

Циклова комісія комп'ютерної інженерії

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
до виконання лабораторних робіт з дисципліни

ОРГАНІЗАЦІЯ БАЗ ДАНИХ

спеціальність

123 Комп'ютерна інженерія

освітня програма

Обслуговування комп'ютерних систем і мереж

(назва освітньої програми)

відділення

Комп'ютерних технологій та систем

(назва відділення)

**Дніпро
2023 рік**

ЗМІСТ

1.	Аналіз предметної області та вимог користувача до проєктованої бази даних. Ви-ділення об'єктів предметної області та встановлення відношень між об'єктами. Побудова концептуальної моделі даних проєктованої бази даних.	3
2.	Спрощення концептуальної моделі даних проєктованої бази даних.	20
3.	Виділення таблиць реляційної бази даних. Перевірка таблиць за допомогою кон-цепції послідовної нормалізації.	27
4.	Побудова ER – діаграми реляційної бази даних.	38
5.	Створення бази даних, таблиць та індексів. Введення даних в таблиці. Модифікація таблиць бази даних.	47
6.	Прості запити. Використання опцій SELECT, FROM та WHERE. Використання опцій DISTINCT, ORDER BY, LIKE, BETWEEN, IN.	61
7.	Багатотабличні запити. Природні, внутрішні й перехресні об'єднання.	72
8.	Складні запити. Створення підзапитів.	80
9.	Генератори. Тригери. Конструкції мови SQL.	89
10.	Збережені процедури	95
11.	Резервування й відновлення баз даних. Резервування й відновлення таблиць бази даних	100
12.	Створення облікових записів. Встановлення рівнів привілей користувачів.	106

КРИТЕРІЇ ОЦІНЮВАННЯ

виконання студентом лабораторної роботи

При виконанні та захисті лабораторної роботи студент повинен:

1. Виконати практичну частину лабораторної роботи згідно його варіанта.
2. Оформити звіт з лабораторної роботи згідно вимог, вказаних в методичних рекомендаціях до виконання лабораторної роботи.
3. При захисті лабораторної роботи дати відповіді на контрольні питання викладача.

Розподіл балів та критерії оцінювання виконаної лабораторної роботи наведено в таблиці:

Лабораторна робота			
Бали	Критерій		
2	Реалізація практичної частини	2	Студент правильно в повному об'ємі реалізує практичну частину згідно завдання лабораторної роботи
		1	Студент в повному об'ємі реалізує практичну частину, але допускає логічні помилки. Або Студент правильно реалізує практичну частину в об'ємі 50%-60%
		0	Студент неправильно реалізує практичну частину
1	Оформлення звіту	1	Студент правильно оформляє звіт з лабораторної роботи згідно до вимог, наданих в методичних рекомендаціях.
		0,5	Студент робить помилки при розробці схеми або програм, які вказані в постановці завдання.
		0	Студент неправильно або в неповному об'ємі оформляє звіт з лабораторної роботи згідно до вимог, наданих в методичних рекомендаціях.
1	Відповідь на контрольне питання	1	Студент правильно в повному об'ємі відповідає на контрольне питання до лабораторної роботи.
		0,5	Студент допускає помилки при відповіді контрольне питання до лабораторної роботи.
		0	Студент неправильно або зовсім не відповідає на контрольне питання до лабораторної роботи.
Максимальна кількість балів: 4			

Методичні рекомендації до виконання лабораторної роботи № 1

Тема: Аналіз предметної області та вимог користувача до проектованої бази даних. Виділення об'єктів предметної області та встановлення відношень між об'єктами. Побудова концептуальної моделі даних проектованої бази даних.

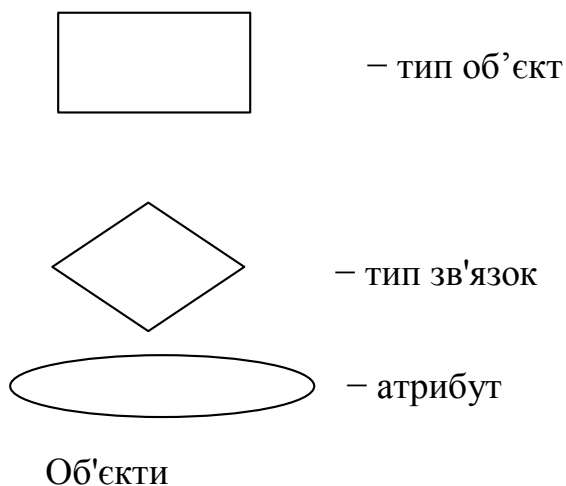
Мета:. Навчитися проводити аналіз предметної області проектованої (БД) виділяючи об'єкти предметної області, властивості об'єктів і встановлювати зв'язку між об'єктами.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

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:



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 моделюємії частини реальності. 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 пойменована 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.

Наприклад, екземпляр об'єкта

ФАКУЛЬТЕТ (Код факультету, Назва_факультету, ПІБ_Декана)

може однозначно ідентифікуватися кожним з перших двох зазначених атрибутів. Третій атрибут не рекомендується використати в якості ідентифікаційного, оскільки не можна гарантувати відсутність повного збігу значень атрибута ПІБ_Декана для декількох екземплярів даного об'єкта. Розглянемо об'єкт:

СТУДЕНТ (Номер залікової книжки, ПІБ_студента, Дата_народження).

Із трьох перерахованих атрибутів у наведеному прикладі тільки атрибут Номер_залікової_книжки однозначно ідентифікує кожен екземпляр об'єкта СТУДЕНТ. Атрибут Дата_Народження, навпроти, не може служити ключем, тому що будь-яка дана дата може бути вдень народження декількох різних людей. 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*. У розглянутому вище прикладі атрибут Код_факультету має меншу довжину, чим атрибут Назва_факультету, тому його варто вибрати як первинний ключ.

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.

Наприклад, об'єкт:

ЛІКУВАННЯ (ПІБ лікаря, ПІБ пацієнта, Дата призначення, Ліки)

однозначно ідентифікувати можна тільки складеним ключем:

(ПІБ лікаря, ПІБ пацієнта, Дата призначення).

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.

Приклад виконання лабораторної роботи

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 долі співробітника в загальному гонорарі за проєкт.

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.

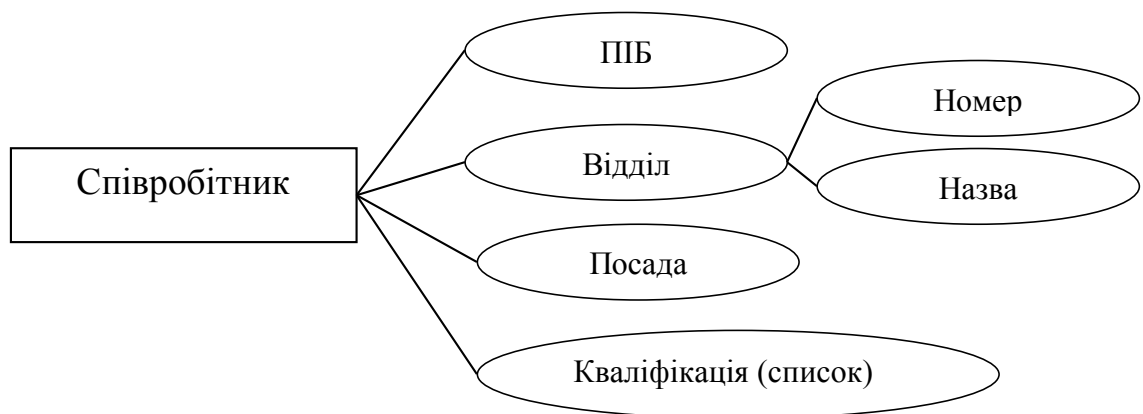
3 Один співробітник може бути керівником декількох проєктів, але кожен проєкт має одного керівника;

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

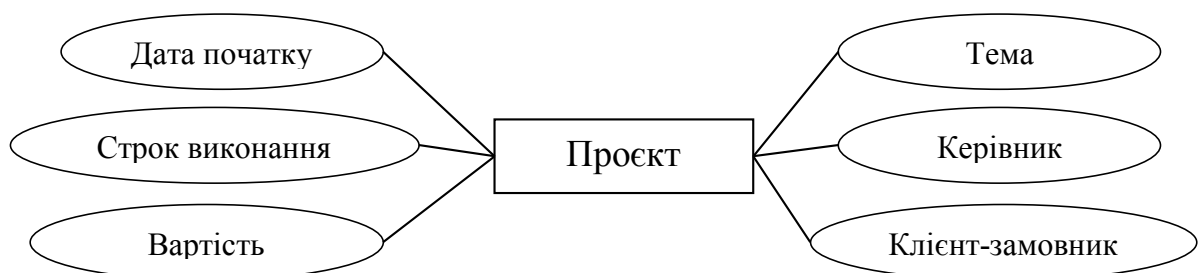
3 Опис об'єктів предметної області

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

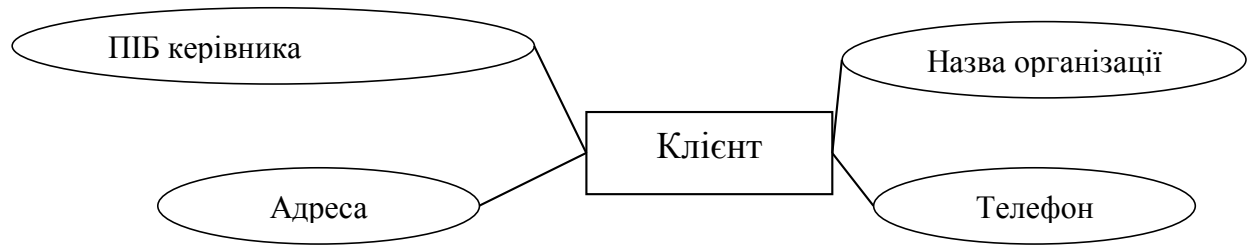
1 Співробітник



2 Проєкт



3 Клієнт

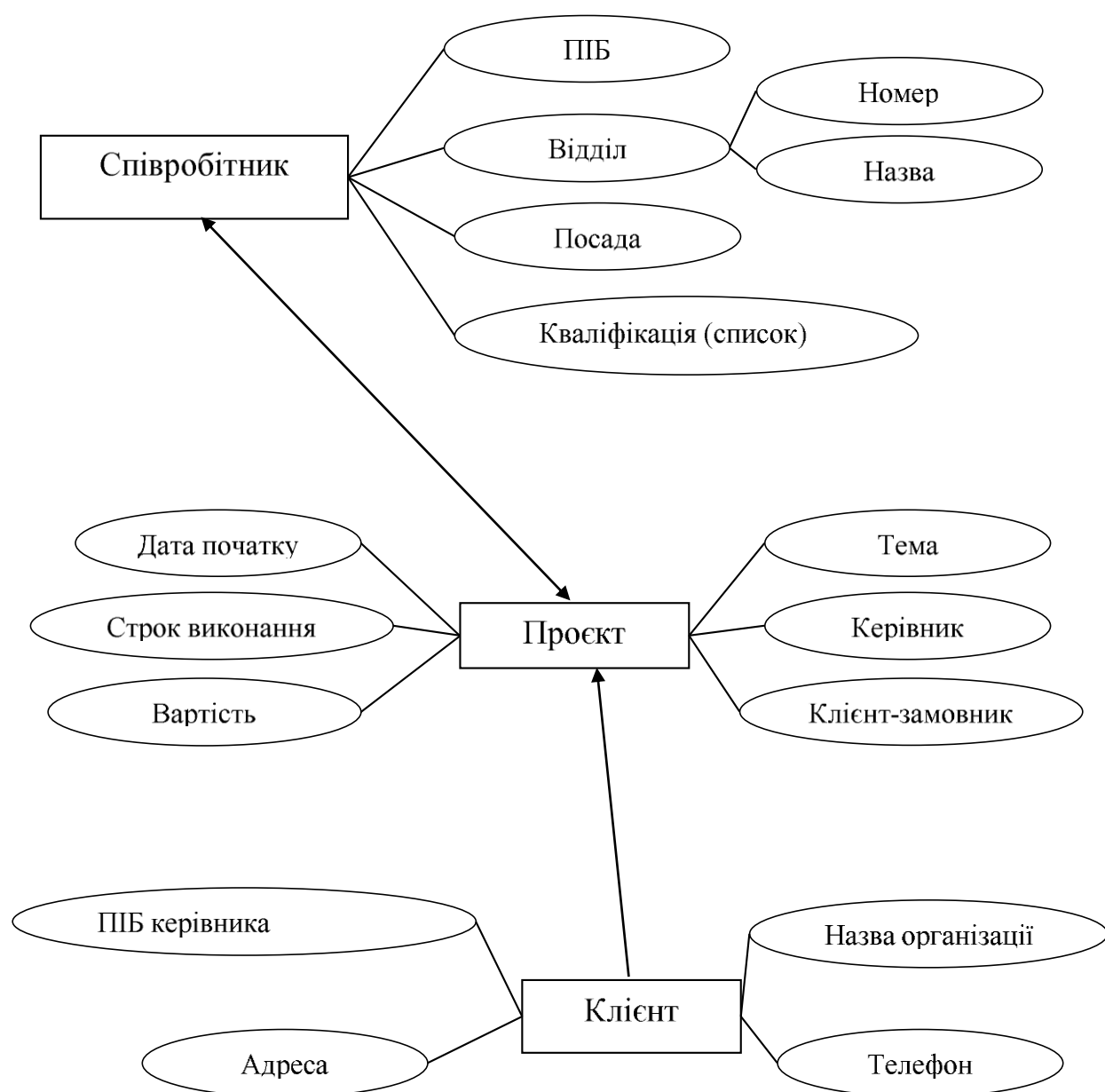


4 Концептуальна схема предметної області

1 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 хто до багатьох"

2 З огляду на, що одна організація - замовник (один клієнт) може замовляти кілька проектів, між об'єктами Клієнт і Проект встановлюємо зв'язок "один до багатьох".

Концептуальна схема предметної області:



ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

- I. Провести аналіз предметної області згідно Вашого варіанта. Уточнити вимоги користувача до проектованої бази даних.
- II. Виділити об'єкти предметної області і їхні атрибути.
- III. Встановити зв'язок між об'єктами. Побудувати концептуальну модель предметної області.
- IV. Оформити звіт по виконанню лабораторної роботи. Звіт повинен включати наступні розділи:
 1. Постановка завдання.
 2. Аналіз предметної області.
 3. Опис об'єктів предметної області з їхніми атрибутами.
 4. Концептуальна схема предметної області.
 5. Висновок.

ВАРІАНТИ ЗАВДАНЬ ЛАБОРАТОРНОЇ РОБОТИ

Варіант 1. Предметна область: *Коледж*.

Необхідна інформація:

A. Відділення коледжу: Назва відділення; завідувач відділенням; предметні комісії, що входять до складу відділення.

Враховувати:

Завідувач відділенням є співробітником коледжу. Відділення має тільки одного завідувача. Один співробітник може бути завідувачем тільки одного відділення.

Відділення може містити кілька предметних комісій. Одна предметна комісія входить до складу тільки одного відділення.

B. Предметна комісія: Назва предметної комісії, голова предметної комісії, співробітники предметної комісії.

Враховувати:

Предметна комісія має тільки одного голову. Один співробітник може бути головою тільки однієї предметної комісії.

У предметній комісії працює кілька співробітників. Один співробітник може працювати в декількох предметних комісіях.

С. Співробітник коледжу: ПІБ співробітника; дата народження; освіта; кваліфікація; стаж роботи; у якій (у яких) предметній комісії працює; ставка.

Варіант 2. Предметна область: *Склад будівельних матеріалів.*

Необхідна інформація:

А. Товари на складі: Найменування товару; виробник; кількість товару на складі; ціна продажу одиниці товару.

Враховувати:

Виробник може постачати на склад різні товари. Один товар може бути поставлений різними виробниками.

Ціна одного й того ж товару від різних виробників може відрізнитись.

В. Виробник: Найменування виробника; країна; список товарів, які виробляються; ціна поставки одиниці товару.

Варіант 3. Предметна область: *Магазин продовольчих товарів.*

Необхідна інформація:

А. Товари в асортименті: Найменування товару; виробник; ціна за одиницю товару; строк зберігання (у днях).

Враховувати:

Виробник може поставляти в магазин різні товари. Один товар може бути поставлений різними виробниками.

Ціна того самого товару від різних виробників може відрізнитись.

В. Товари в магазині: Найменування товару; виробник; кількість товару в магазині; дата поставки.

С. Виробник: Найменування виробника; адреса виробника ; список товарів, які виробляються; ціна поставки одиниці товару.

Варіант 4. Предметна область: *Складальний цех.*

Необхідна інформація:

А. Вироби, що виготовляють у цеху: Найменування виробу; деталі, необхідні для зборки; робітник, що зібрав виріб; контроль якості ("пройшов"/ "не пройшов").

Враховувати:

При виготовленні виробу використовуються різні деталі. Та сама деталь може використатися при виготовленні різних виробів

У переліку деталей необхідно вказати назву деталі й кількість, що використовується для зборки одного виробу.

В. Деталі: Найменування деталі; найменування цеху, що виготовив деталь.

Варіант 5. Предметна область: *Дисципліни коледжу.*

Необхідна інформація:

А. Дисципліни: Найменування дисципліни; кількість годин; вид контролю (залік/іспит); викладач, що викладає дисципліну.

Враховувати:

Ту саму дисципліну можуть викладати різні викладачі. Один викладач може викладати різні дисципліни.

В. Викладачі: ПІБ викладача; дисципліни, які він викладає та групи, в яких викладаються ці дисципліни.

Враховувати:

Одну дисципліну викладач може читати в різних групах. В одній групі читаються різні дисципліни.

Варіант 6. Предметна область: *Бібліотека.*

Необхідна інформація:

А. Книги: Автор (автори); назва книги; рік видання; видавництво; кількість екземплярів.

Враховувати:

Одна книга може мати декілька авторів. Один автор може написати кілька книг.

В. Читачі: ПІБ читача; адреса; телефон.

С. Необхідно зберігати інформацію про видані читачеві книги: яка книга видана; кому книга видана; дата видачі; дата повернення; ознака повернення.

Враховувати:

Та сама книга може бути видана різним читачам. Один читач може взяти кілька книг.

Варіант 7. Предметна область: *Сесія*.

Необхідна інформація:

А. Студенти: ПІБ студента; група.

В. Іспити й заліки: Дисципліна; ознака - іспит/залік; дата; викладач; група; студент; оцінка студента.

Враховувати:

Одну дисципліну здає багато студентів з різних груп.

Один студент здає різні дисципліни.

Один викладач приймає заліки й іспити в різних групах і по різних дисциплінах.

