільне використання, модифікацію та поширення PostgreSQL у будь-якій формі, із закритим або відкритим вихідним кодом. Можна чинити зі зробленими модифікаціями так, як вам буде завгодно. Таким чином, PostgreSQL є не тільки потужною системою управління базами даних, що дає змогу забезпечувати діяльність організації, а й платформою розробки для створення додатків, що потребують використання реляційної СУБД [11].

1.6 SQLServer

Microsoft SQL Server являє собою систему керування реляційними базами даних. Microsoft SQL Server розроблено компанією Microsoft. Як мова запитів використовується мова стандарту ANSI/ISOTransact-SQL з розширеннями. Система працює з базами даних великих і середніх розмірів.

Щоб вийти на ринок корпоративних баз даних, компанія Microsoft використовувала код, заснований на Sybase SQL Server. На цьому ринку баз даних конкурували такі компанії як Oracle, IBM і Sybuse. Компанії Sybase і Microsoft об'єднали зусилля, щоб випустити на ринок програму Ashton-Tate, яку назвали SQLServer 1.0 для OS/2. Вона була еквівалентом Sybase SQL Server 3.0 для Unix, VMS і багатьох інших [10].

Програму Microsoft SQL Server було випущено і включено до складу

операційної системи 1992 року, компанією Microsoft. Пізніше компанія Microsoft намагалася бути абсолютним володарем прав на всі версії SQL Server для Windows.

До моменту виходу Windows ОС у Microsoft і Sybase були власні моделі програмного продукту і власний спосіб маркетингу в здійсненні.

Після цього компанії роз'єдналися, і кожна самостійно зробила релізи програм. Sybase змінила назву продукту на AdaptiveServerEnterprise - це було пов'язано з плутаниною Microsoft SQL Server.

Програма SQL Server 7.0 володіє графічним користувацьким інтерфейсом і була першим сервером, що має такі можливості. Для вирішення спірних питань з компанією Sybase, щоб уникнути порушення авторських прав, програмний код у цій версії було переписано. А вже в листопаді 2005 року було представлено версію SQL Server 2005. Її запуск відбувався одночасно із запуском VisualStudio 2005 [11].

SQL Server - це база даних, наповнена інформацією і яка може розширюватися без втрати швидкості операцій із записами в багатокористувацькому режимі. Нові користувачі можуть додаватися шляхом модернізації обладнання. Під час проведення останнього тесту підтримувалося до 4600 користувачів бази даних.

За рахунок того, що мережева безпека пов'язана з самим сервером безпеки - дані максимально захищені від несанкціонованого доступу. Оскільки функції безпеки реалізовано на рівні користувача, їхній доступ до запису даних може бути обмеженим, захищаючи їх тим самим від пошуку або модифікації, вказавши доступ на рівні користувацьких привілеїв. До того ж, з даними, які зберігаються на окремому сервері, сервер працює як шлюз, що обмежує несанкціонований доступ [9].

SQL Server виконує обробку запитів від користувачів і надсилає їм тільки результати запиту. Таким чином, мережею передається мінімальна інформація, що прискорює час відповіді і дозволяє усувати вузькі місця в мережі. Це також робить SQL Server ідеальною базою даних для використання в інтернеті.

Технічне обслуговування SQL Server дуже просте і не потребує великих знань. Можливі зміни в структурі даних, а також резервне копіювання під час роботи сервера, без зупинки [10].

Дві основні мови розробки додатків використовуються для вилучення інформації з даних SQL Server - це C++ і VisualBasic. Ці мови є частиною VisualStudioMicrosoft. Купівля додатків, розроблених за допомогою цих продуктів, гарантує, що програмне забезпечення буде модернізуватися і розвиватися в майбутньому.

SQL Server є додатком бази даних під час роботи на новітні розробки Microsoft. Вибравши Microsoft SQL Sever як базу даних інформації для компанії, додаток може розширюватися і пристосовуватися в міру зміни бізнес-клімату.

Ця система управління базами даними насамперед вирізняється високою надійністю. Це можливо завдяки застосуванню різних базових технологій, таких як створення відмовостійких кластерів, віддзеркалення, надання різноманітних засобів для роботи з журналами.

Наступна перевага Microsoft SQL Server - це її можливості щодо масштабності та високої продуктивності. Починаючи з редакції SQL Server 2005, у ній з'явилася функція партиціонування. Ця технологія дає змогу розбивати великі таблиці на кілька елементів, які прив'язані до різних файл-груп. У результаті дані фізично розміщуються на декількох жорстких дисках, і таким чином операції читання/запису розподіляються. Ця технологія абсолютно багатозначна для призначеного для користувача додатка, і її робота здійснюється автоматично, за коштами самої СУБД [11].

Для подальшого підвищення швидкодії тут використано технологію стиснення, яка працює як на рівні записів, так і сторінок. Особливу увагу в Microsoft SQL Server приділено питанням безпеки. У системі управління базами даних реалізовано підтримку сучасних криптоалгоритмів. У редакцію SQL Server 2008 було додано засоби шифрування даних. У результаті цього дані тепер зберігаються на жорсткому диску тільки в зашифрованому вигляді, а їхнє

розшифрування відбувається "на льоту" під час читання. Ключі шифрування можуть зберігатися не тільки в самій системі управління базами даних, але також і на зовнішніх апаратних HASP-модулях [11].

Microsoft SQL Server активно застосовується сьогодні для створення корпоративних систем. Ця система управління базами даних посідає перше місце на ринку за кількістю проданих копій. Терабайтні впровадження на її основі давно перестали сприйматися як унікальні події. Останнім часом під час великих впроваджень Microsoft SQL Server все частіше підступають до петабайтного рівня.

2. КОНЦЕПЦІЇ ПРОЕКТУВАННЯ БД

2.1 Життєвий цикл БД

Як і будь-який програмний продукт, база даних має власний життєвий цикл (ЖЦБД). Головною складовою в життєвому циклі БД є створення єдиної бази даних і програм, необхідних для її роботи.

ЖЦБД містить у собі такі основні етапи:

- планування розробки бази даних;
- визначення вимог до системи;
- збір та аналіз вимог користувачів;
- проектування бази даних:
- концептуальне проектування бази даних;
- логічне проектування бази даних;
- фізичне проектування бази даних;

Розробка додатків:

- проектування транзакцій;
- проектування користувацького інтерфейсу;
- реалізація;
- завантаження даних;
- тестування;
- експлуатація та супровід:
- аналіз функціонування та підтримка вихідного варіанта БД;
- адаптація, модернізація та підтримка перероблених варіантів.

Планування розробки бази даних. Зміст цього етапу - розробка стратегічного плану, у процесі якої здійснюється попереднє планування конкретної системи управління базами даних.

