

ІНСТРУКТИВНО-МЕТОДИЧНІ МАТЕРІАЛИ

для виконання лабораторних робіт

з дисципліни

«Бази даних »

ЗІ СПЕЦІАЛЬНОСТІ 121

«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Розробила викладач _____ С.С. Ланська

Розглянуто та затверджено комісією

програмної інженерії

Протокол № 7 від 07.02.2023 р.

Голова ЦК ПІ: _____ С.С. Ланська

Дніпро,
2023

ЛАБОРАТОРНА РОБОТА № 1
з навчальної дисципліни «Бази даних»

Тема Аналіз предметної області проєктованої бази даних. Аналіз вимог користувачів до проєктованої бази даних. Виділення об'єктів предметної області проєктованої бази даних та встановлення відношень між об'єктами.

Мета Навчитися проводити аналіз предметної області проєктованої (БД) виділяючи об'єкти предметної області, властивості об'єктів і встановлювати зв'язку між об'єктами.

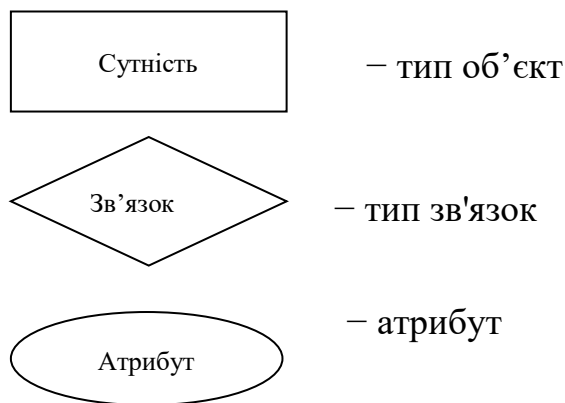
Час –2 години

1 Теоретичні відомості

Для нормального функціонування інформаційної системи необхідно, щоб концептуальна модель адекватно відображала реалії тієї предметної області, для якої вона розробляється. Фундаментальними реаліями в концептуальному моделюванні є дані з їхніми властивостями й зв'язку між ними.

Головними елементами семантичної моделі даних є типи об'єктів, їхні атрибути й типи зв'язків. Типи об'єктів часто представляють у вигляді іменників, а типи зв'язків - у вигляді дієслів.

Семантична модель предметної області зображується у вигляді діаграми з урахуванням прийнятих позначень для її елементів:



Об'єкти.

У процесі концептуального проектування предметна область розглядається як об'єктна система, що має наступні складові:

- об'єкт;
- час;
- засіб;
- зв'язок.

Об'єкти позначають речі, які користувачі вважають важливими в частини реальності, яка моделюється. Об'єкт - це те, про що накопичується інформація в інформаційній системі й що може бути однозначно ідентифіковано. Об'єкти можуть бути атомарними або складеними. Для складеного об'єкта визначається його внутрішня структура. Кожен об'єкт у конкретний момент часу характеризується своїм станом. Цей стан визначається за допомогою обмеженого набору засобів і зв'язків з іншими об'єктами. Складова час дозволяє моделювати динамічні системи.

Об'єкт-тип (надалі просто об'єкт) характеризується незалежним існуванням і представляє безліч об'єктів реального миру з однаковими властивостями. Окремі об'єкти, які входять у даний тип, називають екземплярами об'єкта. Розрізняють реальні й концептуальні об'єкти. Прикладами об'єктів можуть бути люди, товари, будинку, деталі, книги й т.д. Це реальні об'єкти. Концептуальними об'єктами будуть навички, організації, ділові операції, штатний розклад і т.д. Кожен об'єкт має ім'я й зображується на діаграмах у вигляді прямокутника, а екземпляр об'єкта - у вигляді крапки в прямокутнику даного об'єкта.

Атрибути.

Атрибут - це поійменована характеристика об'єкта, за допомогою якої моделюється його властивість. Кожному об'єкту властиві свої атрибути. Наприклад, об'єкт КНИГА повинен мати такі атрибути: найменування, автор, видавництво, рік видання. У людини є ім'я, дата народження, ріст, вага, підлога, колір волосся, мати, батько й так далі. Якщо для деякого екземпляра

об'єкта значення деякого атрибута не визначено, то цей атрибут для даного екземпляра об'єкта має порожнє значення.

Ключі.

Серед атрибутів особливе положення займають ті, за допомогою яких можна ідентифікувати екземпляр об'єкта. Такі атрибути називаються *ключами*. Атрибут або кілька атрибутів, значення яких унікальним образом ідентифікують кожен екземпляр об'єкта, є потенційним ключем даного об'єкта. Потенційних ключів може бути кілька.

Один з потенційних ключів може бути обраний як *первинний ключ*. Звичайно як первинний ключ вибирається той, котрий має найменшу довжину. Інші потенційні ключі називаються *альтернативними*.

Той факт, що атрибут служить первинним ключем, відзначається його підкресленням. Ідентифікацію деяких об'єктів іноді доводиться здійснювати за допомогою складених ключів, які включають кілька атрибутів.

З іншого боку, якщо потрібно, завжди можна створити ідентифікаційний номер, однозначність якого буде закладена в системі. Щоб уникнути зайвої деталізації на діаграмах часто атрибути зовсім не зображують або зображують тільки первинні атрибути, опускаючи інші.

Зв'язки між об'єктами

Два об'єкти можуть бути зв'язані між собою. Подібний зв'язок здійснюється через зв'язок екземплярів одного об'єкта з екземплярами іншого об'єкта, створюючи набір екземплярів зв'язку між двома об'єктами, що називається *типом зв'язку*. Кожному типу зв'язку привласнюється ім'я, що повинне представляти його функцію.

Потужність зв'язку. Важливою характеристикою зв'язку є її потужність, що позначає максимальну кількість екземплярів одного об'єкта, пов'язаних з одним екземпляром іншого об'єкта.

Як приклад візьмемо базу даних компанії, яка займається видавничою діяльністю.

База даних створюється для інформаційного обслуговування редакторів, менеджерів та інших співробітників компанії. БД повинна містити дані про співробітників компанії, книжки, авторів, фінансовий стан компанії і надавати можливість отримувати різноманітні звіти.

Відповідно до предметної області система будується з урахуванням таких особливостей:

- кожна книга видається в рамках контракту;
- книга може бути написана кількома авторами;
- контракт підписується одним менеджером і всіма авторами книги;
- кожен автор може написати кілька книг (за різними контрактами);
- порядок, в якому автори вказані на обкладинці, впливає на розмір гонорару;
- якщо співробітник є редактором, то він може працювати одночасно над кількома книгами;
- у кожній книзі може бути кілька редакторів, один з них - відповідальний редактор;
- кожне замовлення оформляється на одного замовника;
- в замовленні на купівлю може бути перераховано кілька книг.

Виділимо базові сутності цієї предметної області:

- *Співробітники компанії*. Атрибути співробітників - ПІБ, табельний номер, стать, дата народження, паспортні дані, ППН, посаду, оклад, домашню адресу та телефони. Для редакторів необхідно зберігати відомості про редагованих книгах; для менеджерів - відомості про підписаних контрактах;
- *Автори*. Атрибути авторів - ПІБ, ППН (індивідуальний номер платника податків), паспортні дані, домашня адреса, телефони. Для авторів необхідно зберігати відомості про написаних книгах.
- *Книги*. Атрибути книги - автори, назва, тираж, дата виходу, ціна одного примірника, загальні витрати на видання, авторський гонорар.

Контракти будемо розглядати як зв'язок між авторами, книгами і менеджерами. Атрибути контракту - номер, дата підписання та учасники.

Для відображення фінансового положення компанії в системі потрібно враховувати замовлення на книги. Для замовлення необхідно зберігати номер замовлення, замовника, адреса замовника, дату надходження замовлення, дату його виконання, список замовлених книжок із зазначенням кількості примірників.

ER-діаграма видавничої компанії приведена на рисунку 1.1 (базові сутності виділені напівжирним шрифтом).

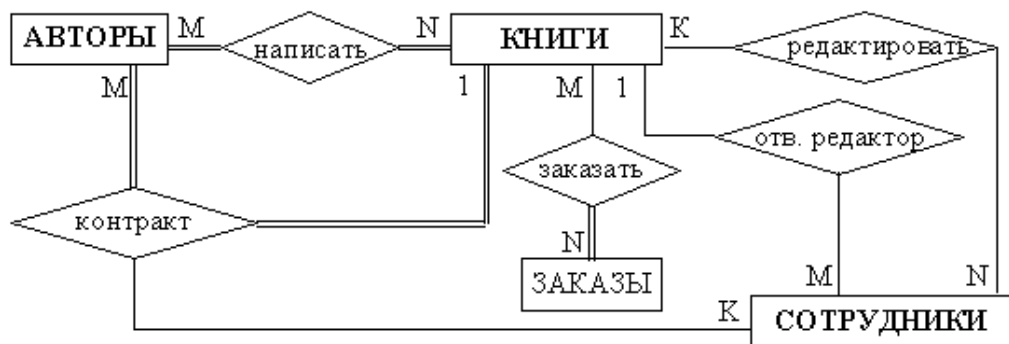


Рисунок 1.1 – ER-діаграма видавничої компанії

2 Аналіз інформаційних завдань і кола користувачів системи

Система створюється для обслуговування таких груп користувачів:

- адміністрація (дирекція);
- менеджери;
- редактори;
- співробітники компанії, що обслуговують замовлення.

3 Приклад виконання лабораторної роботи.

3.1 Постановка завдання

Розглянемо предметну область, пов'язану з роботою організації по розробці програмного забезпечення. У рамках цієї предметної області й залежно від передбачуваних запитів буде необхідна наступна інформація про співробітників організації: Прізвище, ім'я, по батькові співробітника;

відділ, у якому працює співробітник (номер відділу й назва відділу); посада й кваліфікація співробітника. Кваліфікація співробітника являє собою список середовищ і додатків, по яких співробітник має кваліфікаційний сертифікат.

Для оцінки фінансової діяльності організації ведеться наступна інформація про виконуваний організацією проєктах: Тема проєкту; керівник проєкту; клієнт, для якого розробляється проєкт; дата укладання договору; строк виконання проєкту; вартість проєкту; співробітники, зайняті в реалізації проєкту із вказівкою частки співробітника в загальному гонорарі за проєкт.

3.2 Аналіз предметної області

У результаті аналізу проєктної області встановлено:

1 Організація містить кілька відділів. В одному відділі працює кілька співробітників, але один співробітник організації може працювати тільки в одному відділі.

2 Про клієнтів організації зберігається й обробляється наступна інформація: Найменування організації - замовника; прізвище керівника; адреса; контактний телефон. Одна організація - замовник може замовляти кілька проєктів.

3 Один співробітник може бути керівником декількох проєктів, але кожен проєкт має одного керівника;

4 Один співробітник може брати участь у розробці декількох проєктів й в одному проєкті бере участь кілька співробітників.

3.3 Опис об'єктів предметної області

На основі постановки завдання й аналізу предметної області виділимо наступні об'єкти:

1 Об'єкт «Співробітник» представлений на рисунку 1.1.

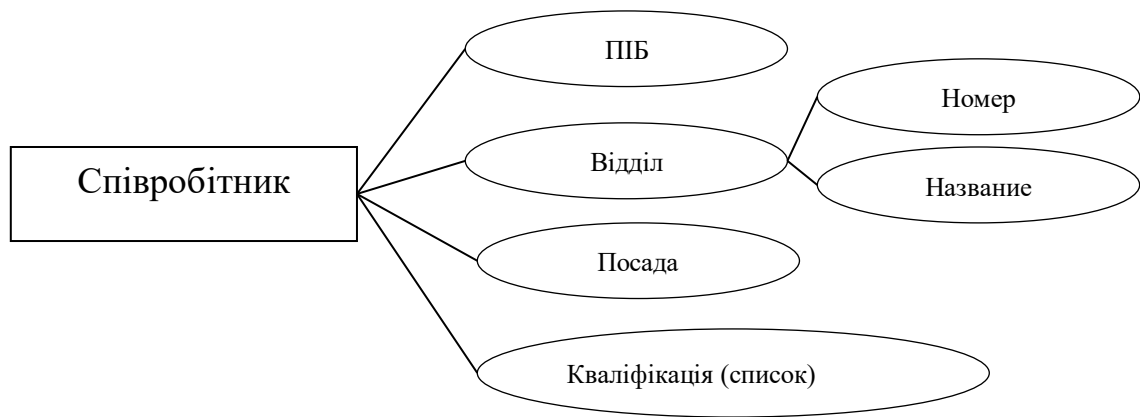


Рисунок 1.1 – Об'єкт «Співробітник»

2 Об'єкт «Проект» представлений на рисунку 1.2.

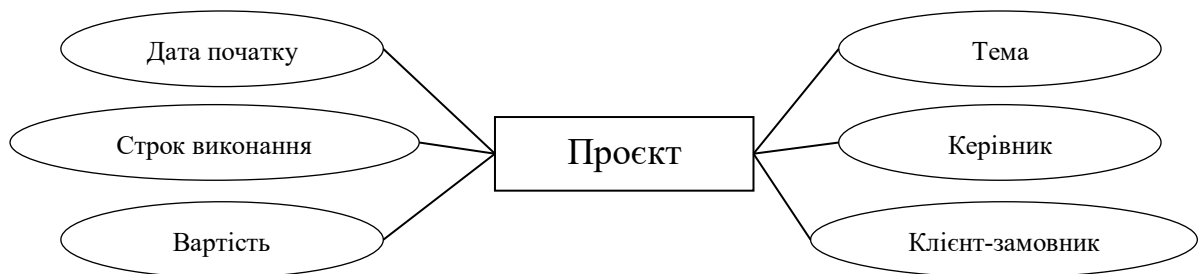


Рисунок 1.2 – Об'єкт «Проект»

3 Об'єкт «Клієнт» представлений на рисунку 1.3.

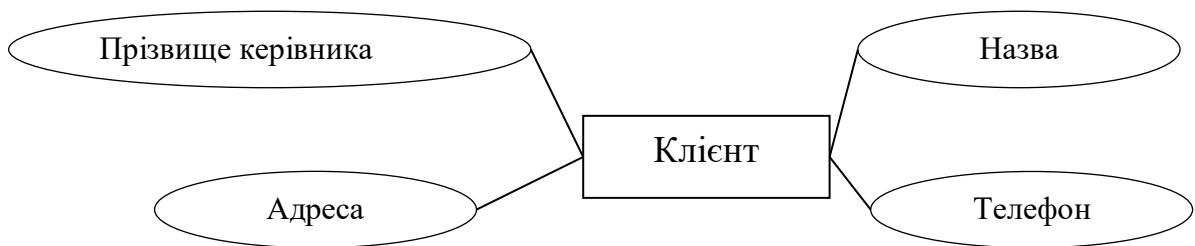


Рисунок 1.3 – Об'єкт «Клієнт»

Хід роботи

I Провести аналіз предметної області згідно варіанта. Уточнити вимоги користувача до проектованої бази даних.

II Виділити об'єкти предметної області і їхні атрибути.

III Оформити звіт по виконанню лабораторної роботи.

Варіанти завдань

Варіант 1	Предметна область: <i>Ювелірна майстерня.</i>
Варіант 2	Предметна область: <i>Супермаркет.</i>
Варіант 3	Предметна область: <i>Салон краси.</i>
Варіант 4	Предметна область: <i>Страхова компанія.</i>
Варіант 5	Предметна область: <i>Прокат авто.</i>
Варіант 6	Предметна область: <i>Облік дисциплін коледжу.</i>
Варіант 7	Предметна область: <i>Прокат спортивного устаткування.</i>
Варіант 8	Предметна область: <i>Облік успішності студентів.</i>
Варіант 9	Предметна область: <i>Продаж авто.</i>
Варіант 10	Предметна область: <i>Кредитування.</i>
Варіант 11	Предметна область: <i>Туристична агенція.</i>
Варіант 12	Предметна область: <i>Ресторан.</i>
Варіант 13	Предметна область: <i>Ремонтне ательє.</i>
Варіант 14	Предметна область: <i>Складський облік.</i>
Варіант 15.	Предметна область: <i>Продаж ж.д. білетів.</i>
Варіант 16	Предметна область: <i>Футбольний клуб.</i>
Варіант 17	Предметна область: <i>Обладнання для підприємств.</i>
Варіант 18	Предметна область: <i>Фірма з продажу запчастин.</i>
Варіант 19	Предметна область: <i>Лікарня.</i>
Варіант 20	Предметна область: <i>Готель.</i>
Варіант 21	Предметна область: <i>Фірма «Меблі».</i>

Звіт повинен містити

- 1 Тему та мету лабораторної роботи.
- 2 Постановка завдання.
- 3 Аналіз предметної області.
- 4 Опис об'єктів предметної області з їхніми атрибутами.
- 5 Висновок.

Контрольні питання

- 1 Поясніть своїми словами зміст термінів:
 - база даних;
 - предметна область;
 - інформаційна система;
 - життєвий цикл бази даних.
- 2 Охарактеризуйте життєвий цикл бази даних. Які основні етапи він включає?
- 3 Назвіть основні цілі процесу проектування бази даних і дайте характеристику його фазам.
- 4 Які стратегії тестування прикладних програм інформаційних систем існують?
- 5 Поясніть своїми словами зміст термінів:
 - об'єкт;
 - атрибут;
 - бінарний зв'язок;
 - "один-до-багатьох";
 - потужність;
 - показник кардинальності;
 - ступінь участі;
 - ключ;
 - рекурсивний зв'язок;

– складовий об'єкт.

6 Поясніть, якою інформацією можна скористатися для визначення наступних конструкцій концептуальної моделі даних:

– об'єктів;

– атрибутів;

Література

1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк

Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.

2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. /

Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.

3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд.

/ К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.

Розглянуто та схвалено

на засіданні циклової комісії

програмної інженерії

Протокол № 7 від 07.02.2023 р.

Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 2
з навчальної дисципліни «Бази даних»

Тема Побудова концептуальної моделі даних проектованої бази даних.

Удосконалення концептуальної моделі даних проектованої бази даних.

Мета Придбання практичних навичок побудови та спрощення
концептуальної схеми предметної області

Час –2 години

Теоретичні відомості

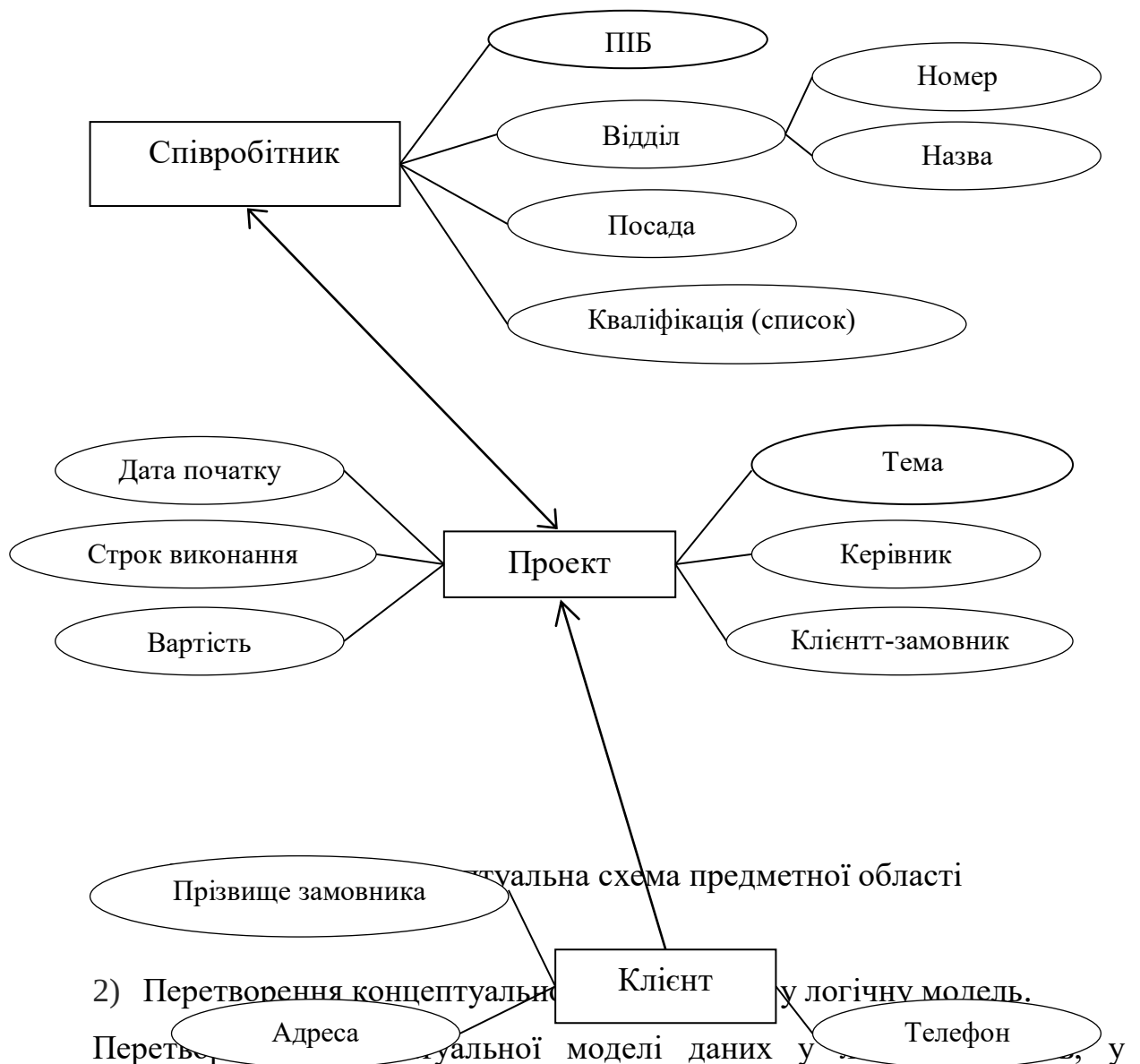
В минулій роботі було проаналізовано ПрО (предметну область), виділені об'єкти і їх атрибути та зв'язки між об'єктами. Виходячи з цього необхідно побудувати концептуальну схему ПрО.

1) Побудова концептуальної схеми предметної області:

– з огляду на, що один співробітник може брати участь у розробці декількох проектів й в одному проекті бере участь кілька співробітників, між об'єктами «Співробітник» і «Проект» встановлюємо зв'язок «багато до багатьох».

– з огляду на, що одна організація – замовник, або фізична особа (один клієнт) може замовляти кілька проектів, між об'єктами «Клієнт» і «Проект» встановлюємо зв'язок «один до багатьох».

Концептуальна схема предметної області представлена на рисунку 2.1.



результаті якого буде визначена схема реляційної моделі даних, може мати два підходи. Один з підходів полягає в тому, що проектувальник працює з концептуальною моделлю прямо, не прибігаючи до її попереднього перетворення. У цьому випадку йому доведеться зіштовхнутися з необхідністю перетворення різноманітних структур даних, причому на шляху перетворення деяких з них він може зустріти ряд труднощів. Другий же підхід полягає в тому, що проектувальник, перш ніж приступитися до процесу переходу від концептуальної моделі до логічної моделі, прагне спочатку даний перехід максимально спростити, провівши попередні перетворення концептуальної моделі, перетворення деяких її, що не підходять для реляційних СУБД, структур даних. До таких структур даних ставляться:

- зв'язку типу «багато до багатьох»;
- складні зв'язки;
- рекурсивні зв'язки;
- зв'язку з атрибутами;
- множинні атрибути;
- надлишкові зв'язки.

Точка зору другого підходу на процес перетворення концептуальної моделі така: якщо в моделі присутні перераховані небажані структури, то вони повинні бути виключені шляхом тотожної їхньої заміни на структури даних, припустимі для логічної моделі. Розглянемо спочатку другий підхід, що припускає наявність проміжного етапу на шляху одержання логічної моделі БД, етапу одержання спрощеної концептуальної моделі.

Виключення зв'язку типу «багато до багатьох».

Перетворення зв'язку типу «багато до багатьох» здійснюється шляхом введення деякого проміжного об'єкта із заміною одного зв'язку двома зв'язками типу «один до багатьох» зі знову створеним об'єктом. При організації нових зв'язків необхідно стежити за тим, щоб максимальна потужність зв'язку «один» була спрямована до вихідного об'єкта, а максимальна потужність зв'язку «багато» - до знову створеного об'єкта. Наприклад, допустимо ситуацію, коли той самий викладач може читати кілька курсів, і той самий курс може викладатися декількома викладачами, як наведено на рисунку 2.2.



Рисунок 2.2 – Схема зв'язку «багато до багатьох»

Уведемо додатковий об'єкт «ЧИТАННЯ» й визначимо нові два зв'язки:
 ВИКЛАДАЧ_ ЧИТАЄ типу 1:N між об'єктами «ВИКЛАДАЧ» і «ЧИТАННЯ»;

КУРС_ ЧИТАЄТЬСЯ типу 1:N між об'єктами «КУРС» і «ЧИТАННЯ»,
 як представлено на рисунку 2.3.



Рисунок 2.3 – Схема зв'язку «один до багатьох»

Виключення складних зв'язків.

Хоча за допомогою бінарних зв'язків можуть бути описані багато ситуацій реального миру, проте неминуче виникнення й такі ситуації, у яких побудова розумної моделі організації не може бути зведена до реалізації тільки таких зв'язків. Наприклад, виникає потреба моделювання тристоронніх зв'язків. Зв'язок, у якій кількість учасників перевищує два, тобто тернарні зв'язки й зв'язки більше високого порядку, називані n-арними зв'язками, вважається складною. Виключення такого зв'язку з концептуальної моделі відбувається по наступному сценарії:

- у модель уводиться новий об'єкт;
- складний зв'язок з бінарними зв'язками типу «один до багатьох» вихідних об'єктів зі знову створеним об'єктом, причому кількість бінарних зв'язків дорівнює ступеню складного зв'язку.

Наприклад, нехай заняття за деяким курсом ведуться деяким викладачем у деякій лабораторії. Концептуальна модель у цій ситуації може містити структуру даних, яка представлена на рисунку 2.4).

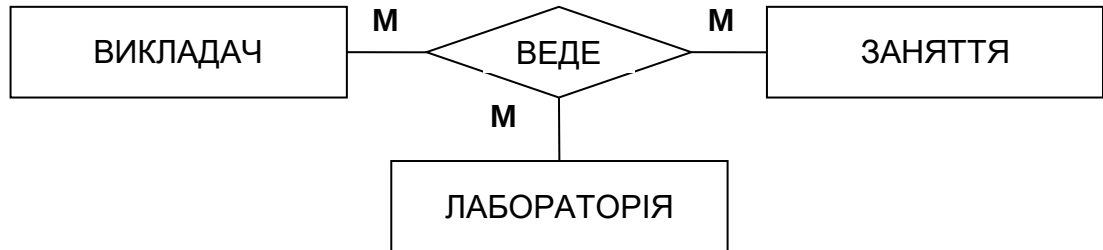


Рисунок 2.4 – Концептуальна схема зі складним зв'язком

Даний тернарний зв'язок, що відбиває відносини між викладачем, заняттями й лабораторією, повинен бути замінений необхідною кількістю бінарних зв'язків типу «один до багатьох». Уведемо додатковий об'єкт: «ЛАБОРАТОРНІ_ЗАНЯТТЯ» й визначимо для нього три бінарні зв'язки типу «один до багатьох» з кожним з вихідних об'єктів: зв'язок «ВИКЛАДАЧ_ВЕДЕ» для об'єктів «ВИКЛАДАЧ», «ЛАБОРАТОРНІ_ЗАНЯТТЯ»; зв'язок «В» для об'єктів «ЛАБОРАТОРІЯ», «ЛАБОРАТОРНІ_ЗАНЯТТЯ»; зв'язок «ВЕДУТЬСЯ» для об'єктів «ЗАНЯТТЯ», «ЛАБОРАТОРНІ_ЗАНЯТТЯ», як представлено на рисунку 2.5.