Варіант 8. Предметна область: *Прокат спортивного обладнання*

Необхідна інформація:

А. Обладнання: Найменування; країна - виробник; кількість в наявності; ціна прокату за добу.

В. Клієнти: ПІБ клієнта; паспортні дані; адреса; телефон.

С. Необхідно зберігати інформацію про видане обладнання: яке обладнання видано; кому видано; дата видачі; дата повернення; ознака повернення; сума оплати за прокат; ознака оплати.

Враховувати:

Обладнання одного виду може бути видано різним клієнтам. Один клієнт може взяти на прокат різне обладнання.

Варіант 9. Предметна область: *Поставка - продаж непродовольчих товарів.*

Необхідна інформація:

А. Товари: Найменування товару; постачальник; кількість в наявності.

Враховувати:

Один постачальник може поставляти в магазин різні товари. Один товар може бути поставлений різними постачальниками.

В. Постачальник: Назва; адреса; телефон.

С. Поставка товару: Найменування товару; постачальник; дата поставки; кількість; ціна поставки за одиницю товару.

Д. Продаж товару: Найменування товару; дата продажу; продана кількість.

Враховувати:

Ціна поставки й ціна продажу того самого товару може залежати від дати(облік інфляції).

Варіант 10. Предметна область: *Банківські кредити.*

Необхідна інформація:

А. Види кредитів: Найменування кредиту; процентна ставка по кредиту; мінімальна сума кредиту; максимальна сума кредиту; максимальний строк погашення кредиту (у місяцях).

В. Клієнти банку: ПІБ клієнта; паспортні дані; адреса; телефон.

С. Інформація з виданих кредитів: який кредит виданий; кому виданий; сума кредиту; дата видачі кредиту; дата погашення кредиту; сума погашення з урахуванням процентної ставки; ознака погашення.

Ураховувати:

Той самий вид кредиту може бути виданий різним клієнтам. Один клієнт може взяти в банку різні кредити.

Варіант 11. Предметна область: *Туристична фірма.*

Необхідна інформація:

А. Клієнти: ПІБ клієнта; паспортні дані; адреса; телефон.

В. Співробітники фірми: ПІБ співробітника; посада; паспортні дані; адреса; телефон.

С. Туристичні маршрути: Найменування маршруту; країна; опис маршруту (список відвідуваних міст); кількість днів; вид транспорту; вартість путівки.

Ураховувати:

На кожен маршрут є по кілька путівок на різні дати від'їзду.

Д. Інформація із проданих путівок: клієнт; маршрут; дата виїзду; співробітник, що продав путівку.

Ураховувати:

Той самий вид маршруту може бути проданий різним клієнтам. Один клієнт може купити путівки на різні маршрути

Один співробітник фірми може обслужити декількох клієнтів. Один клієнт може купити путівки в різних співробітників.

Варіант 12. Предметна область: *Поліклініка.*

Необхідна інформація:

А. Лікарі: ПІБ; посада; спеціалізація; кваліфікація; номер кабінету; графік роботи (по днях тижня із вказівкою часу початку й кінця прийому).

Враховувати:

У поліклініці може працювати кілька лікарів кожної спеціалізації.

В. Пацієнт: ПІБ; паспортні дані; адреса; телефон.

С. Інформація об відвідування пацієнтами лікарів поліклініки: пацієнт, лікар, дата відвідування, час, поставлений діагноз.

Враховувати:

Один пацієнт може відвідати різних лікарів. Один лікар приймає багатьох пацієнтів.

Варіант 13. Предметна область: *Абітурієнти коледжу.*

Необхідна інформація:

А. Анкета абітурієнта: ПІБ; дата народження; закінчений навчальний заклад (назва, номер, населена пункт, номер диплома); дата закінчення навчального закладу;

наявність червоного диплома або золотий/ срібної медалі; адреса; телефон; найменування обраної спеціальності.

Ураховувати:

Один абітурієнт може подавати документи на декілька спеціальностей.

В. Спеціальності: Назва спеціальності; кількість бюджетних місць; кількість контрактних місць; список дисциплін, які здаються при вступі.

С. Вступні іспити: Абітурієнт; дисципліна; екзаменаційна оцінка.

Враховувати:

Ту саму дисципліну здає багато абітурієнтів.

Один абітурієнт здає різні дисципліни.

Варіант 14. Предметна область: *Ательє по пошиттю одягу.*

Необхідна інформація:

А. Майстри: ПІБ; посада; кваліфікація.

В. Клієнти: ПІБ; адреса; телефон.

С. Замовлення: Номер замовлення; клієнт; найменування виробу; майстер, що виконує замовлення; дата замовлення; строк виконання замовлення; вартість.

Враховувати:

Один майстер може обслуговувати декількох клієнтів. Один клієнт може обслуговуватися в різних майстрів.

Варіант 15. Предметна область: *Поставки товарів у мережу магазинів.*

Необхідна інформація:

А. Магазины: Назва магазину; ПІБ власника; адреса.

Враховувати:

Одна людина може бути власником декількох магазинів.

В. Постачальник: Найменування постачальника; телефон.

С. Товари: Найменування товару; країна-виробник;

Д. Умови поставки товарів: Товар; постачальник; мінімальний строк поставки; мінімальна партія поставки; ціна поставки за одиницю товару.

Е. Поставка: Товар; постачальник; магазин; обсяг поставки; дата поставки.

Ураховувати:

Один постачальник може поставляти товари в різні магазини.

Один магазин може одержувати товари від різних постачальників (у тім числі той самий товар).

Варіант 16. Предметна область: **Транспортні перевезення.**

Необхідна інформація:

А. Автомобілі: Марка автомобіля; модель; державний номер; вантажопідйомність; витрата бензину; ПІБ водія.

Ураховувати:

Фірма може мати кілька автомобілів однакової марки й моделі.

В. Заявки: Код заявки; дата й час заявки; Назва вантажу; кількість вантажу; пункт доставки; строк доставки (у годинниках).

С. Виконання заявок (доставка вантажу): Заявка; автомобіль; дата й час відправлення; дата й час прибуття; пройдена відстань.

Ураховувати:

Один автомобіль може доставляти вантаж за різними заявками й у різні дні.

Вантаж по одній заявці може бути доставлений декількома автомобілями.

Варіант 17. Предметна область: **Автосалон.**

Необхідна інформація:

А. Автомобілі: Марка; модель; колір; рік випуску; країна-виробник; ціна за базову комплектацію.

В. Устаткування доукомплектації: Найменування; виробник; ціна.

С. Клієнти: ПІБ клієнта; паспортні дані; адреса; телефон.

Д. Співробітники фірми: ПІБ співробітника; посада; паспортні дані; адреса; телефон.

Е. Продаж автомобілів: який автомобіль проданий; якому клієнтові; хто із співробітників виконав продаж; чи була проведена доукомплектація (чим доукомплектований автомобіль і ціна доукомплектації); дата продажу; ціна продажу.

Варіант 18. Предметна область: *SPA - салон*.

Необхідна інформація:

- А. Процедури (послуги салону): Найменування; тривалість виконання; вартість.
- В. Співробітники: ПІБ співробітника; телефон; графік роботи (по днях тижня з указаним часом початку й кінця роботи); список процедур, які може виконувати співробітник.
- С. Обслуговування клієнтів: ПІБ клієнта; дата відвідування салону; які процедури були виконані й ким зі співробітників.

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;
- зв'язків.

7 Приведіть приклад концептуальної моделі деякої предметної області.

8 Побудуйте концептуальну модель деякої предметної області, використовуючи поняття складеного об'єкта.

9 Поясніть поняття спеціалізації, генералізації, категоризації.

Методичні рекомендації до виконання лабораторної роботи № 2

Тема: Спрощення концептуальної моделі даних проєктованої бази даних.

Мета: Придбання практичних навичок спрощення концептуальної схеми предметної області до реляційної моделі даних.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

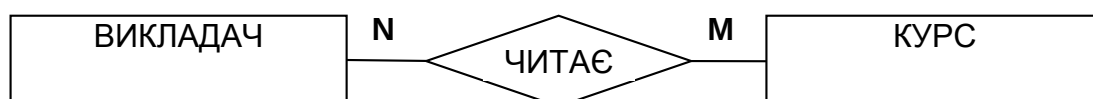
Перетворення концептуальної моделі даних у логічну модель, у результаті якого буде визначена схема реляційної моделі даних, може мати два підходи. Один з підходів полягає в тому, що проєктувальник працює з концептуальною моделлю прямо, не прибігаючи до її попереднього перетворення. У цьому випадку йому 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 проєктувальник, 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 хто до багатьох";
- складні зв'язки;
- рекурсивні зв'язки;
- зв'язку з атрибутами;
- множинні атрибути;
- надлишкові зв'язки.

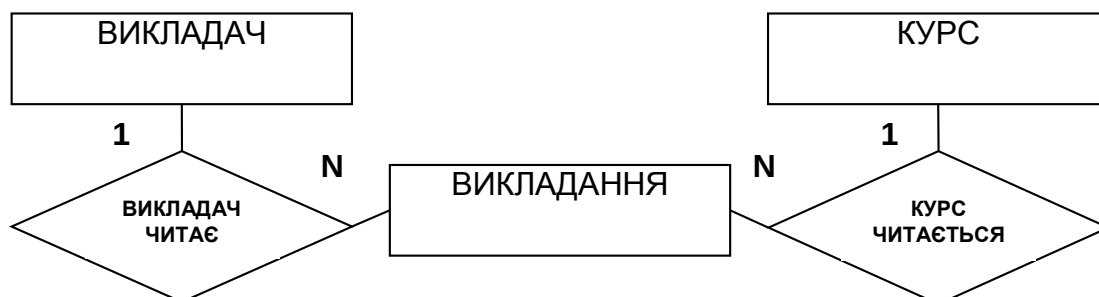
Точка зору другого підходу на процес перетворення концептуальної моделі така: якщо в моделі присутні перераховані небажані структури, то вони повинні бути виключені шляхом тотожної їхньої заміни на структури даних, припустимі для логічної моделі. Розглянемо спочатку другий підхід, що припускає наявність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

Виключення зв'язку типу "багато хто до багатьох"

Перетворення зв'язку типу "багато хто 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.



Введемо додатковий об'єкт ВИКЛАДАННЯ й визначимо нові два зв'язки: ВИКЛАДАЧ_ЧИТАЄ типу 1:N між об'єктами ВИКЛАДАЧ і ВИКЛАДАННЯ; КУРС_ЧИТАЄТЬСЯ типу 1:N між об'єктами КУРС і ВИКЛАДАННЯ.



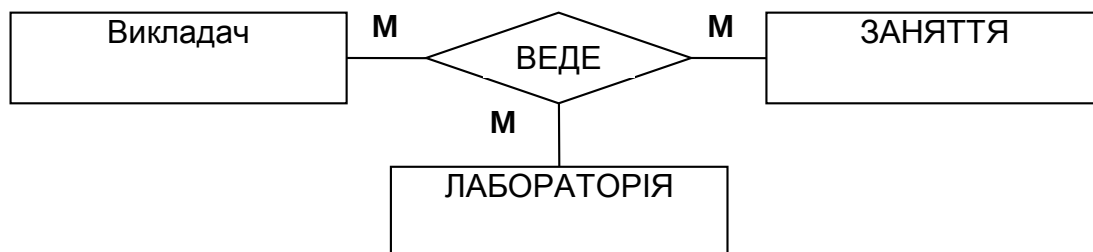
Виключення складних зв'язків

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:

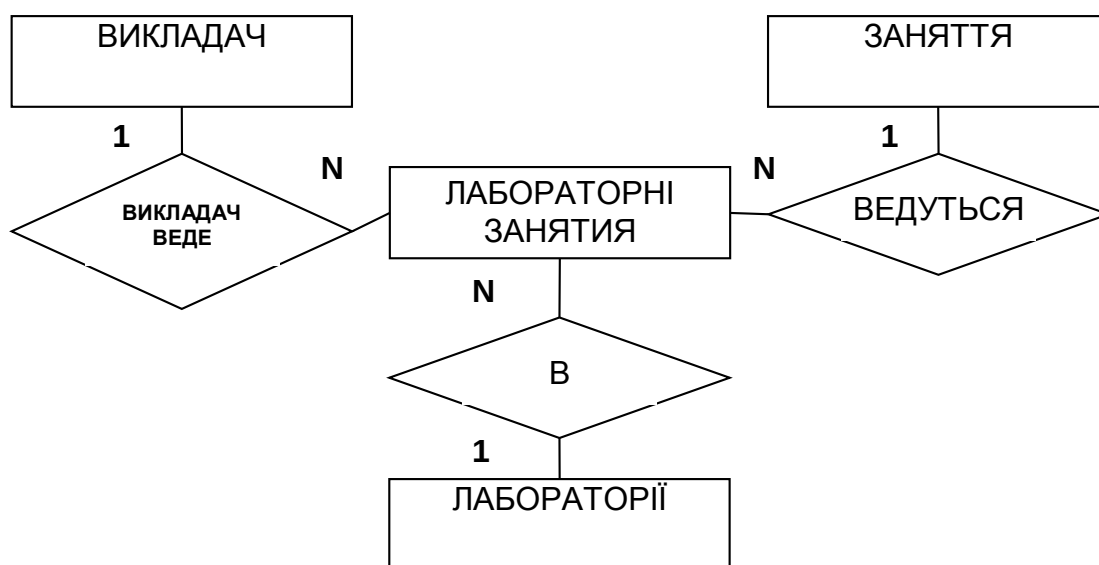
у модель уводиться новий об'єкт;

складний зв'язок з бінарними зв'язками типу "один до багатьох" вихідних об'єктів зі знову створеним об'єктом, причому кількість бінарних з дорівнює ступеня складного зв'язку.

Наприклад, нехай заняття за деяким курсом ведуться деяким викладачем у деякій лабораторії. Концептуальна модель у цій ситуації може містити наступну структуру даних:



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.

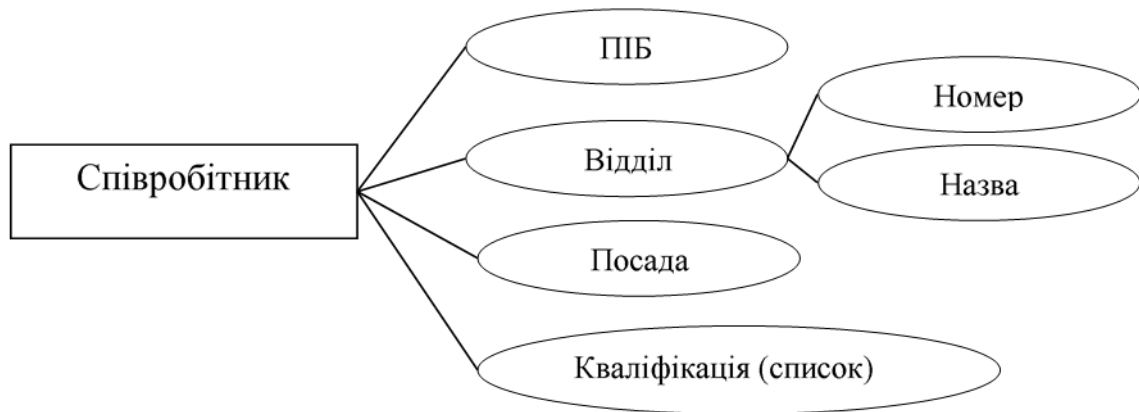


Отримана в результаті даного перетворення концептуальна модель уже не містить небажаних для реляційної схеми бази дані структури.

Приклад виконання лабораторної роботи

I Аналіз атрибутів предметної області

Об'єкт Співробітник

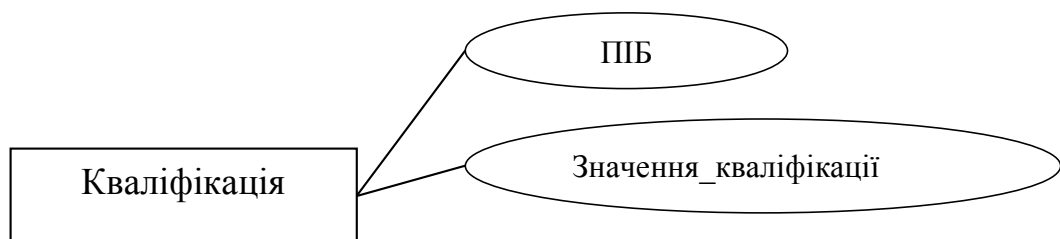


Має дві складові атрибута: Відділ і Кваліфікація.

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 з атрибутами: ФІО співробітника й Значення кваліфікації

Новий об'єкт Кваліфікація



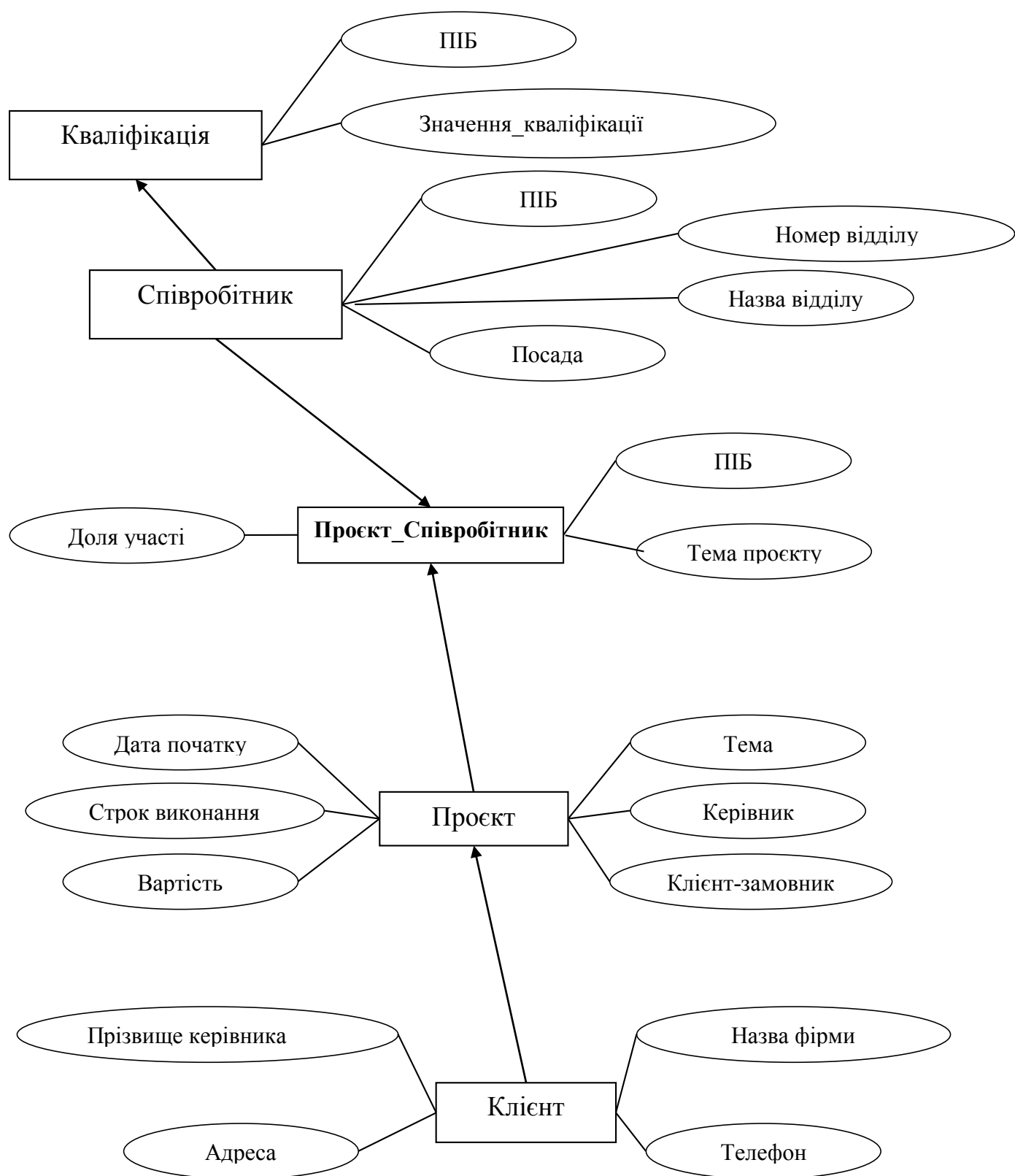
2. Розрив зв'язків "багато до багатьох" проміжними об'єктами.