Планування розробки бази даних полягає у визначенні трьох основних компонентів: обсягу робіт, ресурсів і вартості проєкту.

Важливою частиною розроблення стратегічного плану є перевірка

здійсненності проєкту, що складається з кількох частин.

Перша частина - перевірка технологічної здійсненності. Вона полягає у з'ясуванні питання, чи існує обладнання та програмне забезпечення, що задовольняє інформаційні потреби фірми.

Друга частина - перевірка операційної здійсненності - з'ясування наявності експертів і персоналу, необхідних для роботи БД.

Третя частина - перевірка економічної доцільності здійснення проекту. Під час дослідження цієї проблеми вельми важливо дати оцінку низці чинників, зокрема й таким:

- доцільність спільного використання даних різними відділами;
- величина ризику, пов'язаного з реалізацією системи бази даних;
- очікувана вигода від впровадження додатків, що підлягають створенню;
- час окупності впровадженої БД;
- вплив системи управління БД на реалізацію довгострокових планів організації.

Визначення вимог до системи. На цьому етапі необхідно визначити діапазон дії додатка бази даних, склад його користувачів і сфери застосування. Визначення вимог включає вибір цілей БД, з'ясування інформаційних потреб різних відділів і керівників фірми та вимог до обладнання і програмного забезпечення.

Збір та аналіз вимог користувачів. На цьому етапі необхідно створити для себе модель руху важливих матеріальних об'єктів і усвідомити процес документообігу. За кожним документом необхідно встановити періодичність використання, визначити дані, необхідні для виконання виділених функцій (аналізуючи наявну і плановану документацію, з'ясовують, як отримують кожен елемент даних, ким отримують, де надалі використовують, ким контролюють.

Зібрана інформація про кожну важливу сферу застосування застосунку і групи користувачів має містити такі компоненти: вихідну і генеровану документацію, докладні відомості про виконувані транзакції, а також список вимог із зазначенням їхніх пріоритетів.

Формалізація зібраної на цьому етапі інформації може бути підвищено за допомогою методів складання специфікацій вимог, до числа яких належать, наприклад, технологія структурного аналізу та проєктування, діаграми потоків даних і графіки "вхід - процес - вихід".

Повний цикл розроблення бази даних охоплює концептуальне, логічне та фізичне її проєктування.

Перша фаза процесу проектування бази даних полягає у створенні для аналізованої частини підприємства концептуальної моделі даних.

Проектування складних баз даних з великою кількістю атрибутів здійснюється з використанням так званого низхідного підходу.

Цей підхід починається з розроблення моделей даних, які містять кілька високорівневих сутностей і зв'язків, потім роботу продовжують у вигляді серії низхідних уточнень низькорівневих сутностей, зв'язків і атрибутів, що належать до них.

Низхідний підхід демонструється в концепції моделі

"сутність - зв'язок" (Entity-Relationship model - ER-модель) - найпопулярнішої технології високорівневого моделювання даних, запропонованої П. Ченом.

Модель "сутність - зв'язок" належить до семантичних моделей. Семантичне моделювання даних, пов'язане зі смисловим змістом даних, незалежно від їх подання в ЕОМ.

У побудові загальної концептуальної моделі даних виділяють низку етапів:

- виокремлення локальних уявлень, що відповідають зазвичай відносно незалежним даним. Кожне таке подання проектується як підзадача;

- формулювання сутностей, що описують локальну предметну галузь проєктованої БД, і опис атрибутів, що складають структуру кожної сутності;

- виділення ключових атрибутів;

- специфікація зв'язків між сутностями. Видалення надлишкових зв'язків;

- аналіз і додавання неключових атрибутів;

- об'єднання локальних подань.

Створена концептуальна модель даних підприємства є джерелом інформації для фази логічного проектування бази даних.

Мета другої фази проектування бази даних полягає у створенні логічної моделі даних для досліджуваної частини підприємства.

Логічна модель, що відображає особливості уявлення про функціонування підприємства одночасно багатьох типів користувачів, називається глобальною логічною моделлю даних.

Процес проектування БД має спиратися на певну модель даних (реляційна, мережева, ієрархічна), яка визначається типом передбачуваної для реалізації інформаційної системи СУБД.

Концептуальне та логічне проєктування - це ітеративні процеси, які включають у себе низку уточнень, що тривають доти, доки не буде отримано продукт, який найбільше відповідає структурі підприємства.

Фізичне проектування бази даних. Метою проектування на цьому етапі є створення опису СУБД орієнтованої моделі БД.

Дії, що виконуються на цьому етапі, занадто специфічні для різних моделей даних, тому їх складно узагальнити. Зупинимося на реляційній моделі даних. У цьому разі під фізичним проектуванням мається на увазі:

- створення опису набору реляційних таблиць і обмежень для них на основі інформації, представленої в глобальній

- логічній моделі даних;

- визначення конкретних структур зберігання даних і методів доступу до них, що забезпечують оптимальну продуктивність системи з базою даних;

- розробка засобів захисту створюваної системи.

Паралельно з проектуванням системи бази даних виконується розробка додатків. Головні складові цього процесу - це проектування транзакцій і користувацького інтерфейсу.

2.2 Моделі даних

Моделлю даних називають формалізований опис структури одиниць інформації та операцій над ними в інформаційній системі.

Модель даних - це деяка абстракція, у якій відображаються найважливіші аспекти функціонування виділеної предметної області, а другорядні - ігноруються. Модель даних містить у собі набір понять для опису даних, зв'язків між ними та обмежень, що накладаються на дані.

По суті, модель даних, підтримувана механізмами СУБД, повністю визначає безліч конкретних баз даних, що можуть бути створені засобами цієї системи, а також способи модифікації стану БД з метою відображення тих змін, що відбуваються в предметній області.

Творцем реляційної моделі є співробітник фірми IBM доктор Е. Ф. Кодд. Будучи за освітою математиком, Е. Кодд запропонував використовувати для обробки даних апарат теорії множин. У статті "A Relational Model of Data for Large Shared Data Banks", що вийшла друком 1970 року, він показав, що будь-яке представлення даних зводиться до сукупності двовимірних таблиць особливого виду, відомого в математиці як відношення (relation).

Поклавши теорію відношень в основу реляційної моделі, Е. Кодд обґрунтував реляційну замкненість відношень і низки деяких спеціальних операцій, які застосовують одразу до всієї множини рядків відношення, а не до окремого рядка. Зазначена реляційна замкнутість означає, що результатом виконання операцій над відношеннями є також відношення, над яким своєю чергою можна здійснити деяку операцію. З цього випливає, що в даній моделі можна оперувати реляційними виразами, а не тільки окремими операндами у вигляді простих імен таблиць.

Однією з основних переваг реляційної моделі є її однорідність. Усі дані розглядаються як такі, що зберігаються в таблицях і тільки в таблицях. Кожен рядок такої таблиці має один і той самий формат.