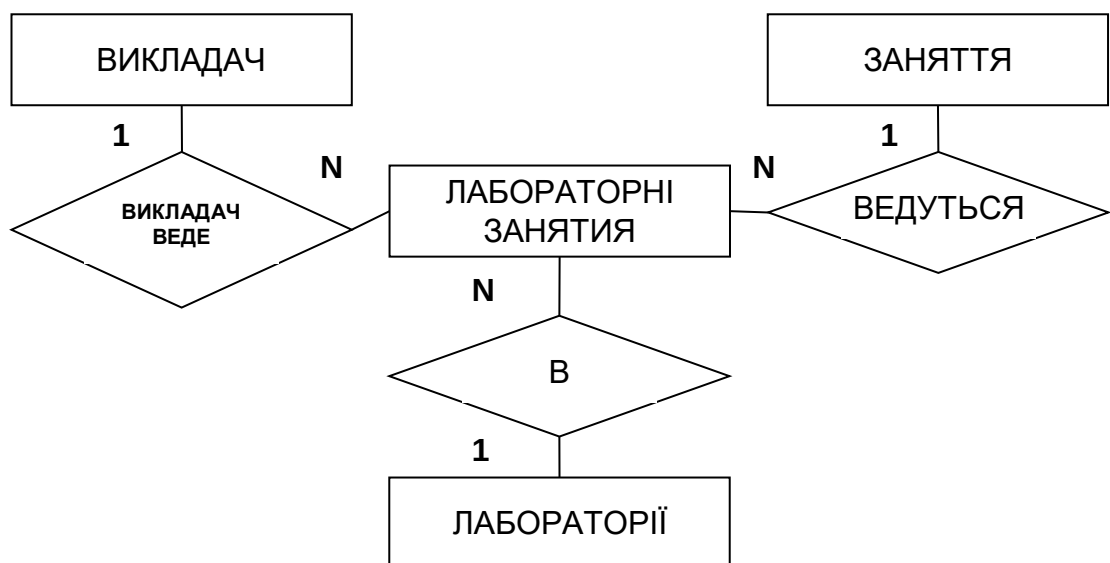


Рисунок 2.5 – Спрощена концептуальна схема

Отримана в результаті даного перетворення концептуальна модель вже не містить небажані для реляційної схеми бази дані структури.

ПРИКЛАД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ.

1) Аналіз атрибутів предметної області.

Об'єкт «Співробітник» представлений на рисунку 2.6.

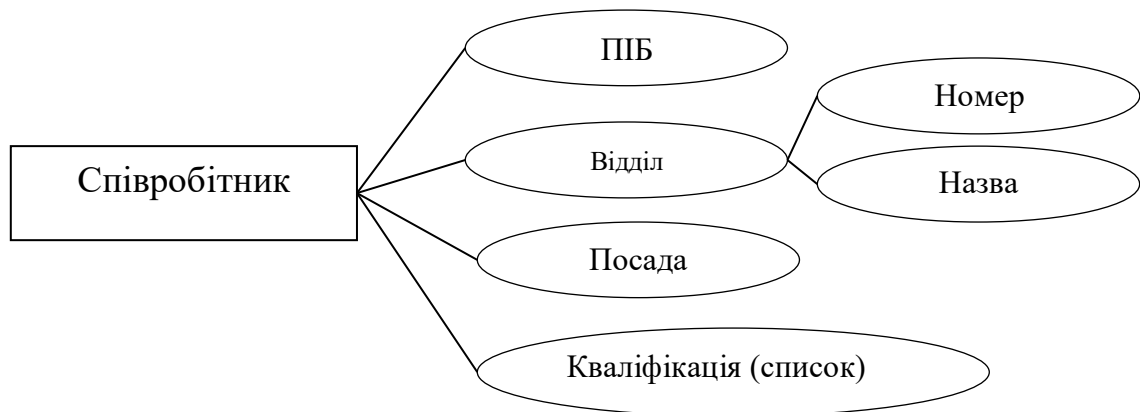


Рисунок 2.6 – Об'єкт співробітник

Має дві не атомарних атрибута: «Відділ» і «Кваліфікація».

Атрибут «Відділ» являє собою сукупність різнотипних значень: «Номер відділу» й «Назва відділу». Замінімо атрибут «Відділ» двома атрибутами: «Номер» і «Назва»

Атрибут «Кваліфікація» являє собою список однотипних значень. Тому виділимо додатковий об'єкт «Кваліфікація», як наведено на рисунку 2.7 з атрибутами: ПІБ співробітника й Значення кваліфікації.

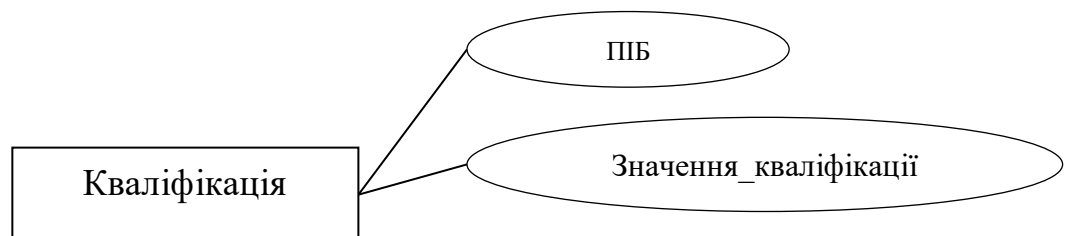


Рисунок 2.7 – Новий об'єкт «Кваліфікація»

2) Розрив зв'язків «багато до багатьох» проміжними об'єктами.

Об'єкти «Співробітник» і «Проект» зв'язані між собою зв'язком «багато до багатьох». Для розриву цього зв'язку введемо додатковий об'єкт

«Проект_Співробітник» з атрибутами «ПІБ співробітника», що працюють у проекті, «Тема проекту» й «Частка участі» співробітника в проекті, як наведено на рисунку 2.8.

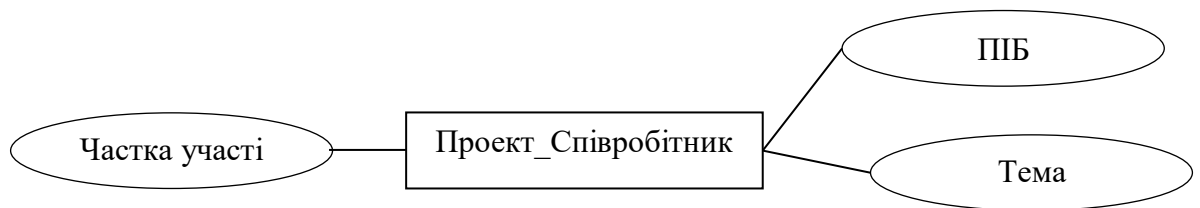


Рисунок 2.8 – Новий проміжний об’єкт «Проект_Співробітник»

3) Спрощена концептуальна схема предметної області представлена на рисунку 2.9.

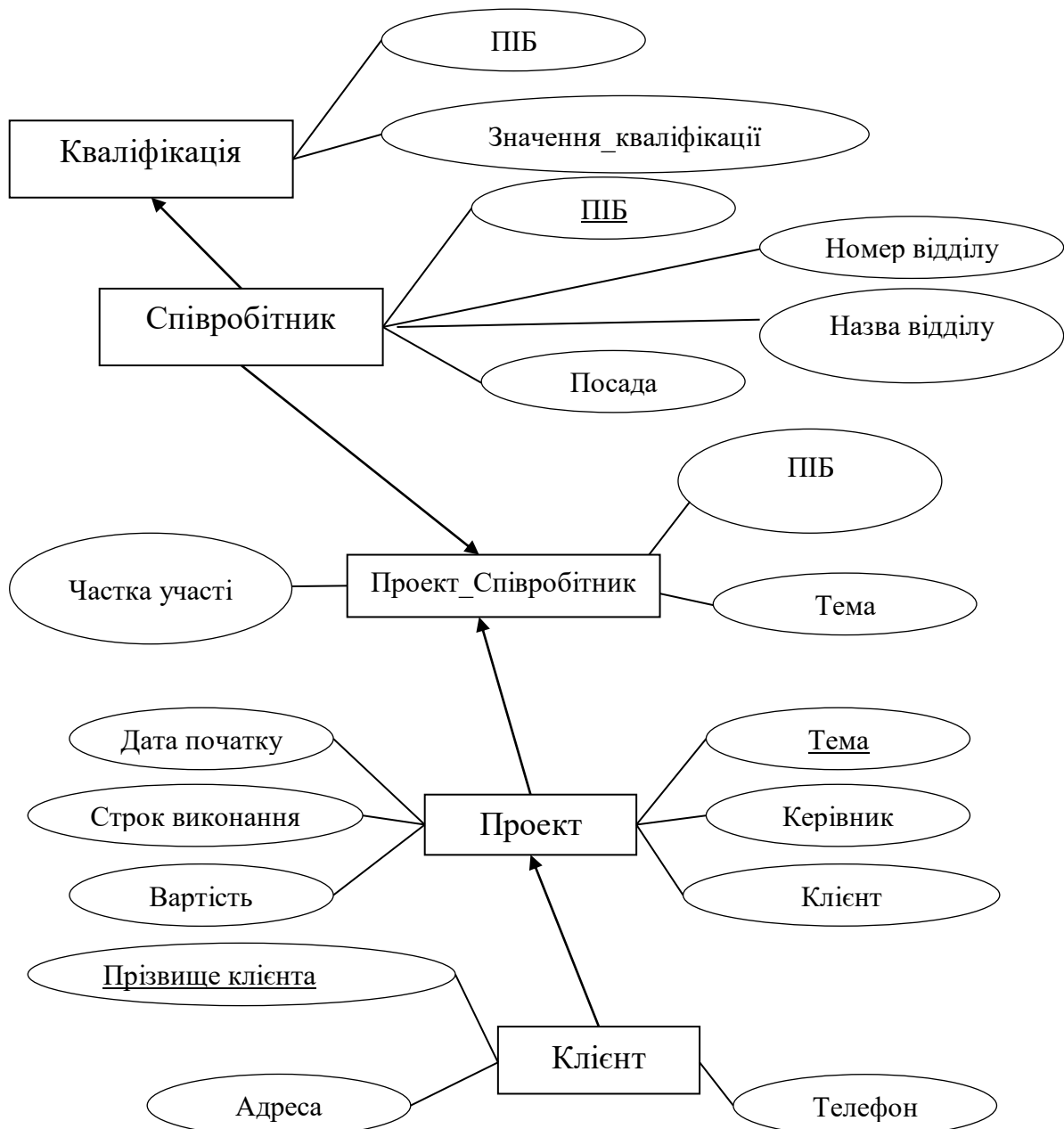


Рисунок 2.9 – Спрощена концептуальна схема предметної області

Хід роботи

I За результатами виконання лабораторної роботи №1 побудувати концептуальну модель предметної області.

II Проаналізувати атрибути виділених об'єктів. Провести заміну складених атрибутів простими.

III Розірвати зв'язку типу «багато до багатьох» проміжними об'єктами.

IV Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1 Тему та мету лабораторної роботи.
- 2 Постановку завдання.
- 3 Концептуальну схему предметної області.
- 4 Аналіз атрибутів предметної області.
- 5 Розрив зв'язків «багато до багатьох» проміжними об'єктами.
- 6 Удосконалену концептуальну схему предметної області.
- 7 Висновок.

Контрольні питання

- 1 Поясніть своїми словами зміст термінів:
 - надмірність даних;
 - аномалії включення, відновлення, видалення;
 - сторонній атрибут;
- 2 Освітите процес проектування реляційної бази даних. Які дії припускає фаза логічного проектування реляційної БД?
- 3 Яким образом можна спростити концептуальну модель даних, щоб полегшити процес переходу від концептуальної моделі до логічної моделі?
- 4 Викладете правила методики перетворення концептуальних структур даних у реляційні структури.

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 3
з навчальної дисципліни «Бази даних»

Тема	Виділення таблиць реляційної бази даних. Перевірка таблиць за допомогою концепції послідовної нормалізації
------	--

Мета	Придбання практичних навичок виділення таблиць реляційної бази даних на основі концептуальної схеми предметної області й приведення таблиць до третьої нормальної форми (3НФ).
------	--

Час – 2 години

Теоретичні відомості

Створенням спрощеної концептуальної моделі предметної області закладається основа для одержання реляційної схеми БД. Оскільки в такій моделі присутнє дуже вузьке коло дозволених структур, перетворення кожної з яких має свої особливості, то методика одержання реляційної схеми бази даних являє собою сукупність правил їхнього перетворення в набір відносин. У спрощеній концептуальній моделі після виключення небажаних структур можуть бути присутнім наступні структури даних:

- об'єкти й атрибути;
- бінарні зв'язки типу 1:1 і типу 1: N.

Розглянемо правила перетворення кожної із зазначених структур.

Перетворення об'єктів й атрибутів.

Нехай проектується БД, призначена для зберігання інформації про викладачів факультету університету й про ті курси, які вони читають. База даних, отже, включає об'єкти «ВИКЛАДАЧ» і «КУРС», як наведено на рисунку 3.1.



Рисунок 3.1 – Об'єкти «ВИКЛАДАЧ» та «КУРС»

Об'єкт «ВИКЛАДАЧ» має в цьому випадку два атрибути, один із яких «Табельний номер» може бути використаний для ідентифікації екземпляра викладача, тобто бути первинним ключем. Таким чином, верхня діаграма може бути перетворена в наступне відношення: ВИКЛАДАЧ (Табельний номер, ПІБ). Об'єкт «КУРС» має тільки один атрибут, якого не можна використати як ключ, оскільки дисципліни з однаковою назвою можуть мати різні робочі програми для різних спеціальностей. Тому в даній ситуації для ідентифікації екземпляра об'єкта необхідно додати ще один атрибут: «Номер_курсу», як наведено на рисунку 3.2.

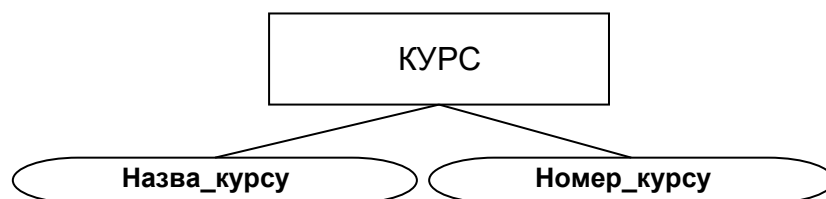


Рисунок 3.2 – Відношення «КУРС»

У результаті маємо наступне відношення: КУРС (Номер курсу, Найменування курсу). Отже, об'єкт із атрибутами може бути перетворений у відношення з ім'ям об'єкта як ім'я відношення й атрибутами об'єкта як атрибуту відношення. Якщо деякий набір атрибутів може бути використаний як ключ відношення, то він призначається ключем відношення. У протилежному випадку у відношення необхідно додати ідентифікуючий

атрибут. Загальний підхід до перетворення об'єктів концептуальної моделі Про у відношення реляційної бази даних полягає в наступному:

- побудувати набір попередніх відносин і вказати первинні ключі для кожного відношення;
- підготувати список всіх атрибутів, що представляють інтерес, (тих з них, які не були перераховані в діаграмі як первинні ключі об'єктів) і призначити кожному із цих атрибутів одному з попередніх відношень з тією умовою, щоб ці відношення перебували в НФБК. На цьому кроці для кожного відношення повинні бути визначені міжатрибутні функціональні залежності, за допомогою яких перевіряється відповідність відносин НФБК. Якщо отримані відношення в підсумку не перебувають у НФБК або якщо деяким атрибутам не перебуває логічно обґрунтованих місць у попередніх відносинах, то в цих випадках діаграми необхідно переглянути.

Перетворення бінарних зв'язків.

Кожен об'єкт перетвориться в певне відношення, а виходить, зв'язок між об'єктами перетвориться у зв'язок між відносинами. Виключення з концептуальної моделі складних зв'язків і зв'язків типу «багато хто до багатьох» досить спрощує модель Про, залишаючи в ній тільки бінарні зв'язки типу «один до одному» й «один до багатьох». Зв'язку між відносинами в реляційній моделі даних реалізуються за допомогою механізму первинних і зовнішніх ключів. Щоб цей механізм діяв, необхідно в першу чергу визначитися з тим, що із двох відносин є батьківським, а яке - дочірнім. Батьківським вважається таке відношення, що передає копію набору значень свого первинного ключа іншому відношенню (дочірньому), де цей набір значень буде представляти зовнішній ключ. Останнє відношення в цьому випадку буде дочірнім відношенням. Розглянемо бінарний зв'язок «ЧИТАЄ» між об'єктами «ВИКЛАДАЧ» і «КУРС», як наведено на рисунку 3.3.



Рисунок 3.3 – Бінарний зв'язок «ЧИТАЄ»

Цей зв'язок може бути зображена за допомогою діаграми, що містить всю інформацію, необхідну для генерації відповідних відносин РБД. Об'єкт «ВИКЛАДАЧ» і «КУРС» однозначно ідентифікуються за допомогою ТН (табельний номер викладача) і НК (номер курсу). Якщо всі елементи даного об'єкта повинні брати участь у зв'язку, то така участь називається обов'язковою. Наприклад, якщо кожен викладач повинен читати один який-небудь курс, то клас приналежності такого об'єкта - обов'язковий. Клас приналежності об'єктів, а також потужність зв'язку між об'єктами є факторами, що визначають структуру проектованої БД.

Попередні відношення для бінарних зв'язків типу 1:1.

Попередні відношення для бінарних зв'язків з показником кардинальності, рівним 1:1 можуть бути отримані внаслідок перегляду декількох логічних альтернатив і вибору з них найбільш підходящої. Приклад наведений на рисунку 3.4. Наприклад, нехай у проектованій БД повинна зберігатися наступна інформація: ТН - табельний номер викладача; ПІБ - прізвище, ім'я, по батькові викладача; Тел_ Викладача - телефон викладача; НК - номер курсу; Назв_ Курсу - назва курсу.

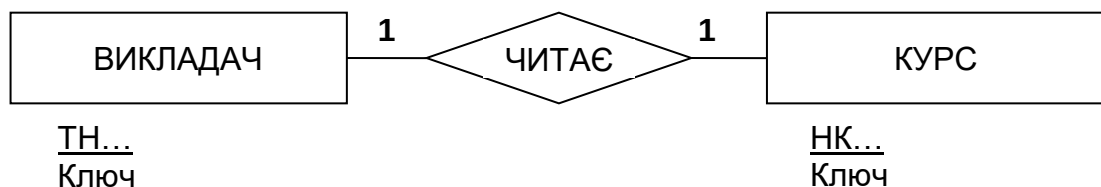


Рисунок 3.4 – Відношення для бінарних зв'язків з показником кардинальності, рівним 1:1

1) Клас приналежності для обох об'єктів - обов'язковий.

Уважаючи, що клас приналежності є обов'язковим для обох об'єктів, одержуємо відношення: ВИКЛАДАЧ (ТН, ПІБ, Тел_ Викладача, НК, Назв_

Курсу). У цьому зведеному відношенні перебуває вся необхідна інформація. Так як показник кардинальності зв'язку дорівнює 1:1 і клас приналежності є обов'язковим для обох об'єктів, то гарантується однократна поява кожного значення ТН і НК у будь-якому екземплярі відношення. Це значить, що відношення ніколи не буде містити ні порожньої інформації, ні повторюваних груп надлишкових даних. Первинним ключем цього відношення може бути ключ кожного із двох об'єктів. Існування єдиного відношення в даній ситуації тим більше переважно тоді, коли вихідні два об'єкти не приймають участі в інших зв'язках. Якщо ж по якихось міркуваннях бажано для кожного об'єкта мати окреме відношення, то вибір батьківського й дочірнього відношень виконується довільно.

2) Клас приналежності одного з об'єктів - обов'язковий.

Ситуація буде інша, якщо клас приналежності одного з об'єктів «ВИКЛАДАЧ» - обов'язковий, другого «КУРС» - ні. У цьому випадку одного відношення в БД недостатньо. Пробіли з'являються в ньому у всіх кортежах, що містять інформацію про курси, що не викладаються жодним з викладачів. Існує тільки один вихід з такої ситуації - побудова двох відношень: по одному під кожен об'єкт. При цьому ключ кожного об'єкта повинен служити первинним ключем для відповідного відношення. Об'єкт, для якого клас приналежності є необов'язковим, перетвориться в батьківське відношення, а об'єкт, що бере участь у зв'язку з обов'язковим класом приналежності, перетвориться в дочірнє відношення. Для зв'язку отриманих відносин ключ батьківського відношення додається як атрибут (зовнішнього ключа) у дочірнє відношення. У результаті одержуємо наступну реляційну схему: ВИКЛАДАЧ (ТН, ПІБ, Тел_ Викладача, НК); КУРС (НК, Назв_ Курсу).

3) Клас приналежності для обох об'єктів - необов'язковий.

У подібній ситуації можна розглянути два альтернативних рішення. Перше рішення припускає існування двох відносин. Причому в цьому випадку зовсім байдуже, що із двох відносин буде визначене як батьківське відношення. Можливий такий варіант: КУРС (НК, Назв_ Курсу); ВИКЛАДАЧ (ТН, ПІБ, Тел_ Викладача, НК). Можливий й інший варіант:

ВИКЛАДАЧ (ТН, ПІБ, Тел_ Викладача); КУРС (НК, Назв_ Курсу, ТН). Але обом цим варіантам властива поява пробілів у всіх кортежах, що містять інформацію про курси, що не читаються жодним з викладачів (перша схема), і у всіх кортежах, що містять інформацію про викладачів, не ведучих жоден курс (друга схема). Кращим рішенням при сформованих обставинах є визначення трьох відносин - по одному для кожного об'єкта й одного для зв'язку: ВИКЛАДАЧ (ТН, ПІБ, Тел_ Викладача); КУРС (НК, Назв_ Курсу); ЧИТАЄ (ТН, НК). У відношенні «ЧИТАЄ» будь-які значення ТН і НК можуть з'являтися тільки один раз, тому що показник кардинальності зв'язку дорівнює 1:1, і в ньому з'являються тільки ті викладачі, які й читають ті курси, які читаються. Відношення «ВИКЛАДАЧ» містить відомості про всіх викладачів, а відношення «КУРС» - про всі курси. Відношення для зв'язку повинне мати серед своїх атрибутів по одному ключі від кожного об'єкта.

Попередні відношення для бінарних зв'язків типу 1: N.

Нехай у розглянутій концептуальній моделі існує бінарний зв'язок типу 1: N. Для таких зв'язків існує тільки два правила. Варіант визначається класом приналежності N-зв'язного об'єкта, клас приналежності 1-зв'язного об'єкта не впливає на кінцевий результат в обох випадках.

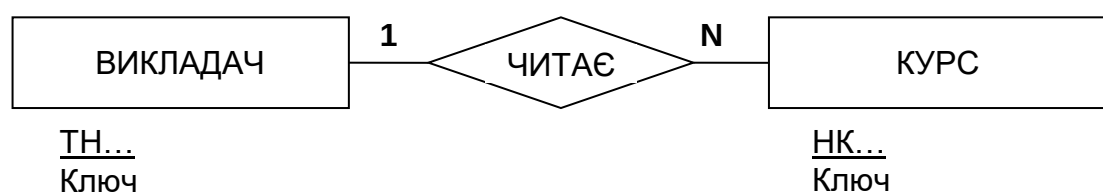


Рисунок 3.5 – Бінарний зв'язок типу 1: N

1) Перший варіант. «ВИКЛАДАЧ» — клас приналежності необов'язковий. «КУРС» — обов'язковий. Якщо в такій ситуації всю інформацію помістити в одне відношення, то в ньому будуть присутні повторювані відомості про викладачів, що читають кілька курсів, і пробіли в полях атрибутів курсу, що не читається жодним викладачем. Якби клас приналежності об'єкта «ВИКЛАДАЧ» був обов'язковий, то зникли б пробіли,

але повторювані дані залишилися б. Рішення цього завдання стає можливим, якщо використати двоє відносин, по одному на кожен об'єкт, за умови, що ключ кожного об'єкта служить як первинний ключ для відповідного відношення. Для такої діаграми існує тільки одне безальтернативне визначення батьківського й дочірнього відношень. У якості батьківського призначається відношення, що відповідає «одиначної» стороні зв'язку, а в якості дочірнього призначається відношення, що представляє «множинну» сторону зв'язку. Для подання цього зв'язку необхідно скопіювати первинний ключ батьківського відношення в дочірнє відношення, де даний ключ повинен бути описаний як зовнішній. Остаточнo в БД увійдуть двоє відносин: ВИКЛАДАЧ (ТН, ПІБ, Тел_ Викладача); КУРС (НК, Назв_ Курсу, ТН).

2) Другий варіант. Клас приналежності для обох об'єктів - необов'язковий. В одиночному відношенні, що моделює таку систему, з'являється ряд проблем: виникають пробіли в атрибутах курсу, де викладач не читає курс; виникають пробіли в атрибутах викладача, де курс не читається жодним викладачем; повторюються відомості про викладачів, що читають більше одного курсу. Якщо скористатися попереднім правилом, то зникнуть перша й третя проблеми, але друга залишиться. Рішення другої проблеми - у формуванні трьох відносин: по одному на кожен об'єкт (причому ключ кожного об'єкта служить первинним ключем відповідного відношення) і одні відносини для зв'язку. Відношення для зв'язку повинне мати серед своїх атрибутів ключ кожного об'єкта: КУРС (НК, Назв_ Курсу); ВИКЛАДАЧ (ТН, П_І_Б, Тел_ Викладача); ЧИТАЄ (ТН, НК).

ПРИКЛАД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1) Таблиці реляційної бази даних.

На основі побудованої та удосконаленої концептуальної схеми предметної області (див. лабораторні роботи №1 та 2) виділимо наступні таблиці реляційної бази даних:

Співробітник (ПІБ_співробітника, Посада, Номер_відділу, Назва_відділу)

Кваліфікація (ПІБ_співробітника, Значення_кваліфікації)

Проект (Тема_проекту, Керівник_проекту, Клієнт_замовник, Дата_початку, Строк_виконання, Вартість)

Проект_Співробітник (Тема_проекту, ПІБ_співробітника, Частка_участі)

Клієнт (Прізвище_клієнта, Адреса, Телефон)

2) Атрибути зв'язку між таблицями:

2.1 Для встановлення зв'язків між таблицями «Співробітник» і «Кваліфікація» й таблицями «Співробітник» і «Проект_Співробітник» введемо атрибут «Код_співробітника». У таблиці «Співробітник» це буде додатковий атрибут — установемо його ключовим атрибутом.

У таблицях «Кваліфікація» й «Проект_співробітник» атрибути «ПІБ_співробітника» замінимо на атрибут «Код_співробітника». У таблиці «Проект» атрибут «Керівник_проекту» теж можемо замінити атрибутом «Код_співробітника», тому що керівник проекту є співробітником організації.

2.2 Для встановлення зв'язку між таблицями «Клієнт» і «Проект» введемо атрибут «Код_клієнта». У таблиці «Клієнт» це буде додатковий атрибут - установемо його ключовим атрибутом.

У таблиці «Проект» атрибут «Клієнт_замовник» замінимо на атрибут «Код_клієнта».

2.3 Для встановлення зв'язку між таблицями «Проект» і «Проект_співробітник» введемо атрибут «Код_проекту». У таблиці «Проект» це буде додатковий атрибут — установемо його ключовим атрибутом.