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 ФІО співробітника, що працюють у проекті, Тема проекту й Частка участі співробітника в проекті.

Новий проміжний об'єкт Проект_Співробітник



3 Спрощена концептуальна схема предметної області:



ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

- I. За результатами виконання лабораторної роботи № 12 проаналізувати атрибути виділених об'єктів. Провести заміну складених атрибутів простими.
- II. Розірвати зв'язку типу "багато до багатьох" проміжними об'єктами.
- III. Оформити звіт по виконанню лабораторної роботи. Звіт повинен включати наступні розділи:
 1. Аналіз атрибутів предметної області.
 2. Розрив зв'язків "багато до багатьох" проміжними об'єктами.
 3. Удосконалена концептуальна схема предметної області.
 4. Висновок.

КОНТРОЛЬНІ ПИТАННЯ

- 1 Поясните 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.

Методичні рекомендації до виконання лабораторної роботи № 3

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).

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

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;

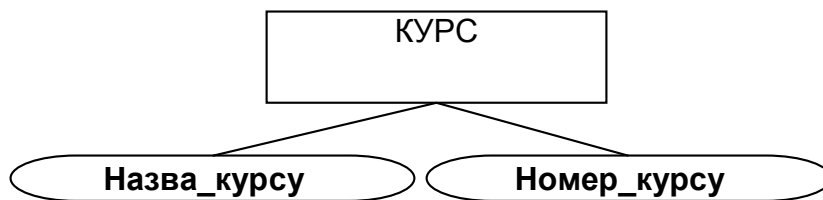
Розглянемо правила перетворення кожної із зазначених структур.

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.



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, П_І_Б). 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



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 атрибути 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.

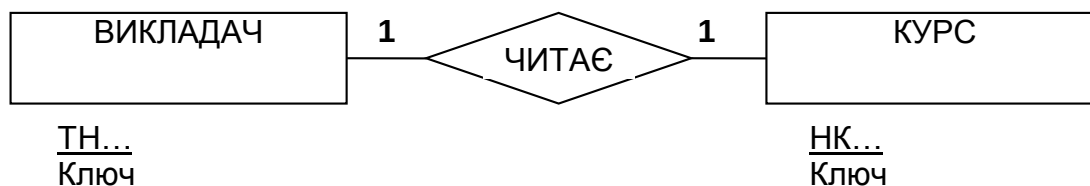


Цей зв'язок може бути зображена за допомогою діаграми, що містить всю інформацію, необхідну для генерації відповідних відносин РБД. Об'єкт ВИКЛАДАЧ і КУРС однозначно ідентифікуються за допомогою ТН (табельний номер викладача) і НК (номер курсу). Якщо всі елементи даного об'єкта повинні брати участь у 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 проекрованої 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. Наприклад, нехай у проектованій БД повинна зберігатися наступна інформація: ТН - табельний номер викладача; П_І_Б - прізвище,

ім'я, по батькові викладача; Тел_ Викладача - телефон викладача; НК - номер курсу; Назв_ Курсу - назва курсу.



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

Уважаючи, що клас приналежності є обов'язковим для обох об'єктів, одержуємо відношення: ВИКЛАДАЧ (ТН, П_І_Б, 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) - немає. 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.

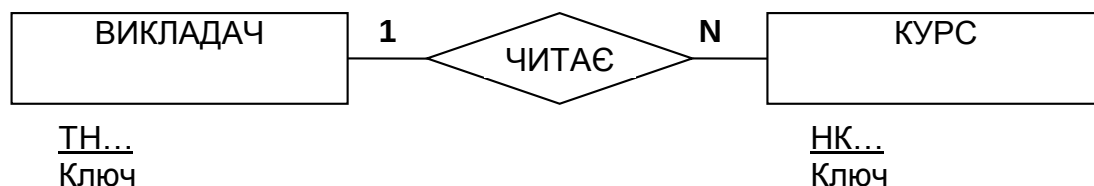
У результаті одержуємо наступну реляційну схему: ВИКЛАДАЧ (ТН, П_І_Б, Тел_Викладача, НК); КУРС (НК, Назв_Курсу).

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, П_І_Б, Тел_Викладача, НК). Можливий й інший варіант: ВИКЛАДАЧ (ТН, П_І_Б, 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, П_І_Б, 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.



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 "одиничної" 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, П_І_Б, 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. Методичні рекомендації до виконання лабораторних робіт №№ 1-2) 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, Доля_участі)

Клієнт(Назва_фірми, Прізвище_керівника, 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 - встановимо його ключовим атрибутом.

У таблицях Кваліфікація й Проект_співробітник атрибути ФІО_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 - встановимо його ключовим атрибутом.

У таблиці Проект атрибут Клієнт_замовник замінимо на атрибут Код_клієнта.

2.3 Для встановлення зв'язку між таблицями Проект і Про-ект_співробітник уведемо атрибут Код_проекту. У таблиці Проект це буде до-полнительный атрибут - встановимо його ключовим атрибутом.

У таблиці Проект_Співробітник атрибут Тема_проекту замінимо на атрибут Код_проекту.

У результаті одержимо наступні таблиці реляционной бази даних (з виділеними ключевими атрибутами й зовнішніми клбчами):

Співробітник(Код_співробітника, ФІО_співробітника, Посада, Але^мер_відділу, Назва_відділу)

Кваліфікація(Код_співробітника , Значення кваліфікації)

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

Проект_Співробітник(Код_проекту, Код_співробітника, Доля_участі)

Клієнт(Код_клієнта, Назва, Прізвище_керівника, Адреса)

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 Доля_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 неключових 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)
та
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.

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

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

II. Встановити 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 включати наступні розділи:

1. Таблиці реляційної бази даних.
2. Атрибути зв'язку між таблицями.
3. Приведення таблиць до 3НФ.
4. Висновок.

КОНТРОЛЬНІ ПИТАННЯ

- 1 Поясните 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 Поясните терміни:
 - відношення;
 - кортеж;
 - атрибут і ступінь відношення;
 - первинний і зовнішній ключі;
 - домен;
 - кардинальне число.
- 5 Поясните властивості й види відношень.
- 6 Поясните зміст використання потенційних і первинних ключів відношень.

Як використовувати ключі для зв'язку відношень?

- 7 Поясните 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 Поясните 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,

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.

Методичні рекомендації до виконання лабораторної роботи № 4

Тема: Побудова ER – діаграми реляційної бази даних.

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

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Модель “сутність-зв'язок” є найбільш розповсюдженим підходом до побудови БД. Її застосування стала діючим стандартом при інфологічному проектуванні БД. Загальноприйнятим стало скорочене найменування ER-модель (Essence – сутність, Relation – зв'язок).

ER-модель добре співвідноситься з концепцією об'єктно-орієнтованого проектування і тому використовується в різних програмних засобах автоматизованого проектування систем (у так званих CASE-системах). Найбільш розповсюдженими засобами інфологічного проектування на даний час є, наприклад, програмні системи ERwin, POWER DESIGNER та ін.

Розглянемо коротко базові поняття ER-моделі (багато з них вам покажуться знайомими, тому що вони зв'язані з поняттями реляційної моделі даних).

Поняттям сутність представляється клас однотипних об'єктів. Передбачається, що існує множина екземплярів даної сутності, що представляють реальні об'єкти предметної області. Об'єкт, якому відповідає дана сутність, має свій набір атрибутів – характеристик, що описують властивості даного об'єкта. При цьому набір атрибутів повинний бути таким, щоб можна було розрізняти конкретні екземпляри даної сутності. Графічно сутність прийнято зображувати прямокутником, у верхній частині якого записується ім'я сутності, а нижче перелічуються всі атрибути даної сутності. На рис. 1 показаний приклад зображення сутності “Співробітник”

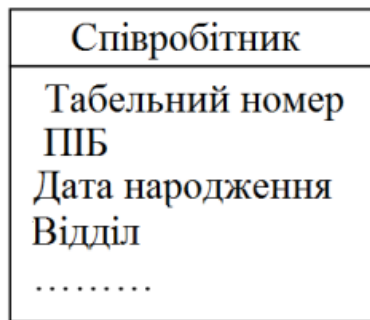


Рисунок 1 - Зображення сутності “Співробітник”

Ключовими атрибутами називається такий набір атрибутів, що однозначно ідентифікує кожен екземпляр сутності. Наприклад, для сутності “Співробітник” це може бути атрибут “Табельний номер”. Ключові атрибути виділяються підкресленням чи іншим шрифтом.

Між сутностями завжди встановлюються зв'язки, тому що всі об'єкти реального світу зв'язані між собою. Зв'язкам привласнюються найменування. Наприклад, між сутностями “Студент” і “Викладач” може існувати зв'язок “Читає лекції”. На схемі моделі зв'язки позначаються лініями, що з'єднують значки відповідних сутностей. Наприклад, так, як це показано на рис.2.

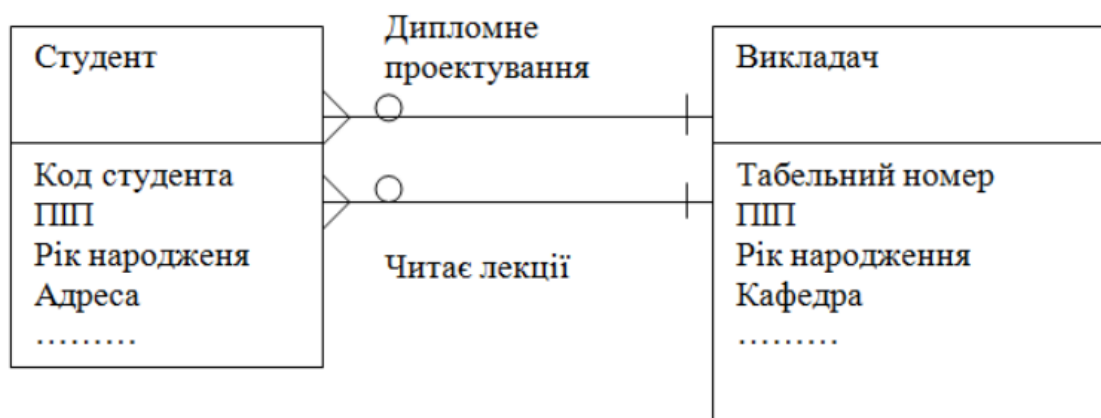


Рисунок .2 - Зображення зв'язку між сутностями

Зв'язки між сутностями можуть бути типів “один до одного” (1:1), “один до кількох” (1:M) чи “кілька до кількох” (M:M).

Зв'язок типу “один до одного” (1:1) означає, що один екземпляр першої сутності зв'язаний тільки з одним екземпляром другої сутності.

Зв'язок типу “один до кількох” (1:M) означає, що один екземпляр першої сутності може бути зв'язаний з декількома екземплярами другої сутності, але кожен екземпляр другої сутності

- зв'язана тільки з одним екземпляром першої сутності.

Зв'язок типу “кілька до кількох” (M:M) означає, що один екземпляр першої сутності може бути зв'язаний з декількома екземплярами другої сутності і навпаки, кожен екземпляр другої сутності може бути зв'язаний з декількома екземплярами першої сутності.

У розглянутому вище прикладі зв'язок “Читає лекції” є зв'язком типу “один до кількох”, причому першою сутністю тут є “Викладач”, а другою – “Студент”.

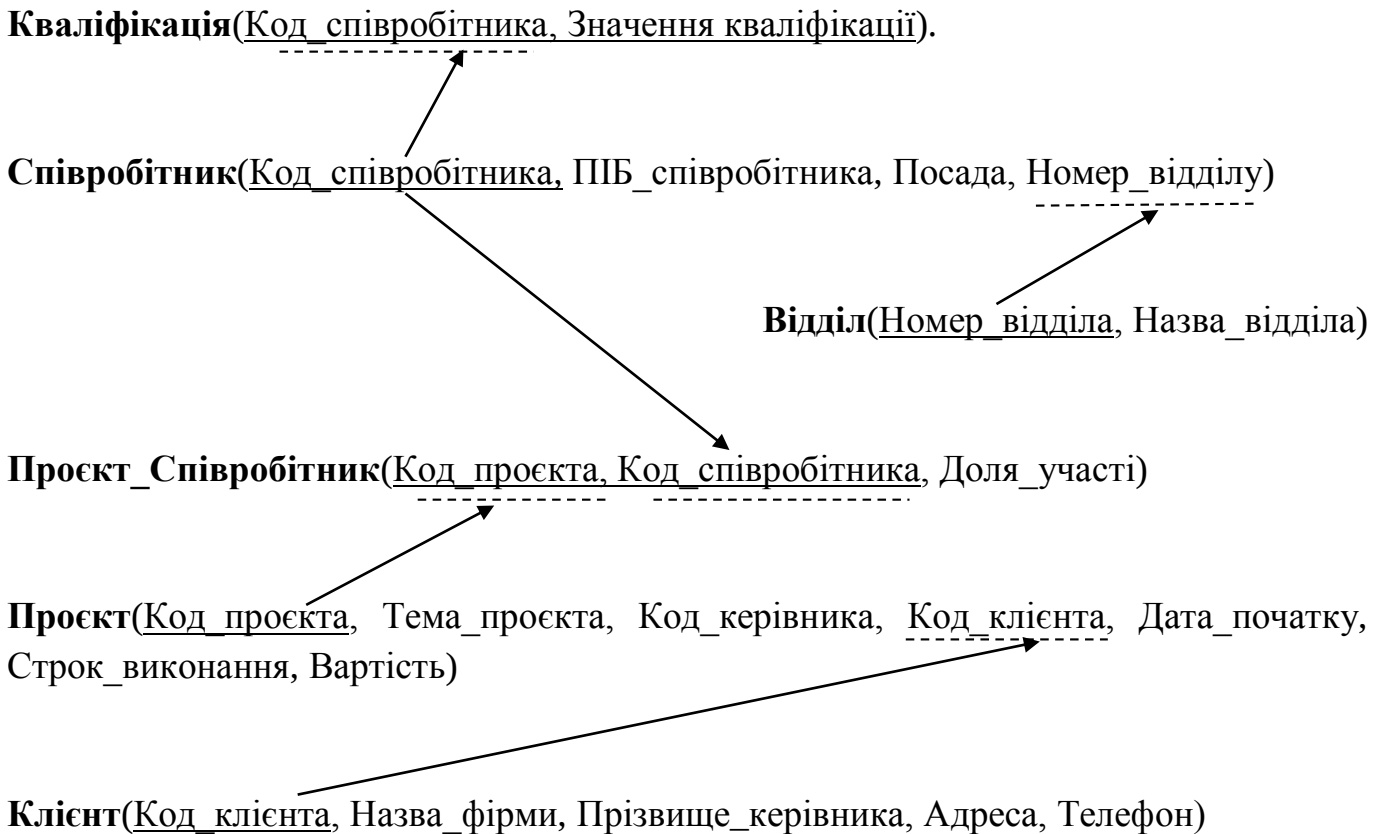
У різних CASE-системах використовуються різні способи позначення типів зв'язків. Наприклад, у системі POWER DESIGNER прийнято зв'язок “один до кількох” позначати потрійною лінією з боку “кілька” (приблизно так, як це показано на рис. 2). Між двома сутностями може бути встановлено як завгодно багато зв'язків. Наприклад, між сутностями “Студент” і “Викладач” ще може бути встановлений зв'язок “Дипломне проектування”. Очевидно, що цей зв'язок так само є зв'язком типу “один до кількох” – один викладач може керувати дипломною роботою декількох студентів, але кожен студент може мати тільки одного керівника дипломного проекту. Зв'язки будь-якого типу можуть бути обов'язковими чи необов'язковими.

Якщо в даному зв'язку повинен брати участь кожен екземпляр сутності, то зв'язок у даному випадку є обов'язковим, а якщо не кожен екземпляр – то необов'язковим. При цьому зв'язок може бути обов'язковим з однієї сторони і необов'язковим з іншої сторони.

Обов'язковий зв'язок на схемі прийнято позначати кружечком, а необов'язковий – перпендикулярною лінією, що перекреслює лінію зв'язку. На рис. 2 зв'язок “Дипломне проектування” з боку студента є обов'язковим, а з боку викладача – необов'язковим. Дійсно, кожен студент-дипломник повинен обов'язково мати керівника дипломного проекту, але не кожен викладач є керівником дипломного проекту.

Умовна схема зв'язків між таблицями бази даних.

Покажемо умовну схему зв'язків між отриманими в попередній лабораторній-роботі таблицями:



ER-діаграма реляційної бази даних.

Таблиці

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.

Таблиці

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.

Таблиці

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

Таблиці

Проект(Код_проекту, Тема_проекту, Код_керівника, 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.

Таблиці

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.