Реляційна база даних - це кінцевий (обмежений) набір відносин. Відносини використовуються для представлення сутностей, а також для представлення зв'язків між сутностями. Відношення - це двовимірна таблиця, що має унікальне ім'я і складається з рядків і стовпців, де рядки відповідають записам, а стовпці - атрибутам. Кожен рядок у таблиці представляє деякий об'єкт реального світу або співвідношення між об'єктами.

Атрибут - це пойменований стовпець відношення. Властивості сутності, її характеристики визначаються значеннями атрибутів. Порядок слідування атрибутів не впливає на саме відношення.

Нехай є відношення r. Схемою відношення r називається скінченна множина імен атрибутів R = {A1, A2, ... Ап}. Заголовки стовпців відношення містять імена його атрибутів і, отже, всі разом відображають його схему.

Схему відношення СТУДЕНТ можна подати так таким чином: {Ф.І.О, Група, Номер телефону} Рисунок 2.1.

Ф.І.О.	Група	Номер телефона
Шевченко Т.Г.	ІТІР-20-1	95546525
Сковорода Г.С	ІТІР-20-1	66789635
Косач Л.П.	ІТІР-20-1	93587452

Рисунок 2.1 – Схема відношення

Стосунки будуються з урахуванням низки чинників. Кожному імені атрибута Ai, 1 >= i <= n ставиться у відповідність множина допустимих для відповідного стовпчика значень. Цю множину Di називають доменом даного імені атрибута.

Кожен рядок відношення є множиною значень, узятих по одному з домену кожного імені атрибута. Домени є довільними непорожніми скінченними або лічильними множинами й утворюють множину:

D = D1 ∪ D1 ∪ D2 ∪ .... ∪ Dn.

Відношення r зі схемою R - це скінченна множина відображень {t1, t2, ... tp) з R у D. Причому кожне відображення t∈ r має задовольняти такому обмеженню:

Ці відображення називаються кортежами. Кожен кортеж відношення відображає екземпляр сутності, а атрибут відношення відображає атрибут сутності.

Множина кортежів називається тілом відношення. Тіло відношення відображає стан сутності, тому в часі воно постійно змінюється. Тіло відношення характеризується кардинальним числом, яке дорівнює кількості кортежів, що містяться в ньому.

Однією з головних характеристик відношення є його ступінь. Ступінь відношення визначається кількістю атрибутів, яка в ньому присутня. Ця характеристика відношення має ще назви: ранг і арність. Відношення з одним атрибутом називається унарним, з двома атрибутами - бінарним, з трьома - тернарним, з n атрибутами n-арним. Визначення ступеня відношення здійснюється за заголовком відношення.

Усі названі характеристики відносин позначені на рис. 2.2

Рисунок 2.2 - Характеристики відносин

Відношення за структурою подібне до таблиці, але таблиці, що має певні властивості. Зведемо воєдино всі властивості відношення:

- відношення має ім'я, яке відрізняється від імен усіх інших відносин;

- відношення подається у вигляді табличної структури;

- кожен атрибут має унікальне ім'я, його значення беруться з одного й того самого домену;

- кожен компонент кортежу є простим, атомарним значенням, що не складається з групи значень;

- упорядкування атрибутів теоретично несуттєве, однак воно може впливати на ефективність доступу до кортежів;

- усі рядки (кортежі) мають бути різними;

- теоретично порядок слідування кортежів не має значення.

У відношенні можуть існувати кілька одиночних або складових атрибутів, які однозначно ідентифікують кортеж відношення. Це - потенційні ключі.

Відношення може мати кілька потенційних ключів. Ключ, що містить два і більше атрибутів, називається складеним ключем. Кожне відношення має хоча б один можливий ключ, оскільки у відношенні не може бути однакових кортежів, а це означає, що щонайменше комбінація всіх його атрибутів задовольняє умову унікальності. Потенційні ключі, даючи змогу гарантовано виділити точно один кортеж, забезпечують основний механізм адресації на рівні кортежів реляційної моделі.

Один із можливих ключів (обраний довільним чином) приймається за його первинний ключ. Зазвичай первинним ключем призначається той можливий ключ, яким найпростіше користуватися під час повсякденної роботи. Решта можливих ключів, якщо вони є, називаються альтернативними ключами. Для індикації зв'язку між відносинами використовують зовнішні ключі.

Зовнішній ключ - це набір атрибутів одного відношення, що є потенційним ключем іншого відношення.

Завдяки наявності зв'язок між потенційними та зовнішніми ключами забезпечується взаємозв'язок кортежів певних відносин. Відношення, що містить зовнішній ключ, називається дочірнім або відношенням, на яке посилаються. А відношення, що містить пов'язаний із зовнішнім ключем потенційний ключ, - батьківським або цільовим відношенням.

Відносини не можна розглядати як статичні об'єкти, оскільки вони призначені для відображення деякої частини реального світу, а ця частина реального світу може змінюватися в часі. Тому і відносини змінюються в часі: кортежі можуть додаватися, видалятися або модифікуватися. Проте,

передбачається, що сама схема відношення є інваріантною в часі. Відношення слід сприймати як множину можливих станів, яких може набувати відношення.

Нехай розглядається концептуальна модель, наведена на рис.2.3. Приклад належить до предметної області, яку можна назвати "Викладацька діяльність". Ця модель містить дві сутності: СТУДЕНТ і ПРЕДМЕТ, між якими встановлено зв'язок СЛУХАЄ типу "багато до багатьох". Характеристики сутностей подано зображеними на рисунку атрибутами. Зв'язок СЛУХАЄ не має власних атрибутів.

Рисунок 2.3 - Концептуальна модель

На рис. 2.4 подано відповідні відносини, заповнені кортежами.

Код стд.	Ф.І.О.	Група	Номер тел.
1	Шевченко	ІТІР-20-1	56204786
2	Гусін	ІТІР-20-1	65489635
3	Ребров	ІТІР-20-1	49625873
4	Зінченко	ІТІР-20-1	45385453

Код предмета	Назва	Кількість годин
1	Інформатика	20
2	Бази даних	24
3	ООП	18
4	C++	40
5	IoT	55
6	Схемотехніка	25

Код предмета	Код стд.
1	2
2	2
3	4
4	1
5	3
6	1

Рисунок 2.4 - Реляційні відносини моделі

Усі наведені відносини є нормалізованими, оскільки атрибути всіх відносин мають атомарні значення. У всіх трьох відношеннях відсутні дублюючі кортежі.

Підтримка цілісності бази даних реалізується за допомогою низки обмежень, що накладаються на дані.