У таблиці «Проект_Співробітник» атрибут «Тема_проекту» замінимо на атрибут «Код_проекту».

У результаті одержимо наступні таблиці реляційної бази даних (з виділеними ключовими атрибутами й зовнішніми ключами):

Співробітник (Код_співробітника, ПІБ_співробітника, Посада, Номер_відділу, Назва_відділу)

Кваліфікація (Код_співробітника, Значення_кваліфікації)

Проект (Код_проекту, Тема_проекту, Код_керівника, Код_клієнта, Дата_початку, Строк_виконання, Вартість)

Проект_Співробітник (Код_проекту, Код_співробітника, Частка_участі)

Клієнт (Код_клієнта, Прізвище_клієнта, Адреса, Телефон)

3) Приведення таблиць до 3НФ.

3.1 Приведення таблиць до 1НФ:

Тому що в лабораторній роботі №2 була зроблена заміна складених атрибутів на атомарні, то всі таблиці бази даних перебувають в 1НФ.

3.2 Приведення таблиць до 2НФ:

Таблиця «Кваліфікація» перебуває в 2НФ тому що не містить неключових атрибутів.

Таблиця «Проект_Співробітник» перебуває в 2НФ тому що неключовий атрибут «Частка_участі» функціонально повно залежить від складеного ключа (частка участі одного співробітника в різних проектах може бути різної).

3.3 Приведення таблиць до 3НФ:

Транзитивна залежність не ключових атрибутів від первинного ключа є тільки в таблиці Співробітник:

Код_співробітника → Номер_відділу → Назва_відділу

Для усунення даної транзитивної залежності розіб'ємо таблицю «Співробітник» на дві таблиці:

Співробітник (Код_співробітника, ПІБ_співробітника, Посада, Номер_відділу)

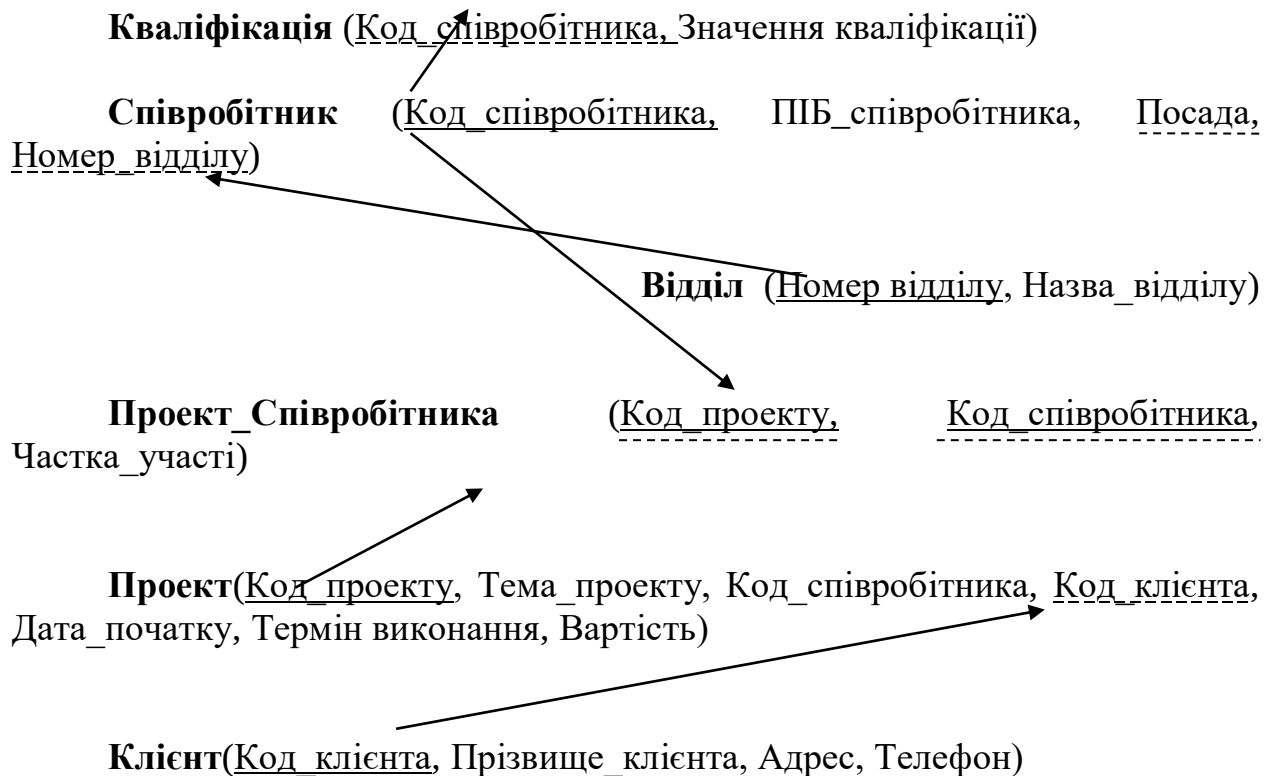
і

Відділ (Номер_відділу, Назва_відділу)

Таблиця «Відділ» буде пов'язана з таблицею «Співробітник» по атрибуту «Номер_відділу». Тип зв'язку: «один до багатьох». У таблиці «Співробітник» атрибут «Номер відділу» буде виконувати роль зовнішнього ключа.

4) Умовна схема зв'язків між таблицями бази даних.

Покажемо умовну схему зв'язків між отриманими таблицями:



Хід роботи

I На основі лабораторної роботи № 2 виділити таблиці реляційної бази даних, що відповідають об'єктам концептуальної схеми предметної області. Установити ключові атрибути таблиць.

II Установити атрибути зв'язку між таблицями. При необхідності, ввести нові атрибути або замінити вже наявні. Виділити зовнішні ключі таблиць.

III Провести процес нормалізації таблиць. Привести таблиці до 3НФ.

IV Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Таблиці реляційної бази даних.
- 3) Атрибути зв'язку між таблицями.
- 4) Приведення таблиць до 3НФ.

5) Умовна схема зв'язків між таблицями бази даних.

6 Висновок.

Контрольні питання

1) Поясніть своїми словами зміст термінів:

- реляційна модель даних;
- реляційна схема бази даних.

2) Охарактеризуйте реляційну модель даних.

3) Дайте розгорнуте пояснення структурної частини реляційної моделі даних.

4) Поясніть терміни:

- відношення;
- кортеж;
- атрибут і ступінь відношення;
- первинний і зовнішній ключі;
- домен;
- кардинальне число.

5) Поясніть властивості й види відношень.

6) Поясніть зміст використання потенційних і первинних ключів відносин. Як використовувати ключі для зв'язку відношень?

7) Поясніть операції відновлення відношень.

8) Яким образом реалізується підтримка цілісності бази даних?

9) Дайте визначення двох основних правил цілісності бази даних.

10) Які існують шляхи збереження цілісності бази даних при різних операціях модифікації даних?

Для узагальнення та перевірки засвоєного матеріалу на лекції

11) Поясніть своїми словами зміст термінів:

- надмірність даних;
- аномалії включення, відновлення, видалення;
- сторонній атрибут;

- нормалізація відносин;
- перша нормальна форма;
- функціональні залежності;
- друга нормальна форма;
- транзитивні залежності;
- третя нормальна форма;

12) Приведіть приклади відносин, що не перебувають у першій нормальній формі.

13) Розглянете відношення з нав'язаною функціональною залежністю. Продемонструйте алгоритм виявлення функціональної залежності.

14) Що таке транзитивна залежність? До яких негативних наслідків приводить наявність транзитивної залежності? Які міри допомагають позбутися від цих наслідків?

15) Якщо таблиця перебуває в другій нормальній формі, то ця таблиця

- а) перебуває в першій нормальній формі;
- б) перебуває в третій нормальній формі;
- в) не містить транзитивних залежностей;
- г) містить атрибути, які не є повністю функціонально залежними від ключа.

ключа.

16) Якщо таблиця перебуває в другій нормальній формі, то ця таблиця

- а) перебуває в нормальній формі Бойс-Кодда;
- б) містить неатомарні атрибути;
- в) не містить транзитивних залежностей;
- г) містить атрибути, які не є повністю функціонально залежними від ключа.

ключа.

17) Трьома видами аномалій є аномалії

- а) вставки, вибору, видалення;
- б) вставки, модифікації, видалення;
- в) вибору, модифікації, видалення.

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Файли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Файли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 4
з навчальної дисципліни «Бази даних»

Тема Побудова ER – діаграми реляційної бази даних.

Мета Придбання практичних навичок побудови ER – діаграми реляційної бази даних

Час –2 години

Теоретичні відомості

Побудувати реляційну базу даних на основі ПрО представленої в лабораторній роботі №3.

а) ER-діаграма реляційної бази даних.

1) Відношенню Співробітник (Код_співробітника, ПІБ_співробітника, Посада, Номер_відділу) привласнимо ім'я **Empl** й введемо наступні імена атрибутів:

- **e_id** - кодовий номер співробітника (ключовий атрибут таблиці);
- **e_name** - прізвище ім'я та по батькові співробітника;
- **job** - посада;
- **d_id** - номер відділу, у якому працює співробітник. Виконує роль

зовнішнього ключа для зв'язку з таблицею «Відділ».

2) Відношенню Кваліфікація (Код_співробітника, Значення кваліфікації) привласнимо ім'я **Skills** й уведемо наступні імена атрибутів:

- **e_id** - кодовий номер співробітника (Код_співробітника);
- **skill** - значення кваліфікації співробітника.

Обидва атрибути є первинним ключем таблиці. Крім того, атрибут **e_id** виконує роль зовнішнього ключа для зв'язку з таблицею «Співробітник».

3) Відношенню Відділ (Номер_відділу, Назва_відділу) привласнимо ім'я **Dep** й введемо наступні імена атрибутів:

- **d_id** - номер відділу. Є ключовим атрибутом таблиці.
- **d_name** - найменування відділу

4) Відношенню Проект (Код_проекту, Тема_проекту, Код_співробітника, Код_клієнта, Дата_початку, Строк_виконання, Вартість) привласнимо ім'я **Projects** й уведемо наступні імена атрибутів:

- **pr_id** - унікальний код проекту (номер укладеного договору на розробку проекту). Буде виконувати роль первинного ключа таблиці;
- **pr_theme** - назва теми проекту;
- **e_id** - код співробітника, що є керівником проекту. Буде виконувати роль зовнішнього ключа, що вказує на таблицю Empl для одержання інформації про керівника проекту;
- **cl_id** - код клієнта, що є замовником даного проекту. Буде виконувати роль зовнішнього ключа, що вказує на таблицю Client для одержання інформації про те, який клієнт є замовником проекту.
- **data_start** - дата укладання договору на розробку проекту;
- **data_finish** - строк виконання проекту;
- **price** - вартість проекту.

5) Відношенню Проект_Співробітник (Код_проекту, Код_співробітника, Частка_участі) привласнимо ім'я **Proj_Emp** й уведемо наступні імена атрибутів:

- **pr_id** - код проекту;
- **e_id** - код співробітника, зайнятого в проекті;

Ці два атрибути будуть формувати складений первинний ключ, а окремо кожний з них буде зовнішнім ключем, що вказує на відповідні таблиці Projects й Empl.

- **royalty_share** буде показувати частку співробітника в загальному гонорарі за проект.

6) Відношенню Клієнт (Код_клієнта, Прізвище_керівника, Адреса, Телефон) привласнимо ім'я Client й уведемо наступні імена атрибутів:

- **cl_id** - унікальний код клієнта. Є ключовим атрибутом таблиці;
- **cl_name** - найменування клієнта;
- **addr** - адреса фірми-клієнта;

– **tel** -телефон.

Дано бази даних ім'я **Organization**. Схема бази даних наведена на рисунку 4.1.

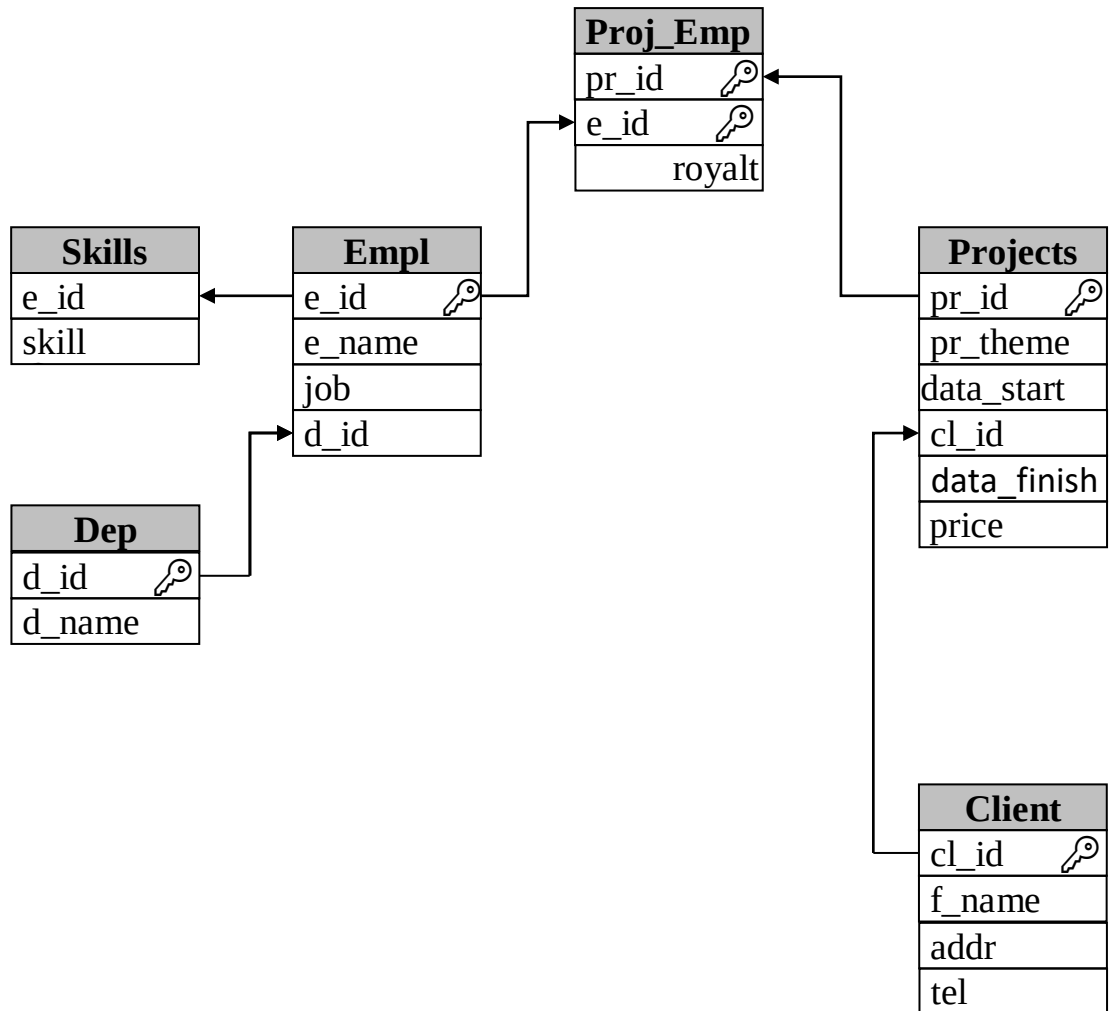


Рисунок 4.1 - ER-діаграма ПрО

Хід роботи

I На основі лабораторної роботи № 3 побудувати ER-діаграму реляційної бази даних

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) ER-діаграма реляційної бази даних.

3) Висновок.

Контрольні питання

- 1) Де і навіщо потрібно створення ER-діаграми?
- 2) У чому полягає відмінність між двома шляхами дослідження предметної області?
- 3) На основі яких підстав можна встановити ключові атрибути об'єктів?
- 4) Що таке супроводження та оновлення системи баз даних?
- 5) На яких етапах важливо врахування вимог замовника?

Література

- | |
|--|
| 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с. |
| 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил. |
| 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с. |

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 5
з навчальної дисципліни «Бази даних»

Тема Створення бази даних, таблиць та індексів

Мета Засобами MySQL навчитися створювати базу даних і визначати
структуру таблиць, використовуючи оператори CREATE DATABASE й
CREATE TABLE.

Час –2 години

Теоретичні відомості

1) Створення бази даних.

Після розробки структури було б логічно повідомити MySQL про те, що ми хочемо створити нову базу даних. Це робиться за допомогою оператора SQL.

```
CREATE DATABASE ім'я бази даних;  
create database organization;
```

Переконавшись в тому, що цей оператор виконав своє завдання, можна за допомогою команди: `show databases;` Ви повинні побачити базу даних `organization` у списку баз даних, наявних у системі.

Отже, тепер ми маємо порожню базу даних, що очікує створення своїх таблиць.

Перш ніж одержати можливість створювати таблиці або робити щось інше з нашою новою базою даних, ми повинні повідомити MySQL, що хочемо працювати із цією базою даних. Для цього використається оператор `use`:

```
use organization;
```

Тепер база даних `organization` обрана, і всі дії, які ми будемо вживати надалі, за замовчуванням будуть застосовуватися саме до цієї бази даних.

2) Створення таблиць

Для створення таблиць у базі даних використається оператор SQL

CREATE TABLE.

У своїй типовій формі цей оператор виглядає так:

```
create table ім'я_таблиці (визначення таблиці) [type=тип_таблиці];
```

Як бачите, необхідно почати зі слів `create table` (створити таблицю), за яких повинне впливати ім'я таблиці, а потім указати деякий набір визначень для стовпців. Наприкінці оператора можна вказати тип механізму зберігання даних, що ми воліємо використати.

Тепер, після того як ми вивчили приклад, розглянемо повний синтаксис оператора `CREATE TABLE`. У посібнику з MySQL приводиться наступна загальна форма цього оператора:

```
create [temporary] table [if not exists] ім'я_таблиці  
[(визначення_стовпця, ...)] [опції_таблиці] [оператор_select]  
або
```

```
create [temporary] table [if not exists] ім'я_таблиці  
like ім'я_старої_таблиці;
```

визначення_стовпця:

```
ім'я_стовпця тип [not null | null] [default значення]  
[auto_increment] [primary key] [визначення_посилання]  
або primary key (ім'я_стовпця_індексу, ...)  
або key [ім'я_індексу] (ім'я_стовпця_індексу, ...)  
або index [ім'я_індексу] (ім'я_стовпця_індексу, ...)  
або unique [index] [ім'я_індексу] (ім'я_стовпця_індексу, ...)  
або fulltext [index] [ім'я_індексу] (ім'я_стовпця_індексу, ...)  
або [constraint символ, foreign key [ім'я_індексу]  
(ім'я_стовпця_індексу, ...)] [визначення_посилання]  
або CHECK (вираження)
```

В якості прикладу розглянемо команди створення таблиць бази даних Organization, яка була спроектована в лабораторній роботі №4.

а) Опис таблиць бази даних Organization.

```
empl ( e_id, e_name, job, d_id )  
dep ( d_id, d_name ) -----
```

skills(e_id, skill)

Client(cl_id, f_name, addr, tel)

projects(pr_id, pr_theme, e_id, cl_id, data_start, data_finish, price)

proj_emp(pr_id, e_id, royālty_shāre)

Створення таблиці dep

```
mysql> create table dep
-> (d_id int(2) not null primary key,
-> d_name varchar(30) not null);
Query OK, 0 rows affected (0.60 sec)
```

```
mysql> describe dep;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| d_id  | int(2)        | NO   | PRI | NULL    |       |
| d_name| varchar(30)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.05 sec)
```

Створення таблиці empl

```
mysql> create table empl
-> (e_id int(3) not null primary key,
-> e_name varchar(15) not null,
-> job varchar(15) not null,
-> d_id int(2) not null references dep(d_id));
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> describe empl;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| e_id  | int(3)        | NO   | PRI | NULL    |       |
| e_name| varchar(15)   | NO   |     | NULL    |       |
| job   | varchar(15)   | NO   |     | NULL    |       |
| d_id  | int(2)        | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)
```


Створення таблиці **skills**

```
mysql> create table skills
-> (e_id int(3) not null references empl(e_id),
-> skill varchar(10),
-> primary key (e_id, skill));
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> describe skills;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| e_id  | int(3)        | NO   | PRI | NULL    |       |
| skill | varchar(10)   | NO   | PRI |         |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)
```

Створення таблиці **client**

```
mysql> create table client
-> (cl_id int(2) not null primary key,
-> f_name varchar(15) not null,
-> addr varchar(50),
-> tel char(10));
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> describe client
-> ;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cl_id | int(2)        | NO   | PRI | NULL    |       |
| f_name | varchar(15)   | NO   |     | NULL    |       |
| addr  | varchar(50)   | YES  |     | NULL    |       |
| tel   | char(10)      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Створення таблиці **projects**

```
mysql> create table projects
-> (pr_id int(3) not null primary key,
-> pr_theme varchar(50) not null,
-> e_id int(3) not null references empl(e_id),
-> cl_id int(3) not null references client(cl_id),
-> data_start date not null,
-> data_finish date,
-> price float(7,2));
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> describe projects
-> ;
```

Field	Type	Null	Key	Default	Extra
pr_id	int(3)	NO	PRI	NULL	
pr_theme	varchar(50)	NO		NULL	
e_id	int(3)	NO		NULL	
cl_id	int(3)	NO		NULL	
data_start	date	NO		NULL	
data_finish	date	YES		NULL	
price	float(7,2)	YES		NULL	

```
7 rows in set (0.01 sec)
```

Створення таблиці **proj_emp**

```
mysql> create table proj_emp
-> (pr_id int(3) not null references projects(pr_id),
-> e_id int(3) not null references empl(e_id),
-> royalty_share float(2,1));
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> describe proj_emp;
```

Field	Type	Null	Key	Default	Extra
pr_id	int(3)	NO		NULL	
e_id	int(3)	NO		NULL	
royalty_share	float(2,1)	YES		NULL	

```
3 rows in set (0.01 sec)
```

Виведення списку таблиць бази даних.

```
mysql> show tables;
```

Tables_in_organization
client
dep
empl
proj_emp
projects
skills

```
6 rows in set (0.02 sec)
```

Хід роботи

I Використовуючи результат виконання лабораторної роботи № 4, створити відповідну Вашому варіанту базу даних і таблиці бази даних. За допомогою команди `describe ім'я_таблиці` одержати MySQL-інформацію про структуру створених таблиць.

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Опис таблиць бази даних.
- 3) Для кожної створюваної таблиці - лістинг команди `create table ім'я_таблиці;` і скриншот результату виконання команди `describe ім'я_таблиці;`.
- 4) Висновок.

Контрольні питання

- 1) Яке з наступних імен неприпустимо для таблиць MySQL?
 - a. `employee;`
 - б. `select;`
 - в. `employee . skill;`
 - г. `employeeSkills.`
- 2) Які з наступних тверджень відносно CHAR й VARCHAR коректні?
 - a. стовпець CHAR, незалежно від його вмісту, завжди займає той самий обсяг дискового простору;
 - б. значення VARCHAR доповнюються пробілами, коли зберігаються на диску;
 - в. стовпець CHAR у середньому займає менше місця, чим аналогічний стовпець VARCHAR;
 - г. стовпець VARCHAR, незалежно від його вмісту, завжди займає той самий обсяг дискового простору.

3) Перш ніж створювати таблиці в базі даних, необхідно:

- а. створити індекси для таблиці;
- б. створити цю базу даних;
- в. створити цю базу даних і вибрати її для використання;
- г. створити всі стовпці таблиці.

4) Який з наступних операторів CREATE TABLE синтаксично коректний?

а. create table department

departmenti int not null auto_increment

primary key,

name varchar(20)

type=InnoDB;

б. create table department type=InnoDB

(departmenti int not null auto_increraent

primary key,

name varchar(20));

в. create department

(departmenti int not null auto_increment

primary key,

name varchar(20)) type=InnoDB;

г. create table department

(departmenti int not null auto_increment

primary key,

name varchar(20)) type=InnoDB.

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Файли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Файли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 6
з навчальної дисципліни «Бази даних»

Тема Модифікація таблиць бази даних

Мета Використовуючи оператор ALTER TABLE навчитися змінювати
структуру таблиць бази даних

Час –2 години

Теоретичні відомості

Зміна структури таблиць бази даних – оператор ALTER TABLE/

Для зміни структури існуючої таблиці стандартом ISO передбачено оператор ALTER TABLE. Визначення цього оператора містить шість параметрів для виконання таких операцій:

- вставка в таблицю нового стовпчика;
- видалення стовпчика з таблиці;
- видалення з визначення таблиці існуючого обмеження;
- визначення для зазначеного стовпчика значення за змовчуванням;
- відміна встановленого значення за змовчуванням для даного

стовпчика.

Оператор ALTER TABLE має такий формат:

ALTER TABLE table_name

[ADD [COLUMN] column_name data_type[NOT NULL] [UNIQUE]

[DEFAULT default_option] [CHECK (search_condition)]]

[DROP [COLUMN] column_name[RESTRICT | CASCADE]]

[ADD [CONSTRAINT [constrain_name]]table_constraint_definition]

[DROP CONSTRAINT constrain_name[RESTRICT | CASCADE]]

[ALTER [COLUMN] SET DEFAULT default_option]

[ALTER [COLUMN] DROP DEFAULT]

Тут параметри мають ті ж призначення, що і в операторі CREATE TABLE. Параметр table_constraint_definition може приймати одне з таких

значень: PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK. У фразі DROP COLUMN *column_name* задається ім'я стовпчика, який видаляється з таблиці. Кваліфікатор RESTRICT означає, що якщо на даний стовпчик посилається який небудь інший об'єкт бази, він з таблиці не видалиться; кваліфікатор CASCADE означає, що крім стовпчика вказаної таблиці будуть видалені всі посилання на даний стовпчик в базі. За змовчуванням діє кваліфікатор RESTRICT.

Оператор ALTER TABLE реалізовано не у всіх діалектах SQL. В деяких діалектах підтримується урізаний варіант цього оператора (не дозволяється видаляти вже існуючі стовпчики).

Оператор ALTER TABLE використовується для додавання або зміни параметрів стовпців бази даних. Для додавання стовпців використовується умова ADD, а для зміни – CHANGE. При додаванні стовпчика необхідно вказати ім'я таблиці, ім'я стовпчика, його тип та довжину. В операторі також можна задати місце додавання стовпчика. Якщо стовпчик вставляється на перше місце задається опція FIRST, якщо після деякого стовпчика, то задається опція AFTER <назва стовпчика після якого вставляємо>. Якщо опцій розташування стовпчика не вказано, то він розташовується на останньому місці. Приклади

```
ALTER TABLE student ADD id2 INT NOT NULL AFTER student_id ;
```

```
ALTER TABLE student ADD id_final INT(6) ;
```

```
ALTER TABLE student ADD id1 INT(8) FIRST ;
```

Модифікація типу стовпчика можлива лише тоді, коли стовпчик пустий. Якщо стовпчик заповнений, то його довжину не можна зменшувати. При зміні стовпчика спочатку вказується попередня назва стовпчика, а потім – його нова назва. Приклади

```
ALTER TABLE student CHANGE id2 id3 VARCHAR(8) NOT NULL ;
```

```
ALTER TABLE student CHANGE id3 id3 VARCHAR(20) ;
```

Для усунення стовпчиків використовують умову DROP