Таблиці

Клієнт(Код_клієнта, Назва_фірми, 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;

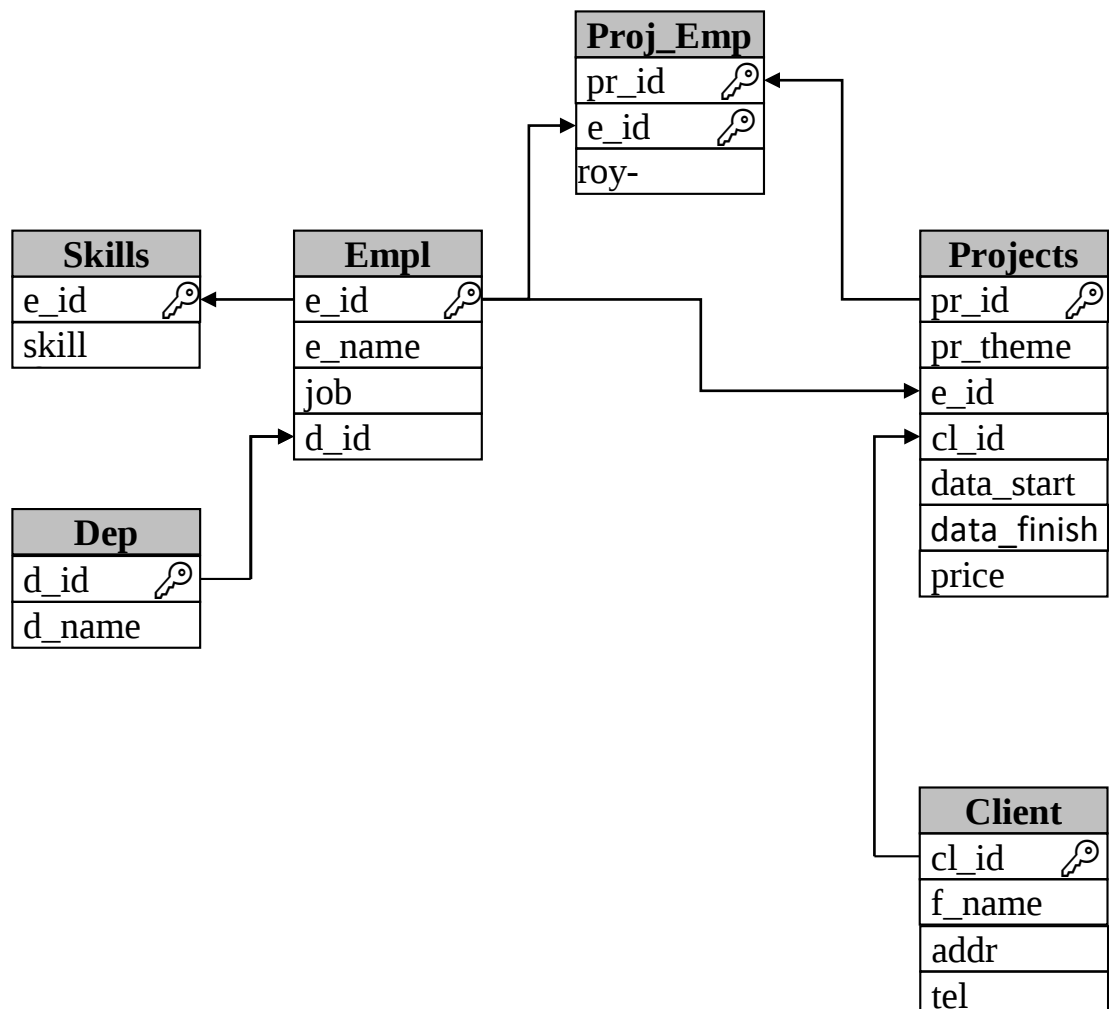
f_name - фіо керівника клієнта;

addr - адреса фірми-клієнта;

tel - телефон.

Дамо базі даних ім'я **Organization**

ER - діаграма бази даних



ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

I. На основі лабораторної роботи № 3 побудувати умовну схему зв'язків між таблицями бази даних.

II. Побудувати ER-діаграму реляційної бази даних.

III. Оформити звіт по виконанню лабораторної роботи.

Звіт повинен включати наступні розділи:

1. Умовна схема зв'язків між таблицями бази даних.

2. ER-діаграма реляційної бази даних.

3. Висновок.

КОНТРОЛЬНІ ПИТАННЯ

1 Поясните 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 Поясните терміни:

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

5 Поясните властивості й види відношень.

6 Поясните зміст використання потенційних і первинних ключів відношень.

Як використовувати ключі для зв'язку відношень?

7 Поясните 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 Розгляньте 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,
- 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.

17 Трьома видами аномалій є аномалії

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

Методичні рекомендації до виконання лабораторної роботи №№ 5-6

Тема: Створення бази даних, таблиць та індексів. Введення даних в таблиці. Модифікація таблиць бази даних.

Мета: Засобами MySQL навчитися створювати базу даних і визначати структуру таблиць, використовуючи оператори CREATE DATABASE й CREATE TABLE. Використовуючи оператор ALTER TABLE навчитися змінювати структуру таблиць бази даних. Навчитися вводити дані використовуючи оператори INSERT й LOAD DATA INFILE.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

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;
```

Переконалися в тім, 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 лабораторних роботах №№ 1-4.

1. *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, royalty_share)

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)
```

Перегляд структури таблиці

```
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)
```

Перегляд структури таблиці

```
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)
```

Перегляд структури таблиці

```
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)
```

Перегляд структури таблиці

```
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)
```

Перегляд структури таблиці

```
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)
```

Перегляд структури таблиці

```
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)
```

3 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_стовбця, . . . )]
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, . . .]
```

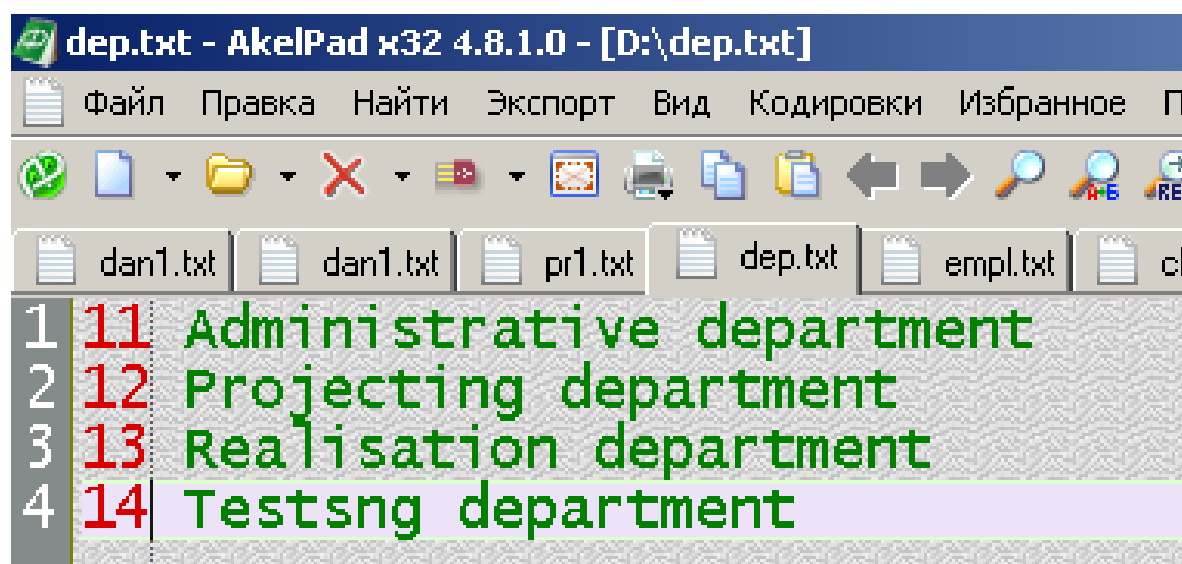
Завантаження даних за допомогою LOAD DATA INFILE

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`.

Таблица **dep**

Вміст текстового файлу з даними таблиці:

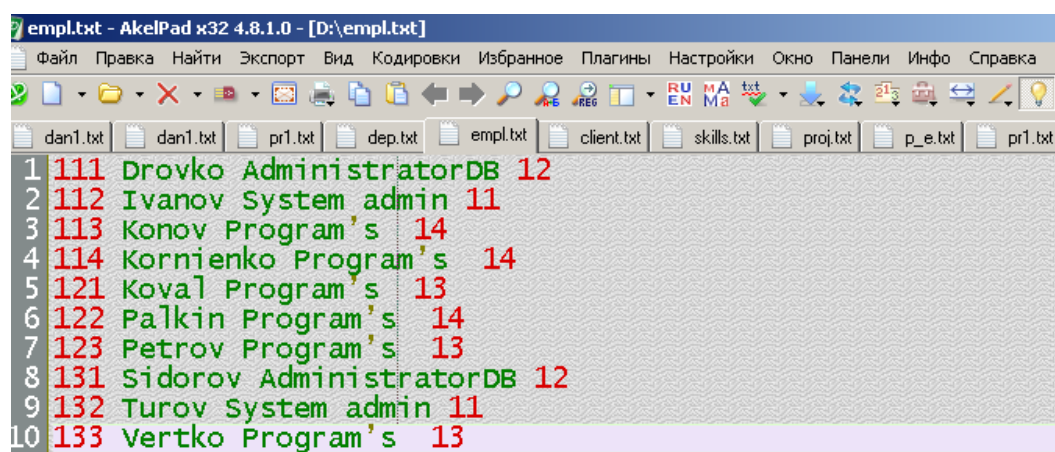


Перегляд вмісту таблиці після занесення в неї даних:



Таблица **empl**

Вміст текстового файлу з даними таблиці:



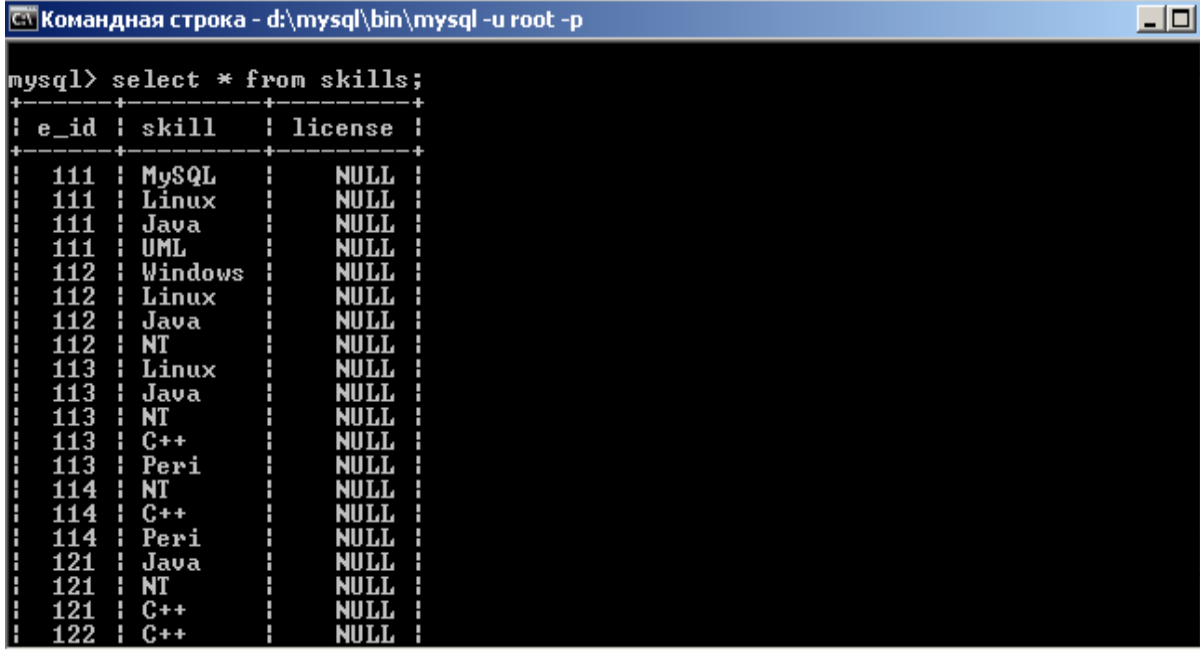
Перегляд вмісту таблиці після занесення в неї даних:

```
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    | ProgramTs      | 14   |
| 114  | Kornienko | ProgramTs      | 14   |
| 121  | Koval    | ProgramTs      | 13   |
| 122  | Palkin   | ProgramTs      | 14   |
| 123  | Petrov   | ProgramTs      | 13   |
| 131  | Sidorov  | AdministratorDB | 12   |
| 132  | Turov    | System admin   | 11   |
| 133  | Uertko   | ProgramTs      | 13   |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

Таблиця skills

Перегляд вмісту таблиці після занесення в неї даних:



```
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>

```

Створення таблиці **client**

Вміст текстового файлу з даними таблиці:

```

1 25 Strogof 10945 Brigge Rd. Oacland 0563657298
2 48 Tvorogenko 578 Bljnde St. Rockville 0456783458
3 77 Dragov 5768 Seventh Av. Gary 0567234852

```

Перегляд вмісту таблиці після занесення в неї даних:

```

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

mysql>

```

Таблиця **projects**

Вміст текстового файлу з даними таблиці:

	pr_id	e_id	cl_id	data_start	data_finish	price
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

Перегляд вмісту таблиці після занесення в неї даних:

```
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>
```

Таблиця proj_emp

Вміст текстового файлу з даними таблиці:

	pr_id	e_id	royalty_share
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

Перегляд вмісту таблиці після занесення в неї даних:

```
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)

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

I. Використовуючи результат виконання лабораторних робіт №№ 1-4, 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. При необхідності, за допомогою оператора `ALTER TABLE`, ввести зміни в структуру створених таблиць.

III. 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`.

IV. Оформити звіт по виконанню лабораторної роботи.

Звіт повинен включати наступні розділи:

1. Опис таблиць бази даних.
2. Для кожної створюваної таблиці - лістинг команди `create table ім'я_таблиці`; і скриншот результату виконання команди `describe ім'я_таблиці`;
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. 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;

5 Щоб видалити базу даних dbname разом з усім її вмістом, варто ввести:

- а. drop all tables on dbname;
- б. drop database dbname;
- в. drop dbname;
- г. delete database dbname;

6 Який з наступних операторів успішно вставить рядок у таблицю employee?

а) insert into employee Values
set employeei=NULL, name='Laura Thomson1, job='Programmer',
departmenti=128;

б) insert employee values
(NULL, 'Laura Thomson1, 'Programmer1, 128) ;

в) insert into employee values
(NULL, Laura Thomson, Programmer, 128);

г) insert employee values
(NULL, 'Laura O'Leary', 'Programmer1, 128) ;

7 Оператор REPLACE

а) 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.

8 За замовчуванням значення полів у файлах даних розділяються

а) комами;

б) пробілами;

в) знаками табуляції;

г) каналами.

9 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;

Методичні рекомендації до виконання лабораторної роботи №№ 7-8

Тема: Прості запити. Використання опцій SELECT, FROM та WHERE.
Використання опцій DISTINCT, ORDER BY, LIKE, BETWEEN, IN.

Мета: Одержати навички використання оператора SELECT для реалізації простих запитів.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

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 бази даних Organisanion.

Занум 1: Виведення всіх записів таблиці empl:

```
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>
```

Занум 2: Відбір певних стовпців таблиці (опція FROM):

```
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)
```

Занум 3: Псевдоніми для стовпців (опція AS):

```
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)
```

Занум 4: Вибір рядків за допомогою опції WHERE.

Виведення списку співробітників - програмістів:

```
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)
```

Занум 5: Видалення повторень за допомогою DISTINCT:

Виведення без повторень списку посад організації:

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

Занум 6: Використання GROUP BY:

Підрахунок числа службовців по групах посад:

```
mysql> select count(*), job from empl
-> group by job;
+-----+-----+
| count(*) | job           |
+-----+-----+
| 2        | AdministratorDB |
| 6        | Programmer     |
| 2        | System admin   |
+-----+-----+
3 rows in set (0.00 sec)
```

Занум 7: Використання HAVING

Виведення наявних у компанії посад, на яких працює рівно по два службовці:

```
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)
```

Занум 8: Сортювання результатів пошуку за допомогою ORDER BY

Виведення списку співробітників, відсортованого за посадою:

```
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  | Vertko   | Programmer     | 13   |
| 123  | Petrov   | Programmer     | 13   |
| 122  | Palkin   | Programmer     | 14   |
| 121  | Koval    | Programmer     | 13   |
| 114  | Kornienko | Programmer     | 14   |
| 113  | Konov    | Programmer     | 14   |
| 132  | Turov    | System admin   | 11   |
| 112  | Ivanov   | System admin   | 11   |
+-----+-----+-----+-----+
10 rows in set (0.01 sec)
```

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.

Заняття 9: Обмеження результатів пошуку за допомогою LIMIT

Виведення перших трьох співробітників організації з відсортованого за абеткою списку всіх співробітників.

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

mysql>
```

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

I. Реалізувати за допомогою оператора SELECT прості запити до таблиць бази даних (згідно Вашого варіанта).

II. Оформити звіт по виконанню лабораторної роботи. Звіт повинен включати наступні розділи для кожного запиту:

1. Формулювання запиту.
2. Виведення вмісту таблиці, для якої сформований запит.
3. Оператор SELECT до відповідного запиту.
4. Копія вікна виконання оператора SELECT.
5. Висновок.

ВАРІАНТИ ЗАВДАНЬ ЛАБОРАТОРНОЇ РОБОТИ

Варіант 1. Предметна область: **Коледж**.

- 1) Одержати список відділень коледжу. Стовпцю додати псевдонім Name_of_department.
- 2) Одержати список всіх співробітників коледжу, відсортований за абеткою , із вказівкою їхньої дати народження.
- 3) Одержати список всіх співробітників коледжу, стаж роботи яких перевищує задане число років. Список відсортувати по зростанню стажу роботи.
- 4) Підрахувати кількість співробітників у кожній предметній комісії.
- 5) Одержати список перших п'яти співробітників коледжу, що мають найбільші ставки.

Варіант 2. Предметна область: **Склад будівельних матеріалів**.

- 1) Одержати список виробників товарів, відсортований за абеткою .
- 2) Одержати список товарів із вказівкою із ціни, відсортована за ціною в порядку зростання.
- 3) Підрахувати кількість виробників, що виготовляють продукцію даного виду.
- 4) Одержати список товарів, кількість яких лежить у заданому діапазоні.
- 5) Одержати список перших чотирьох найдорожчих товарів.

Варіант 3. Предметна область: **Магазин продовольчих товарів**.

- 1) Одержати відсортований за абеткою список виробників із вказівкою їхньої адреси.
- 2) Одержати список товарів, поставлених у магазин до зазначеної дати.
- 3) Підрахувати, кількість товарів, що поставляють кожним виробником.
- 4) Одержати список товарів, ціни яких лежать у зазначеному діапазоні. Список відсортувати по убутуванню ціни.
- 5) Одержати список шести найдешевших товарів.