Перший тип обмежень виникає з того факту, що кожен атрибут визначається на своєму домені, або навпаки: домен атрибута задає множину значень, які може приймати атрибут. Зазначене обмеження називається обмеженням атрибута.

Важливими поняттями в теорії реляційних баз даних є категорна цілісність і цілісність на рівні посилань.

Категорна цілісність обмежує набір значень первинних ключів базових відносин. Такого роду цілісність полягає в такому: кортеж не може записуватися в БД доти, доки значення його ключових атрибутів не будуть повністю визначені. Іншими словами: жоден ключовий атрибут будь-якого кортежа відношення не може містити відсутнього значення, позначуваного визначником NULL.

Цілісність на рівні посилань. Під час побудови відносин для зв'язування рядків однієї таблиці з рядками іншої таблиці використовуються зовнішні ключі. База даних, у якій усі непорожні зовнішні ключі посилаються на поточні значення ключів іншого відношення, має цілісність на рівні посилань.

Цілісність на рівні посилань. Під час побудови відносин для зв'язування рядків однієї таблиці з рядками іншої таблиці використовуються зовнішні ключі. База даних, у якій усі непорожні зовнішні ключі посилаються на поточні значення ключів іншого відношення, має цілісність на рівні посилань.

Як правило, визначення умови цілісності даних здійснюють під час установлення взаємозв'язку між батьківським і дочірнім відношеннями; при цьому користувачеві часто надають можливість самому вирішити, яким шляхом зберігати цілісність бази даних і наскільки жорстко потрібно дотримуватися умови цілісності під час різних операціях зміни даних.

Так, наприклад, під час зміни значення первинного ключа в батьківському відношенні часто дозволені такі варіанти дій, з яких два перших - забезпечують знаходження бази даних у цілісному несуперечливому стані.

2.3 Керування реляційною базою даних

Для управління реляційною базою даних Е. Ф. Кодд ввів у систему Ф. Кодд запровадив реляційні мови опрацювання даних - реляційну алгебру та реляційне числення.

Реляційна алгебра - це процедурна мова обробки реляційних таблиць. Це означає, що в реляційній алгебрі використовується покроковий підхід до створення реляційних таблиць, що містять відповіді на запити.

Реляційне числення - непроцедурна мова. У реляційному обчисленні запит створюється шляхом визначення таблиці запиту за один крок.

Кодд показав логічну еквівалентність реляційної алгебри та реляційного числення. Це означає, що будь-який запит, який можна сформулювати за допомогою реляційного числення, також можна сформулювати, користуючись реляційною алгеброю, і навпаки.

І реляційна алгебра, і реляційне числення в тому вигляді, як їх сформулював Кодд, є теоретичними мовами.

Раніше було визначено основні операції з оновлення інформації в реляційній базі даних. Дані операції оновлення - це операції не над відношеннями, а над кортежами відношення. Оператори реляційної алгебри використовують одне або два з наявних відношень для створення нового відношення. Реляційна алгебра (або алгебра відношень) являє собою сукупність операцій високого рівня над відношеннями. Реляційна алгебра визначає такі операції:

- об'єднання;
- різниця;
- добуток;
- перетин;
- проекція;
- вибір;

- з'єднання;

- поділ.

Перші чотири операції взяті Коддом із математичної теорії множин і практично збігаються з операціями теорії множин. Наступні чотири - нові операції, що стосуються тільки креляційної моделі даних.

3. РОЗРОБКА БД

3.1 Створення БД в PostgreSQL

База даних буде містити 1 таблицю зі студентами, 1 таблицю з дисциплінами та таблицю яка виводить інформацію який студент що вивчає. В подальшому цю БД можна легко розширити та додати інші групи та курси

Створюємо таблицю зі студентами та наповнюємо її (рисунок 3.1).

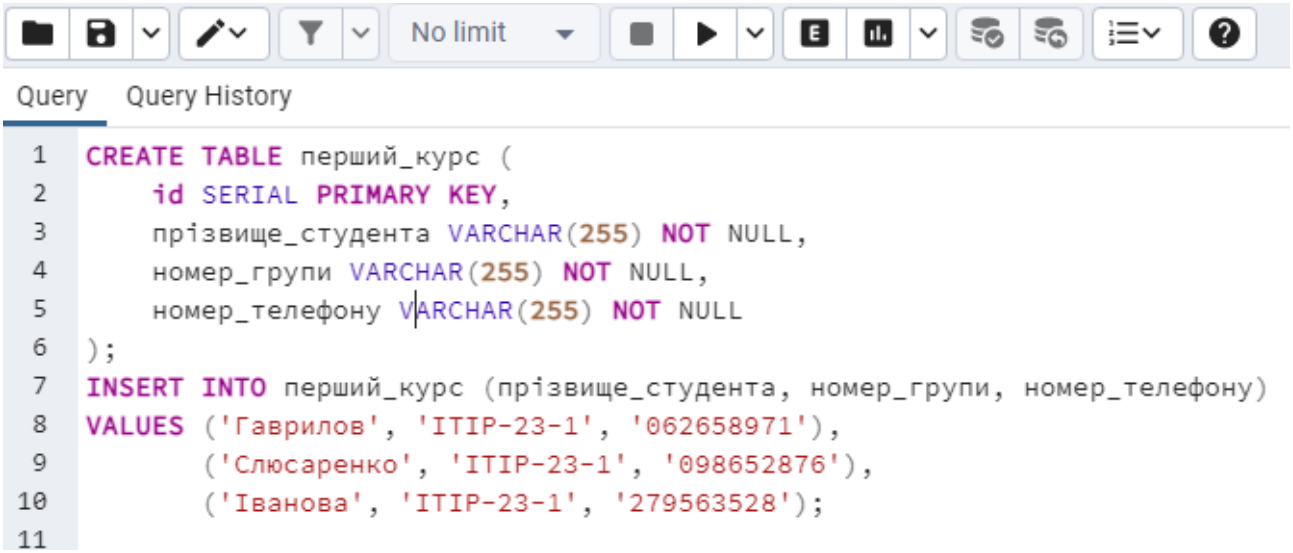

Рисунок 3.1 – Створення на наповнення таблиці перший_курс

Далі створюємо та наповнюємо таблицю з дисциплінами (рисунок 3.2).