```
ALTER TABLE student DROP id2;
```

Хід роботи

I Використовуючи результат виконання лабораторної роботи № 5, за допомогою оператора `ALTER TABLE`, ввести зміни в структуру створених таблиць.

III Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Опис таблиць бази даних.
- 3) Для кожної зміни таблиці - лістинг команди `alter table ім'я_таблиці`; і скриншот результату виконання команди `describe ім'я_таблиці`;
- 4) Висновок.

Контрольні питання

- 1) Яке з наступних імен неприпустимо для таблиць MySQL?
 - а. `employee`;
 - б. `select`;
 - в. `employee . skill`;
 - г. `employeeSkills`.
- 2) Які з наступних тверджень відносно `CHAR` й `VARCHAR` коректні?
 - а. стовпець `CHAR`, незалежно від його вмісту, завжди займає той самий обсяг дискового простору;
 - б. значення `VARCHAR` доповнюються пробілами, коли зберігаються на диску;
 - в. стовпець `CHAR` у середньому займає менше місця, чим аналогічний стовпець `VARCHAR`;
 - г. стовпець `VARCHAR`, незалежно від його вмісту, завжди займає той самий обсяг дискового простору.
- 3) Перш ніж створювати таблиці в базі даних, необхідно:
 - а. створити індекси для таблиці;

б. створити цю базу даних;

в. створити цю базу даних і вибрати її для використання;

г. створити всі стовпці таблиці.

4) Який з наступних операторів CREATE TABLE синтаксично коректний?

а. create table department

departmenti int not null auto_increment

primary key,

name varchar(20)

type=InnoDB;

б. create table department type=InnoDB

(departmenti int not null auto_increraent

primary key,

name varchar(20));

в. create department

(departmenti int not null auto_increment

primary key,

name varchar(20)) type=InnoDB;

г. create table department

(departmenti int not null auto_increment

primary key,

name varchar(20)) type=InnoDB.

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Файли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Файли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії програмної
інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 7
з навчальної дисципліни «Бази даних»

Тема Введення даних в таблиці

Мета Навчитися вводити дані використовуючи оператори INSERT й
LOAD DATA INFILE

Час –2 години

Теоретичні відомості

1 Введення даних у таблиці

Використання INSERT.

Оператор SQL INSERT використовується для додавання рядків у таблиці.

Загальна форма оператора INSERT з посібника з MySQL виглядає так:

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]

[INTO] ім'я_таблиці [(ім'я_стовпця, . . .)]

VALUES ((вираження | DEFAULT), . . .), (. . .), . . .

[ON DUPLICATE KEY UPDATE ім'я_стовпця=вираження, . . .]

або *INSERT [LOW_PRIORITY | DELAYED] [IGNORE]*

[INTO] ім'я_таблиці [(ім'я_стовпця, . . .)]

SELECT . . .

або *INSERT [LOW_PRIORITY | DELAYED] [IGNORE]*

[INTO] ім'я_таблиці

SET ім'я_стовпця=(вираження | DEFAULT), . . .

[ON DUPLICATE KEY UPDATE ім'я_стовпця=вираження, . . .]

Завантаження даних за допомогою LOAD DATA INFILE

Команда LOAD DATA INFILE дозволяє вставляти дані з текстового файлу в одну таблицю без необхідності використання операторів INSERT. Наприклад, можна було б заповнити даними таблицю dep, використовуючи наступний підхід. На вставці показаний уміст файлу dep.txt з інформацією про відділи.

В якості прикладу розглянемо введення даних в таблиці бази даних Organization.

Таблиця **dep**

Вміст текстового файлу з даними таблиці представлено на рисунку 7.1:

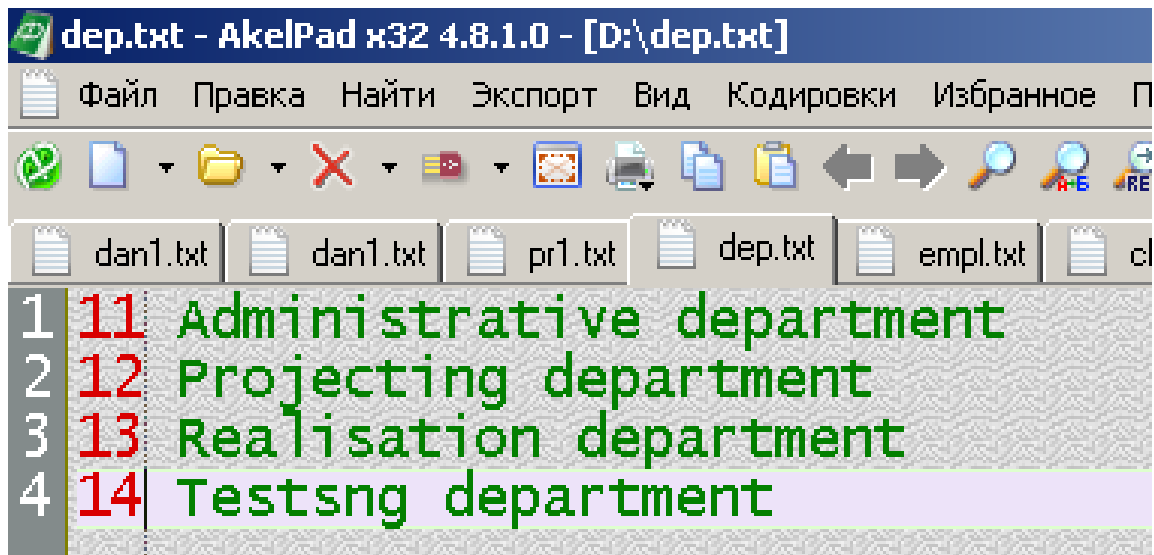


Рисунок 7.1 – Вміст текстового файлу dep

Перегляд вмісту таблиці після занесення в неї даних представлено на рисунку 7.2.



Рисунок 7.2 – Результат занесення даних в таблицю dep

Таблиця **empl**

Вміст текстового файлу з даними таблиці наведено на рисунку 7.3.

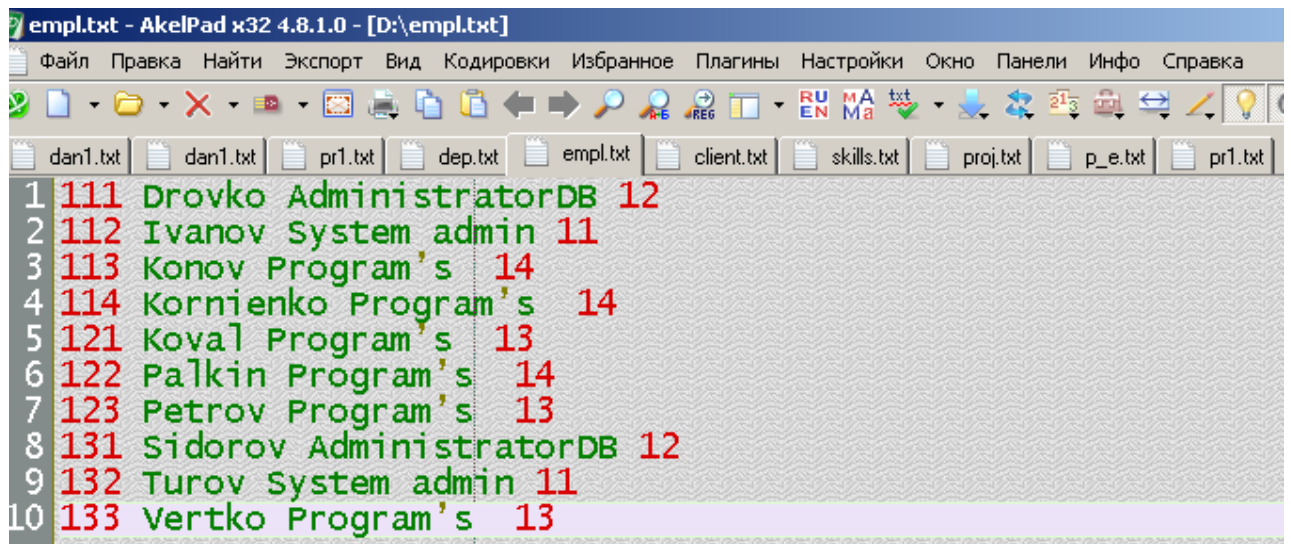


Рисунок 7.3 – Вміст текстового файлу empl

Перегляд вмісту таблиці після занесення в неї даних представлено на рисунку 7.4.

```

mysql> load data infile 'd:\empl.txt' into table empl
Query OK, 10 rows affected (0.05 sec)
Records: 10 Deleted: 0 Skipped: 0 Warnings: 0

mysql> select * from empl;
+----+-----+-----+-----+
| e_id | e_name   | job              | d_id |
+----+-----+-----+-----+
| 111 | Drovko   | AdministratorDB  | 12   |
| 112 | Ivanov   | System admin    | 11   |
| 113 | Konov    | Program's       | 14   |
| 114 | Kornienko | Program's       | 14   |
| 121 | Koval    | Program's       | 13   |
| 122 | Palkin   | Program's       | 14   |
| 123 | Petrov   | Program's       | 13   |
| 131 | Sidorov  | AdministratorDB  | 12   |
| 132 | Turov    | System admin    | 11   |
| 133 | Vertko   | Program's       | 13   |
+----+-----+-----+-----+
10 rows in set (0.00 sec)

```

Рисунок 7.4 – Результат занесення даних в таблицю empl

Таблиця **skills**

Перегляд вмісту таблиці після занесення в неї даних представлено на рисунку 7.5.

```
Командная строка - d:\mysql\bin\mysql -u root -p

mysql> select * from skills;
+----+-----+-----+
| e_id | skill | license |
+----+-----+-----+
| 111 | MySQL | NULL |
| 111 | Linux | NULL |
| 111 | Java | NULL |
| 111 | UML | NULL |
| 112 | Windows | NULL |
| 112 | Linux | NULL |
| 112 | Java | NULL |
| 112 | NT | NULL |
| 113 | Linux | NULL |
| 113 | Java | NULL |
| 113 | NT | NULL |
| 113 | C++ | NULL |
| 113 | Peri | NULL |
| 114 | NT | NULL |
| 114 | C++ | NULL |
| 114 | Peri | NULL |
| 121 | Java | NULL |
| 121 | NT | NULL |
| 121 | C++ | NULL |
| 122 | C++ | NULL |
+----+-----+-----+
| 122 | MySQL | NULL |
| 122 | UML | NULL |
| 123 | Java | NULL |
| 123 | NT | NULL |
| 123 | C++ | NULL |
| 123 | Peri | NULL |
| 131 | MySQL | NULL |
| 131 | Linux | NULL |
| 131 | Java | NULL |
| 131 | NT | NULL |
| 132 | NT | NULL |
| 132 | Java | NULL |
| 132 | Linux | NULL |
| 132 | UML | NULL |
| 133 | UML | NULL |
| 133 | C++ | NULL |
| 133 | Peri | NULL |
+----+-----+-----+
37 rows in set (0.00 sec)

mysql>
```

Рисунок 7.5 – Результат занесения данных в таблицу skills

Створення таблиці client

Вміст текстового файлу з даними таблиці представлено на рисунку 7.6.

```
client.txt - AkelPad x32 4.8.1.0 - [D:\client.txt]
Файл Правка Найти Экспорт Вид Кодировки Избранное Плагины Настройки Окно Панели Инфо Справка
dan1.txt dan1.txt pr1.txt dep.txt empl.txt client.txt skills.txt proj.txt p_e.txt pr1.txt
1 25 Strogof 10945 Brigge Rd. Oacland 0563657298
2 48 Tvorogenko 578 Bljnde St. Rockville 0456783458
3 77 Dragov 5768 Seventh Av. Gary 0567234852
```

Рисунок 7.6 – Вміст текстового файлу client

Перегляд вмісту таблиці після занесення в неї даних представлено на рисунку 7.7.

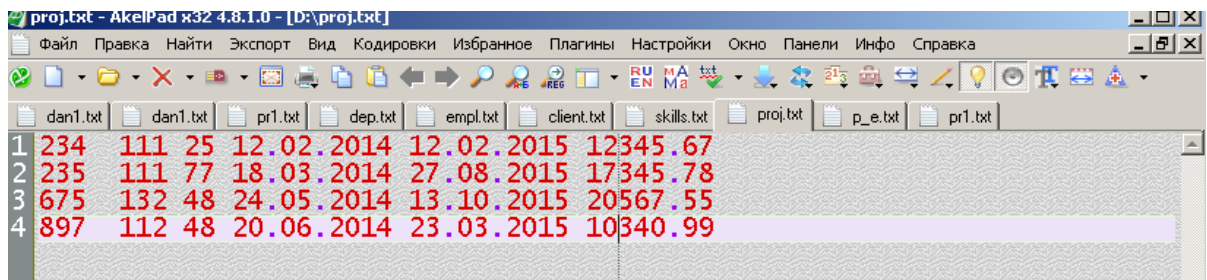
```
mysql> select * from client;
+-----+-----+-----+-----+
| cl_id | f_name | addr | tel |
+-----+-----+-----+-----+
| 25 | Strogof | 10945 Brigge Rd. Oacland | 0563657298 |
| 48 | Ivorogenko | 578 Bljnde St. Rockville | 0456783458 |
| 77 | Dragov | 5768 Seventh Av. Gary | 0567234852 |
+-----+-----+-----+-----+
3 rows in set (0.07 sec)

mysql>
```

Рисунок 7.7 – Результат занесення даних в таблицю client

Таблиця projects

Вміст текстового файлу з даними таблиці представлено на рисунку 7.8.



```
proj.txt - AkelPad v32 4.8.1.0 - [D:\proj.txt]
Файл Правка Найти Экспорт Вид Кодировки Избранное Плагины Настройки Окно Панели Инфо Справка
dan1.txt dan1.txt pr1.txt dep.txt empl.txt client.txt skills.txt proj.txt p_e.txt pr1.txt
1 234 111 25 12.02.2014 12.02.2015 12345.67
2 235 111 77 18.03.2014 27.08.2015 17345.78
3 675 132 48 24.05.2014 13.10.2015 20567.55
4 897 112 48 20.06.2014 23.03.2015 10340.99
```

Рисунок 7.8 – Вміст текстового файлу projects

Перегляд вмісту таблиці після занесення в неї даних представлено на рисунку 7.9.

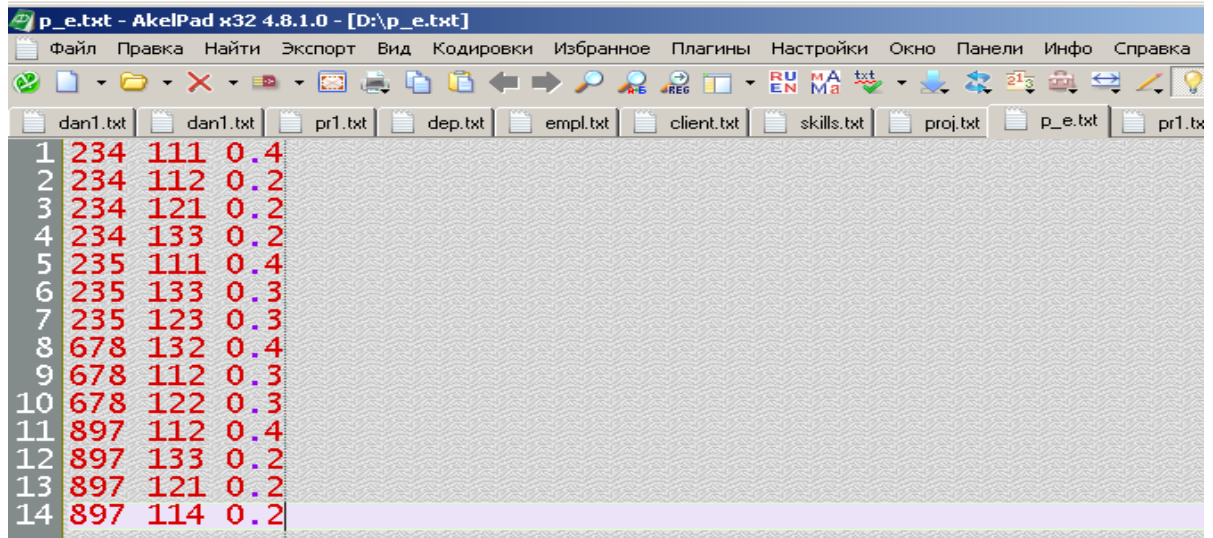
```
mysql> select * from projects;
+-----+-----+-----+-----+-----+-----+-----+
| pr_id | pr_theme | e_id | cl_id | data_start | data_finish | price |
+-----+-----+-----+-----+-----+-----+-----+
| 234 | | 111 | 25 | 2012-02-20 | 2012-02-20 | 12345.67 |
| 235 | | 111 | 77 | 2018-03-20 | 2027-08-20 | 17345.78 |
| 675 | | 132 | 48 | 2024-05-20 | 2013-10-20 | 20567.55 |
| 897 | | 112 | 48 | 2020-06-20 | 2023-03-20 | 10340.99 |
+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.08 sec)

mysql>
```

Рисунок 7.9 – Результат занесення даних в таблицю projects

Таблиця **proj_emp**

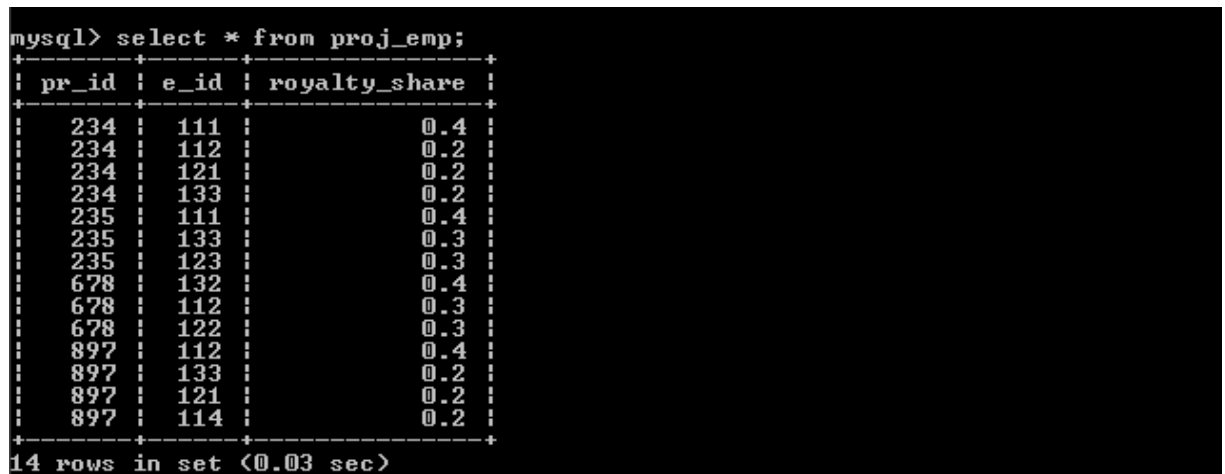
Вміст текстового файлу з даними таблиці представлено на рисунку 7.10.



1	234	111	0.4
2	234	112	0.2
3	234	121	0.2
4	234	133	0.2
5	235	111	0.4
6	235	133	0.3
7	235	123	0.3
8	678	132	0.4
9	678	112	0.3
10	678	122	0.3
11	897	112	0.4
12	897	133	0.2
13	897	121	0.2
14	897	114	0.2

Рисунок 7.10 – Вміст текстового файлу **proj_emp**

Перегляд вмісту таблиці після занесення в неї даних представлено на рисунку 7.11.



```
mysql> select * from proj_emp;
```

pr_id	e_id	royalty_share
234	111	0.4
234	112	0.2
234	121	0.2
234	133	0.2
235	111	0.4
235	133	0.3
235	123	0.3
678	132	0.4
678	112	0.3
678	122	0.3
897	112	0.4
897	133	0.2
897	121	0.2
897	114	0.2

14 rows in set (0.03 sec)

Рисунок 7.11 – Результат занесення даних в таблицю **proj_emp**

Хід роботи

І Використовуючи результати виконання лабораторних робіт № 6 та 7, ввести дані в таблиці бази даних за допомогою оператора INSERT з командного рядка MySQL-вікна або з текстового файлу за допомогою оператора LOAD DATA INFILE.

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Опис таблиць бази даних.
- 3) Для кожної створюваної таблиці - лістинг команди insert або скриншот текстового файлу й скриншот результату виконання команди `Select * from ім'я_таблиці`.
- 4) Висновок.

Контрольні питання

1) Який з наступних операторів успішно вставить рядок у таблицю employee?

- a) `insert into employee Values
set employeei=NULL, name='Laura Thomson1, job='Programmer',
departmenti=128;`
- б) `insert employee values
(NULL, 'Laura Thomson1, 'Programmer1, 128) ;`
- b) `insert into employee values
(NULL, Laura Thomson, Programmer, 128);`
- r) `insert employee values
(NULL, 'Laura 0'Leary', 'Programmer1, 128) .`

2) Оператор REPLACE

- a) аналогічний операторові INSERT, за винятком того, що при конфлікті ключів він замінює старий рядок нової;
- б) аналогічний операторові INSERT, за винятком того, що при конфлікті ключів він залишає старий рядок, а нову відкидає;
- в) аналогічний операторові UPDATE, за винятком того, що при конфлікті ключів він замінює старий рядок нової;
- г) аналогічний операторові UPDATE, за винятком того, що при

конфлікті ключів він залишає старий рядок, а нову відкидає.

3) За замовчуванням значення полів у файлах даних розділяються

- а) комами;
- б) пробілами;
- в) знаками табуляції;
- г) каналами.

4) Опція LOCAL в операторі LOAD DATA INFILE повідомляє про те, що:

- а) клієнт і сервер виконуються на одній машині;
- б) файл із даними перебуває на сервері;
- в) файл із даними перебуває на машині клієнта.

Література

1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк

Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.

2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. /

Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.

3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд.

/ К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.

Розглянуто та схвалено

на засіданні циклової комісії

програмної інженерії

Протокол № 7 від 07.02.2023 р.

Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 8
з навчальної дисципліни «Бази даних»

Тема Використання умовних конструкцій IN, LIKE, BETWEEN та
обчислень в SQL запитах. Використання функцій обробки символьних даних.

Мета Одержати навички використання функцій обробки символьних
даних в операторі SELECT.

Час –2 години

Теоретичні відомості

Загальна форма оператора SELECT виглядає так:

SELECT стовпці

FROM таблиці

[WHERE умови]

[GROUP BY група]

[HAVING групові_умови]]

[ORDER BY сортування_стовпців]

[LIMIT межі];

Розглянемо використання оператора SELECT на прикладі реалізації
простих запитів до таблиць баз даних.

Запит 1: виведення всіх записів таблиці empl представлено на
рисунку 8.1.

```
mysql> select * from empl;
+----+-----+-----+-----+
| e_id | e_name   | job              | d_id |
+----+-----+-----+-----+
| 111 | Drovko   | AdministratorDB  | 12   |
| 112 | Ivanov   | System admin    | 11   |
| 113 | Konov    | Programmer       | 14   |
| 114 | Kornienko | Programmer       | 14   |
| 121 | Koval    | Programmer       | 13   |
| 122 | Palkin   | Programmer       | 14   |
| 123 | Petrov   | Programmer       | 13   |
| 131 | Sidorov  | AdministratorDB  | 12   |
| 132 | Turov    | System admin    | 11   |
| 133 | Vertko   | Programmer       | 13   |
+----+-----+-----+-----+
10 rows in set (0.09 sec)
```

Рисунок 8.1 –Результат виконання запиту 1

Запит 2: відбір певних стовпців таблиці (опція FROM) представлено на рисунку 8.2.

```
mysql> select e_name, e_id from empl;;
+-----+-----+
| e_name | e_id |
+-----+-----+
| Drovko | 111  |
| Ivanov | 112  |
| Konov  | 113  |
| Kornienko | 114 |
| Koval  | 121  |
| Palkin | 122  |
| Petrov | 123  |
| Sidorov | 131 |
| Turov  | 132  |
| Vertko | 133  |
+-----+-----+
10 rows in set (0.02 sec)
```

Рисунок 8.2 –Результат виконання запиту 2

Запит 3: псевдоніми для стовпців (опція AS) представлено на рисунку 8.3.

```
mysql> select e_name as employee_Name from empl;
+-----+
| employee_Name |
+-----+
| Drovko        |
| Ivanov        |
| Konov         |
| Kornienko     |
| Koval         |
| Palkin        |
| Petrov        |
| Sidorov       |
| Turov         |
| Vertko        |
+-----+
10 rows in set (0.00 sec)
```

Рисунок 8.3 –Результат виконання запиту 3

Вибір рядків за допомогою опції WHERE.

Запит 4: виведення списку співробітників – програмістів представлено на рисунку 8.3.

```
mysql> select e_id, e_name from empl
-> where job='Programer';
+-----+-----+
| e_id | e_name |
+-----+-----+
| 113  | Konov  |
| 114  | Kornienko |
| 121  | Koval  |
| 122  | Palkin |
| 123  | Petrov |
| 133  | Vertko |
+-----+-----+
6 rows in set (0.01 sec)
```

Рисунок 8.4 –Результат виконання запиту 4

Видалення повторень за допомогою DISTINCT.

Запит 5: виведення без повторень списку посад організації представлено на рисунку 8.5.

```
mysql> select distinct job from empl;  
+-----+  
| job |  
+-----+  
| AdministratorDB |  
| System admin |  
| Programmer |  
+-----+  
3 rows in set (0.12 sec)
```

Рисунок 8.5 –Результат виконання запиту 5

Обмеження результатів пошуку за допомогою LIMIT

Запит 6: висновок перших трьох співробітників організації з відсортованого за абеткою списку всіх співробітників представлено на рисунку 8.6.

```
mysql> select e_name, job from empl  
-> order by e_name asc  
-> limit 3;  
+-----+-----+  
| e_name | job |  
+-----+-----+  
| Drovko | AdministratorDB |  
| Ivanov | System admin |  
| Konov | Programmer |  
+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

Рисунок 8.6 –Результат виконання запиту 6

Використання оператора IN.

Запит 7: вивести ПІБ батьків, які працюють водіями та поварами.

SELECT FIO_ROD FROM RODITELI WHERE RABOTA IN
(‘ВОДИТЕЛЬ’, ‘ПОВАР’)

Використання оператора BETWEEN.

Запит 8: вивести інформацію про батьків з кодом от 2 до 9.

SELECT * FROM RODITELI WHERE KOD_RODITEL BETWEEN 2
AND 10;

Використання оператора LIKE.

Запит 8: вибрати відомості про батьків, ПІБ яких починається на букву «П».

```
SELECT * FROM RODITELI WHERE FAM LIKE 'П%'
```

Хід роботи

I Реалізувати за допомогою оператора SELECT запити до таблиць бази даних з використання функцій символьної обробки даних (запити отримати у викладача).

II Оформити звіт по виконанню лабораторної роботи.