Варіант 4. Предметна область: **Складальний цех.**

- 1) Одержати відсортований за абеткою список виробів із вказівкою прізвища робітника, що виготовив виріб.
- 2) Одержати список деталей, що виготовляють у кожному цеху.
- 3) Підрахувати кількість деталей, необхідних для виготовлення кожного виробу.
- 4) Вивести список перших трьох виробів, на виготовлення яких потрібно найбільше деталей.
- 5) Одержати список деталей, що виготовляють у конкретному цеху.

Варіант 5. Предметна область: **Дисципліни коледжу.**

- 1) Одержати відсортований за абеткою список викладачів.
- 2) Одержати список дисциплін, по яких здають залік.
- 3) Одержати список викладачів, які викладають у зазначеній групі.
- 4) Для кожного викладача підрахувати кількість дисциплін, які він читає.
- 5) Одержати список перших чотирьох дисциплін з найбільшою кількістю годин.

Варіант 6. Предметна область: **Бібліотека.**

- 1) Одержати відсортований за абеткою список читачів із вказівкою їхнього телефону.
- 2) Одержати список книг, виданих у зазначений період.
- 3) Підрахувати, скільки книг було видано кожним видавництвом.
- 4) Одержати список перших п'яти книг, кількість яких найменше.
- 5) Знайти ФІО читача по зазначеному телефоні.

Варіант 7. Предметна область: **Сесія.**

- 1) Одержати для кожної групи список студентів, відсортований за абеткою .
- 2) Одержати список дисциплін із вказівкою прізвища викладача, по яких були здані іспити.
- 3) Підрахувати, скільки іспитів і заліків приймав кожен викладач.

- 4) Підрахувати для зазначеної групи й дисципліни, скільки студентів здали іспит (або залік) на відмінно, добре, задовільно й незадовільно.
- 5) Вивести список дисциплін, які здавали студенти в зазначений день.

Варіант 8. Предметна область: ***Прокат спортивного встаткування.***

- 1) Одержати відсортований за абеткою список клієнтів із вказівкою їхніх телефонів.
- 2) Одержати список устаткування, ціна прокату якого лежить у зазначеному діапазоні.
- 3) Підрахувати кількість устаткування для кожної країни - виробника.
- 4) Одержати список устаткування із вказівкою виробника й кількості, відсортована по країні за абеткою й по кількості в порядку убутання.

Варіант 9. Предметна область: ***Поставка - продаж непродовольчих товарів.***

- 1) Одержати відсортований за абеткою список постачальників із вказівкою їхніх телефонів.
- 2) Одержати відсортований за абеткою список товарів, згрупованих по постачальниках.
- 3) Підрахувати кількість товарів, проданих у зазначений період.
- 4) Одержати список перших трьох товарів, кількість яких мінімально.

Варіант 10. Предметна область: ***Банківські кредити.***

- 1) Одержати відсортований за абеткою список клієнтів.
- 2) Одержати список кредитів із зазначеною максимальною сумою кредиту.
- 3) Одержати список кредитів, згрупованих по процентній ставці.
- 4) Одержати список кредитів, у яких різниця між максимальною й мінімальною ставками дорівнює заданій величині.
- 5) Знайти телефон заданого клієнта.

Варіант 11. Предметна область: ***Туристична фірма.***

- 1) Одержати відсортований за абеткою список маршрутів, згрупованих по країнах.
- 2) Одержати список посад співробітників із вказівкою кількості співробітників по кожній посаді.
- 3) Одержати список маршрутів у зазначену країну, вартість путівки яких лежить у зазначеному інтервалі.
- 4) Знайти телефон заданого клієнта.
- 5) Одержати список перших трьох маршрутів із заданим видом транспорту, вартість яких мінімальна.

Варіант 12. Предметна область: ***Поліклініка.***

- 1) Одержати відсортований за абеткою список пацієнтів.
- 2) Одержати відсортований за абеткою список лікарів із вказівкою номера кабінету, згрупованих по спеціалізації.
- 3) Одержати список лікарів зазначеної кваліфікації.
- 4) Одержати список посад лікарів поліклініки й підрахувати, скільки лікарів займають кожну з посад.

Варіант 13. Предметна область: ***Абітурієнти коледжу.***

- 1) Одержати відсортований за абеткою список абітурієнтів із вказівкою їхніх телефонів.
- 2) Одержати відсортований по кількості бюджетних місць список спеціальностей коледжу.
- 3) Одержати відсортований за абеткою список абітурієнтів, що закінчили навчальний заклад у заданому році.
- 4) Підрахувати кількість абітурієнтів, що здали задану дисципліну на зазначену оцінку.

Варіант 14. Предметна область: *Ательє по пошиттю одягу.*

Необхідна інформація:

- 1) Одержати відсортований за абеткою список клієнтів із вказівкою їхніх телефонів.
- 2) Одержати список посад із вказівкою кількості майстрів по кожній посаді.
- 3) Одержати відсортований за абеткою список майстрів заданої кваліфікації.
- 4) Указати кваліфікацію заданого майстра.

Варіант 15. Предметна область: *Поставки товарів у мережу магазинів.*

- 1) Для кожного власника одержати список його магазинів.
- 2) Одержати відсортований за абеткою список постачальників й їхніх телефонів.
- 3) Одержати відсортований за абеткою список товарів зазначеної країни - виробника.
- 4) Підрахувати кількість товарів, поставлених у заданий магазин, у зазначений період часу.

Варіант 16. Предметна область: *Транспортні перевезення.*

- 1) Одержати згрупований по марці список автомобілів із вказівкою їхнього державного номера й ФІО водія.
- 2) Одержати список вантажів, заявлених на перевезення в зазначений період часу.
- 3) Одержати список перших трьох автомобілів, у яких витрата бензину мінімальний.
- 4) Підрахувати кількість автомобілів, які виконували заявки на задану дату.

Варіант 17. Предметна область: *Автосалон.*

- 1) Одержати відсортований за абеткою список автомобілів із вказівкою марки, моделі, країни - виробника й ціни за базову комплектацію.

- 2) Одержати відсортований за абеткою список устаткування доукомплектації, згрупований по країнах - виробникам.
- 3) Одержати список перших п'яти найдорожчих автомобілів.
- 4) Одержати список посад з підрахунком кількості співробітників по кожній посаді.
- 5) Одержати телефон заданого клієнта.

Варіант 18. Предметна область: **SPA - салон.**

- 1) Одержати відсортований за вартістю список послуг.
- 2) Одержати відсортований за абеткою список співробітників.
- 3) Одержати відсортований за абеткою список клієнтів, що відвідували салон у зазначений період часу.
- 4) Одержати список перших чотирьох процедур, тривалість виконання яких максимальна.

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=' Programmer';
```

в)

```
select *  
from empl  
where job=' Programmer'  
group by job  
having job=' Programmer' ;
```

г)

```
select job  
from empl;
```

3. Який з наступних запитів не поверне загальне число службовців з таблиці empl?

- а)

```
select count(e_id) from empl;
```
- б)

```
select count(e_id) as total from empl;
```
- в)

```
select count(distinct e_id) from empl;
```
- г)

```
select count(e_id) from empl group by e_id;
```

4. Не допускається використати псевдоніми

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

5. 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
```

Методичні рекомендації до виконання лабораторної роботи № 9

Тема: Багатотабличні запити. Природні, внутрішні й перехресні об'єднання.

Мета: Охарактеризувати використання об'єднань у запитах до декількох таблиць, включаючи.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Розглянемо таблиці бази даних Organization, які будуть використовуватись:

```
mysql> select * from dep;
+-----+-----+
| d_id | d_name      |
+-----+-----+
| 11   | Administrative |
| 12   | Projecting    |
| 13   | Realization   |
| 14   | Testing       |
+-----+-----+
4 rows in set (0.37 sec)
```

```
mysql> select * from empl;
+-----+-----+-----+-----+
| e_id | e_name      | job              | d_id |
+-----+-----+-----+-----+
| 111  | Drovko      | Director         | 11   |
| 112  | Ivanov      | Sysadmin         | 11   |
| 113  | Konov       | Sysadmin         | 11   |
| 121  | Kornienko   | AdministratorDB  | 12   |
| 122  | Koval       | Anaiyst          | 12   |
| 123  | Palkin      | Anaiyst          | 12   |
| 124  | Gromov      | Programer        | 12   |
| 131  | Petrov      | Programer        | 13   |
| 132  | Sidorov     | Programer        | 13   |
| 133  | Tronov      | Programer        | 13   |
| 134  | Todorov     | Coder            | 13   |
| 135  | Abramov     | Coder            | 13   |
| 141  | Stogov      | Programer        | 14   |
| 142  | Vetrov      | Coder            | 14   |
+-----+-----+-----+-----+
14 rows in set (0.07 sec)
```

Запити, розглянуті на попередній лабораторній роботі, повертали дані тільки з однієї таблиці. В умовах нормалізованої бази даних, коли інформація зберігається не в одній, а в цілому наборі таблиць, можливість відблру даних тільки з однієї таблиці, звичайно ж, є занадто вкликим обмеженням. Добре спроектована реляційна база даних стає привабливим об'єктом через існуючих у ній відносин, тобто через зв'язки

між таблицями. При виборі інформації з декількох таблиць такі зв'язки називають об'єднаннями. А тепер розглянемо запити, у яких поєднуються дві таблиці.

Об'єднання двох таблиць

Розглянемо наступний запит:

```
select empl.e_name, dep.d_name  
from empl, dep  
where empl.d_id = dep.d_id;
```

Тут у виразі FROM замість однієї таблиці зазначені дві. У цьому випадку ми хотіли б витягти з бази даних імена співробітників разом з назвами відділів, у яких вони працюють. Результат буде наступним:

```
mysql> select empl.e_name, dep.d_name  
-> from empl, dep  
-> where empl.d_id=dep.d_id;  
+-----+-----+  
| e_name | d_name |  
+-----+-----+  
| Drovko | Administrative |  
| Ivanov | Administrative |  
| Konov | Administrative |  
| Kornienko | Projecting |  
| Koval | Projecting |  
| Palkin | Projecting |  
| Gromov | Projecting |  
| Petrov | Realization |  
| Sidorov | Realization |  
| Tronov | Realization |  
| Todorov | Realization |  
| Abramov | Realization |  
| Stogov | Testing |  
| Vetrov | Testing |  
+-----+-----+  
14 rows in set (0.11 sec)
```

Як вийшов такий результат? Спочатку ми вибрали стовпці, що належать двом різним таблицям. (Як бачите, ми застосували нотацію, що використовує крапку-роздільник, щоб розрізнити поле e_name з таблиці empl і поле d_name з таблиці dep). Для цього треба було включити обидві таблиці у вираження FROM.

Ясно, що вираз WHERE важливий з погляду одержання потрібного нам результату. Набір умов, використовуваних при створенні зв'язку для об'єднання таблиць, називають **умовою об'єднання**. У цьому випадку ми використали умову `empl.d_id = dep.d_id`, що зв'яже таблиці по зовнішніх ключах нашої оригінальної схеми.

```
mysql> select empl.e_name, dep.d_name
-> from empl, dep
-> where dep.d_name='Realization' and dep.d_id=empl.d_id;
```

e_name	d_name
Petrov	Realization
Sidorov	Realization
Tronov	Realization
Todorov	Realization
Abramov	Realization

5 rows in set (0.00 sec)

Коли потрібно витягти інформацію відразу з декількох таблиць, для знаходження цієї інформації доводиться використати наявні між таблицями зв'язки. Іноді це означає необхідність знайти шлях від уже наявної інформації до інформації, потрібної вам.

Об'єднання декількох таблиць

Об'єднання декількох таблиць аналогічно об'єднанню двох таблиць. Розглянемо випадок, коли потрібно з'ясувати, **яким службовцем і з яких відділів було доручено працювати над проектом 'College'**. Як знайти цю інформацію? Ми знаємо назву проекту, і, знайшовши його в таблиці проектів (projects), можна з'ясувати його кодовий номер (pr_id). Це можна використати для того, щоб знайти відповідні рядки в таблиці proj_empl і побачити, які службовці працювали над даним проектом - ми одержимо кодові номери службовців (e_id), а по таблиці службовців (empl) одержати прізвища службовців і з'ясувати номери відділів, у яких ці службовці працюють. Із цією інформацією ми можемо звернутися до таблиці відділів (dep) і знайти назву відповідного відділу!

Маючи план у вигляді зазначеного шляху через чотири таблиці, ми повинні створити запит, що відбиває нашу логіку. Такий запит може виглядати в такий спосіб:

```

select dep .d_name, empl.e_name
from projects , proj_emp, empl, dep
where projects .pr_name='College'
and projects .pr_id = proj_emp .pr_id
and proj_emp .e_id = empl. e_id
and empl. d_id = dep .d_id;

```

Нижче наведений результат виконання цього запиту.

```

mysql> select empl.e_name, dep.d_name
-> from projects, proj_emp, empl, dep
-> where projects.pr_theme='College'
-> and projects.pr_id=proj_emp.pr_id
-> and proj_emp.e_id=empl.e_id
-> and empl.d_id=dep.d_id;
+-----+-----+
| e_name | d_name |
+-----+-----+
| Koval  | Projecting |
| Kornienko | Projecting |
| Petrov | Realization |
| Sidorov | Realization |
| Tronov | Realization |
+-----+-----+
5 rows in set (0.06 sec)

```

Вивчаючи представлений тут запит, ви можете помітити, що нам довелося вказати спочатку всі таблиці спланованого нами шляхи, а потім умови об'єднань, що зв'язують ці таблиці одну з іншою. Як бачите, для зв'язування чотирьох таблиць нам знадобилися три умови об'єднання.

Це спостереження можна використати для перевірки того, що в запиті присутні всі умови, необхідні для об'єднання. Якщо потрібно об'єднати n таблиць, те, як правило, кожна умова буде зв'язувати пари таблиць, тому потрібно вказати $n-1$ умова об'єднання.

1.3 Самооб'єднання таблиць

Аналогічно об'єднанню таблиці з іншою таблицею, можна об'єднати таблицю саму із собою. Але навіщо це робити? Іноді вас можуть цікавити зв'язку між рядками

однієї й тієї ж таблиці. *Припустимо, що ви хочете з'ясувати імена службовців відділу, у якому працює Копов.* Для цього необхідно знайти в таблиці empl номер відділу (d_id), у якому працює Копов, а потім подивитися в таблиці empl, хто ще працює в тім же відділі.

```
mysql> select e2.e_name
-> from empl as e1, empl as e2
-> where e1.e_name='Koval'
-> and e1.d_id=e2.d_id;
+-----+
| e_name |
+-----+
| Kornienko |
| Koval      |
| Palkin     |
| Gromov     |
+-----+
4 rows in set (0.02 sec)
```

Це можна зробити так:

```
select e2.e_name
from empl e1, empl e2
where e1.e_name = 'Konov'
and e1.d_id = e2.d_id;
```

У цьому запиті для таблиці empl ми визначили два різних псевдоніми. По суті ми повідомили системі MySQL, що хотіли б мати дві окремих таблиці, e1 й e2, які повинні містити ті самі дані. Після цього ми поєднуємо їх так само, як будь-які інші таблиці. Тепер спочатку знаходимо рядок 'Konov' у таблиці e1 (e1.e_name='Konov'), а потім у таблиці e2 шукаємо рядка з тим же значенням номера відділу, що й в Konov (e1.d_id = e2.d_id). Спочатку це може здаватися незвичним, але в роботі з декількома таблицями в запитах ідея саме об'єднання таблиць не повинна викликати серйозних труднощів.

Нижче наведені результати попереднього запиту.

```
mysql> select e2.e_name
-> from empl e1, empl e2
-> where e1.e_name='Konov'
-> and e1.d_id=e2.d_id;
+-----+
| e_name |
+-----+
| Konov  |
| Kornienko |
| Palkin  |
+-----+
3 rows in set (0.00 sec)
```

Дуже просто додати умову, що виключить Konov з результуючого списку:

```
select e2.name
from empl e1, empl e2
where e1.e_name = 'Konov'
and e1.d_id = e2.d_id
and e2.e_name != 'Konov';
```

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

I. Реалізувати багатотаюличні запити до таблиць бази даних (згідно Вашого варіанта). При виконанні лабораторної роботи необхідно реалізувати 3 запити:

- 1 Об'єднання двох таблиць
- 2 Об'єднання декількох таблиць (3-4 таблиці)
- 3 Самооб'єднання таблиць

II. Оформити звіт по виконанню лабораторної роботи.

Звіт повинен включати наступні розділи *для кожного запиту*:

1. Формулювання запиту.
2. Виведення вмісту таблиць, для яких сформований запит.
3. Оператор SELECT до відповідного запиту.
4. Копія вікна виконання оператора SELECT.
5. Висновок по результатам виконання лабораторної роботи.

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 рядки декількох таблиць;
 - б) зв'язує рядки в одній таблиці;
 - в) зв'язує два об'єднання;
 - г) зв'язує рядки 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 left JOINn (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.

Запит 5.1:

```
select employee.name  
from employee left JOINn assignment  
on employee.employeeel = assignment.employeeel  
where client is null;
```

Запит 5.2:

```
select e.name, e.employeeel from employee e where not exists  
(select *  
from assignment  
where employeeel = e.employeeel);
```

ВПРАВИ

1. Створіть запит, що повертає ім'я службовця й всю інформацію про його (або неї) кваліфікації (skill).
2. Створіть запит, що використає LEFT JOIN і повертає список клієнтів, для яких не було виділено службовців, що працюють із ними.
3. Перепишіть свій запит із вправи 2, щоб тепер у ньому використалося ключове слово EXISTS.

Методичні рекомендації до виконання лабораторної роботи №№ 10-11

Тема: Складні запити. Створення підзапитів.

Мета: Отримати навички створення підзапитів похідних таблиць, підзапитів з одним значенням, підзапитів в логічних виразах.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

1 Створення підзапитів

Підзапит - це запит усередині іншого запиту, тобто запит, у якому використовуються результати іншого запиту. Іноді підзапити називають вкладеними запитами. Підзапити є новою можливістю в MySQL 4.1. Цієї можливості користувачі чекали давно. Підзапити не додають нових функціональних можливостей, але з ними запити часто виявляються більше зручними для читання, чим зі складними наборами умов об'єднання.

В MySQL були додані два основних види підзапитів:

- підзапити похідних таблиць;
- підзапити-вирази.

Підзапити-вирази застосовуються у вираженні WHERE оператора SELECT, Вони бувають двох типів:

- підзапит, що повертають одне значення або рядок;
- підзапит, використовуваний в логічному вираженні.

Ми розглянемо приклади підзапитів обох зазначених типів.