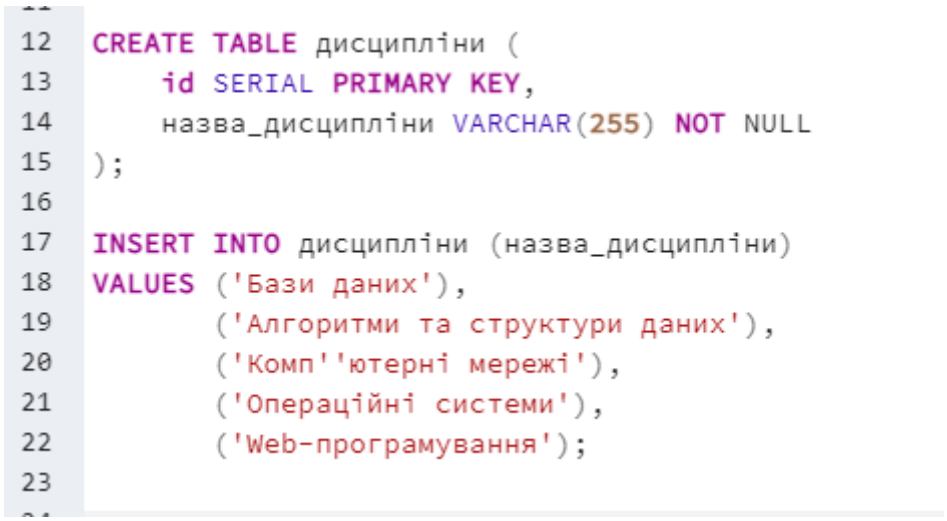

Рисунок 3.2 - Створення на наповнення таблиці дисципліни

Для подальшого виводу інформації хто що вивчає нам потрібно Створити таблицю для зберігання відношення багато до багатьох між студентами і дисциплінами (рисунок 3.3).

```
        -- Створення таблиці для зберігання відношення багато до багатьох між
CREATE TABLE студенти_дисципліни (
    id SERIAL PRIMARY KEY,
    id_студента INTEGER REFERENCES перший_курс(id),
    id_дисципліни INTEGER REFERENCES дисципліни(id)
);

-- Додавання зв'язків між студентами та дисциплінами
INSERT INTO студенти_дисципліни (id_студента, id_дисципліни)
VALUES (1, 1), (1, 4),  -- Гаврилов вивчає Бази даних та Операційні системи
       (2, 2), (2, 1),  -- Слюсаренко вивчає Алгоритми та структури даних та
       (3, 5), (3, 2); -- Іванова вивчає Web-програмування та Алгоритми та ст
```

Рисунок 3.3 - Таблиця для зберігання відношення багато до багатьох

З розробленою базою даних можна виконувати різноманітні запити для отримання корисної інформації. Ось деякі приклади операцій, які можна виконати з вашою БД:

Отримання списку студентів та їх дисциплін: Ви можете використовувати JOIN для отримання повної інформації про студентів та дисципліни, які вони вивчають.

Фільтрація результатів: Використовуйте WHERE для фільтрації результатів запитів. Наприклад, ви можете вибрати лише тих студентів, які вивчають певну дисципліну.

Оновлення записів: Ви можете використовувати UPDATE, щоб оновити інформацію про студентів або дисципліни.

Видалення записів: Використовуйте DELETE, щоб видалити студентів або дисципліни з бази даних.

Додавання нових записів: Ви можете використовувати INSERT, щоб додати нових студентів або дисципліни до бази даних.

Агрегація даних: Використовуйте функції агрегації, такі як COUNT, SUM, AVG для отримання статистичної інформації про ваші дані.

Сортування результатів: Використовуйте ORDER BY, щоб відсортувати

результати вашого запиту за певними критеріями, наприклад, за іменем студента або назвою дисципліни.

Групування даних: Використовуйте GROUP BY, щоб згрупувати результати вашого запиту за певними критеріями, наприклад, за номером групи студентів.

Ці операції дозволяють вам ефективно керувати та використовувати вашу базу даних для отримання потрібної інформації.

3.2 Управління БД

Отримання списку студентів та їх дисциплін. Цей запит використовує JOIN, щоб об'єднати таблиці студенти_дисципліни, перший_курс і дисципліни за їх зовнішніми ключами, тобто id_студента та id_дисципліни. INNER JOIN вибирає лише ті рядки, які мають відповідність у обох таблицях (рисунок 3.4).