Варіанти завдань

Варіант 1	1 Вивести ПІБ майстрів, які мають телефони оператора МТС. 2 Вивести інформацію по замовленням, які склалися в березні 2022 року. 3 Вивести інформацію по виробам, вартість яких дорівнює від 200 до 600 грн. 4 Вивести номери замовлення в яких було замовлено браслети з певною вагою. 5 Вивести інформацію про ціни на срібло.
Варіант 2	1 Вивести інформацію про товари, ціна яких лежить в діапазоні від 50 до 500 грн. 2 Вивести номери товарів, які поставлялись в вересні 2022 року. 3 Вивести інформацію по продажам, які здійснювались в січні 2023 року. 4 Вивести номери поставок, які здійснювались в термін з січня по вересень 2022 року. 5 Вивести номери поставок, в яких були продані товари з артиклем, в якому присутнє число 48.
Варіант 3	1 Вивести ПІБ клієнтів, які мають телефони оператору Київстар. 2 Вивести інформацію про співробітників з певним ім'ям. 3 Вивести назви послуг, вартість яких від 1000 до 3000 грн. 4 Вивести інформацію по відвідуванням, які були здійснені в квітні 2022 року. 5 Вивести ПІБ клієнтів, які замовляли в якості послуги будь-який вид масажу.
Варіант 4	1 Вивести ПІБ страхувальників, які мають телефони оператору МТС. 2 Вивести на екран інформацію про клієнтів, у яких в номері ПІН присутні цифри 22. 3 Вивести інформацію по договорам, які склалися с лютому 2023

	року. 4 Вивести інформацію по видам страховок, вартість яких від 5 до 10 тис. грн. 1 Вивести інформацію по клієнтам, які проживають за певно. адресою.
Варіант 5	1 Вивести ПІБ водіїв, які мають телефони оператора МТС. 2 Вивести моделі авто, вік яких більше за 5 років. 3 Вивести інформацію по контрактам, які були складені в січні 2023 року. 4 Вивести інформацію по авто, вартість оренди яких за годину складається від 150 до 300 грн. 5 Вивести номери контрактів, в яких орендується авто, в держ.номері якого присутнє число 37 .
Варіант 6	1 Вивести назви дисциплін, в яких кількість годин відведених на лабораторні роботи від 50 до 80 годин. 2 Вивести назви дисциплін, заліки та іспити по яким проходили в травні 2023 року. 3 Вивести інформацію по студентам, які поступили в 2020 році до навчального закладу. 4 Вивести ПІБ студентів, яким виповнилось 16 років. 5 Вивести інформацію про студентів, які проживають в певному місті.
Варіант 7	1 Вивести ПІБ клієнтів, які мають телефони оператора МТС. 2 Вивести номери телефонів всіх клієнтів з певним ім'ям. 3 Вивести дані про оплату, які були здійснені в березні 2023 року. 4 Вивести інформацію про устаткування, вартість оренди яких від 100 до 150 грн. за добу. 1 Вивести дані про усі велосипеди, які надаються в оренду.
Варіант 8	1 Вивести інформацію про групи, в шифрі яких присутня аббревіатура ПЗ. 2 Вивести інформацію про спеціальності, які належать до відділення КПП. 3 Вивести абітурієнтів, які успішно здали сесію. 4 Визначити групи, у яких куратор з ім'ям Світлана. 6 Визначити групи, у яких в кодї спеціальності присутня цифра 12.
Варіант 9	1 Вивести ПІБ співробітників, які мають телефони оператора МТС. 2 Вивести на екран інформацію про клієнтів, з певним ім'ям. 3 Вивести інформацію про авто, ціна яких лежить в діапазоні від 40 до 60 тисяч. доларів 4 Вивести інформацію про договори, які склалися в березні 2023 року. 5 Вивести інформацію про доукомплектацію, в складі назви якої присутнє певне слово.
Варіант 10	1 . Вивести ПІБ співробітників, які проживають на певній вулиці. 2 Вивести дані по кредитах, максимальна сума кредитування по яким лежить в діапазоні від 100 до 500 тисяч. 3 Вивести на екран інформацію про клієнтів, у яких номер телефону оператора МТС. 4 Вивести інформацію по договорам, які склалися в березні 2023 року.

	5 Вивести ПІБ клієнтів в номері ПІН яких присутнє число 48.
Варіант 11	1 Вивести ПІБ співробітників, які мають телефони оператора МТС. 2 Вивести інформацію по клієнтам вік яких від 20 до 40 років. 3 Вивести інформацію по клієнтам ПІН яких має в номері число 29. 4 Вивести інформацію по контрактам, які складались в січні 2023 року. 5 Вивести назви турів, ціна яких лежить в діапазоні від 500 до 1000 доларів.
Варіант 12	1 Вивести інформацію про блюда, вартість яких лежить в діапазоні від 100 до 250 грн. 2 Вивести ПІБ клієнтів, які мають телефони оператора МТС. 3 Вивести інформацію про замовлення, які були здійснені в квітні 2023 року. 4 Вивести інформацію по замовленням клієнтів з певним ім'ям. 5 Вивести інформація по блюдам, які мають енергетичну цінність 200, 250 та 300 кКал
Варіант 13	1 Вивести інформацію на екран про ціну усіх видів костюмів. 2 Вивести ПІБ співробітників, які мають телефони оператора Київстар. 3 Вивести інформацію про всіх клієнтів, які проживають в певному місті. 4 Вивести назву усіх виробів, ціна яких лежить в діапазоні від 2 до 5 тисяч грн. 5 Вивести номер всіх замовлень, які будуть оброблятися не більше 20 днів.
Варіант 14	1 Вивести інформацію про товари, ціна яких лежить в діапазоні від 100 до 300 грн. 2 Вивести номери поставок, які здійснювалися в термін з вересня по грудень 2022 року. 3 Вивести інформацію по продажам, які здійснювались в січні 2023 року. 4 Вивести номери поставок, які здійснювалися в термін з січня по вересень 2022 року. 5 Вивести номери поставок, в яких були продані товари з артиклем, в якому присутнє число 23.
Варіант 15.	1 Вивести ПІБ клієнтів, які мають телефони оператора Київстар. 2 Вивести інформацію про співробітників з певним ім'ям. 3 Вивести назви рейсів, вартість білетів на які від 1000 до 3000 грн. 4 Вивести інформацію по рейси, які були здійснені в квітні 2023 року. 5 Вивести номери рейсів, які відправляються до Києву.
Варіант 16	1 Вивести інформацію про тренерів, які проживають в місті Дніпро. 2 Вивести ПІБ гравців, які мають телефони оператора МТС. 3 Вивести інформацію вартість гравців, які від 10000 до 40000 грн. 4 Вивести інформацію про ігри які відбулись з січня по квітень 2023 року. 5 Вивести інформацію про клуби, які тренує певний тренер.
Варіант 17	1 Вивести інформацію про обладнання в назві якого є слово «холодильне».

	2 Вивести інформацію по договорам покупки обладнання, які склалися в січні 2023 року. 3 Вивести номери цехів, в які закуповувалось обладнання в 2023 році. 4 Вивести інформацію по обладнанню, вартість якого від 50000 до 100000 грн. 6 Вивести інформацію про постачальників, юридична адреса яких знаходиться у м. Дніпро.
Варіант 18	1 Вивести інформацію про співробітників, які проживають на певній вулиці. 2 Вивести інформацію про товар, в яких є слово колесо. 3 Вивести назву товарів ціна яких лежить в діапазоні від 5000 до 20000 грн. 4 Вивести інформацію по контрактам, які були складені в лютому 2023 року. 5 Вивести ПІБ клієнтів, які мають телефони оператора Київстар.
Варіант 19	1 Вивести ПІБ лікарів, які мають телефони оператора Київстар. 2 Вивести інформацію по всіх прийомах, які відбулись в певний термін (від ____ дати до ____). 3 Вивести інформацію по усім пацієнтам, які проживають в певному місті. 4 Вивести коди лікарів, які працюють у вихідні дні. 5 Вивести інформацію по пацієнтам, яким виповнилось більше 40 років
Варіант 20	1 Вивести інформацію про співробітників, які проживають на певній вулиці. 2 Вивести інформацію про номери класу «ЛЮКС». 3 Вивести номера номерів, ціна яких лежить в діапазоні від 1000 до 2000 грн./добу. 4 Вивести інформацію про оренду номерівв лютому 2023 року. 5 Вивести ПІБ клієнтів, які мають телефони оператора Київстар.
Варіант 21	1 Вивести ПІБ клієнтів, які мають телефони оператора МТС. 2 Вивести номери телефонів всіх клієнтів з певним ім'ям. 3 Вивести дані про продажі, які були здійснені в березні 2023 року. 4 Вивести інформацію про меблі, вартість яких від 2000 до 5000 грн. 5 Вивести дані про усі дивани, які мають у продажу.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Формулювання запиту.
- 3) Виведення вмісту таблиці, для якої сформований запит.
- 4) Оператор SELECT до відповідного запиту.
- 5) Копія вікна виконання оператора SELECT.
- 6) Висновок.

Контрольні питання

1) Який з наступних запитів вибере всі рядки з таблиці client?

- a) `select * from client
where cl_id=2;`
- б) `select cl_id, f_name, addr, tel
from client;`
- в) `select * from client
limit 1;`
- г) `select all from client;`

2) Який з наступних запитів вибере всіх програмістів з таблиці empl?

- a) `select *
from empl
where job='Programer' ;`
- б) `select *
from empl
having job=' Programer';`
- в) `select *
from empl
where job=' Programer'
group by job
having job=' Programer' ;`
- г) `select job
from empl;`

3) Не допускається використати псевдоніми

- а) для стовпців;
- б) для таблиць;
- в) у вираженні WHERE;
- г) у вираженні SELECT.

4) Якщо необхідно за допомогою запиту повернути рядка 15-20, коректним вираженням LIMIT буде

- а) LIMIT 15, 20
- б) LIMIT 14, 19
- в) LIMIT 14, 5
- г) LIMIT 15, 5

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М.: СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 9
з навчальної дисципліни «Бази даних»

Тема Особливості використання сортування та групування даних в SQL
запитах, підзапити

Мета Одержати навички використання сортування та групування даних в
SQL запитах та створення простих підзапитів

Час –2 години

Теоретичні відомості

Загальна форма оператора SELECT виглядає так:

SELECT стовпці

FROM таблиці

[WHERE умови]

[GROUP BY група

[HAVING групові_умови]]

[ORDER BY сортування_стовпців]

[LIMIT межі];

Розглянемо використання оператора SELECT на прикладі реалізації запитів з групуванням та сортуванням даних в таблицях.

Використання GROUP BY.

Після ORDER BY <поле> можна вказати напрямок сортування. За замовчуванням записи сортуються по зростанню (ASC), ключове слово DESC задає сортування в порядку, зворотному алфавітному.

Запит 1: підрахунок числа службовців по групах посад. Запит 1 та результат його виконання наведений на рисунку 9.1.

```
mysql> select count(*), job from empl
-> group by job;
```

count(*)	job
2	AdministratorDB
6	Programer
2	System admin

```
3 rows in set (0.00 sec)
```

Рисунок 9.1 – Запит 1 та результат його виконання

Використання HAVING.

Запит 2: виведення наявних у компанії посад, на яких працює рівно по два службовці. Запит 2 та результат його виконання наведений на рисунку 9.2.

```
mysql> select count(*), job from empl
-> group by job having count(*)=2;
```

count(*)	job
2	AdministratorDB
2	System admin

```
2 rows in set (0.00 sec)
```

Рисунок 9.2 – Запит 2 та результат його виконання

Сортування результатів пошуку за допомогою ORDER BY

Запит 3: виведення списку співробітників, відсортованого за посадою. Запит 3 та результат його виконання наведений на рисунку 9.3.

```
mysql> select * from empl
-> order by job asc, t_name desc;
ERROR 1054 (42S22): Unknown column 't_name' in 'order clause'
mysql> select * from empl
-> order by job asc, e_name desc;
```

e_id	e_name	job	d_id
131	Sidorov	AdministratorDB	12
111	Drovko	AdministratorDB	12
133	Uertko	Programer	13
123	Petrov	Programer	13
122	Palkin	Programer	14
121	Koval	Programer	13
114	Kornienko	Programer	14
113	Konov	Programer	14
132	Turov	System admin	11
112	Ivanov	System admin	11

```
10 rows in set (0.01 sec)
```

Рисунок 9.3 – Запит 3 та результат його виконання

У результаті обрані всі рядки з таблиці empl і відсортовані за значенням job за абеткою. Якщо при цьому два або трохи службовці будуть мати однакові посади, то відповідні рядки відсортовані по імені у

зворотному порядку.

Використання функцій AVG, MIN, MAX.

Запит 4: вивести середнє оцінок, максимальну оцінку, мінімальну оцінку студентів згрупованих за номером групи, з вказівкою імені кожного стовпця.

```
SELECT KOD_STUDENT, AVG(OCENKA) AS СРЕДНЯЯ,  
MAX (OCENKA) AS МАКСИМАЛЬНАЯ, MIN (OCENKA) AS  
МИНИМАЛЬНАЯ FROM USPEV GROUP BY KOD_STUDENT
```

Підзапит - це запит на вибірку даних, вкладений в інший запит.

```
SELECT <поля> FROM <таблиця> WHERE (HAVING) УСЛОВИЕ  
(SELECT <поля> FROM <таблиця> WHERE <умова>)
```

В свою чергу підзапит може містити інший підзапит, але в першу чергу виконується підзапит, який має найглибший рівень вкладення.

Часто, але не завжди, зовнішній запит звертається до однієї таблиці, а підзапит - до іншої.

Прості підзапити характеризуються тим, що вони формально ніяк не пов'язані із зовнішніми запитами, які в них містяться, що дозволяє спочатку виконати підзапит, результат якого використовується для виконання зовнішнього запиту.

Три види простих підзапитів:

- підзапити, які повертають єдине значення;
- підзапити, які повертають список значень, з одного стовпчика таблиці;
- підзапити, які повертають набір записів.

Підзапити, які повертають єдине значення

Запит5: вивести оцінки, які більше за середнє значення всіх оцінок.

```
SELECT * FROM USPEV WHERE OCENKA > (SELECT  
AVG(OCENKA) FROM USPEV)
```

Підзапити, які повертають список значень, з одного стовпчика таблиці

Запит 6: визначити коди батьків студента Маркова та Іванова.

```
SELECT      KOD_RODITEL      FROM      RODDETI      WHERE  
KOD_STUDENT IN (SELECT KOD_STUDENT FROM DANNIE WHERE  
FAM='МАРКОВ' OR FAM='ИВАНОВ')
```

Запит 7: визначити назву дисциплін, які здавав студент з кодом 2.

```
SELECT      NAZVANIE      FROM      DISCIPLINA      WHERE  
KOD_DISCIPLINA IN (SELECT KOD_DISCIPLINA FROM USPEV  
WHERE KOD_STUDENT='2')
```

Запит 8: вивести назву дисциплін, за якими отримана оцінка 5.

```
SELECT NAZVANIE FROM DISCIPLINA WHERE 5 IN (SELECT  
OCENKA      FROM      USPEV      WHERE      KOD_DISCIPLINA=  
DISCIPLINA.KOD_DISCIPLINA)
```

Такий підзапит відрізняється від простого підзапиту тим, що вкладений підзапит не може бути оброблений перш, ніж буде оброблятися зовнішній підзапит. Це пов'язано з тим, що вкладений підзапит залежить від значення DISCIPLINA.KOD_DISCIPLINA, а воно змінюється в міру того, як система перевіряє різні рядки таблиці DISCIPLINA. Отже, з концептуальної точки зору обробка здійснюється наступним чином:

1) Система перевіряє перший рядок таблиці DISCIPLINA. Припустимо, що це рядок дисципліни з номером 1. Тоді значення DISCIPLINA.KOD_DISCIPLINA буде в даний момент має значення, рівне 1, і система обробляє внутрішній запит:

```
(SELECT OCENKA FROM USPEV WHERE KOD_DISCIPLINA= 1)
```

отримуючи в результаті безліч (1, 4, 3, 5, 2, 4, 3, 3, 5). Тепер система може завершити обробку для дисципліни з номером 1. Вибірка значення NAZVANIE для KOD_DISCIPLINA = 1 (База даних) буде проведена тоді і тільки тоді, коли OCENKA = 5 буде належати цій множині, що, очевидно, справедливо.

2) Далі система буде повторювати обробку такого роду для наступної дисципліни і т.д. до тих пір, поки не будуть розглянуті всі рядки таблиці DISCIPLINA.

Подібні підзапити називаються пов'язаними/співвіднесеними/корельованими, так як їх результат залежить від значень, визначених в зовнішньому підзапиті. Обробка пов'язаного підзапиту, отже, повинна повторюватися для кожного значення витягується з зовнішнього підзапит, а не виконуватися раз і назавжди.

Хід роботи

I Реалізувати за допомогою оператора SELECT запити з використання групування та сортування даних таблиць бази даних, а також з використанням підзапитів (отримати у викладача згідно ПрО).

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Формулювання запиту.
- 3) Виведення вмісту таблиці, для якої сформований запит.
- 4) Оператор SELECT до відповідного запиту.
- 5) Копія вікна виконання оператора SELECT.
- 6) Висновок.

Контрольні питання

1) Який з наступних запитів не поверне загальне число службовців з таблиці empl?

- a) `select count(e_id) from empl;`
- b) `select count(e_id) as total from empl;`
- c) `select count(distinct e_id) from empl;`
- d) `select count(e_id) from empl group by e_id.`

2) У реченні GROUP BY:

a) визначаються імена таблиці-джерела даних або декількох таблиць;

b) виконується фільтрація рядків об'єкта відповідно до заданих

умов;

с) фільтруються групи рядків об'єкта відповідно до зазначеної

умови;

д) утворюються групи рядків, що мають те саме значення в зазначеному стовпці;

е) утворюються групи рядків, що мають значення, що ввів користувач.

3) Яка SQL команда використовується для упорядкування результатів?

а) ORDER BY;

б) SORT BY;

с) SORT;

д) ORDER;

е) GROUP BY.

4) Як вибрати всі записи з таблиці "Persons", упорядковані по полю "FirstName" в зворотному порядку?

а) SELECT * FROM Persons SORT 'FirstName' DESC;

б) SELECT * FROM Persons ORDER BY FirstName DESC;

с) SELECT * FROM Persons ORDER FirstName DESC;

д) SELECT * FROM Persons SORT BY 'FirstName' DESC;

е) SELECT * FROM PersonsGroup BY 'FirstName' DESC.

5) Конструкція HAVING у запитах/відображеннях задає...

а) поля, що відображатимуться;

б) критерії фільтрів для груп рядків;

с) критерії впорядкування результатів;

д) критерії відображення груп;

е) критерії відображення запитів.

Література

1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк

Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.

2 Файли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. /

Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.

З Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд.

/ К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.

Розглянуто та схвалено

на засіданні циклової комісії

програмної інженерії

Протокол № 7 від 07.02.2023 р.

Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 10
з навчальної дисципліни «Бази даних»

Тема	Реалізація складних запитів SQL: З'єднання таблиць, об'єднання запитів
------	--

Мета	Навчитися з'єднувати таблиці різними способами
------	--

Час –2 години

Теоретичні відомості

Природне з'єднання (NATURAL JOIN).

Декартовим добутком двох таблиць називається таблиця, складена з різноманітних поєднань записів обох таблиць.

Запит 1: вивести декартовий добуток таблиць DANNIE і USPEV.

```
SELECT * FROM DANNIE, USPEV
```

Спільним стовпцем цих таблиць є стовпець kod_student.

В отриманому декартовому добутку цікавлять не всі дані, а тільки ті, в яких ідентичні стовпці мають однакові значення. Крім того, в результативній таблиці не потрібні два ідентичних стовпчика, досить лише одного з них.

```
SELECT DANNIE.*, USPEV.OCENKA FROM DANNIE, USPEV  
WHERE DANNIE.KOD_STUDENT=USPEV.KOD_STUDENT
```

В результаті виконання цього запиту виходить таблиця природним з'єднанням.

Даний запит можна записати, використовуючи псевдоніми.

```
SELECT T1.*, T2.OCENKA FROM DANNIE T1, USPEV T2 WHERE  
T1.KOD_STUDENT=T2.KOD_STUDENT
```

Еквівалентний запит можна скласти за допомогою оператора NATURAL JOIN.

```
SELECT T1.*, T2.OCENKA FROM DANNIE T1 NATURAL JOIN  
USPEV T2.
```

При природному з'єднанні, що виконується за допомогою NATURAL JOIN, перевіряється рівність всіх одноіменних стовпців з'єднуючих таблиць.

Умовне з'єднання (JOIN ... ON).

Умовне з'єднання схоже на з'єднання з умовою рівності. Як умова може виступати будь-який логічний вираз, яке записується після ключового слова ON (при), а не WHERE. Якщо умова виконується для поточного запису декартового добутку, то вона входить у результативну таблицю.

Запит 2. Вивести інформацію про студентів і їх оцінках з дисципліни з кодом 2.

```
SELECT * FROM DANNIE JOIN USPEV ON  
(DANNIE.KOD_STUDENT = USPEV.KOD_STUDENT) AND  
(USPEV.KOD_DISCIPLINA=2)
```

З'єднання по іменах стовпців (JOIN ... USING).

З'єднання по іменах стовпців схоже на природне з'єднання. Відмінність полягає в тому, що можна вказати, які саме однойменні стовпці повинні перевірятися, а при природному перевіряються всі однойменні стовпці.

Запит 3. Вивести в якому місті які вулиці.

```
SELECT * FROM GOROD JOIN ULICA USING (KOD_GOROD)
```

Дані таблиці містять більше одного ідентичного поля, тому природне з'єднання поверне порожній результат.

Запит 4. Запит можна сформулювати інакше:

```
SELECT * FROM GOROD INNER JOIN ULICA ON  
(GOROD.KOD_GOROD= ULICA. KOD_GOROD)
```

З таблиці, одержуваної при внутрішньому сполученні, відбраковуються всі записи, для яких немає відповідного запису одночасно в обох таблицях.

При зовнішньому з'єднанні такі невідповідні записи зберігаються.

Ліве з'єднання (LEFT OUTER JOIN).

При лівому зовнішньому з'єднанні невідповідні записи, наявні в лівій таблиці, зберігаються в результативній, а наявні в правій - видаляються.

Розглянутий в *Запиті 1* запит виводить не всі записи, той же самий запит при зовнішньому лівому об'єднанні виглядає так:

```
SELECT T1.*, T2.OCENKA FROM DANNIE T1 LEFT OUTER JOIN  
USPEV T2
```

ON T1.KOD_STUDENT=T2.KOD_STUDENT

Праве з'єднання (RIGHT OUTER JOIN).

При правом зовнішньому з'єднанні невідповідні записи, наявні в правій таблиці, зберігаються в результативній, а наявні в лівій - видаляються.

Розглянутий в Запиті 1 запит виводить не всі записи, той же самий запит при зовнішньому правому об'єднанні виглядає так:

```
SELECT T1.*, T2.OCENKA FROM DANNIE T1 RIGHT OUTER JOIN  
USPEV T2 ON T1.KOD_STUDENT=T2.KOD_STUDENT
```

Повне з'єднання (FULL JOIN).

Повне з'єднання виконує і ліве, і праве зовнішні з'єднання.

```
SELECT * FROM DANNIE FULL JOIN USPEV
```

Хід роботи

I Реалізувати об'єднання полів з декількох таблиць (отримати у викладача згідно ПрО).

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Формулювання запиту.
- 3) Виведення вмісту таблиць, для яких сформований запит.
- 4) Оператор SELECT до відповідного запиту.
- 5) Копія вікна виконання оператора SELECT.
- 6) Висновок.

Контрольні питання

- 1) Декартовий добуток

а) представляє всі можливі комбінації рядків двох або декількох таблиць;

б) представляє всі можливі комбінації співпадаючих рядків двох або декількох таблиць;

в) представляє рядки однієї таблиці в парі з відповідними рядками іншої таблиці, а де це неможливо, замість рядка другої таблиці використовується рядок зі значень NULL;

г) не описується жодним з попередніх варіантів.

2) Ліве об'єднання

а) представляє всі можливі комбінації рядків двох або декількох таблиць;

б) представляє всі можливі комбінації співпадаючих рядків двох або декількох таблиць;

в) представляє рядки однієї таблиці в парі з відповідними рядками іншої таблиці, а де це неможливо, замість рядка другої таблиці використовується рядок зі значень NULL;

г) не описується жодним з попередніх варіантів.

3) Об'єднання по еквівалентності

а) представляє всі можливі комбінації рядків двох або декількох таблиць;

б) представляє всі можливі комбінації співпадаючих рядків двох або декількох таблиць;

в) представляє рядки однієї таблиці в парі з відповідними рядками іншої таблиці, а де це неможливо, замість рядка другої таблиці використовується рядок зі значень NULL;

г) не описується жодним з попередніх варіантів.

4) Зв'язаний підзапит називається так тому, що він:

а) зв'язує рядка таблиць;

б) зв'язує рядка в одній таблиці;

в) зв'язує два об'єднання;

г) зв'язує рядка зовнішнього запиту з рядками внутрішнього.

5) Різниця між наведеними нижче запитам 5.1 й 5.2 полягає в тім, що:

а) ніякої різниці немає;

- б) вони повертають різні дані;
- в) вони повертають ті самі дані, але leftjoin (запит 5.1), швидше за все, буде виконуватися швидше;
- г) вони повертають ті самі дані, але подзапрос (запит 5.2), швидше за все, буде виконуватися швидше.

Література

- | |
|--|
| 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с. |
| 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил. |
| 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с. |

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 12
з навчальної дисципліни «Бази даних»

Тема Складні запити з використанням опцій ALL, ANY

Мета Навчитися з'єднувати таблиці різними способами

Час –2 години

Теоретичні відомості

Оператори =IN, =ANY істинні, коли є співпадіння хоча б із одним значенням внутрішнього запиту, якому вони передують.

=NOT IN істинне, коли нема спів падіння із жодним значенням, отриманим у запиті.

> ANY, > = ANY істинні коли значення більше або більше-рівне довільного значення внутрішнього запиту.

< ANY, < = ANY істинні коли значення менше або менше-рівне довільного значення внутрішнього запиту

= ALL істинний, коли значення рівне всім значенням внутрішнього запиту. Така ситуація може спостерігатися, коли внутрішній запит повертає лише одне значення.

<> ALL істинний, коли значення не рівне жодному значенню внутрішнього запиту.

> ALL, > = ALL істинні, коли значення більші або більші-рівні всіх значень внутрішнього запиту

< ALL, < = ALL істинні, коли значення менші або менші-рівні всіх значень внутрішнього запиту

Слід враховувати, що коли результат внутрішнього запиту пустий, то оператори (=, >, <) ALL приймають істинне значення, а аналогічні оператори із ANY – хибне.

Нехай потрібно вибрати дані про студентів, які проживають в містах де розташований університет, в якому вони навчаються


```
SELECT * FROM student s WHERE city IN (SELECT city FROM university u WHERE u.univ_id = s.univ_id);
```

Запит, що вибирає назви університетів з рейтингом вищим, ніж рейтинг довільного університету в Тернополі

```
SELECT * FROM university WHERE rating > ALL ( SELECT rating FROM university WHERE city = 'Тернопіль');
```

Хід роботи

I Реалізувати складні запити з використанням опцій ALL, ANY (отримати у викладача згідно ПрО).

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Формулювання запиту.
- 3) Виведення вмісту таблиць, для яких сформований запит.
- 4) Оператор SELECT до відповідного запиту.
- 5) Копія вікна виконання оператора SELECT.
- 6) Висновок.

Контрольні питання

- 1) У яких випадках у підсумковому запиті не використається GROUPBY?
- 2) У чому відмінність використання WHERE від HAVING?
- 3) Які елементи можна використати в списку SELECT підсумкового запиту?
- 4) У яких випадках варто використати оператори IN або NOT IN у підзапитах? Приведіть приклади.
- 5) У яких випадках варто використати оператори EXISTS або NOT EXISTS у підзапитах? Приведіть приклади.

б) Чим SOME й ANY відрізняється від ALL при використанні у вкладених запитах? Приведіть приклади.

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 12
з навчальної дисципліни «Бази даних»

Тема Обробка транзакцій

Мета Отримати навички використання транзакцій

Час –2 години

Теоретичні відомості

Під транзакцією розуміється неподільна з точки зору впливу на БД послідовність операторів маніпулювання даними (читання, видалення, вставки, модифікації), що приводить до одного з двох можливих результатів: або послідовність виконується, якщо всі оператори правильні, або вся транзакція відкочується, якщо хоча б один оператор не може бути успішно виконаний. Обробка транзакцій гарантує цілісність інформації в базі даних. Таким чином, транзакція переводить базу даних з одного цілісного стану в інший.