1.1 Підзапити похідних таблиць

Маємо наступні таблиці:

```
mysql> select * from empl;
```

e_id	e_name	job	d_id
111	Drovko	Director	11
112	Ivanov	Sysadmin	11
113	Konov	Sysadmin	11
121	Kornienko	AdministratorDB	12
122	Koval	Analyst	12
123	Palkin	Analyst	12
124	Gromov	Programer	12
131	Petrov	Programer	13
132	Sidorov	Programer	13
133	Tronov	Programer	13
134	Todorov	Coder	13
135	Abramov	Coder	13
141	Stogov	Programer	14
142	Vetrov	Coder	14

14 rows in set (0.00 sec)

```
mysql> select * from proj_empl;
```

pr_id	e_id	royalty_share
234	122	45.00
234	121	30.15
234	131	10.00
234	132	10.00
234	133	4.75
134	123	30.00
134	112	25.00
134	124	25.00
134	133	20.00
235	123	25.00
235	113	25.00
235	124	15.00
235	135	15.00
235	142	10.00
235	141	10.00
456	123	40.00
456	112	10.00
456	121	10.00
456	132	20.00
456	135	20.00
156	122	40.00
156	124	10.00
156	134	10.00
156	141	10.00
156	142	15.00
156	133	15.00

26 rows in set (0.07 sec)

```
mysql> _
```

Підзапити похідних таблиць дають можливість указати запит у виразі FROM іншого запиту. По суті це дозволяє створити тимчасову таблицю й додати її в запит.

Наприклад, розглянемо наступний простий запит:

```
select empl_id, e_name from empl where job='Programer';
```

Повинне бути очевидним, що цей запит поверне імена й кодові номери всіх програмістів. Можна використати цей запит у рамках іншого, щоб одержати додаткову інформацію:

```
select programmer.e_name
from (select e_id, e_name from empl where job='Programer')
as programmer,
proj_emp
where programmer.e_id = proj_emp.e_id;
```

```
mysql> select e_id, e_name from empl
-> where job='Programer';
+-----+-----+
| e_id | e_name |
+-----+-----+
| 124  | Gromov |
| 131  | Petrov |
| 132  | Sidorov|
| 133  | Tronov |
| 141  | Stogov |
+-----+-----+
5 rows in set (0.38 sec)
```

У цьому випадку ми використаємо підзапит (select e_id, e_name from empl where job='Programer') для створення похідної таблиці, що містить тільки e_id й e_name, що ми призначаємо псевдонім programmer. Після цього до створеної таблиці можна звернутися із запитом, як до будь-якої іншої. У цьому випадку ми використаємо цю таблицю для того, щоб з'ясувати, хто із програмістів працював над виконанням проектів, і одержуємо наступний результат:

```
mysql> select distinct programmer.e_name
-> from (select e_id,e_name from empl where job='Programer')
-> as programmer, proj_emp
-> where programmer.e_id=proj_emp.e_id;
+-----+
| e_name |
+-----+
| Petrov |
| Sidorov|
| Tronov |
| Gromov |
| Stogov |
+-----+
5 rows in set (0.03 sec)
```

Розглянемо запит 1: В проєктах працював співробітник Тіонов і з якою долею участі

```
mysql> select proj_empl.pr_id, proj_empl.royalty_share
-> from empl, proj_empl
-> where e_name='Tronov'
-> and empl.e_id=proj_empl.e_id;
+-----+-----+
| pr_id | royalty_share |
+-----+-----+
| 234   | 4.75          |
| 134   | 20.00         |
| 156   | 15.00         |
+-----+-----+
3 rows in set (0.00 sec)
```

Розглянемо запит 2: В скількох проєктах приймав участь співробітник Тіонов та яке середнє значення долі його участі в цих проєктах:

```
mysql> select count(emp.pr_id), avg(emp.royalty_share)
-> from (select proj_empl.pr_id, proj_empl.royalty_share
->       from empl, proj_empl
->       where e_name='Tronov'
->       and empl.e_id=proj_empl.e_id) as emp;
+-----+-----+
| count(emp.pr_id) | avg(emp.royalty_share) |
+-----+-----+
| 3                | 13.250000              |
+-----+-----+
1 row in set (0.04 sec)

mysql> _
```

1.2 Підзапити з одним значенням

Як й у попередньому розділі, ми починаємо із простого запиту

```
select max (royalty_share) from proj_empl;
```

```
mysql> select max(royalty_share)
-> from proj_empl;
+-----+
| max(royalty_share) |
+-----+
| 45.00              |
+-----+
1 row in set (0.06 sec)
```

Він повертає одне значення, що представляє собою максимальну процентну

ставку, що працівник одержить за виконання завдання по проєкту. Тут використається функція MySQL, що ми не ще згадували, - функція `max ( )`, що повертає максимальне значення відповідного стовпця. Використання результатів, повернутих такими функціями, являє типовий приклад застосування підзапитів з одним значенням.

Можна піти далі й використати даний запит у рамках іншого.

Підзапити з одним значенням повертають одне значення стовпця й звичайно використовуються для порівняння.

Ми намагаємося знайти той, кого можна назвати "ломовим конем" компанії: хто з службовців одержав більше всіх грошей за виконання завдання по проєкті?

```
mysql> select empl.e_name
-> from empl, proj_empl
-> where empl.e_id=proj_empl.e_id
-> and proj_empl.royalty_share=(select max(royalty_share) from proj_empl);
+-----+
| e_name |
+-----+
| Koval  |
+-----+
1 row in set (0.00 sec)
```

Можна також створити підзапит, що повертає не значення, а рядок, але користь від такого підзапита досить обмежена, тому відповідний приклад ми не приводимо.

1.3 Підзапити в логічних виразах

Підзапити в логічних виразах використовуються для перевірки істинності результату деяких спеціальних функцій, що повертають значення типу `BOOLEAN`. Такими спеціальними функціями є `IN`, `EXISTS`, а також `ALL`, `ANY` й `SOME`.

Ключове слово `IN` використовується для перевірки приналежності до деякої множини значень. Розглянемо наступний запит:

```
select e_name from empl
where e_id not in (select e_id from proj_emp);
```

Цей запит дасть той же результат, що й запит, розглянутий вище в прикладі застосування `LEFT JOIN`. Він дозволяє знайти службовців, які не мали завдань при

виконанні проектів. Ключове слово IN дозволяє провести пошук у деякій безлічі значень. Тут ми одержимо те ж, що й від вищенаведеного запиту:

```
mysql> select e_name, job from empl
-> where e_id not in (select e_id from proj_empl);
+-----+-----+
| e_name | job      |
+-----+-----+
| Drovko | Director |
+-----+-----+
1 row in set (0.00 sec)
```

Досить цікавим є наступна можливість використання ключового слова IN для перевірки приналежності до певного набору значень:

```
select e_name from empl
where d_id not in (13, 14);
```

```
mysql> select e_name, d_id from empl
-> where d_id not in (13,14);
+-----+-----+
| e_name | d_id |
+-----+-----+
| Drovko | 11    |
| Ivanov | 11    |
| Konov  | 11    |
| Kornienko | 12   |
| Koval  | 12    |
| Palkin | 12    |
| Gromov | 12    |
+-----+-----+
7 rows in set (0.03 sec)
```

Ключове слово EXISTS працює лише небагато по-іншому в порівнянні зі словом IN. При використанні EXISTS ми насправді використаємо в підзапиті дані деякого зовнішнього запиту. Такий запит іноді називають зв'язаним підзапитом.

Розглянемо наступний приклад: потрібно одержати список службовців, які ніколи не працювали над проектами.

```
mysql> select empl.e_name, empl.job
-> from empl
-> where not exists (select * from proj_empl
->                      where empl.e_id=proj_empl.e_id);
```

e_name	job
Drovko	Director

```
1 row in set (0.00 sec)
```

У підзапиті ми шукаємо рядки таблиці proj_empl, у яких значення e_id збігається зі значенням empl.e_id. Значення e.e_id отримано від зовнішнього запиту. Насправді MySQL тут робить наступне. Для кожного рядка з таблиці empl перевіряються результати підзапиту, і у випадку відсутності відповідності (WHERE NOT EXISTS) інформація про службовця додається в результуючу безліч.

Деякі користувачі знаходять такий синтаксис більше простим для розуміння, але той же результат можна одержати й за допомогою LEFT JOIN, як це було зроблено вище. Використання LEFT JOIN, крім того, буде більше ефективним, і, таким чином, що відповідає запит буде виконаний швидше.

Ключові слова ALL, ANY й SOME використовуються для порівняння з набором значень, повернутих підзапитом.

Отримаємо спочатку долі участі програмістів в розробці проектів:

```
mysql> select proj_empl.royalty_share
-> from proj_empl, empl
-> where proj_empl.e_id=empl.e_id and empl.job='Programer';
```

royalty_share
10.00
10.00
4.75
25.00
20.00
15.00
10.00
20.00
10.00
10.00
15.00

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

А тепер отримаємо список тих співробітників, які працювали над проектами більше всіх програмістів:

```
mysql> select empl.e_name, proj_empl.royalty_share
-> from empl, proj_empl
-> where empl.e_id=proj_empl.e_id and proj_empl.royalty_share > all
-> (select proj_empl.royalty_share
-> from empl, proj_empl
-> where empl.e_id=proj_empl.e_id and empl.job='Programer');
+-----+-----+
| e_name | royalty_share |
+-----+-----+
| Koval  | 45.00         |
| Kornienko | 30.15        |
| Palkin  | 30.00         |
| Palkin  | 40.00         |
| Koval  | 40.00         |
+-----+-----+
5 rows in set (0.00 sec)
```

1.4 Використання агрегатних функцій

Дійсний розділ знайомить із функціями агрегатів, або агрегатними функціями, або функціями, аргументами яких є групи рядків, називані агрегатами, і на виході яких завжди одне-єдине значення, що резюмує. В один агрегат ви, за своїм розсудом, можете включити довільну кількість рядків, що може складатися з:

- всіх рядків довільної таблиці;
- тільки рядків, заданих якоюсь пропозицією WHERE;
- тільки рядків, створених якоюсь пропозицією GROUP BY.

Незалежно від того, скільки саме рядків ви включите у свій агрегат, агрегатна функція видасть для нього тільки одне значення, наприклад статистичне (сума, мінімум або середнє).

Функція	Результат
MIN(expr)	Мінімальне значення в expr
MAX(expr)	Максимальне значення в expr
SUM(expr)	Сума всіх значень в expr
AVG(expr)	Середнє всіх значень в expr
COUNT(expr)	Число всіх значень, що не є значеннями null, в expr
COUNT(*)	Число рядків у довільній таблиці або довільному наборі рядків

Можна також створити підзапит, що повертає не значення, а рядок, але користь від такого підзапита досить обмежена, тому відповідний приклад ми не приводимо.

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

I. Реалізувати підзапити до таблиць бази даних (згідно Вашого варіанта).

При виконанні лабораторної роботи необхідно реалізувати 4 запити:

- 1 Підзапити похідних таблиць
- 2 Підзапити з одним значенням
- 3 Підзапити в логічних вираженнях
- 4 Використання агрегатних функцій

II. Оформити звіт по виконанню лабораторної роботи.

Звіт повинен включати наступні розділи *для кожного запиту*:

1. Формулювання запиту.
2. Виведення вмісту таблиць, для яких сформований запит.
3. Оператор SELECT до відповідного запиту.
4. Копія вікна виконання оператора SELECT.
5. Висновок по результатам виконання лабораторної роботи.

КОНТРОЛЬНІ ПИТАННЯ

- 1 У яких випадках у підсумковому запиті не використовується GROUP BY?
- 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.
- 6 Чим SOME й ANY відрізняється від ALL при використанні у вкладених запитах? Приведіть приклади.

Методичні рекомендації до виконання лабораторної роботи № 12

Тема: Генератори. Тригери. Конструкції мови SQL.

Мета: Отримати навички створення генераторів, тригерів та основних операторів SQL.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Генератор – це механізм, який створює послідовний унікальний номер, який автоматично вставляється в стовпець під час таких операцій, як INSERT або UPDATE. Генератори зазвичай використовуються для створення унікальних значень, які можуть бути вставлені в стовпець, який використовується як первинний ключ.

1. Створення генератора

```
CREATE GENERATOR ім'я_генератора;
```

При створенні генератора за замовчуванням його початкове значення рівнює нулю.

2. Ініціалізація генератора

```
SET GENERATOR ім'я_генератора TO ціле_число;
```

3. Функція GEN_ID

```
GEN_ID (ім'я_генератора , ціле_число);
```

Дана функція збільшує поточне значення генератора на ціле число.

Приклад:

Створити генератор, який використовується для створення унікальних значень поля SNOM, який є первинним ключем таблиці, починаючи з 101.

```
CREATE GENERATOR GEN_STUDENTS;
```

```
SET GENERATOR GEN_STUDENTS TO 100;
```

```
INSERT INTO STUDENTS (SNOM, SFAM)
```

```
VALUES(GEN_ID(GEN_STUDENTS,1), 'ІВАНОВ');
```

Тригери – це частина програмного коду, що асоціюється з таблицею або виглядом, яка автоматично виконує дії при додаванні, зміні або видаленні запису в таблиці.

1. Створення тригера

```
CREATE TRIGGER ім'я_тригера FOR ім'я_таблиці  
[ACTIVE | INACTIVE]  
{BEFORE | AFTER}  
{DELETE | INSERT | UPDATE}  
[POSITION номер]  
AS тіло_тригера  
термінатор
```

[ACTIVE | INACTIVE] – обов'язковий параметр, який визначає дію ера після завершення транзакції. ACTIVE – тригер використовується (по замовчуванню). INACTIVE – тригер не використовується.

{BEFORE | AFTER} – обов'язковий параметр, який визначає коли відбудеться активізація тригера до події (BEFORE) чи після (AFTER).

{DELETE | INSERT | UPDATE} – визначає операцію над таблицею, яка викликає тригер для виконання.

[POSITION номер] – визначає порядок виклику тригера для обробки однієї події, наприклад при додаванні запису в таблицю. Номер має бути цілим від 0 до 32 767, по замовчуванню дорівнює нулю. Тригер, який має менший номер, виконується раніше.

Тіло_тригера – складається з двох частин:

1) блок опису локальних змінних, які використовуються в тригері. Кожна змінна описується конструкцією

```
DECLARE VARIABLE ім'я_змінної тип_змінної ;
```

і закінчується крапкою з комою (;);

2) блок програмного коду, який починається оператором BEGIN і завершується оператором END

```
BEGIN
```

```
команди_InterBase
```

```
END
```

Кожна команда завершується крапкою з комою (;).

У блоці програмного коду тригера використовуються дві контекстні змінні: OLD і NEW. Змінна OLD.ім'я_стовпця відповідає за старі значення стовпця, змінна NEW.ім'я_стовпця – за нові значення.

Оскільки, символ крапка з комою, який використовується зазвичай для завершення SQL-команди, використовується також і всередині тригера, то для завершення команди CREATE TRIGGER слугує певний символ – термінатор. Це може бути будь-який символ, який не використовується в даній команді,

наприклад ^ або !! .

Термінатор визначається перед створенням тригера командою

```
SET TERM термінатор ;
```

Після команди CREATE TRIGGER дію крапки з комою потрібно відновити командою

```
SET TERM ; термінатор
```

Приклад:

Створити генератор GEN_STUDENTS і створити тригер STUDENTS_BI, який використовує даний генератор як лічильник для поля SNOM при додаванні записів у таблицю STUDENTS.

```
CREATE GENERATOR GEN_STUDENTS;
```

```
SET TERM ^ ;
```

```
CREATE TRIGGER STUDENTS_BI FOR STUDENTS
```

```
ACTIVE BEFORE INSERT POSITION 0
```

```
AS
```

```
BEGIN
```

```
IF (NEW.SNOM IS NULL) THEN
```

```
NEW.SNOM = GEN_ID(GEN_STUDENTS,1);
```

```
END ^
```

```
SET TERM ; ^
```

2. Модифікація тригера

```
ALTER TRIGGER ім'я_тригера  
[ACTIVE | INACTIVE]  
[ {BEFORE | AFTER} {DELETE | INSERT | UPDATE} ]  
[POSITION номер]  
[ AS тіло_тригера ]  
[термінатор]
```

При модифікації можна змінити статус тригера, порядок і спосіб його виконання, внести зміни у тіло тригера. Термінатор вказується, якщо модифікується тіло тригера.

Приклад:

Змінити статус активного тригера TR1 на пасивний.

```
ALTER TRIGGER TR1 INACTIVE;
```

3. Видалення тригера

```
DROP TRIGGER ім'я_тригера;
```

Конструкції мови SQL

1. Коментар

```
/*коментар*/
```

2. Конкатенація

```
‘симв_рядок1’||‘симв_рядок2’
```

3. Оператор умови IF ... THEN ... ELSE ...

```
IF (умова)
```

```
THEN оператор_1
```

```
[ELSE оператор_2];
```

умова – логічний вираз, який має значення TRUE або FALSE;

оператор_1 – виконується, якщо умова приймає істинне значення. Замість оператор_1 може бути блок операторів, обмежений конструкцією BEGIN ... END;

оператор_2 – виконується, якщо умова хибна. Замість оператор_2 може бути

блок операторів, обмежений конструкцією BEGIN ... END.

4. Оператор циклу *WHILE ... DO ...*

WHILE (умова) DO оператор;

умова – логічний вираз, значення якого перевіряється перед кожним виконанням циклу;

оператор – оператор чи блок операторів BEGIN ... END, який виконується, якщо умова приймає істинне значення.

5. Оператор *SELECT*

SELECT_команда INTO список_змінних;

Синтаксис даного оператора подібний до синтаксису команди SELECT. На відміну від звичайної команди SELECT результатом даної команди є один рядок, отримані дані записуються у змінні, вказані після ключового слова INTO.

6. Оператор циклу *FOR ... DO ...*

FOR SELECT_команда INTO список_змінних DO оператор;

SELECT_команда отримує дані з бази даних. Дана команда отримує за один раз тільки один рядок, і результат записується у відповідні змінні, вказані після ключового слова INTO. Перед кожною змінною після INTO ставиться двокрапка (:) для того, щоб відрізнити ім'я стовпця від імені змінної;

оператор – оператор чи блок операторів BEGIN ... END, який виконується для кожного рядка, отриманого командою SELECT.

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

- 1) Завантажте програму IBExpert.
- 2) Відкрийте базу даних Univer.
- 3) Створіть генератор GEN1 з початковим значенням, рівним 20.
- 4) Створіть за допомогою тригера лічильник для поля NOM для додавання нових записів у таблицю USPISH.
- 5) Створіть тригер, який перетворює назви предметів таблиці PREDMET у прописні літери при модифікації чи додаванні нових записів у дану таблицю.