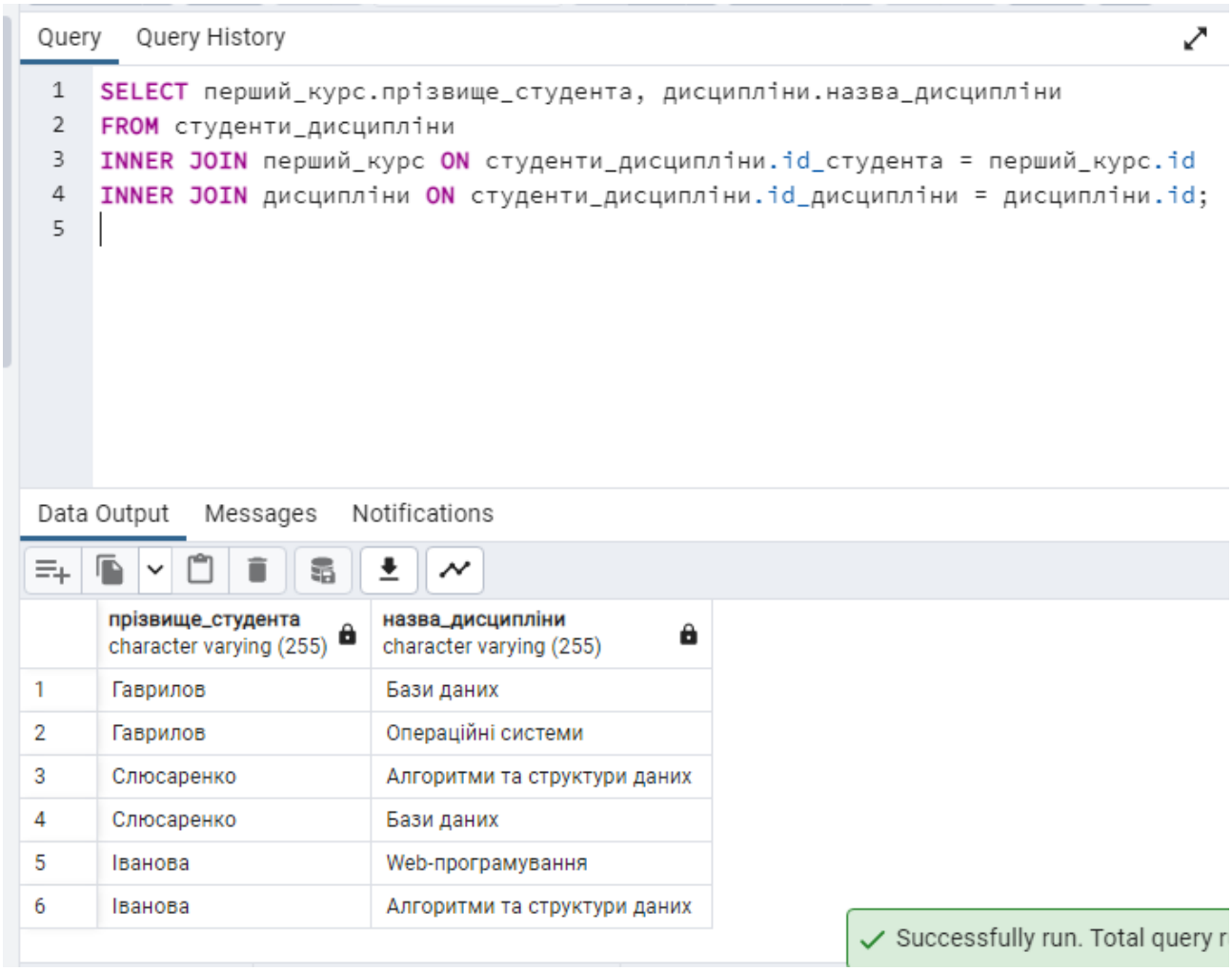

Рисунок 3.4 - Отримання списку студентів та їх дисциплін

Фільтрація результатів. WHERE використовується для фільтрації рядків у результаті запиту. У цьому випадку ми вибираємо тільки ті рядки, де id_студента дорівнює 1, тобто лише дані для студента з id рівним 1 (рисунок 3.5).

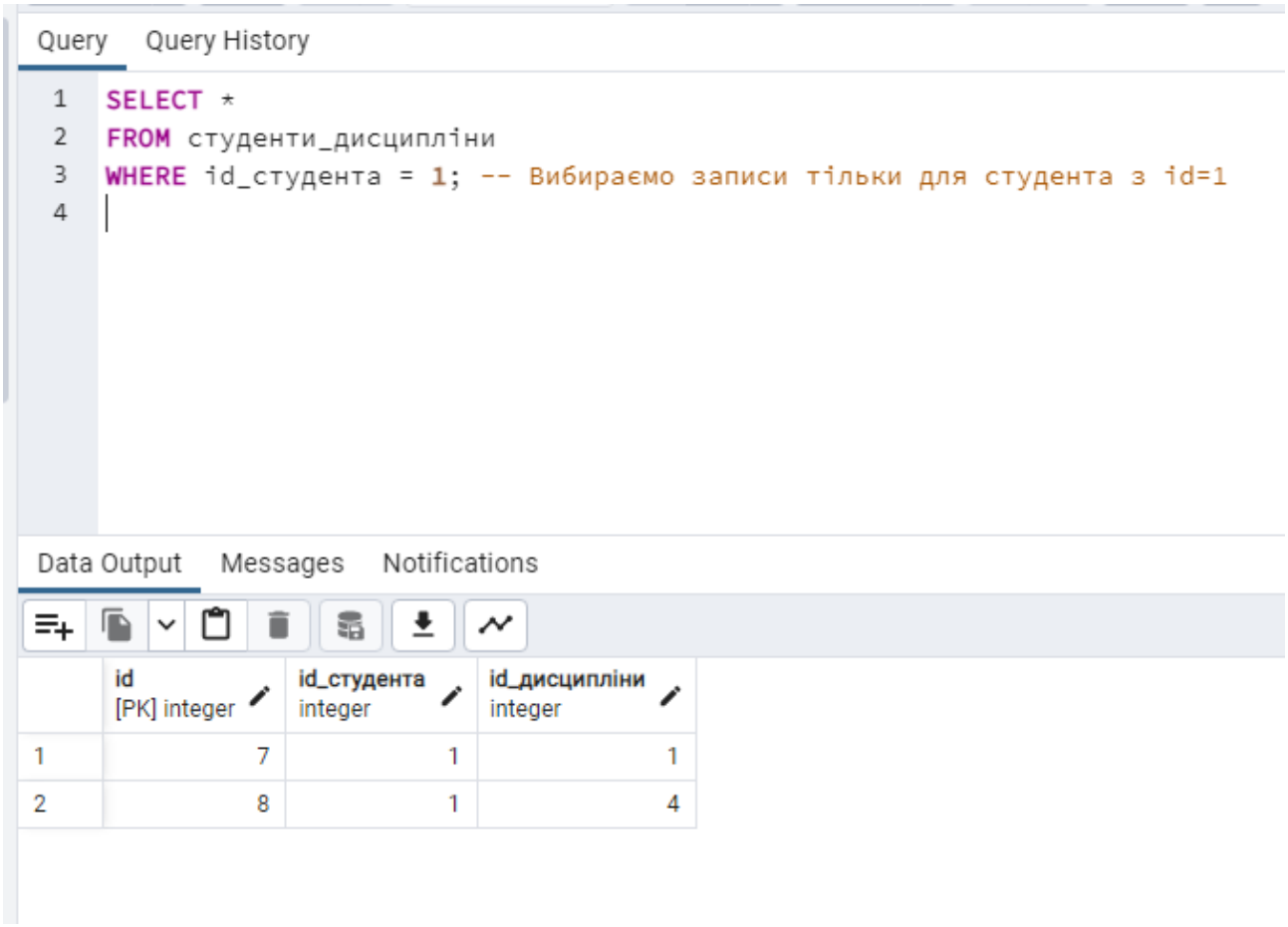

Рисунок 3.5 - Фільтрація результатів

UPDATE використовується для оновлення вже існуючих рядків у таблиці. У цьому випадку ми змінюємо значення поля номер_телефону для студента з id рівним 1 на нове значення (рисунок 3.6).

Видалення записів. DELETE використовується для видалення рядків з таблиці. У цьому випадку ми видаляємо студента з id рівним 1 з таблиці перший_курс.

DELETE FROM перший_курс

WHERE id = 1; -- Видаляємо студента з id=1

Додавання нових записів. INSERT INTO використовується для додавання нових рядків у таблицю. У цьому випадку ми додаємо нового студента з

вказаними значеннями для кожного поля (рисунок 3.7).