Методом контролю за транзакціями є ведення журналу, в якому фіксуються всі зміни, що здійснюються транзакцією в БД. Якщо під час обробки транзакції відбувається збій, транзакція відкочується – з журналу встановлюється стан БД на момент початку транзакції.

У СУБД початок транзакції може задаватися явно, наприклад, командою `BEGIN TRANSACTION`, або передбачатися неявно, тобто чергова транзакція відкривається автоматично одразу ж після вдалого чи невдалого завершення попередньої. Для завершення транзакції зазвичай використовують команди SQL:

- `COMMIT` – успішно завершити транзакцію
- `ROLLBACK` – відкотити транзакцію, тобто повернути БД в стан, в якому вона перебувала на момент початку транзакції.

Стандарт SQL визначає, що транзакція починається з першого SQL-оператора, ініційованого користувачем або міститься в прикладній програмі.

Усі наступні SQL-оператори складають тіло транзакції. Транзакція завершується одним з можливих способів:

- оператор COMMIT означає успішне завершення транзакції, всі зміни, внесення в базу даних робляться постійними;
- оператор ROLLBACK перериває транзакцію і скасовує всі внесені нею зміни;
- успішне завершення програми, яка ініціювала транзакцію, означає успішне завершення транзакції (як використання COMMIT);
- помилкове завершення програми перериває транзакцію (як ROLLBACK).

Приклад явно заданої транзакції:

```
BEGIN TRANSACTION;          /*Почати транзакцію*/  
DELETE ...;                  /*Зміни*/  
UPDATE ...;                  /*даних*/  
if (знайдена помилка) ROLLBACK;  
else COMMIT;                 /*Завершити транзакцію*/
```

Приклад неявно заданої транзакції:

```
COMMIT;                      /*Закінчення попередньої транзакції*/  
DELETE ...;                  /*Зміни*/  
UPDATE ...;                  /*даних*/  
if (знайдена помилка) ROLLBACK;  
else COMMIT;                 /*Завершити транзакцію*/
```

Описаний механізм транзакцій гарантує забезпечення цілісного стану бази даних тільки в тому випадку, коли всі транзакції виконуються послідовно, тобто в кожному одиницю часу активна тільки одна транзакція. Якщо роботу з даними ведуть одночасно кілька користувачів, такий спосіб організації обробки запитів не прийнятний, тому що це призведе до збільшення часу реакції системи. У той же час, якщо одночасно виконуються дві транзакції, можуть виникнути такі помилкові ситуації:

- брудне читання (Dirty Read) – транзакція T1 модифікувала якийсь елемент даних. Після цього інша транзакція T2 прочитала вміст цього

елементу даних до завершення транзакції T1. Якщо T1 завершується операцією ROLLBACK, це означає, що транзакція T2 прочитала неіснуючі дані;

- неповторюване (розмите) читання (Non-repeatable or Fuzzy Read) – транзакція T1 прочитала вміст елемента даних. Після цього інша транзакція T2 модифікувала або видалила цей елемент. Якщо T1 прочитає вміст цього елемента знову, то вона отримає інше значення або виявить, що елемент даних більше не існує;

- Фантом (фіктивні елементи) (Phantom) – транзакція T1 прочитала вміст декількох елементів даних, які відповідають деякій умові. Після цього T2 створила елемент даних, який задовольняє цій умові і стала фіксованою. Якщо T1 повторить читання з цією ж умовою, тоді вона отримає інший набір даних.

Жодна з цих ситуацій не може виникнути при послідовному виконанні транзакцій. Звідси виникло поняття серіалізація – здатність до впорядкування паралельної обробки транзакцій. Тобто почергове (паралельне) виконання заданої множини транзакцій буде вірним, якщо при його виконанні буде отриманий той самий результат, як і при послідовному виконанні тих самих транзакцій.

Такі ситуації виникають тільки тому, що виконання транзакцій T1 і T2 не було впорядковано, тобто не було еквівалентно виконанню спочатку транзакції T1, а потім T2, або, навпаки, спочатку транзакції T2, а потім T1.

Примусове впорядкування транзакцій забезпечується за допомогою механізму блокувань. Суть цього механізму в наступному: якщо для виконання деякої транзакції необхідно, щоб деякий об'єкт бази даних (кортеж, набір кортежів, ставлення, набір відносин, і т.д.) не змінювався непередбачувано і без відома цієї транзакції, такий об'єкт блокується.

Основними видами блокувань є:

- блокування з взаємним доступом, також має назву S-блокування (від Shared locks) і блокуванням із читання;

– монопольне блокування (без взаємного доступу), також має назву X-блокування (від eXclusive locks) або блокування із запису. Цей режим використовується при операціях зміни, додавання та видалення об'єктів.

При цьому:

1 Якщо транзакція накладає на об'єкт X-блокування, то будь-який запит іншої транзакції з блокуванням цього об'єкту буде відкинутий.

2 Якщо транзакція накладає на об'єкт S-блокування, тоді:

– запит з боку іншої транзакції з X-блокуванням на цей об'єкт буде відкинутий;

– запит з боку іншої транзакції з S-блокуванням цього об'єкту буде прийнятий.

Доведено, що серіалізація транзакцій або, інакше, їх ізоляція забезпечується при використанні двофазного протоколу блокувань 2LP – Two-Phase Locks, згідно з яким всі блокування, здійснені транзакцією, знімаються тільки при її завершенні. Тобто виконання транзакції розбивається на дві фази: (1) – накопичення блокувань, (2) – звільнення блокувань в результаті фіксації.

Застосування механізму блокування призводить до уповільнення обробки транзакцій, оскільки система змушена очікувати поки звільняться дані, захоплені конкуруючої транзакцією. Вирішити цю проблему можна за рахунок зменшення фрагментів даних, які захоплені транзакцією. Залежно від захоплених об'єктів розрізняють кілька рівнів блокування:

- блокування усієї бази даних;
- блокування окремих таблиць;
- блокування сторінок;
- блокування записів;
- блокування окремих полів.

Сучасні СУБД, можуть здійснювати блокування на рівні записів або сторінок.

Мова SQL також надає спосіб непрямого управління швидкістю виконання транзакцій за допомогою зазначення рівня ізоляції транзакції. Під

рівнем ізоляції транзакції розуміють можливість виникнення однієї з описаних вище помилкових ситуацій. У стандарті SQL визначено 4 рівня ізоляції:

Таблиця 12.1 – Рівні ізоляції

Рівень ізоляції	Брудне читання	Розмите читання	Фантом
Незафіксоване читання (READ UNCOMMITTED)	можливо	можливо	можливо
Зафіксоване читання (READ COMMITED)	можливо	можливо	можливо
Повторюване читання (REPEATABLE READ)	можливо	неможливо	можливо
Серіалізація (SERIALIZABLE)	можливо	неможливо	неможливо

Для визначення характеристик транзакції використовується оператор SET TRANSACTION <режим доступу>, <рівень ізоляції>

Список рівнів ізоляції наведено в таблиці 12.1. Режим доступу за замовчуванням використовується READ WRITE (читання запис), якщо заданий рівень ізоляції READ UNCOMMITTED, то режим доступу повинен бути READ ONLY (тільки читання).

Одним з найбільш суттєвих недоліків методу серіалізації транзакцій на основі механізму блокувань є можливість виникнення глухих кутів (тупиків) (dead locks) між транзакціями. Наприклад, транзакція T1 наклала монопольне блокування на об'єкт O1 і претендує на доступ до об'єкту O2, який вже монопольно заблокований транзакцією T2, яка чекає доступу до об'єкта O1. У цьому випадку жодна з транзакцій тривати не може, отже, блокування об'єктів O1 і O2 ніколи не будуть зняті. Природного виходу з такої ситуації не існує, тому тупикові ситуації виявляються і усуваються штучно.

Хід роботи

I Реалізувати транзакцію з блокуванням таблиць (завдання отримати у викладача згідно Про).

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Завдання та текст транзакції.
- 3) Виведення вмісту таблиць, для яких сформована транзакція, до та після виконання транзакції.
- 4) Висновок.

Контрольні питання

- 1) Що таке транзакція?
- 2) Що означає аббревіатура ACID?
- 3) Якими властивостями володіє транзакція?
- 4) Які проблеми виникають при роботі з транзакціями?
- 5) Які є рівні ізоляції транзакцій?
- 6) Назвіть статуси роботи транзакцій.
- 7) Як може завершуватись транзакція?
- 8) Які параметри можна задавати транзакціям?

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Фиаيلي, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиаيلي – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 13
з навчальної дисципліни «Бази даних»

Тема	Розробка збережених процедур для введення /виведення даних.
Мета	Отримати навички створення та виклику користувацьких процедур

Час –2 години

Теоретичні відомості

Збережені процедури (stored procedure) - це іменований набір команд TransactSQL (або будь-якого іншого мови процедур, асоційованого з СУБД), що зберігається безпосередньо на сервері і представляє собою самостійний об'єкт бази даних.

Без збережених процедур користувачеві довелося б вводити весь набір команд щоразу, коли він хоче виконати будь-яку дію. Перерахуємо гідності використання збережених процедур:

1) Використання збережених процедур підвищує швидкість виконання операцій, так як процедура попередньо компілюється на сервері, і при повторному виклику процедура вже завантажена в пам'ять (кеш), де знайти її можна набагато швидше, ніж на диску, до того ж не потрібне повторне компіляція та оптимізація .

2) Збережені процедури можуть складатися з десятків і сотень команд, але для їх запуску достатньо вказати лише ім'я потрібної збереженої процедури. Це дозволяє зменшити розмір запиту, що посилається по мережі від клієнта на сервер, так як весь набір команд знаходиться в тому місці, де він повинен бути виконаний. Таким чином, при використанні збережених процедур можливе зменшення навантаження на мережу.

3) Використання збережених процедур реалізує принцип модульного проектування, так як процедури дозволяють розбивати великі завдання на самостійні, більш дрібні і зручні в управлінні частини. Типи збережених процедур.

У SQL Server 2000 підтримується кілька типів збережених процедур:

1) Системні збережені процедури (system stored procedures). Це збережені процедури, що поставляються в складі SQL Server 2000 і призначені для виконання різних адміністративних дій — створення облікових записів (sp_addlogin і інші), отримання інформації про об'єкти бази даних, управління властивостями сервера і баз даних, управління підсистемою реплікації і автоматизації і безліч інших завдань. Практично всі дії з адміністрування SQL Server 2000 виконуються за допомогою збережених процедур цього типу.

2) Користувальницькі збережені процедури (user-defined stored procedures). Крім системних збережених процедур, розроблених програмістами Microsoft, при роботі з SQL Server 2000 користувачі також можуть створювати свої власні процедури, що реалізують ті чи інші дії.

3) Тимчасові (temporary stored procedures) і локальні тимчасові процедури, що зберігаються (local temporary stored procedures). Такі процедури існують лише деякий час, після чого автоматично знищуються сервером.

Створення збереженої процедури передбачає вирішення наступних завдань:

- визначення типу створюваної процедури, що: тимчасова або призначена для користувача. Крім цього, можна створити свою власну системну збережену процедуру, призначивши їй ім'я з префіксом sp_ і помістивши її в системну базу даних. Така процедура буде доступна в контексті будь-якої бази даних локального сервера;

- планування прав доступу. При створенні збереженої процедури слід враховувати, що вона буде мати ті ж права доступу до об'єктів бази даних, що і створив її користувач;

- визначення параметрів процедури. Подібно процедурам, які входять до складу більшості мов програмування, збережені процедури можуть мати вхідними та вихідними параметрами;

- розробка коду збереженої процедури. Код процедури може містити послідовність будь-яких команд SQL, включаючи виклик інших процедур.

Створення нової та зміна наявної процедури, що здійснюється за допомогою наступної команди:

```
<визначення_процедури>::=
{CREATE | ALTER } PROC[EDURE] ім'я_процедури
    [;номер]
    [{@ім'я_параметру типу_даних } [VARYING ]
    [=default][OUTPUT] ][,...n]
    [WITH { RECOMPILE | ENCRYPTION | RECOMPILE,
    ENCRYPTION }]
    [FOR REPLICATION]
AS
    sql_оператор [...n]
```

Розглянемо параметри даної команди.

Використовуючи префікси `sp_`, `#`, `##`, створювану процедуру можна визначити як системної або тимчасовою. Як видно з синтаксису команди, не допускається вказувати ім'я власника, якому належатиме створювана процедура, а також ім'я бази даних, де вона повинна бути розміщена. Таким чином, щоб розмістити створювану збережену процедуру в конкретній базі даних, необхідно виконати команду `CREATE PROCEDURE` в контексті цієї бази даних. При зверненні з тіла збереженої процедури до об'єктів тієї ж бази даних можна використовувати укорочені імена, т. Е. Без вказівки імені бази даних. Коли ж потрібно звернутися до об'єктів, розташованих в інших базах даних, вказівка імені бази даних обов'язково.

Номер в імені — це ідентифікаційний номер збереженої процедури, однозначно визначає її в групі процедур. Для зручності управління процедурами логічно однотипні процедури, можна групувати, привласнюючи їм однакові імена, але різні ідентифікаційні номери.

Для передачі вхідних і вихідних даних в створюваній збереженій процедурі можуть використовуватися параметри, імена яких, як і імена локальних змінних, повинні починатися з символу `@`. В одній збереженій процедурі можна задати безліч параметрів, розділених комами. У тілі процедури не повинні застосовуватися локальні змінні, чиї імена збігаються з іменами параметрів цієї процедури.

Для визначення типу даних, який буде мати відповідний параметр збереженої процедури, годяться будь-які типи даних SQL, включаючи певні

користувачем. Однак тип даних CURSOR може бути використаний тільки як вихідний параметр збереженої процедури, тобто із зазначенням ключового слова OUTPUT.

Наявність ключового слова OUTPUT означає, що відповідний параметр призначений для повернення даних з збереженої процедури. Однак це зовсім не означає, що параметр не підходить для передачі значень в збережену процедуру. Вказівка ключового слова OUTPUT наказує сервера при виході з процедури, що привласнити поточне значення параметра локальній змінній, яка була вказана при виклику процедури в якості значення параметра. Відзначимо, що при вказівці ключового слова OUTPUT значення відповідного параметра при виклику процедури може бути поставлено лише за допомогою локальної змінної. Не дозволяється використання будь-яких виразів або констант, допустимий для звичайних параметрів.

Ключове слово VARYING застосовується спільно з параметром OUTPUT, що має тип CURSOR. Воно визначає, що вихідним параметром буде результуюча безліч.

Ключове слово DEFAULT є значення, яке буде приймати відповідний параметр за замовчуванням. Таким чином, при виклику процедури можна не вказувати явно значення відповідного параметра.

Так як сервер кешує план виконання запиту і компілює код, при наступному виклику процедури будуть використовуватися вже готові значення. Однак в деяких випадках все ж потрібно виконувати перекомпіляцію коду процедури. Вказівка ключового слова RECOMPILE наказує системі створювати план виконання процедури при кожному її виклику.

Параметр FOR REPLICATION затребуваний при реплікації даних і включенні створюваної процедури, як статті в публікацію.

Ключове слово ENCRYPTION наказує сервера виконати шифрування коду збереженої процедури, що може забезпечити захист від використання авторських алгоритмів, що реалізують роботу збереженої процедури.

Ключове слово AS розміщується на початку власне тіла збереженої процедури, тобто набору команд SQL, за допомогою яких і буде реалізовуватися ту чи іншу дію. У тілі процедури можуть застосовуватися практично всі команди SQL, оголошуватися транзакції, встановлюватися блокування і викликатися інші процедури, що зберігаються. Вихід з процедури, що можна здійснити за допомогою команди RETURN.

Видалення збереженої процедури здійснюється командою:

```
DROP PROCEDURE {ім'я_процедури} [, ...n]
```

Виконання процедури,

Для виконання процедури використовується команда:

```
[[ EXEC [ UTE] ім'я_процедури [;номер]  
[[@ім'я_параметру={значення | @ ім'я_змінної}  
[OUTPUT ]|[DEFAULT ]][, ...n]
```

Якщо виклик збереженої процедури не є єдиною командою в пакеті, то присутність команди EXECUTE обов'язково. Більш того, ця команда потрібна для виклику процедури з тіла іншої процедури або тригера.

Використання ключового слова OUTPUT при виклику процедури дозволяється тільки для параметрів, які були оголошені присозданні процедури з ключовим словом OUTPUT.

Коли ж при виклику процедури для параметра вказується ключове слово DEFAULT, то буде використано значення за замовчуванням. Природно, вказане слово DEFAULT дозволяється тільки для тих параметрів, для яких визначено значення за замовчуванням.

З синтаксису команди EXECUTE видно, що імена параметрів можуть бути опущені при виклику процедури. Однак в цьому випадку користувач повинен вказувати значення для параметрів в тому ж порядку, в якому вони перераховувалися при створенні процедури. Присвоїти параметру значення за замовчуванням, просто пропустивши його при перерахуванні можна. Якщо ж потрібно опустити параметри, для яких визначено значення за замовчуванням, досить явної вказівки імен параметрів при виклику збереженої процедури. Більш того, таким способом можна перераховувати параметри і їх значення в довільному порядку.

Відзначимо, що при виклику процедури вказуються або імена параметрів зі значеннями, або тільки значення без імені параметра. Їх комбінування не допускається.

Курсори

Прості курсори забезпечуються всередині збережених процедур і функцій. Синтаксис як у упровадженому SQL. Курсори в даний час тільки для читання, не підтримують прокрутку і нечутливі. Останнє означає, що сервер може або не може робити копію таблиці результату.

Курсори повинні бути оголошені перед оголошенням драйверів і змінних, а умови повинні бути оголошені перед оголошенням курсорів і драйверів.

Приклад:

```
CREATE PROCEDURE curdemo()  
BEGIN  
    DECLARE done INT DEFAULT 0;  
    DECLARE a CHAR(16);  
    DECLARE b,c INT;  
    DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;  
    DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;  
    DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done =  
1;  
  
    OPEN cur1;  
    OPEN cur2;  
    REPEAT  
        FETCH cur1 INTO a, b;  
        FETCH cur2 INTO c;  
        IF NOT done THEN  
            IF b < c THEN INSERT INTO test.t3 VALUES (a,b);  
            ELSE INSERT INTO test.t3 VALUES (a,c);  
            END IF;  
        END IF;  
    UNTIL done END REPEAT;  
    CLOSE cur1;  
    CLOSE cur2;
```

END

Оголошення курсорів.

```
DECLARE cursor_name
```

```
CURSOR FOR select_statement
```

Ця інструкція оголошує курсор. Багато курсорів може бути оголошено в підпрограмі, але кожен курсор в даному блоці повинен мати унікальне ім'я.

Інструкція SELECT не може мати пропозицію INTO.

Інструкція OPEN.

```
OPEN cursor_name
```

Ця інструкція відкриває попередньо оголошений курсор.

Інструкція FETCH.

```
FETCH cursor_name
```

```
INTO var_name [, var_name] ...
```

Ця інструкція вибирає наступний рядок (якщо рядок існує), використовуючи певний відкритий курсор, і просуває покажчик курсору.

Якщо більше немає доступних рядків, відбувається умова No Data зі значенням SQLSTATE 02000. Щоб виявити цю умову, Ви можете встановити драйвер для цього.

Інструкція CLOSE.

```
CLOSE cursor_name
```

Ця інструкція закриває попередньо відкритий курсор. Якщо курсор не закрито явно, він все одно закриється в кінці складовою інструкції, в якій він був оголошений.

Хід роботи

I Реалізувати користувальницьку процедуру (завдання отримати у викладача згідно Про).

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Завдання та текст процедури.
- 3) Виведення вмісту таблиць, з якими працює процедура, до та після її виконання.
- 4) Висновок.

Контрольні питання

- 1) Чи може збережена процедура викликати іншу збережену процедуру?
- 2) Чи може збережена процедура звертатися до таблиць?
- 3) Чи можна передавати або скасовувати транзакції усередині збереженої процедури?
- 4) Чи можна роздруковувати значення змінної всередині збереженої підпрограми для цілей налагодження?
- 5) Чи можна повертати курсор як параметр OUT з збереженої процедури?
- 6) Чи можна передавати курсор як параметр IN для збереженої процедури?

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 14
з навчальної дисципліни «Бази даних»

Тема Створення тригерів для забезпечення активної цілісності

Мета Отримати навички створення тригерів

Час –2 години

Теоретичні відомості

Тригери – це спеціальний вид зберезувальних процедур, що може викликатися SQL-сервером при доданні, вилученні або поновленні запису у таблиці бази даних. Тригер неможливо викликати із прикладної програми, він не має вхідних і вихідних параметрів. Відносно до випадку, який ініціалізує тригера, розрізняють тригери, що виконуються до виконання випадку, та після нього. Переваги використання тригерів можна відмітити такі: - Оскільки тригер фізично знаходиться і виконується на сервері, це знижає навантаження мережі при передачі поновлених даних. - Часто тригери використовуються для обробки каскадних поновлень записів у дочірніх таблицях. - Зміни в тригерах не призводять до необхідності змін прикладних клієнтських програм і не потребують тиражування нових версій клієнтських програм. За допомогою тригерів також часто реалізують так звані «бізнес-правила». Бізнес-правила задають спеціалізовані обмеження цілісності, які були визначені на етапі аналізу предметної області. Обробку частини бізнес-правил можна перенести на сервер. Синтаксис операторів, що задають тригер, може варіюватися залежно від СУБД. Наведемо синтаксис оператора створення тригера, прийнятий в СУБД MS SQL Server v.6.0 та вище:

```
CREATE TRIGGER trigger_name
ON { table | view }
[WITH ENCRYPTION]
{
    { {FOR|AFTER|INSTEAD OF} { [INSERT] [, ] [UPDATE] [, ] [DELETE] }
```

```

[WITH APPEND]
[NOT FOR REPLICATION]
AS
[ {IF UPDATE(column)
[ {AND|OR} UPDATE(column) ]
    [ ...n ]
|IF(COLUMNS_UPDATED() {bitwise_operator} updated_bitmask)
    {comparison_operator} column_bitmask [ ...n ]
} ]
sql_statement [ ...n ]
}
}

```

де trigger_name – ім'я тригера;

table|view – ім'я таблиці або виду, операції додання/вилучення/поновлення для якої повинні викликати тригер.

FOR|AFTER|INSTEAD OF – «для» / «після» / «замість».

INSERT, UPDATE, DELETE – тригер буде реагувати на «додання» / «поновлення» / «вилучення» записів із таблиці/виду.

IF UPDATE(column)[{AND|OR} UPDATE(column)] – тригер буде реагувати також і на додання або поновлення даних (але не на видалення) у конкретних полях таблиці/вида.

IF (COLUMNS_UPDATED() {bitwise_operator} updated_bitmask – функція COLUMNS_UPDATED() повертає значення типа varbinary – бітовий масив, з одиницями на місцях полів, дані в яких були поновлені або додані; у тригера є можливість перевірити, які поля змінилися, за допомогою маски.

sql_statement [...n] – речення Transact-SQL (процедурної мови, що окрім SQL включає оператори BEGIN...END, IF...[ELSE], WHILE...[BREAK][CONTINUE]

Тригери мають доступ до двох внутрішніх таблиць deleted та inserted. Структура таких таблиць відтворює структуру таблиці, для якої створювався тригер, і зберігають видалені або додані записи під час реакції тригера. Структура тіла тригера, що задана реченнями sql_statement зазвичай така:

```
BEGIN
```

```
[<визначення локальних змінних>]
<оператори Transact-SQL>
END
```

Визначення локальних змінних та оператори в тілі тригера задаються на процедурній мові, яка також має деякі відмінності, залежно від СУБД. Ця процедурна мова застосовується для написання як тригерів, так і хранимих процедур, і має оператори повтору, умовного вибору.

Оператор

```
DECLARE      @variableName1      variableType1[,      @variableName2
variableType2...]
```

задає локальну змінну `variableName1` типу `variableType1`.

Приклад 1.

```
DECLARE @ numrows int,
@errno int,
@errmsg varchar(255)
```

Ініціалізація локальної змінної виконується оператором `SELECT`.

Приклад 2.

```
SELECT @numrows = count(*)
FROM Tovar
```

На відміну від зберезувальних процедур, тригер має доступ до таблиць: `inserted` (якщо обробляється випадок додання запису), та `deleted` (якщо обробляється випадок вилучення запису).

Приклад 3.

```
SELECT @numrows = count(*)
FROM Storing, inserted
WHERE (Storing.nnum = inserted.nnum)
AND (Storing.num = inserted.num)
```

Цей фрагмент визначає, скільки в таблиці `Storing` є записів, де значення атрибутів `nnum` і `num` співпадає із значеннями цих же атрибутів в запису, що додається.

Приклад 4.

Наведемо приклад тригера. Нехай в базі даних складу є так званий журнал операцій `OperationList`, в якому фіксуються всі операції по переміщенню товарів на складі. Формат такого журналу може бути такий:

```
OperationDate, OperationType ("income" / "sale"), TovarNNum,  
TovarName, Edizm, Amount, Price, TotalCost
```

Розробимо тригер, який би викликався при доданні запису у таблицю `Income` (тобто, який обробляв би ситуацію із приходом нової партії товару на склад).

```
CREATE TRIGGER tI_Insert ON Income FOR insert  
AS  
BEGIN  
INSERT INTO OperationList (OperationDate, OperationType,  
TovarNNum, TovarName, Edizm, Amount, Price, TotalCost)  
SELECT getdate(),"income", inserted.nnum, Tovar.name, edizm,  
amount, iprice, amount*price  
FROM Tovar, inserted  
WHERE Tovar.nnum = inserted.nnum  
END
```

Аналогічні тригери можна написати і для таблиці расходів товару зі складу.

Видалення тригера:

```
DROP TRIGGER {trigger_name} [. . . n]
```

Хід роботи

I В роботі обов'язково відобразити:

- 1) тригер на операцію модифікації даних `INSERT`;
- 2) тригер на операцію модифікації даних `DELETE`;
- 3) тригер на операцію модифікації даних `UPDATE`;

II Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Завдання та текст процедури.

3) Виведення вмісту таблиць, з якими працюють тригери, до та після виконання.

4) Висновок.

Контрольні питання

- 1) Дайте визначення терміну «тригер»?
- 2) Для чого використовуються тригери?
- 3) Як створити тригер?
- 4) Які обмеження накладаються при створенні тригерів?
- 5) Порівняйте тригера та збережені процедури. Що в них спільного та відмінного?
- 6) Опишіть команду CREATE TRIGGER..
- 7) Яку кількість операторів може містити тригер?
- 8) Які операції можуть ініціювати виконання тригера?
- 9) Яка опція дозволяє користувачам прочитати текст тригера після розміщення його на сервері?
- 10) Наведіть приклад створення тригера. Які дії необхідно для цього виконати?
- 11) Для чого призначені тригери операторів INSERT й UPDATE?
- 12) Для чого призначений тригер оператора DELETE?
- 13) Що означає вкладення тригерів?
- 14) Як заборонити вкладення тригерів?
- 15) Як видалити тригери з таблиці?

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Файли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Файли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

ЛАБОРАТОРНА РОБОТА № 15
з навчальної дисципліни «Бази даних»

Тема Створення облікових записів. Встановлення рівнів привілей користувачів.

Мета Ознайомитися із засобами надання повноважень на використання баз даних і таблиць й основами роботи із зовнішніми базами даних

Час –2 години

Теоретичні відомості

Надання доступу до баз даних.