- 6) Створити тригер, який при знищенні запису з таблиці STUDENTS буде знищувати пов'язані з ним записи з таблиці USPISH.
- 7) Створити тригер, який при додаванні нових записів до таблиці USPISH, перевіряє чи є студент з відповідним номером в таблиці STUDENTS.
- 8) Створити тригер, який при видаленні записів з таблиці VUKLAD замінює значення зовнішнього ключа відповідних записів таблиці PREDMET на NULL.

КОНТРОЛЬНІ ЗАПИТАННЯ:

- 1) Що таке генератор?
- 2) Якими командами задати створення і використання генератора?
- 3) Де може використовуватись функція GEN_ID?
- 4) Що таке тригер?
- 5) Які є види тригерів?
- 6) Чи може тригер використовуватись як окрема програма?
- 7) До яких дій по відношенню до таблиць приводить виконання тригерів?
- 8) Які оператори використовуються в тригерах?
- 9) Що означає слово POSITION 0 в описовій частині тригера?
- 10) Чи буде виконуватись тригер в описовій частині якого вказане слово INACTIVE?
- 11) Який зміст мають змінні NEW та OLD в тілі тригера?

Методичні рекомендації до виконання лабораторної роботи № 13

Тема: Збережені процедури

Мета: Отримати практичні навички в організації та використання збережених процедур

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Збережені процедури – частини програмного коду, які зберігаються у базі даних і виконуються на стороні сервера. Збережену процедуру можна викликати з клієнтських додатків, тригерів та інших збережених процедур.

Основою могутніх можливостей, які закладені в збережені процедури є не лише модифіковані команди звичайного SQL (SELECT, UPDATE, INSERT) та засоби організації розгалужень та циклів (IF, WHILE, FOR), але й засоби обробки помилок у виключних ситуаціях. Мова збережених процедур дозволяє реалізувати складні процедури роботи з даними та завдяки орієнтованості на роботу з реляційними даними збережені процедури виходять більш компактними ніж процедури на традиційних мовах програмування.

Синтаксис збережених процедур описується наступним чином:

```
CREATE PROCEDURE ім'я_процедури  
[ (вхідний_параметр тип [, вхідний_параметр тип ...] ) ]  
[RETURNS (вихідний_параметр тип [, вихідний_параметр тип...])]  
AS  
тіло_процедури;  
[термінатор]
```

Синтаксис збереженої процедури складається з двох частин: заголовка і тіла. Заголовок включає команду CREATE PROCEDURE, ім'я процедури, список вхідних параметрів і список параметрів, які повертаються з процедури. Дані списки можуть бути відсутні.

Тіло процедури може включати DML-SQL-конструкції, а також, програмні конструкції FOR SELECT...DO, IF-THEN, WHILE...DO та інші.

Існує два види збережених процедур: SELECT і EXECUTE.

SELECT-процедури обов'язково повертають одне або декілька значень і можуть використовуватись при створенні запитів разом з таблицями і представленнями.

EXECUTE-процедури можуть повертати або не повертати значення у викликаючу програму.

SELECT-процедури і EXECUTE-процедури описуються однаково, але викликаються по різному.

SELECT-процедури викликаються за допомогою конструкції:

```
SELECT * FROM ім'я_процедури [(фактичний_параметр [, фактичний_параметр...])];
```

EXECUTE -процедури викликаються за допомогою конструкції:

```
EXECUTE PROCEDURE ім'я_процедури [(фактичний_параметр [, фактичний_параметр...])]
```

Конструкції мови SQL

1. Оператор виходу EXIT

EXIT;

Виконується вихід з циклу і перехід на останній END в процедурі.

Використовується тільки в збережених процедурах.²⁸

2. Оператор SUSPEND

SUSPEND;

Використовується тільки в збережених процедурах в циклах. Призупиняє роботу збереженої процедури до тих пір, поки додаток не відновить запит, повертає значення вихідних параметрів.

Приклад:

Приклад збереженої процедури, який ілюструє використання даних операторів.

```
SET TERM !!;
```

```
CREATE PROCEDURE P RETURNS (R INTEGER)
```

```
AS
```

```
BEGIN
```

```

R = 0;
WHILE (R < 5) DO
BEGIN
R = R + 1;
SUSPEND;
IF (R = 3) THEN
EXIT;
END
END;
Set Term ;!!

```

Після виклику SELECT * FROM P; процедура поверне значення 1, 2 и 3 у ви-
кликаючий додаток.

Приклад 1:

Визначити максимальну стипендію.

```

SET TERM ^ ;
CREATE PROCEDURE PR1 RETURNS (Max_stip DECIMAL (8,2))
AS
BEGIN
SELECT MAX(STIP) FROM STUDENTS
INTO :Max_stip
END

```

^

```
SET TERM ; ^
```

Виклик

```
SELECT * FROM PR1;
```

Приклад 2:

Вивести прізвища студентів, які отримують вказану стипендію.

```

SET TERM ^ ;
CREATE PROCEDURE PR2 (ST DECIMAL (8,2))
RETURNS (Fam VARCHAR(15), Stip DECIMAL (8,2))
AS

```

```

BEGIN
FOR SELECT SFAM, STIP FROM STUDENTS WHERE STIP=:ST
INTO :Fam, Stip
DO
SUSPEND;
END
^
SET TERM ; ^
Виклик
SELECT * FROM PR1;

```

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

- 1) Завантажте програму IBExpert.
- 2) Відкрийте базу даних Univer.
- 3) Створіть процедуру, яка визначає мінімальну стипендію групи.
- 4) Створіть процедуру, що визначає прізвища студентів, які отримують максимальну стипендію в групі.
- 5) Створіть процедуру, яка індексує стипендію на 0,25. (Коефіцієнт індексації може змінюватись).
- 6) Створіть процедуру, яка обчислює сумарну кількість годин, прочитаних кожним викладачем.
- 7) Створіть процедуру, яка нараховує стипендію за результатами сесії.

КОНТРОЛЬНІ ЗАПИТАННЯ:

- 1) Що таке збережені процедури?
- 2) Якою командою створюються збережені процедури?
- 3) Які команди і оператори може включати тіло процедури?
- 4) Які є види збережених процедур?

- 5) Яким чином викликаються збережені процедури?
- 6) Чим відрізняються SELECT-процедури від EXECUTE-процедур?

Методичні рекомендації до виконання лабораторної роботи №№ 14-15

Тема: Резервування й відновлення баз даних. Резервування й відновлення таблиць бази даних

Мета: Навчитися створювати дампи (бекапи) бази даних за різними параметрами з використанням серверного додатку MySQLdump.

ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАД ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Встановлення MySQLdump

MySQLdump – це безкоштовний серверний додаток, який дозволяє робити резервне копіювання (далі дампи) баз даних і зберігати їх в окремому файлі. При цьому можна здійснювати гнучкі налаштування дампу: декілька або всі бази даних, архівація в gzip-архів, додавання команд lock, drop і т.д.. Також можливий зворотний імпорт резервних копій БД. Засоби phpmyadmin дозволяють здійснювати бекап бази даних і призначені для невеликих проектів, які мають малий об'єм даних. Програма mysql_dump зручна при реалізації експорту і імпорту даних з БД, стандартно встановлюється на mysql сервер хостингу, а також на denwer. Додаток mysql_dump дозволяє отримати дамп вмісту бази даних або сукупності баз для створення резервної копії або пересилки даних на будь-який SQL-сервер. Дамп міститиме набір команд SQL для створення і/або заповнення таблиць. Так само mysql_dump має можливість розгортання баз даних із створеного SQL-файла.

Порядок установки на локальний сервер.

1. Скопіювати файл MySQLdump.exe на локальний диск у папку, де встановлено Denwer: C:\WebServers\usr\local\mysql5\bin\ При цьому у вас може бути інша назва папки mysql5: mysql-5.1 чи інша.

2. Запускаємо Denwer іконкою на робочому столі "Start Denwer".

3. Запускаємо консоль: Пуск – Выполнить – cmd.exe чи в операційній системі Windows7: Пуск – Поиск – Вводим cmd.exe – Enter.

4. Перевірити роботу, створивши тестовий дамп:

За допомогою команд в консолі, переходимо на віртуальний диск denwer (W:\) і в папку із додатком MySQLdump. Для підтвердження виконання команди тиснемо Enter.

Вводимо команди:

W: - заходимо на віртуальний диск денвера;

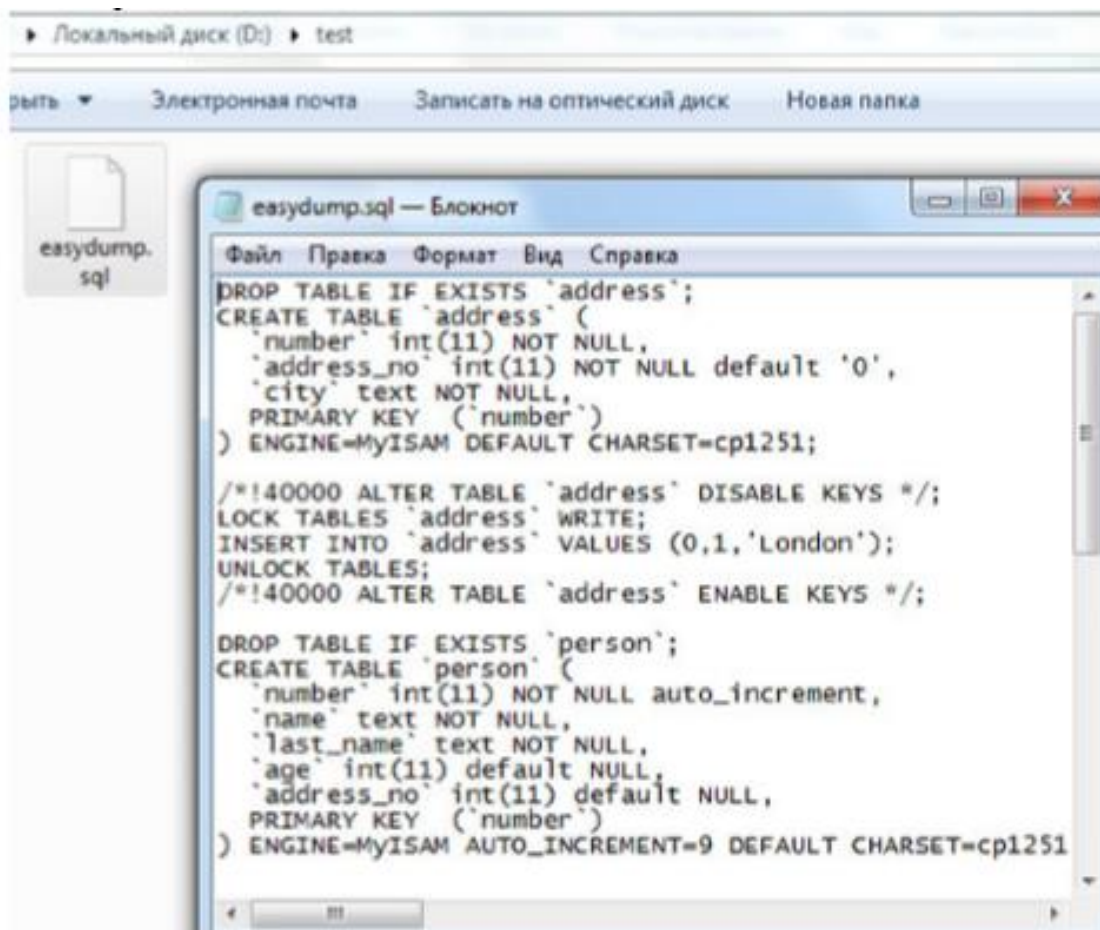
cdusr\local\mysql5\bin – заходимо в папку із додатком;

MySQLdump -uroot ім'я_вашої_бд> ім'я_файлу.sql – тестуємо, робимо дамп довільної бази даних у файл, який збережеться в папку bin.

У папці bin знаходимо файл резервної копії БД.

Експорт та імпорт бази даних

Для виконання простого дампу бази даних в потрібний каталог для створення раніше використовувану БД під ім'ям “test”. Вона знаходиться на локальному сервері Denwer. Нижче надані покрокові команди консолі для дампу БД test в потрібну теку і потрібний файл. W:\cdusr\local\mysql5\bin MySQLdump -uroot test>D:\test\easydump.sql На рисунку показаний дамповий файл в папці бази даних експорт виконаний успішно.



Для імпорту файлу БД назад на сервер, необхідно очистити БД в phpmyadmin, і скористуватися наступною командою в командному рядку cmd.exe:

```
mysql -uroot test <D:\ test \easydump.sql
```

Якщо при експорті було використано MySQLdump, то при імпорті потрібно починати команду з mysql. В даному прикладі показано базове використання застосування MySQLdump, для створення резервних копій (бэкапа) баз даних.

Приклади використання параметрів MySQLdump

Нижче показано найбільш актуальні приклади використання MySQLdump. За допомогою яких можна не тільки робити бэкап, але і додавати деякі параметри резервного копіювання: стиснення за допомогою gzip, додавання дати бэкапа, робити дампи тільки декількох таблиць або структури БД, використовувати гнучкі настройки. Ці параметри дозволяють збільшити швидкість виконання дампу і економно використовувати місце дискового простору.

Найпростіше створення дампу бази даних «database» за допомогою перенаправлення потоку у файл «database.SQL»:

Приклад 1. `mysqldump -uroot -h82.82.82.82 -p database > database.SQL`

Приклад 2. `mysqldump uUSER -h82.82.82.82`

`pPASSWORD database > /path/to/file/dump.sql` де:

- `-u` або `--user=...` – ім'я користувача;
- `-h` або `--host=...` – видалений хост (для локального хосту цей параметр можна опустити);
- `-p` або `--password` – запитати пароль;
- `database` – ім'я бази даних, що експортується;
- `database.SQL` – файл для дампу (`/path/to/file/dump.SQL` – шлях і файл для дампу).

Для того, щоб зробити дамп декількох баз даних, необхідно використати параметр `--databases` (чи скорочено `-B`), приклад:

`mysqldump -uroot -h82.82.82.82 -p -B database1 database2 database3 > databases.SQL`

Для того, щоб зробити дамп всіх баз даних, що існують на сервері, необхідно використати параметр `-all-databases` (або скорочено `-A`), наприклад:

`mysqldump -uroot -h82.82.82.82 -p -A > all-databases.SQL`

Для розгортання дампу перенаправляємо потік у зворотний бік і розгортаємо базу даних:

`mysql -uroot -h82.82.82.82 -p database < database.SQL`

Через `mysql-console` створення дампу буде виглядати так:

`mysql> use database; mysql> source database.SQL`

Для даних з існуючої версії фрагмент бази для розробника, потрібно БД для тестування, наприклад не більше 100 записів:

`mysqldump -uroot -h82.82.82.82 -p --where="true limit 100" database > database.SQL`

Створення дампу *тільки структури*, без даних необхідно використати параметр `--no-data` як показано на прикладі нижче:

```
mysqldump -uroot -h82.82.82.82 -p --no-data database > database.SQL
```

Створення дампу *тільки тригерів, процедур і подій*:

```
mysqldump --no-create-info --no-data --triggers --routines --events -uroot -p database | gzip > ~/database.SQL.gz
```

Для створення дампу із вказуванням дати створення файлу в імені файлу:

```
mysqldump -uUSER
```

```
pPASSWORD database | gzip > `date +dump.sql.%Y%m%d.%H%M%S.gz`
```

Додаткові атрибути зменшують підсумковий розмір файлу і прискорюють процес резервного копіювання.

```
MySQLdump -Q -c -i -uUSER
```

```
pPASSWORD DATABASE > /path/to/file/dump.sql
```

де:

- `-Q` – обертає імена зворотними лапками;
- `-c` – робить повну вставку, включаючи імена колонок;
- `-i` – робить розширену вставку.

Для створення дампу і одночасного його архівування в `gzip`-архів:

```
mysqldump -u USER -pPASSWORD DATABASE | gzip > /path/to/outputfile.sql.gz
```

При розгортанні дампу бази даних із `gzip`-архіву, то:

```
zcat database.SQL.gz | mysql -uroot -h82.82.82.82 -p database
```

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

- I. Встановити серверний додаток `MySQLdump` для резервування даних.
- II. Виконати імпорт та експорт однієї з таблиць з `MSAccess`.
- III. Створити резервну копію бази даних `MySQL` на сервері із використанням `MySQLdump` за різними параметрами.
- IV. Виконати відновлення бази даних з створених дамтів.
- V. Оформити звіт з лабораторної роботи

КОНТРОЛЬНІ ПИТАННЯ

- 1 Що таке дамп (бэкап) бази даних?
- 2 Які головні параметри використовуються при створенні дампу БД?
- 3 Назвіть додаткові параметри, що використовуються при створенні дампу БД.
- 4 Як виконати розгортання дампу?
- 5 Як виконати експорт і імпорт таблиць бази даних ?

Методичні рекомендації до виконання лабораторної роботи № 16

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.

ТЕОРЕТИЧНІ ВІДОМОСТІ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

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:

База даних: mysql 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	

База даних: mysql 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	

База даних: mysql 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 запущений 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. Принципами 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 (Logini), 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 (Useri), 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 зворотнє: 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.