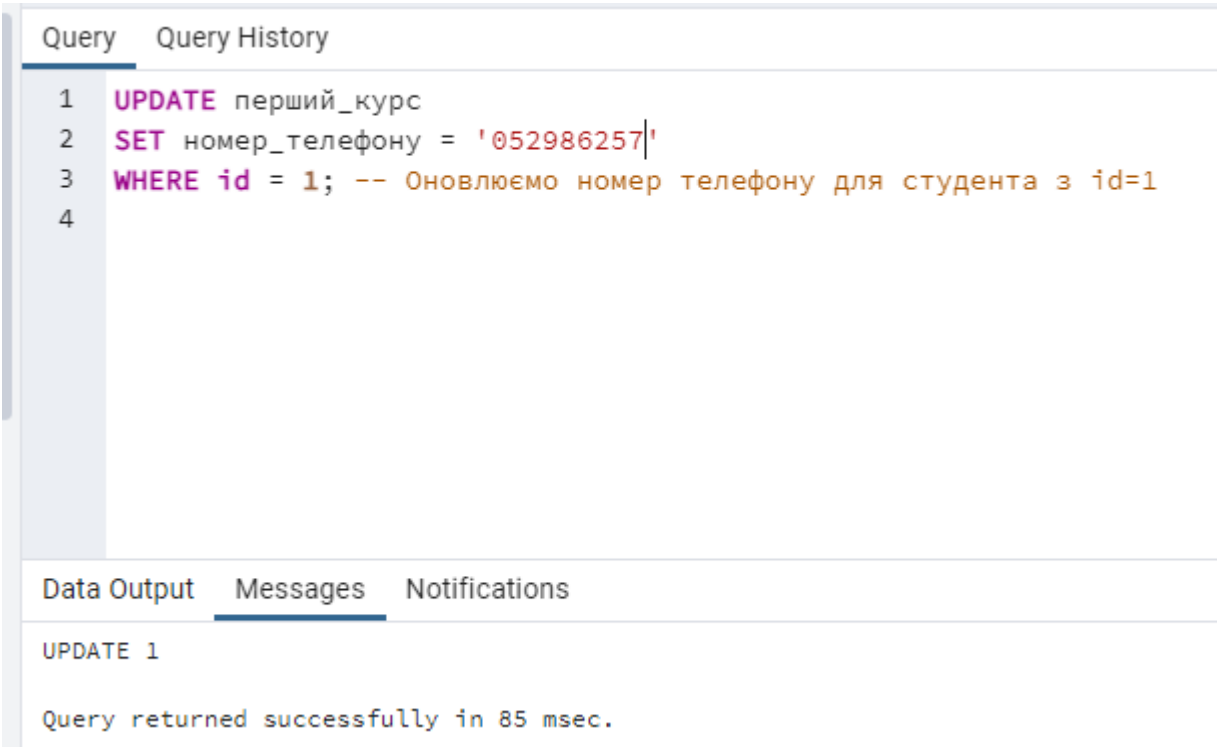

Рисунок 3.6 – UPDATE

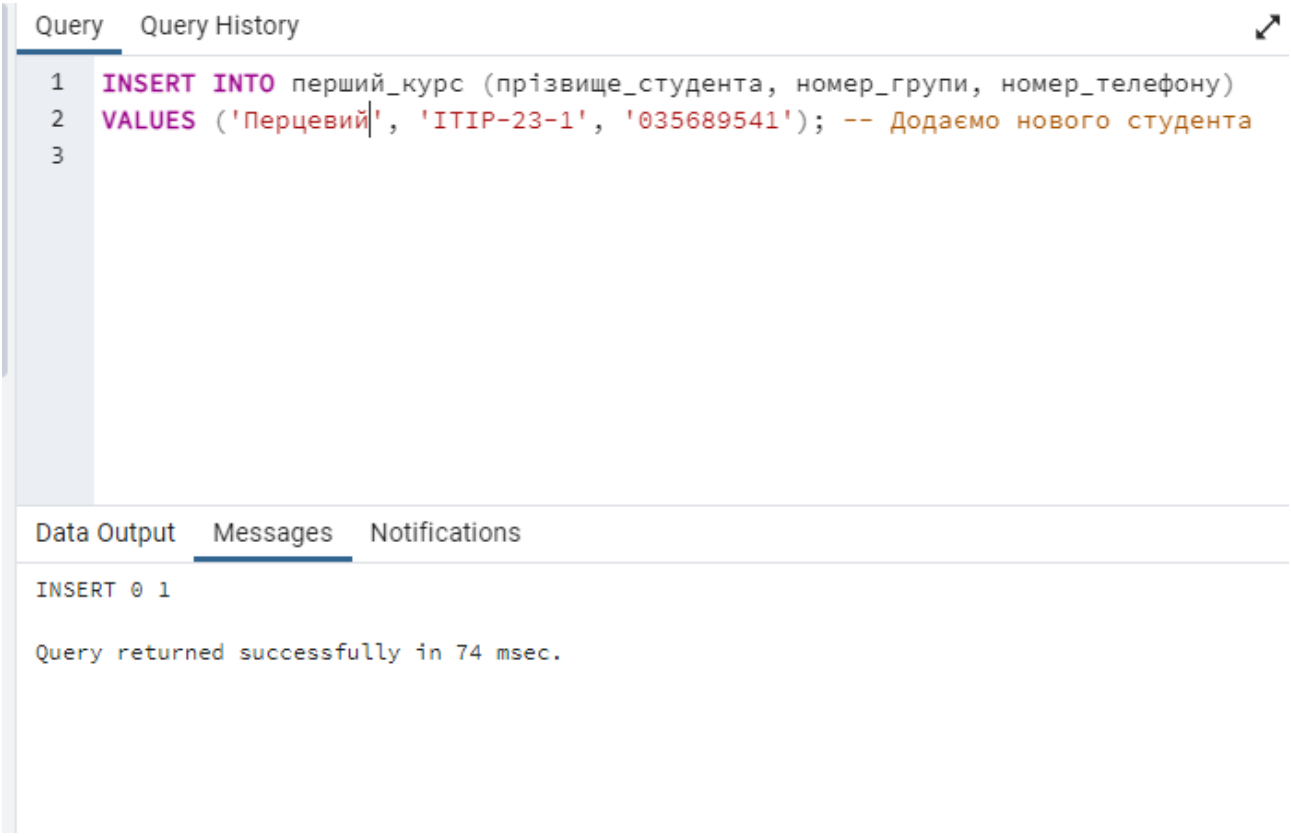

Рисунок 3.7 - Додавання нових записів

Агрегація даних: Функція COUNT(*) використовується для підрахунку кількості рядків у таблиці. У цьому випадку ми підраховуємо кількість студентів у таблиці перший_курс. (рисунок 3.8).