СУБД MySQL використовує спеціальну базу даних для надання прав доступу до своїх баз даних. Ці права можуть базуватися на іменах серверів й/або користувачів і надаватися для однієї або декількох баз даних

Користувальницькі облікові записи можуть бути постачені паролями. При звертанні до бази даних, пароль шифрується. Тому він не може бути перехоплений і використаний стороннім користувачем. СУБД MySQL має три таблиці, а саме:

- таблиця db, вміст якої наведено в таблиці 15.1;
- таблиця host, вміст якої наведено в таблиці 15.2;
- таблиця user, вміст якої наведено в таблиці 15.3.

Таблиця 15.1 – Таблиця db

Поле	Тип	Null	Ключ	Умолчание	Extra
Хост	char(60)		PRI		
Db	char(32)		PRI		
Пользователь	char(16)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

Таблиця 15.2 – Таблиця host

Поле	Тип	Null	Ключ	Умолчание	Extra
Хост	char(60)		PRI		
Db	char(32)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

Таблиця 15.3 – Таблиця user

Поле	Тип	Null	Key	Умолчание	Extra
Хост	char(60)		PRI		
Пользователь	char(16)		PRI		
Пароль	char(8)				
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	
Reload_priv	char(1)			N	
Shutdown_priv	char(1)			N	
Process_priv	char(1)			N	
File_priv	char(1)			N	

Поточною базою даних називається база даних, відкрита за допомогою операторів `use Database` або за допомогою утиліти `mysqladmin`. Будь-яка інша база даних називається зовнішньою. Для посилання на таблицю в зовнішній базі даних необхідно вказати ім'я цієї бази даних як частину імені таблиці, наприклад, `salesdb:contracts`, де `salesdb` - ім'я зовнішньої бази даних, `contracts` - ім'я таблиці. До імені бази даних можна додати ім'я

сервера, тобто мережної машини, де запусканий ще один сервер баз даних mysql, і в такий спосіб у випадку розподіленої бази даних звертання до таблиці contracts бази даних salesdb, розміщеної на сервері central, буде виглядати в такий спосіб: salesdb@central:contracts.

Система безпеки SQL Server.

Система безпеки SQL Server заснована на концепції об'єктів, що захищають, (securables), тобто об'єктів, на які можна призначати дозволу, і принципалів (principles), тобто об'єктів, яким можна призначати дозволу. Принципалами можуть бути логіни на рівні сервера, користувачі й ролі на рівні бази даних. Ролі призначаються користувачам. Дозволу на доступ до об'єктів можуть надаватися як безпосередньо користувачам, так і через ролі. Кожен об'єкт має свого власника, і права власності також впливають на дозволи.

SQL Server використовує двоетапну схему аутентифікації. На рівні сервера користувач розпізнається по своєму ідентифікаторі (Login), що може бути або ім'ям входу SQL Server, або групою або обліковим записом Windows. Після входу на сервер користувач одержує ті права, які були призначені йому адміністратором на рівні сервера, зокрема за допомогою фіксованих серверних ролей. Якщо користувач належить ролі sysadmin, то він має повний доступ до всіх функцій сервера, а також до всіх баз даних й об'єктам на ньому.

Для одержання доступу до бази даних логін користувача повинен бути зіставлений з відповідним йому ідентифікатором користувача (User), що специфічний для кожної бази даних. Цілком можлива ситуація, коли користувач був розпізнаний в SQL Server, але в нього немає доступу ні до однієї з баз даних. Також можливо й зворотне: користувачеві відкритий доступ до баз даних, але він не був розпізнаний сервером. Переміщення бази даних й її дозволів на інший сервер без паралельного переміщення імен входу сервера може привести до виникнення таких "осиротілих" користувачів.

На рівні бази даних користувачеві може бути наданий певний набір дозволів за допомогою призначення йому фіксованих ролей бази даних. Всі користувачі автоматично стають членами стандартної ролі public, у якої за замовчуванням немає ніяких дозволів. Користувальницькі ролі - це додаткові ролі, що служать як групи. Ролі може бути дозволений доступ до об'єктів бази даних, а користувачеві можуть бути призначені ролі.

Дозволу до об'єктів призначаються за допомогою інструкцій GRANT (надати), REVOKE (відкликати) і DENY (заборонити). Заборона привілею заміщає собою її надання, а надання привілею заміщає собою її відкликання. Користувачеві може бути надана безліч дозволів до об'єкта (індивідуальних, успадкованих від ролі, забезпечених приналежністю до ролі public). Якщо яка-небудь із цих привілеїв заборонена, для користувача блокується доступ до об'єкта. У протилежному випадку, якщо яка-небудь із привілеїв надає дозвіл, користувач одержує доступ до об'єкта.

Дозволу об'єкта досить деталізовані. Існують окремі дозволи для кожної з можливих дій (SELECT, INSERT, UPDATE, RUN і т.д.) над об'єктом.

Вибір типу логіну й налаштування режиму аутентифікації.

SQL Server підтримує два типи логінів (імен входу):

- логін Windows (логін для локального облікового запису Windows, логін для доменного облікового запису Windows, логін для групи Windows);
- логін SQL Server.

При використанні логінів Windows у системні таблиці бази даних master записується інформація про ідентифікатор облікового запису або групи Windows (але не пароль). Аутентифікація (тобто перевірка імені користувача й пароля) виробляється звичайними засобами Windows при вході користувача на свій комп'ютер.

При використанні логіна SQL Server пароль для цього логіна (точніше, його хешоване значення) зберігається разом з ідентифікатором логіна в базі даних master. При підключенні користувача до сервера йому доведеться вказати ім'я логіна й пароль.

Кращий варіант логіна для користувача - це логін Windows, при цьому не для облікового запису, а для групи (найкраще для локальної доменної групи). Переваг у такого рішення безліч:

- користувачеві досить пам'ятати один пароль - для входу на свій комп'ютер;
- підвищується рівень захищеності SQL Server. Це відбувається, принаймні, за рахунок того, що пароль не буде передаватися по мережі відкритим текстом, як це відбувається за замовчуванням при використанні команд CREATE LOGI й ALTER LOGI. Крім того, хеши Windows більше захищені, чим хеши логінів SQL Server;
- перевірка при вході користувача відбувається швидше.

Ці переваги справедливі для будь-яких логінів Windows: як для облікових записів, так і для груп. Але при використанні логінів для груп Windows з'являються додаткові переваги:

- знижується розмір системних таблиць бази даних master, у результаті чого аутентифікація відбувається швидше. На одному сервері SQL Server цілком може бути кілька тисяч логінів для користувачів Windows або, що значно зручніше, усього пари десятків логінів для груп;
- значно спрощується надання дозволів для нових облікових записів.

Використання логінів SQL Server може бути обумовлено наступною причиною: дуже часто на підприємствах адмініструванням SQL Server і домена Windows займаються різні люди, яким складно погоджувати свої дії. Логіни SQL Server дозволяють адміністраторові бази даних бути незалежним від адміністратора домена. Крім того, у логінів SQL Server є й інші переваги, які приймаються в увагу розроблювачами:

- на підприємстві цілком може не виявитися домена Windows (якщо, наприклад, мережа побудована на основі NetWare або UNI);
- користувачі SQL Server можуть не входити в домен (наприклад, якщо вони підключаються до SQL Server з філій або через Web-інтерфейс із домашнього комп'ютера).

Таким чином, вибір використовуваних типів логінів залежить від багатьох факторів й у кожному конкретному випадку рішення приймається індивідуально. Логіни Windows - це зручність і захищеність, логіни SQL Server- це більша гнучкість і незалежність від адміністратора мережі.

При установці SQL Server одним з рішень, які варто прийняти, є вибір використовуваного режиму аутентифікації.

У режимі аутентифікації Windows SQL Server повністю довіряє (делегує) аутентифікацію операційній системі.

У змішаному режимі аутентифікація Windows і самого сервера співіснують незалежно друг від друга.

Третього варіанта, у якому використання логінів Windows було б заборонене, не передбачено: логіни цього типу доступні завжди.

Установлений при інсталяції режим аутентифікації можна змінити в утиліті Management Studio вибравши потрібний перемикач у групі "Серверна перевірка дійсності" на сторінці "Безпека" діалогового вікна "Властивості сервера", як наведено на рисунку 15.1.

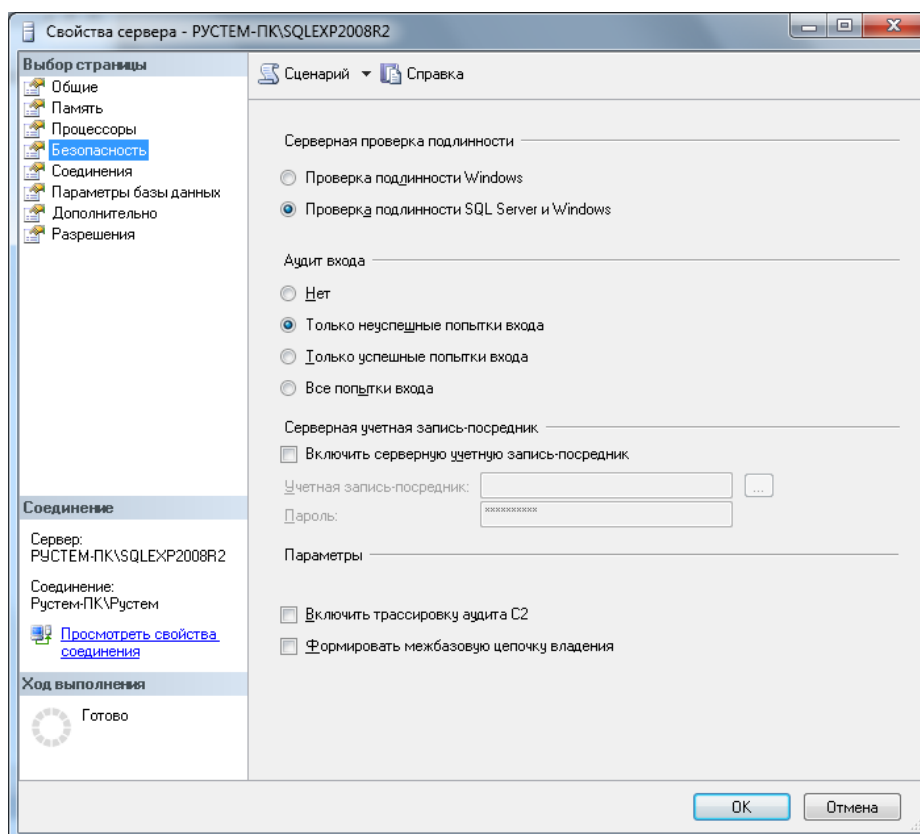


Рисунок 15.1 – Настройки «Властивостей сервера»

Установите переключатель в положение "Проверка действительности SQL Server и Windows". Таким чином, ви включите змішаний режим аутентифікації, у якому й будуть виконуватися інші вправи. Для того щоб зміна режиму аутентифікації набуло чинності сервер потрібно запустити знову.

Створення логіна й налаштування його параметрів.

Логіни будь-якого типу створюються однаково.

1) За допомогою графічного інтерфейсу - з вікна "Створення імені входу". Це вікно відкривається за допомогою команди "Створити ім'я входу..." контекстного меню вузла "Безпека | Імена входу" дерева оглядача об'єктів в SQL Server Management Studio, як наведено на рисунку 15.2.

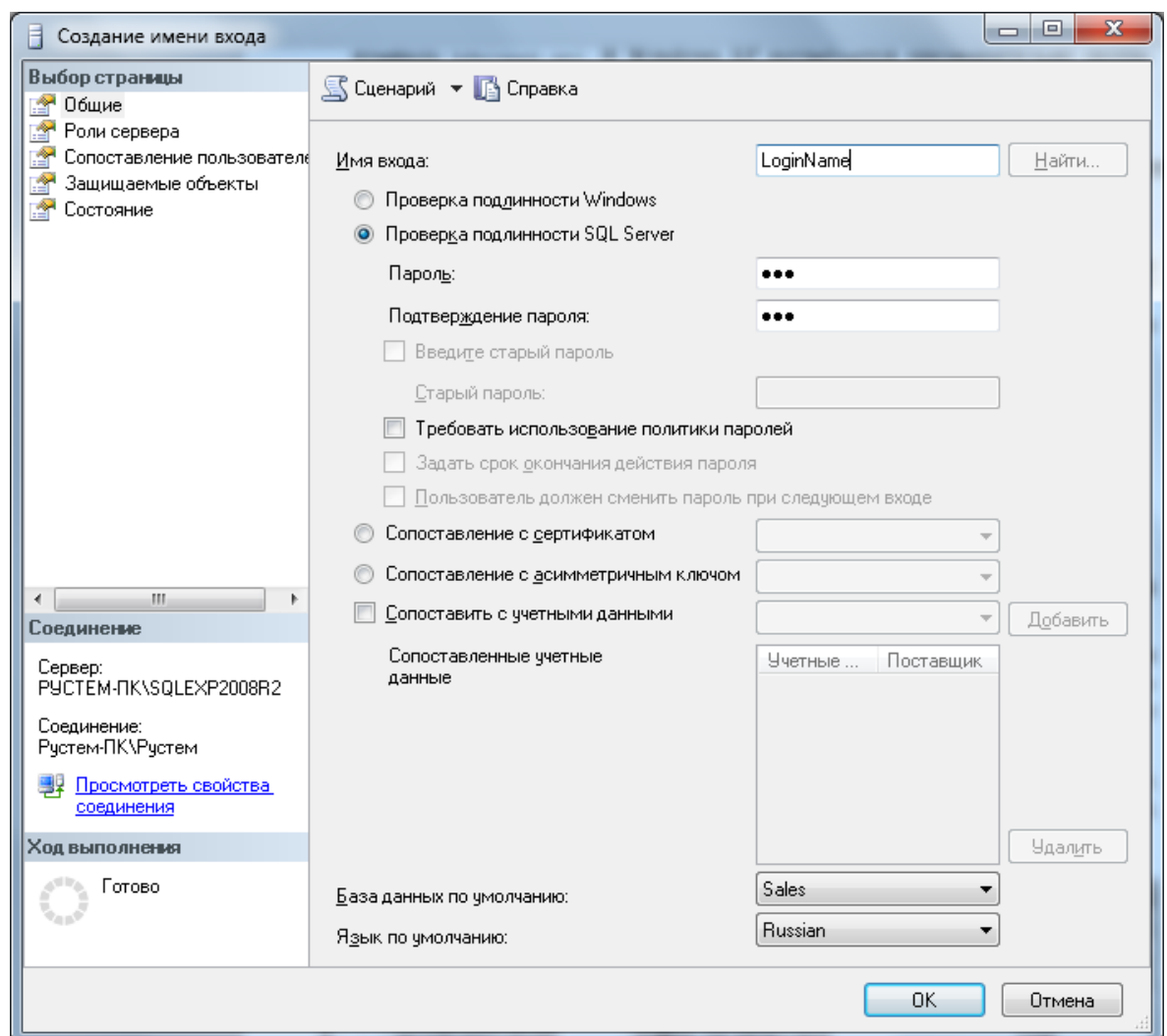


Рисунок 15.2 – Створення логіну

2) Зі скрипта - за допомогою команди CREATE LOGIN.

Наприклад, команда на створення логіна SQL Server з ім'ям User1 і паролем P@sswOrd (для всіх інших параметрів будуть прийняті значення за замовчуванням) може виглядати так:

```
CREATE LOGIN User1 WITH PASSWORD = 'P@sswOrd';
```

Якщо ви створюєте логін Windows, вам буде потрібно вибрати відповідний обліковий запис або групу Windows.

Якщо ви створюєте логін SQL Server, вам доведеться ввести його ім'я й пароль. Пароль завжди чутливий до регістра, а логін - тільки тоді, коли чутливість до регістра була визначена при установці SQL Server. Звичайно, крім імені й пароля для логінів можна визначити безліч інших параметрів. Деякі з них перераховані нижче.

База даних за замовчуванням, до якої за замовчуванням буде підключатися користувач при вході на SQL Server. За замовчуванням використовується база даних master. Як правило, міняти цей параметр не треба: код для переходу до потрібної бази даних при підключенні забезпечує клієнтський додаток.

Мова за замовчуванням - мова, що буде використатися за замовчуванням даним користувачем під час сеансів. В основному він впливає на формат дати й часу, які повертає SQL Server. У більшості випадків для цього параметра залишається значення за замовчуванням (тобто мова, настроєна на рівні всього сервера), якщо про інше значення спеціально не просить розроблювач.

На вкладці "Стан" властивостей логіну можна настроїти для цього логіну додаткові параметри:

Дозвіл на підключення до ядра СУБД - за замовчуванням для всіх логінів установлюється значення "Надати", тобто підключатися до SQL Server дозволено. Значення "Заборонити", як правило, використовується тільки в одному випадку - коли ви надаєте доступ на SQL Server логіну для групи Windows, а одному або декільком членам цієї групи доступ потрібно заборонити. Оскільки явна заборона завжди має пріоритет перед дозволом, то

досить буде створити свої власні логіни Windows для цих користувачів й установити для них значення "Заборонити".

Ім'я входу (Включене/Відключено) - звичайно, всі логіни за замовчуванням включені. Звичайно відключати їх доводиться тільки в ситуації, коли якийсь користувач звільняється або переходить на іншу роботу. Щоб заощадити час, досить просто відключити даний логін, а з появою користувача зі схожими робочими обов'язками перейменувати цей логін, поміняти пароль і включити. Займатися наданням дозволів заново в цьому випадку не прийде.

Ім'я входу заблоковане - установити цей прапорець ви не можете (тільки зняти його). Обліковий запис користувача блокується автоматично після декількох спроб невірного введення пароля для логіну SQL Server, якщо таке блокування настроєне не з операційної системи, а для логіну встановлений прапорець "Вимагати використання політики паролів". Приклад наведено на рисунку 15.3.

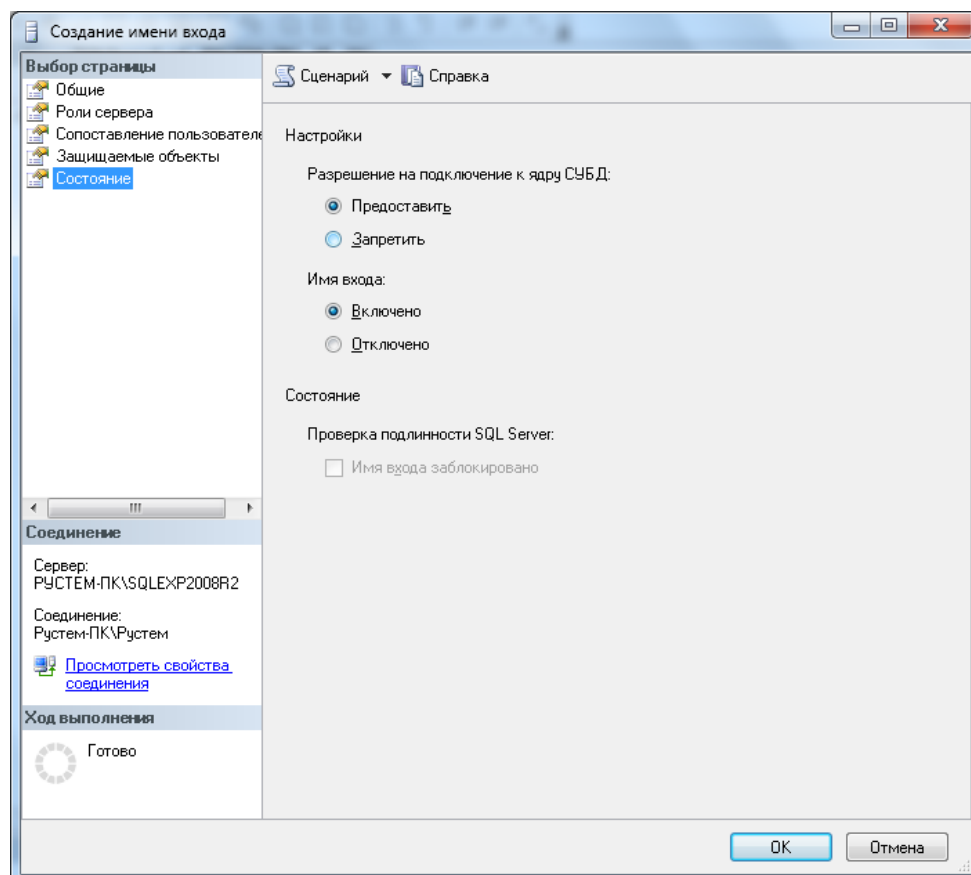


Рисунок 15.3 – Дозвіл на підключення

Дозволи на рівні сервера. Фіксовані серверні ролі.

Створивши логіни, ви забезпечуєте користувачам можливість входу на SQL Server. Але сам по собі вхід на сервер нічого не дає: користувачеві потрібні також права на виконання певних дій. Звичайно для цієї мети створюються користувачі або ролі баз даних й їм надаються дозволи (як це зробити, буде розглянуто в наступному розділі). Однак є й інший спосіб. Якщо вам потрібно надати користувачеві права на рівні всього сервера, а не окремої бази даних, можна скористатися серверними ролями.

На графічному екрані робота з ролями сервера виробляється або із властивостей логіну (вкладка "Ролі сервера"), або із властивостей самої серверної ролі (вузол "Ролі сервера" дерева оглядача об'єктів Management Studio), як наведено на рисунку 15.4.

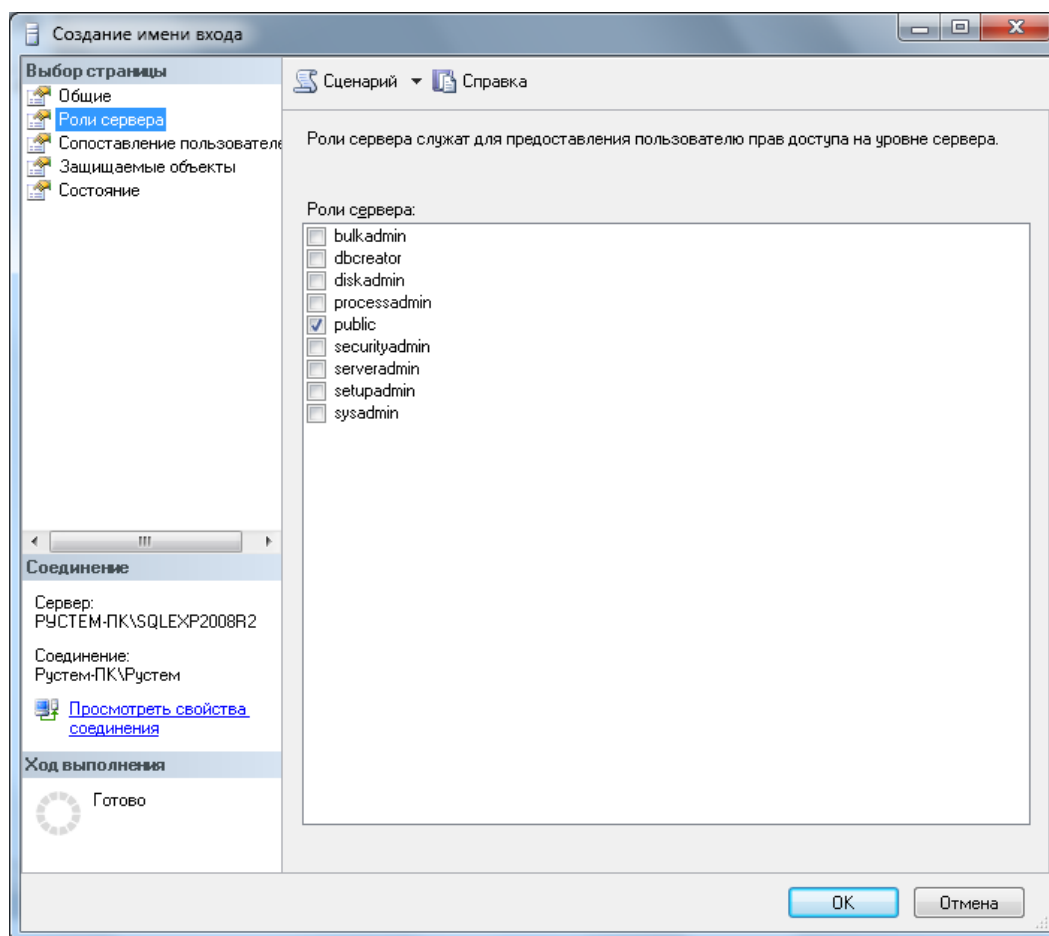


Рисунок 15.4 – Встановлення ролі

Відзначимо кілька моментів, пов'язаних із серверними ролями:

- набір серверних ролей є фіксованим. Ви не можете створювати свої серверні ролі (на відміну від ролей бази даних);
- для надання прав на рівні всього сервера необов'язково використати серверні ролі. Ви цілком можете надати ці права прямо логіну (за допомогою вкладки "Дозволу" вікна властивостей сервера). За замовчуванням кожен логін володіє на рівні всього сервера двома правами: CONNECT SQL (тобто підключатися до сервера, це право надається логіну прямо) і VIEW ANY DATABASE (тобто переглядати список баз даних, це право користувач одержує через серверну роль public), як наведено на рисунку 15.5;
- серверні ролі використовуються тільки в спеціальних випадках. Для більшості користувачів набудовувати їх не потрібно.