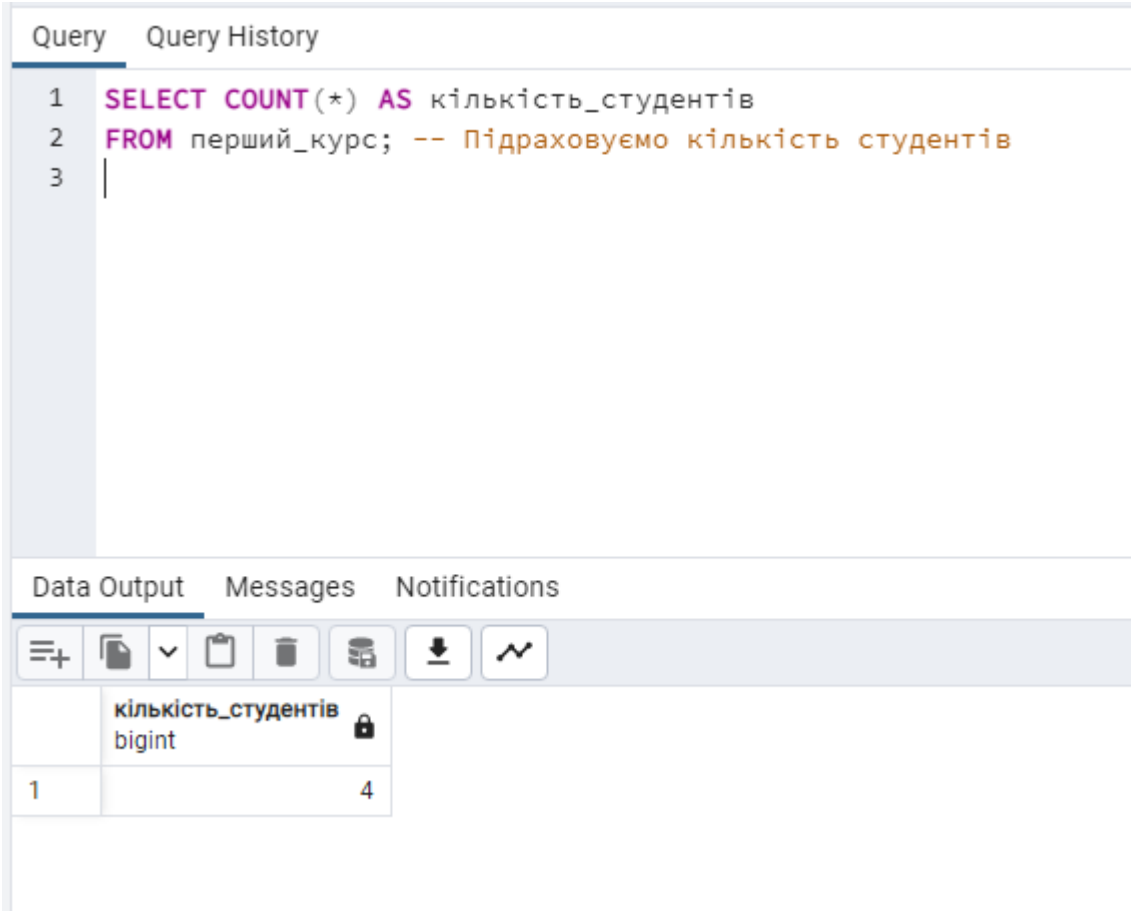

Рисунок 3.8 - Агрегація даних

Сортування результатів. Використовуйте ORDER BY, щоб відсортувати результати вашого запиту за певними критеріями, наприклад, за іменем студента або назвою дисципліни/

```
SELECT *
FROM перший_курс
ORDER BY прізвище_студента ASC; -- Сортуємо студентів за алфавітом.
```

Додамо ще одну таблицю та виведемо інформацію (рисунок 3.9)

```
INSERT INTO четвертий_курс (прізвище_студента, номер_групи, номер_телефону)
VALUES ('Петров', 'ІТІР-20-1', '0991234567'),
    ('Сидоров', 'ІТІР-20-1', '0987654321'),
```

('Іваненко', 'ITIP-20-1', '0972468135');

Query Query History

```
1  SELECT *
2  FROM четвертий_курс
```

Data Output Messages Notifications

	id [PK] integer	прізвище_студента character varying (255)	номер_групи character varying (255)	номер_телефону character varying (255)
1	1	Петров	ITIP-20-1	0991234567
2	2	Сидоров	ITIP-20-1	0987654321
3	3	Іваненко	ITIP-20-1	0972468135

Рисунок 3.9 – Додавання нової таблиці

В процесі нашої роботи ми створили базу даних для курсу з чотирьох років, що включає таблиці для студентів, дисциплін та зв'язків між ними. Кожен студент має прізвище, номер групи та номер телефону, що зберігаються в таблиці перший_курс. Для кожного курсу також визначено дві дисципліни, які вивчають студенти, та створено таблицю студенти_дисципліни для зберігання цих зв'язків.

Крім того, ми розширили базу даних, додавши таблицю для четвертого курсу та зв'язки з дисциплінами. Також було внесено зміни у інформацію про студентів першого курсу, змінивши їхні прізвища та номери телефонів.

Ця база даних може бути використана для ведення обліку студентів, груп та дисциплін на кожному курсі, а також для відстеження того, які студенти вивчають які дисципліни. Вона дозволяє легко виконувати запити для отримання різних статистичних даних та аналізу навчального процесу.

ВИСНОВКИ

У ході роботи була розроблена база даних для ведення обліку студентів, груп та дисциплін на кожному курсі університету. Ця база даних є актуальною та важливою для сучасної освітньої системи, де необхідно забезпечувати ефективне управління та аналіз даних про навчальний процес.

Впровадження такої бази даних дозволить покращити організацію навчального процесу, забезпечити зручний доступ до інформації для викладачів та адміністрації навчального закладу, а також підвищить якість управління та контролю над навчальним процесом.

Дослідження у галузі баз даних для управління навчальним процесом має велике значення для покращення ефективності та якості управління різноманітною інформацією. Вирішення задачі створення бази даних для управління навчальним процесом є перспективним напрямом в області інформаційних технологій та сприятиме покращенню якості освіти в цілому.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Балик Н.Р., Мандзюк В.І. Бази даних MySQL: Навчальний посібник. – Тернопіль: Навчальна книга – Богдан, 2010. – 160 с.

2. Gaina G.A. Fundamentals of database design: A textbook / G.A. Hayna - Kyiv: KNUBA, 2005. 204 p.

3. Atre S. Structural approach to the organisation of databases. - Finance and Statistics, 1983. - 320 c.

4. Golitsyna O.L., Maksimov N.V., Popov I.I. Databases: Study guide. - MOSCOW: FORUM: INFRA-M, 2003. 352 p.

5. Pasichnyk V.V. Organisation of databases and knowledge / V.V. Pasichnyk, V.A. Reznichenko - Kyiv: BHV Publishing Group, 2006. 384 p.

6. Пономаренко В. С. Інформаційні системи в управлінні персоналом.. Навчальний посібник / В. С. Пономаренко, І. В. Журавльова, І. Л. Латишева. – Харків: Вид. ХНЕУ, 2008. – 336 с. (Укр. мов.) Електрон. аналог друк. вид.: режим доступу: http://www.repository.hneu.edu.ua/ (дата звернення 18.04.2024 р).

7. Інформаційно-пошукові системи. [Електронний ресурс] – Режим доступу: http://esu.com.ua/ (дата звернення 28.04.24р)

8. About SQLite. [Електронний ресурс] – Режим доступу: https://www.sqlite.org/index.html (дата звернення 10.05.24р)