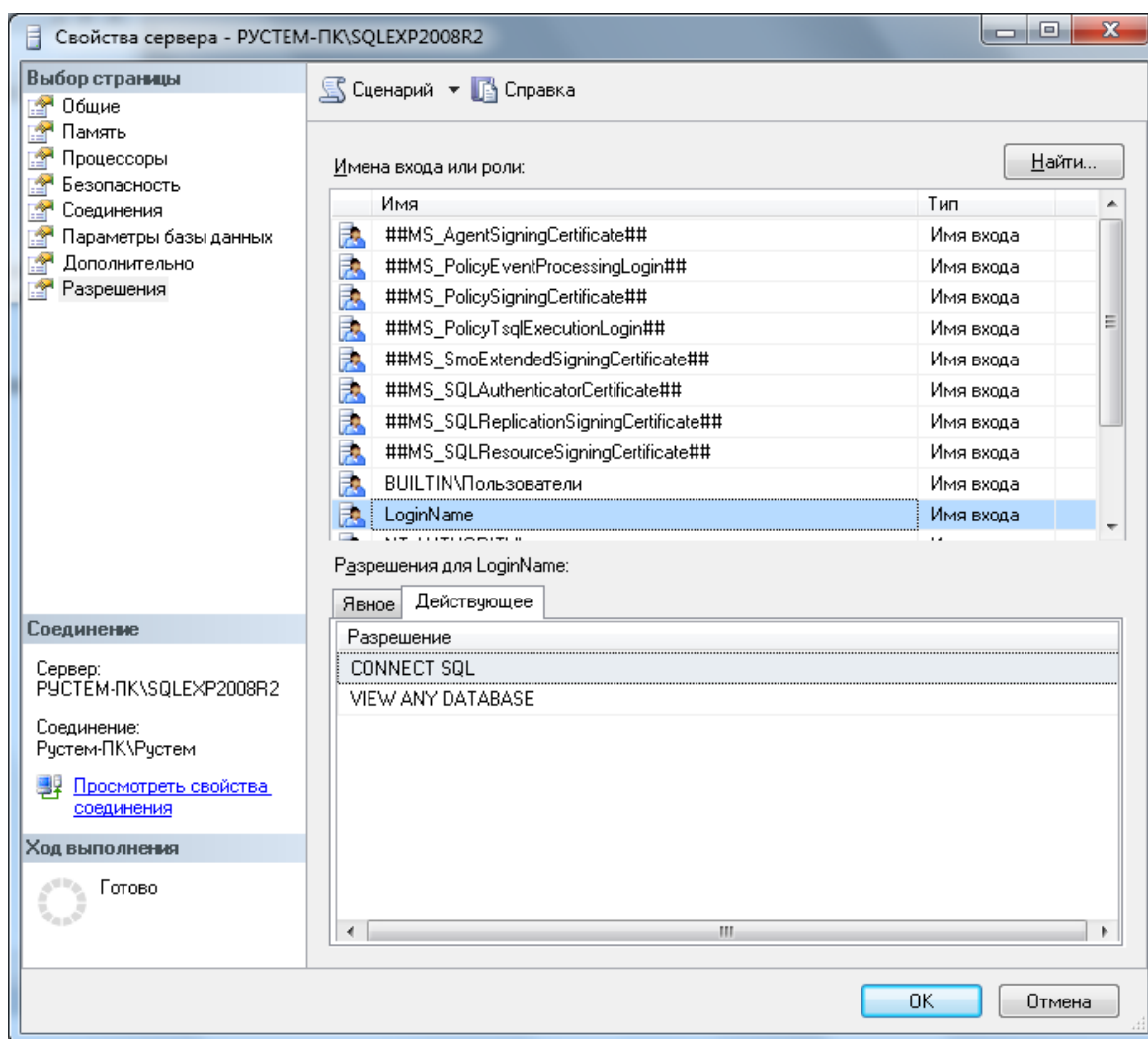


Рисунок 15.5 – Надання прав на рівні серверу

Серверних ролей не так багато, тому приведемо їхній повний список з коментарями:

`public` - права цієї ролі автоматично одержують усі, хто підключився до SQL Server, і позбавити користувача членства в цій ролі не можна. Звичайно ця роль використовується для надання дозволів всім користувачам даного сервера;

`sysadmin` - логін, якому призначена ця роль, дістає повні права на весь SQL Server (і можливість передавати ці права іншим користувачам);

`serveradmin` - ця роль для оператора, що обслуговує сервер. Можна змінювати будь-які налаштування роботи сервера й відключати сервер, але одержувати доступ до даних і змінювати дозволи не можна;

`securityadmin` - ця роль для того, хто видає дозволи користувачам, не втручаючись у роботу сервера. Він може створювати логін для звичайних користувачів, змінювати їхні паролі, надавати дозволу в базах даних. Однак надати кому-небудь адміністративні права або змінювати обліковий запис адміністратора ця роль не дозволяє;

`bulkadmin` - роль для співробітників, які виконують масові завантаження даних у таблиці SQL Server;

`dbcreator` - ця роль дозволяє створювати бази даних на SQL Server (а той, хто створив базу даних, автоматично стає її власником);

`diskadmin` - ця роль дозволяє виконувати будь-які операції з файлами баз даних на диску;

`processadmin` - ця роль призначена для виконання єдиного обов'язку: закриття користувацьких підключень до сервера (наприклад, що зависли);

`setupadmin` - права цієї ролі дозволяють підключати зовнішні сервери SQL Server.

Користувачі баз даних. Схеми.

Після створення логінів наступне завдання - спуститися на рівень бази даних і створити користувачів бази даних. Користувачі баз даних - це спеціальні об'єкти, які створюються на рівні бази даних і використовуються для

надання дозволів у базі даних (на таблиці, подання, збережені процедури й т.д.).

Логіни й користувачі баз даних - це зовсім різні об'єкти. Поділ логінів і користувачів баз даних забезпечує більшу гнучкість. Наприклад, користувач, що входить від імені того самого логіна, зможе працювати в різних базах даних від імені різних користувачів.

Створити користувача бази даних можна:

У середовищі Management Studio викликавши команду "Створити користувача..." у контекстному меню підвузла "Безпека | Користувачі" вузла конкретної бази даних дерева оглядача об'єктів. У вікні, що відкрився, "Користувач бази даних - Створити", знімок екрана з яким наведений нижче, необхідно вказати два обов'язкових параметри: ім'я нового користувача й вибрати відповідний йому логін (Windows або SQL Server), як наведено на рисунку 15.6.

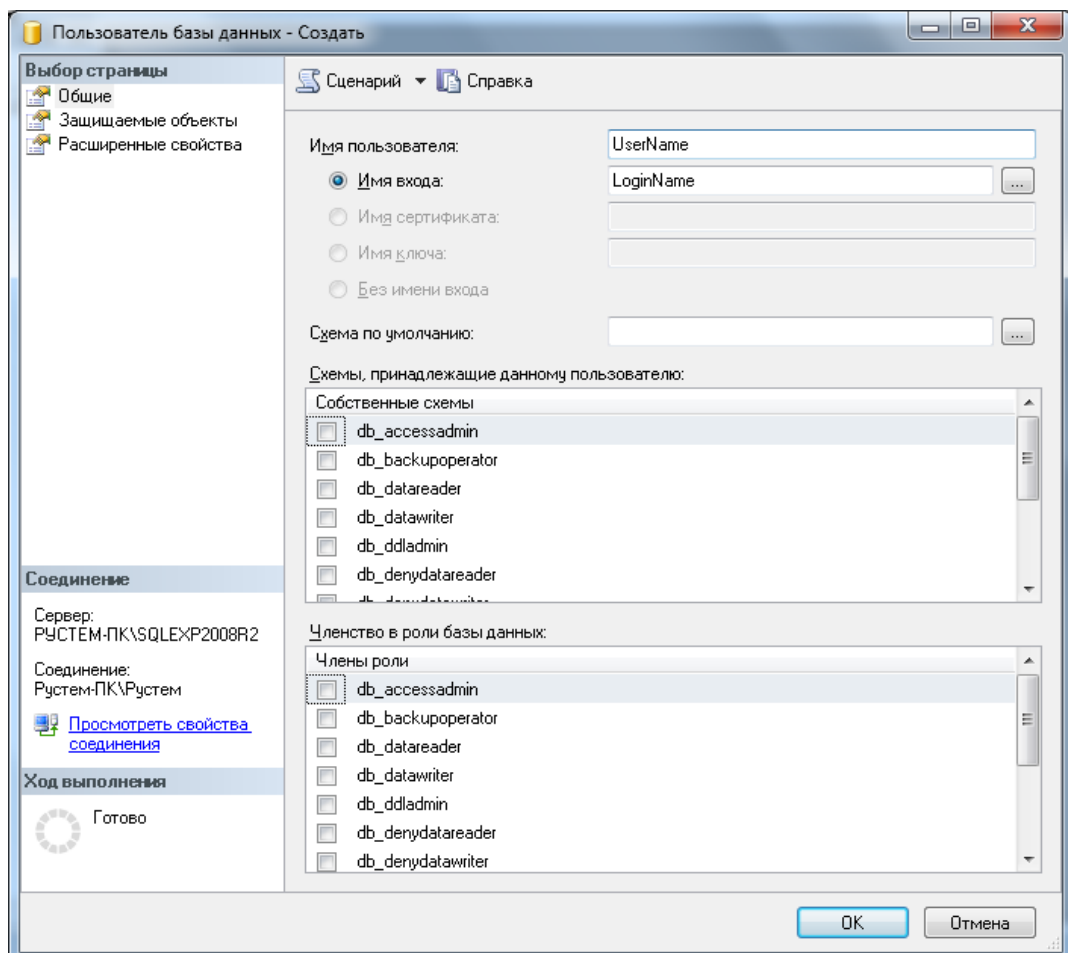


Рисунок 15.6 – Створення користувача

Зміна властивостей користувача і його видалення виробляється з того ж контейнера в Management Studio, що й створення користувача, а також за допомогою команд ALTER USER/DROP USER.

В SQL Server 2000 й у більше старих версіях об'єкт користувача бази даних ніс на собі подвійне навантаження: по-перше, воно використовувався для надання дозволів у базі даних, а по-друге, ім'я користувача бази даних використалося для ідентифікації об'єктів. Наприклад, звертання до таблиці по повному її імені могло виглядати так:

```
SELECT * FROM MyServer.MyDatabase.User1.Table1;
```

Якщо розроблювач використав у коді додатка просте ім'я об'єкта (наприклад, SELECT * FROM Table1), то при підключенні до сервера із цього додатка від імені іншого користувача могли виникнути проблеми, оскільки замість User1 підставлялося поточне ім'я користувача (а якщо об'єкт із таким повним ім'ям не був виявлений, то ім'я спеціального користувача dbo).

Починаючи з версії SQL Server 2005 ці дві функції розділені. Для надання дозволів у базі даних, як і раніше, використовується об'єкт користувача, а для іменування об'єктів у базі даних використовується спеціальний об'єкт схема. Запит з використанням повного формату імені в SQL Server тепер повинен виглядати так:

```
SELECT * FROM MyServer.MyDatabase.Schema1.Table1;
```

Схема формально визначається як набір об'єктів у базі даних, об'єднаних загальним простором імен. Найпростіше представити собі схему як якийсь логічний контейнер у базі даних, якому можуть належати таблиці, подання, збережені процедури, користувальницькі функції, обмеження цілісності, користувальницькі типи даних й інші об'єкти бази даних. Цей контейнер зручно використати як для іменування об'єктів й їхнього логічного угруповання, так і для надання дозволів. Наприклад, якщо в базі даних є набір таблиць зі зв'язаними даними, зручно помістити їх в одну схему й надавати користувачам дозволу на цю схему (тобто на цей набір таблиць).

Користувачеві можна призначити схему за замовчуванням. У цю схему SQL Server буде за замовчуванням поміщати об'єкти, які створює цей

користувач. Крім того, шукати об'єкти, до яких звертається користувач (наприклад, у випадку запиту виду `SELECT * FROM Table1`), SQL Server також буде в першу чергу в його схемі за замовчуванням.

Застосування схеми дає ряд додаткових переваг у порівнянні зі старим підходом:

- декільком користувачам можна призначити ту саму схему за замовчуванням, що може бути зручно при розробці додатків;
- декілька користувачів (через групи Windows або ролі баз даних) можуть володіти однієї й тією же схемою. При цьому один користувач може бути власником відразу декількох схем;
- при видаленні користувача з бази даних не прийде перейменовувати його об'єкти;
- спрощується надання дозволів для наборів об'єктів у базі даних.

Список схем можна побачити в підвузлі "Безпека | Схеми" вузла конкретної бази даних дерева оглядача об'єктів Management Studio.

При створенні будь-якої бази даних у ній автоматично створюються чотири спеціальних користувача:

`dbo` (від `database owner`) - власник бази даних. Він автоматично створюється для того логіна, від імені якого була створена ця база даних. Звичайно ж, як власник, він дістає повні права на свою базу даних (за допомогою убудованої ролі бази даних `db_owner`);

`guest` (гість) - цей спеціальний користувач призначений для надання дозволів всім логінам, яким не відповідає жоден користувач у базі даних. За замовчуванням у цього користувача немає права `login` для бази даних, і, отже, працювати він не буде. Цей користувач використовується найчастіше для надання дозволів логінам на якісь навчальні/тестові бази даних або на бази даних-довідники, доступні тільки на читання;

`INFORMATION_SCHEMA` - цьому користувачеві не може відповідати жоден логін. Його єдине значення - бути власником схеми `INFORMATION_SCHEMA`, у якій зберігаються подання системної інформації для бази даних;

sys - цьому спеціальному користувачеві, як й INFORMATION_SCHEMA, не можуть відповідати логін. Він є власником схеми sys, що належать системні об'єкти бази даних.

Ролі бази даних.

Звичайно після створення логіна й користувача бази даних наступне, що потрібно зробити, - надати користувачеві дозволу в базі даних. Один зі способів зробити це - скористатися ролями бази даних.

Ролі бази даних - це спеціальні об'єкти, які використовуються для спрощення надання дозволів у базах даних. На відміну від серверних ролей, які можуть бути тільки убудованими, ролі баз даних можуть бути як убудованими, так і користувальницькими. Убудовані ролі баз даних мають визначений набір дозволів, а користувальницькі ролі можна використати для угруповання користувачів при наданні дозволів. Звичайно користувальницькі ролі використовуються тільки для логінів SQL Server, оскільки для угруповання логінів Windows звичайно зручніше й простіше використати групи Windows.

Спочатку перелічимо убудовані ролі баз даних:

public - ця спеціальна роль призначена для надання дозволів відразу всім користувачам бази даних. Спеціально зробити користувача членом цієї ролі або позбавити його членства неможливо. Всі користувачі бази даних дістають права цієї ролі автоматично.

db_owner - цієї ролі автоматично надаються повні права на базу даних. Споконвічно права цієї ролі надаються тільки спеціальному користувачеві dbo, а через нього - логину, що створив цю базу даних;

db_accessadmin - роль для співробітника, відповідального за користувачів бази даних. Цей співробітник одержить можливість створювати, змінювати й видаляти об'єкти користувачів баз даних, а також створювати схеми. Інших прав у базі даних у нього немає;

db_securityadmin - ця роль доповнює роль db_accessadmin. Співробітник із правами цієї ролі одержує можливість призначати дозволу на об'єкти бази даних і змінювати членство в убудованих і користувальницьких ролях. Прав на створення й зміну об'єктів користувачів у цієї ролі немає;

`db_backupoperator` - ця роль надає право виконувати резервне копіювання бази даних;

`db_ddladmin` - ця роль застосовується в рідких ситуаціях, коли користувачеві необхідно дати право створювати, змінювати й видаляти будь-які об'єкти в базі даних, не надаючи прав на інформацію, що втримується в існуючих об'єктах;

`db_datareader` й `db_datawriter` - ці убудовані ролі надають право переглядати й змінювати відповідно (у тому числі додавати й видаляти) будь-яку інформацію в базі даних. Дуже часто користувачеві необхідно дати права на читання й запис інформації у всіх таблицях бази даних, не надаючи йому зайвих адміністративних дозволів (на створення й видалення об'єктів, зміна прав і т.п.). Найпростіший варіант у цій ситуації - скористатися цими двома ролями.

`db_denydatareader` й `db_denydatawriter`- ці ролі протилежні ролям `db_datareader` й `db_datawriter`. Роль `db_denydatareader` явно забороняє переглядати які-небудь дані, а `db_denydatawriter` забороняє внесення змін. Явна заборона завжди має пріоритет перед явно наданими дозволами. Звичайно ці ролі використовуються в ситуації, коли "дозволяємо всім, а потім деяким забороняємо".

Як уже говорилося раніше, на відміну від серверних ролей, ролі баз даних ви можете створювати самостійно. Це можна зробити з контекстного меню вузла "Безпека | Ролі | Ролі бази даних" оглядача об'єктів в Management Studio, як наведено на рисунку 15.7 або за допомогою команди `CREATE ROLE`.

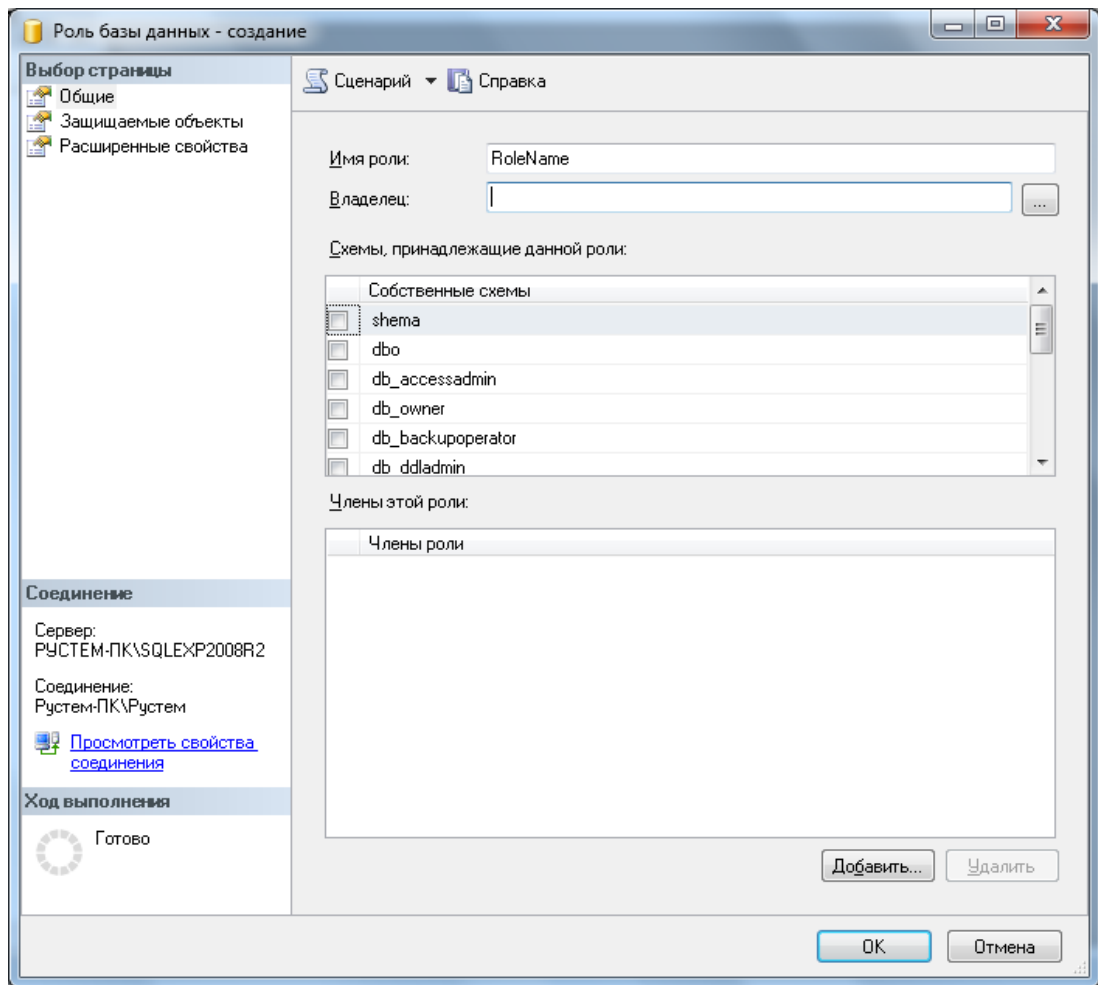


Рисунок 15.7 – Створення ролі

Крім імені ролі, при створенні можна також вказати її власника (якщо це не зазначено, власником ролі автоматично стане її користувач, що створив, бази даних). Якщо ви створюєте роль за допомогою графічного інтерфейсу, ви можете відразу призначити роль власником схеми в базі даних і призначити користувачів, які будуть мати права цієї ролі.

Убудованим ролям призначені визначені права, змінити які неможливо. Надання прав користувальницьким ролям виробляється точно так само, як і звичайним користувачам бази даних.

Надання прав на об'єкти в базі даних.

Робота з дозволами виробляється однаково для всіх об'єктів бази даних:

- на вкладці "Дозволу" властивостей цього об'єкта (ця вкладка передбачена не для всіх об'єктів, для яких можна надати дозволу);
- на сторінці "Об'єкти, що захищуються" вікна властивостей користувача або ролі;

– за допомогою команд GRANT (надати дозвіл), DENY (явно заборонити щось робити) і REVOKE (скасувати явно наданий дозвіл або заборона).

Починаючи з версії 2005, в SQL Server з можливість надавати дозволи на рівні схеми. До схеми в SQL Server можуть ставитися таблиці, подання, збережені процедури, користувальницькі функції, обмеження цілісності, користувальницькі типи даних й інші об'єкти, на які доводиться надавати дозволу найчастіше. Якщо ви призначите користувачеві дозволу на схему, то він одержить дозволи на всі об'єкти цієї схеми.

Далі перераховані дозволи, які можна надати на зі схеми. Ми приведемо тільки дозволу для цього об'єкта, оскільки дозволу схеми містять у собі дозволу, які можна надати іншим об'єктам, наприклад, дозволу SELECT для таблиць і подань й EXECUTE для збережених процедур.

ALTER - можливість вносити будь-які зміни у властивості об'єкта (за винятком зміни власника).

CONTROL - той, кому надане такий дозвіл, дістає повні права як на сам об'єкт, так і на інформацію в ньому.

DELETE - можливість видаляти існуючу інформацію в таблицях. Застосовується до таблиць, поданням і стовпцям.

EXECUTE - право запускати на виконання. Застосовується до збережених процедур і функцій.

INSERT - право на вставку нових даних у таблиці. Застосовується до таблиць, поданням і стовпцям.

REFERENCES - дозвіл, яке можна надати для перевірки обмежень цілісності. Наприклад, користувач може додавати дані в таблицю із зовнішнім ключем, а на таблицю з первинним ключем йому не можна надавати права на перегляд. У цьому випадку на таблицю з первинним ключем йому можна надати право REFERENCES - і він зможе робити вставку даних у таблицю із зовнішнім ключем, не одержуючи зайвих прав на головну таблицю. Це дозвіл можна надавати на таблиці, подання, стовпці й функції.

SELECT - право на читання інформації. Надається для таблиць, подань, стовпців і табличних функцій.

TAKE OWNERSHIP - право на прийняття на себе володіння даним об'єктом. Власник автоматично має повні права на свій об'єкт. Таке право можна призначити для будь-яких об'єктів.

UPDATE - можливість вносити зміни в існуючі записи в таблиці. Надається на таблиці, подання й стовпці.

VIEW DEFINITION - право на перегляд визначення для даного об'єкта. Передбачено для таблиць, подань, процедур і функцій.

Для кожного дозволу ми можемо встановити три значення:

GRANT - дозвіл наданий явно;

WITH GRANT - дозвіл не тільки наданий даному користувачеві, але він також одержав право надавати цей дозвіл іншим користувачам. Можна надавати, звичайно, тільки разом з GRANT;

DENY - явна заборона на виконання дії, певного даним дозволом. Як у будь-яких системах безпеки, явна заборона має пріоритет перед явно наданими дозволами.

З коду Transact-SQL дозволу можна надавати командами GRANT, DENY й REVOKE. Наприклад, щоб надати користувачеві User1 можливість переглядати дані в таблиці Table1 у схемі dbo, можна скористатися командою: GRANT SELECT ON dbo.Table1 TO User1;

Позбавити його раніше наданого права можна за допомогою команди: REVOKE SELECT ON dbo.Table1 TO User1;

Деякі рекомендації, пов'язані з наданням дозволів.

У більшості реальних завдань використовуються десятки й навіть сотні таблиць й інших об'єктів бази даних. Надавати кожному користувачеві дозволу на кожний із цих об'єктів дуже незручно. Якщо є можливість, зручніше використати дозволи на рівні схеми або всієї бази даних.

Існує загальний принцип: не варто звертатися із клієнтського додатка до таблиць бази даних прямо. Для зміни даних краще використати збережені процедури, а для запитів на читання - збережені процедури або подання.

Причина проста: якщо буде потрібно поміняти структуру вашої бази даних (наприклад, якусь таблицю поділити на поточну й архівну або додати новий стовпець) не буде потрібно вносити зміни в клієнтський додаток. Це варто пам'ятати й при наданні дозволів. Відзначимо також, що за допомогою збережених процедур можна дуже просто реалізувати додаткові перевірки в додавання до звичайних дозволів;

SQL Server дозволяє набудовувати дозволу на рівні окремих стовпців. На практиці краще не користуватися такими дозволами через падіння продуктивності й ускладнення системи дозволів. Якщо користувачеві можна бачити не всі стовпці в таблиці (наприклад, йому не потрібні домашні телефони співробітників), то вірніше буде створити подання або збережену процедуру, які будуть відфільтровувати непотрібні стовпці.

Хід роботи

- 1) Створіть логин SQL Server «Admin» і призначте їй роль sysadmin;
- 2) Створіть у базі даних роль «_____» і призначте їй дозволу на вибірку даних із всіх таблиць, зміна даних у декількох певних таблицях і запуск збереженої процедури;
- 3) Створіть логин SQL Server "Ivanov" і зіставте його з однойменним користувачем у базі даних. Призначте створеному користувачеві роль «_____».
- 4) Оформити звіт по виконанню лабораторної роботи.

Звіт повинен містити

- 1) Тему та мету лабораторної роботи.
- 2) Команди та результати їх виконання.
- 3) Висновок.

Контрольні питання

- 1) Хто є власником бази даних?
- 2) Якими правами володіють інші користувачі стосовно Вашої бази

даних?

3) Якими правами володіє адміністратор бази даних стосовно Вашої бази даних?

4) Яким образом надаються права на користування базою даних й окремих її таблиць?

5) Яким образом вилучаються права на користування базою даних й окремих її таблиць?

6) Що таке зовнішня база даних?

7) Як ідентифікується таблиця зовнішньої бази даних?

8) Як ідентифікується таблиця зовнішньої розподіленої бази даних?

9) Привілей GRANT OPTION дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) обновляти привілею.

10) Привілей USAGE дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) обновляти привілею.

11) Привілей RELOAD дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) обновляти привілею.

12) Привілей FILE дозволяє користувачеві

- а) завантажувати дані з файлу;
- б) передавати свої привілеї іншим користувачам;
- в) зареєструватися в системі;
- г) обновляти привілею.

Література

- 1 Веллинг, Люк. MySQL Учебное пособие [Текст]: Пер. с англ. / Люк Веллинг, Лора Томпсон – М.: Издательский дом «Вильямс», 2005. – 304 с.
- 2 Фиайли, К. SQL Руководство по изучению языка [Текст]: Пер. с англ. / Крис Фиайли – М.: ДМК Пресс; СПб.: Питер, 2004, – 464с.: ил.
- 3 Дейт, К.Дж. Введение в системы баз данных [Текст]: Пер. с англ., 7-е изд. / К. Дж. Дейт – М., СПб., Киев: Вильямс, 2001. – 171 с.

Розробив: викладач Ланська С.С.
Розглянуто та схвалено
на засіданні циклової комісії
програмної інженерії
Протокол № 7 від 07.02.2023 р.
Голова комісії _____ Ланська С.С.

Критерії оцінювання захисту лабораторної роботи

з дисципліни «Бази даних»

Захист лабораторної роботи вміщує:

- виконання індивідуального завдання згідно теми лабораторної роботи;
- оформлення звіту з лабораторної роботи, згідно з вимогами, наданими в методичних рекомендаціях до виконання лабораторних робіт;
- відповідь студента на контрольні запитання викладача.

Максимальна кількість балів за виконання та захист лабораторної роботи – 3 балів.

Критерії оцінювання та система нарахування балів за виконання та захист лабораторної роботи наведено у таблиці № 1:

Таблиця № 1

<i>Види завдань</i>	<i>Кількість балів</i>	<i>Критерії оцінювання виконання завдання</i>	<i>Максимальна кількість балів</i>
Реалізація програми	2	Студент правильно в повному об'ємі виконує індивідуальне завдання згідно завдання лабораторної роботи	2
	1	Студент в повному об'ємі виконує індивідуальне завдання, але допускає логічні помилки. Або Студент правильно виконує індивідуальне завдання в об'ємі 50%-60%	
	0	Студент неправильно виконує індивідуальне завдання.	
Оформлення звіту	0,5	Студент правильно оформляє звіт з лабораторної роботи згідно до вимог, наданих в методичних рекомендаціях.	0,5
	0,25	Студент робить помилки при розробці блок-схеми алгоритму рішення задачі.	
	0	Студент неправильно або в неповному об'ємі оформляє звіт з лабораторної роботи згідно до вимог, наданих в методичних рекомендаціях.	
Відповідь на контрольне питання	0,5	Студент правильно в повному об'ємі відповідає на контрольний запит до лабораторної роботи.	0,5
	0,25	Студент допускає помилки при відповіді контрольний запит до лабораторної роботи.	
	0	Студент неправильно або зовсім не відповідає на контрольний запит до лабораторної роботи.	
Разом			3

Відповідність кількості набраних студентом балів оцінці за 4 – бальною шкалою оцінювання наведена в таблиці № 2.

Таблиця № 2

Бальна	Національна
2,5-3	відмінно
2-2,4	добре
1,5	задовільно
0-1	незадовільно