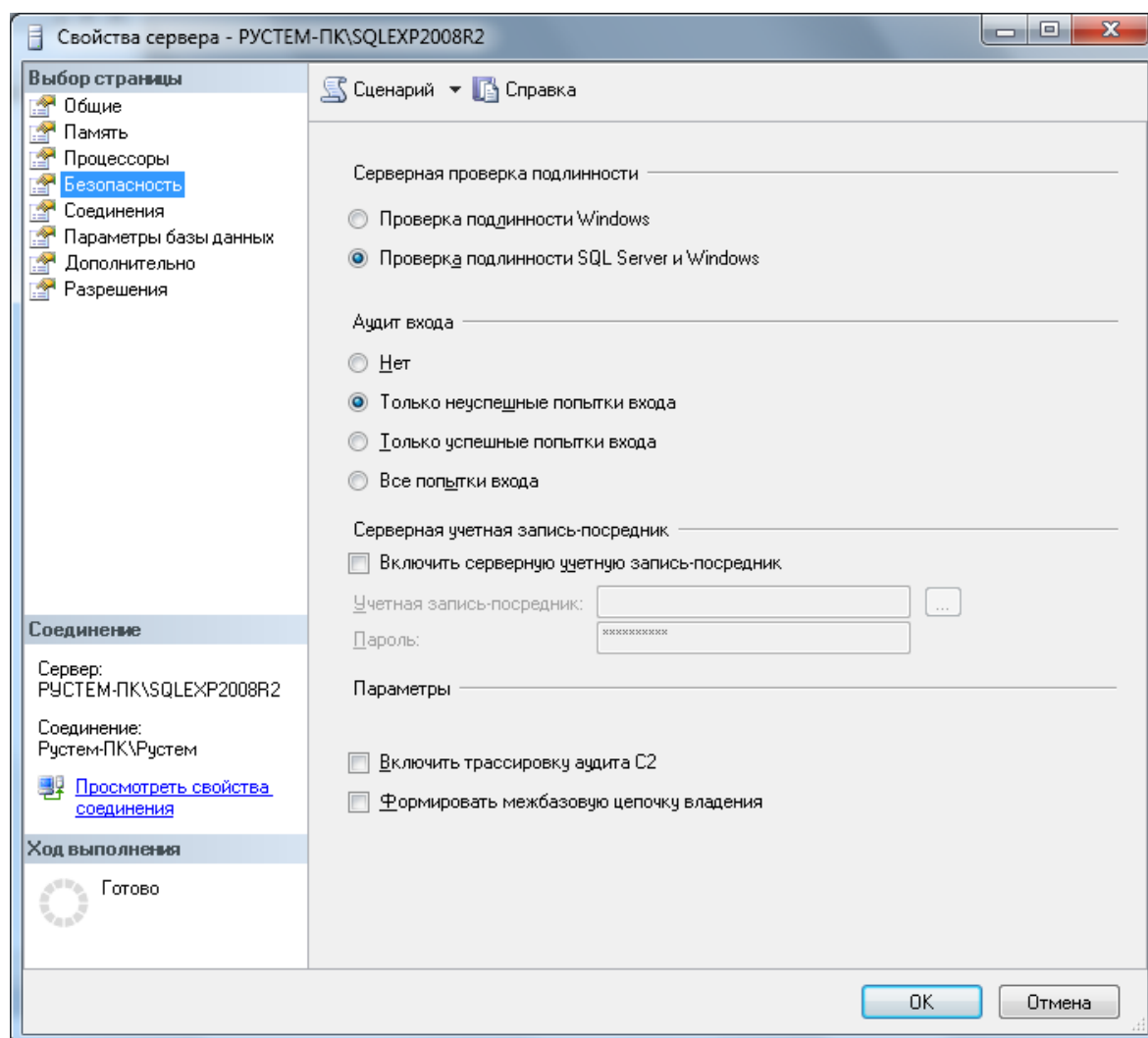
При установці SQL Server одним з рішень, які варто прийняти, є вибір використовуваного режиму аутентифікації.

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".

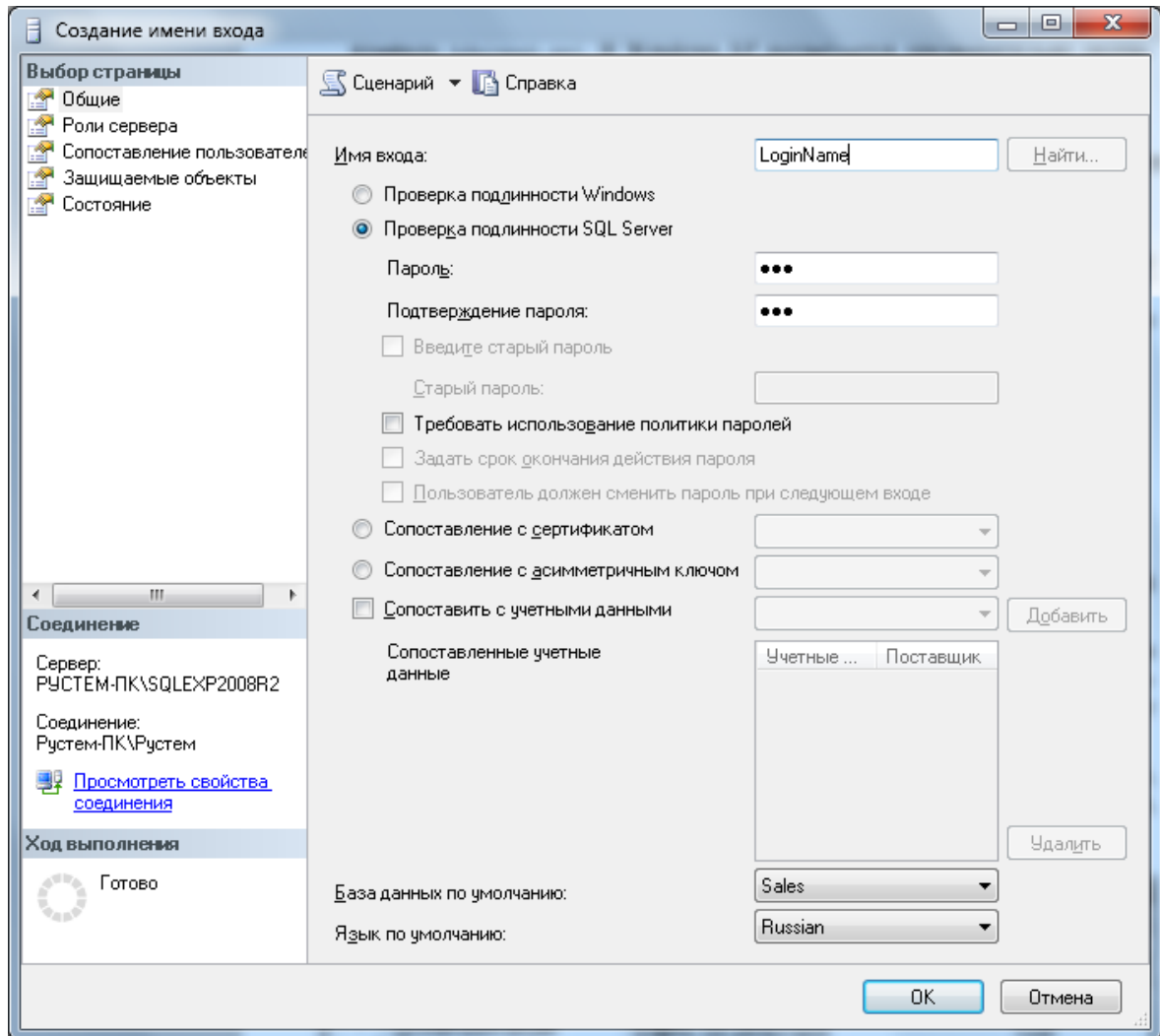


Установите перемикач у положения "Перевірка дійсності SQL Server й Windows". Таким чином, ви включите змішаний режим аутентифікації, у якому й будуть виконуватися інші вправи. Для того щоб зміна режиму аутентифікації набуло чинності сервер потрібно запустити знову.

Створення логіна й настроювання 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;



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 використається 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.

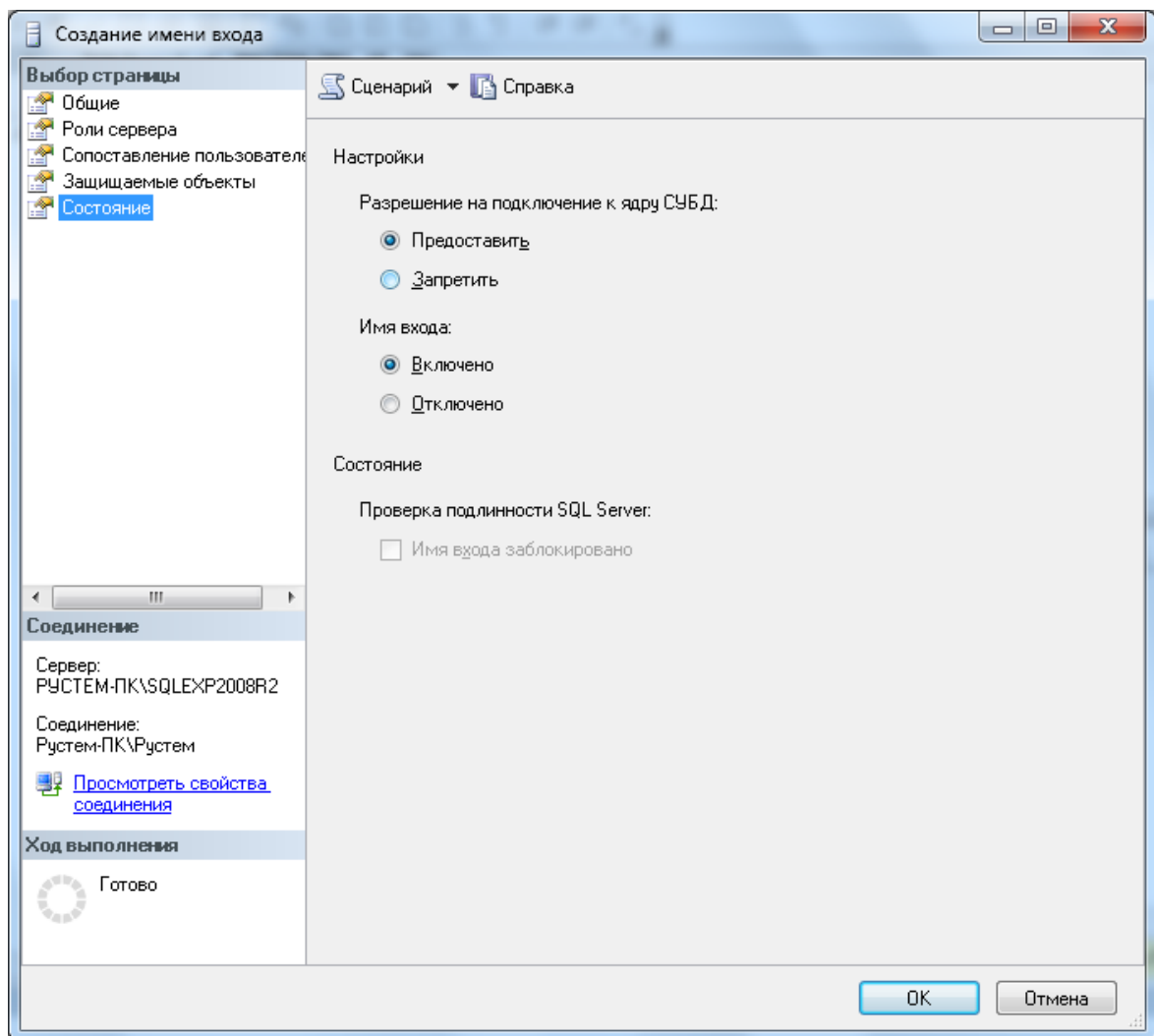
На вкладці "Стан" властивостей логіну можна настроїти для цього логіну додаткові параметри:

Дозвіл на підключення до ядра СУБД - за замовчуванням для всіх логінів установлюється значення "Надати", тобто підключатися до SQL Server дозволено. Значення "Заборонити", як правило, використається 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

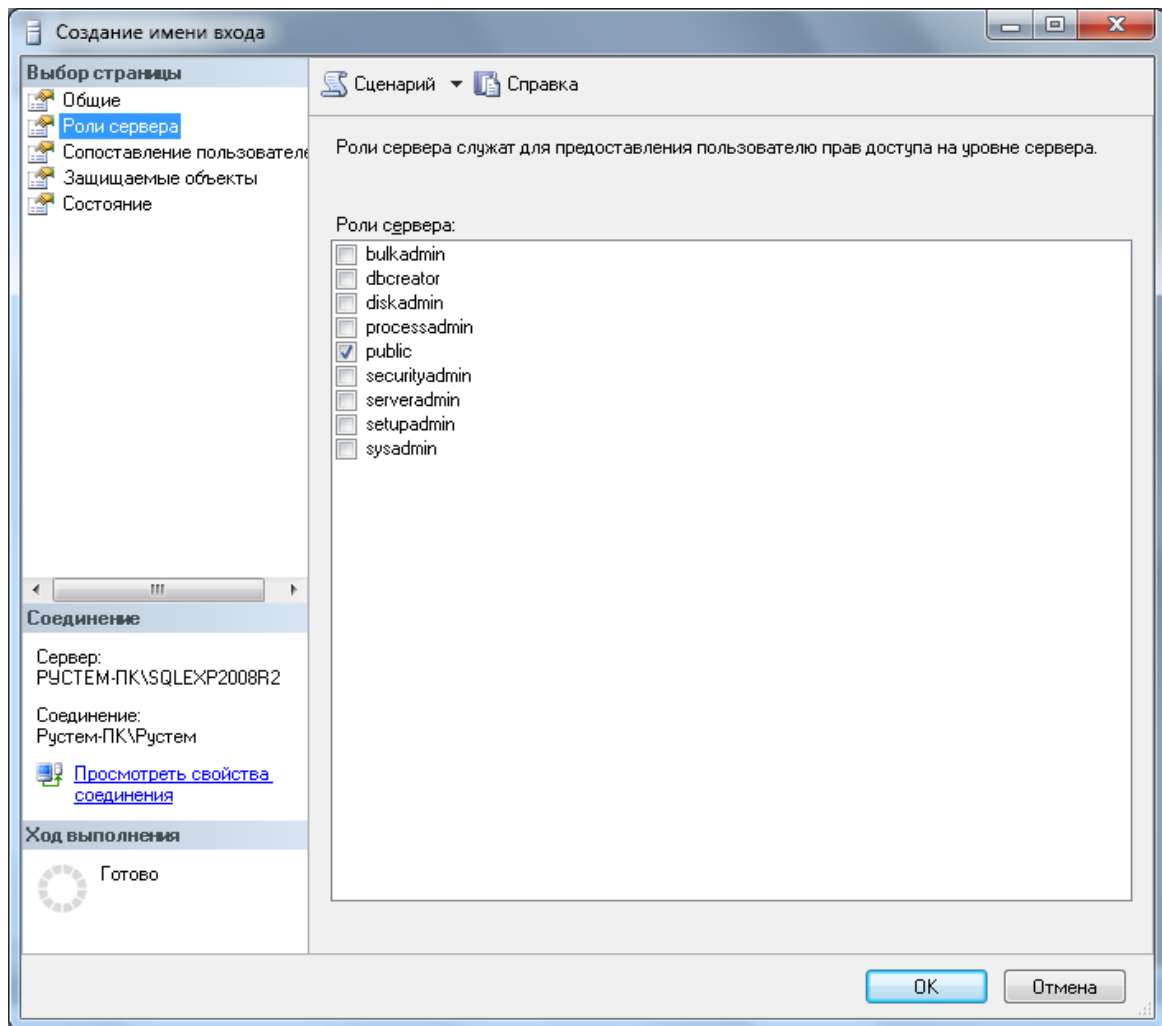
на з операційної системи, а для логіну встановлений прапорець "Вимагати використання політики паролів".



Дозволи на рівні сервера. Фіксовані серверні ролі

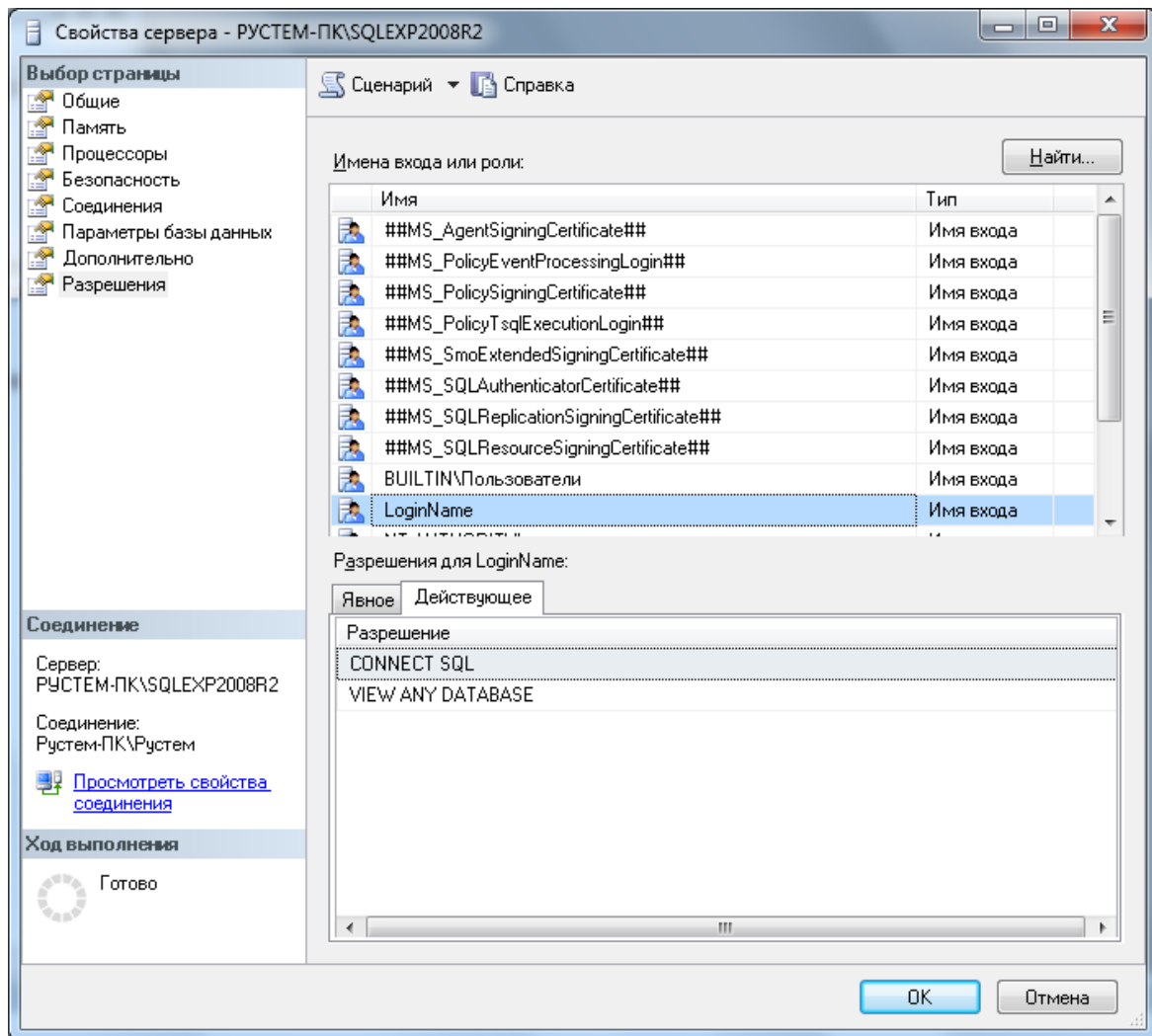
Створивши логіни, ви забезпечуєте користувачам можливість входу на SQL Server. Але сам по собі вхід на сервер нічого не дає: користувачеві потрібні також права на виконання певних дій. Звичайно для цієї мети створюються користувачі або ролі баз даних й їм надаються дозволи (як це зробити, буде розглянуто 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).



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);
- серверні ролі використовуються тільки в спеціальних випадках. Для більшості користувачів набудовувати їх не потрібно.



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

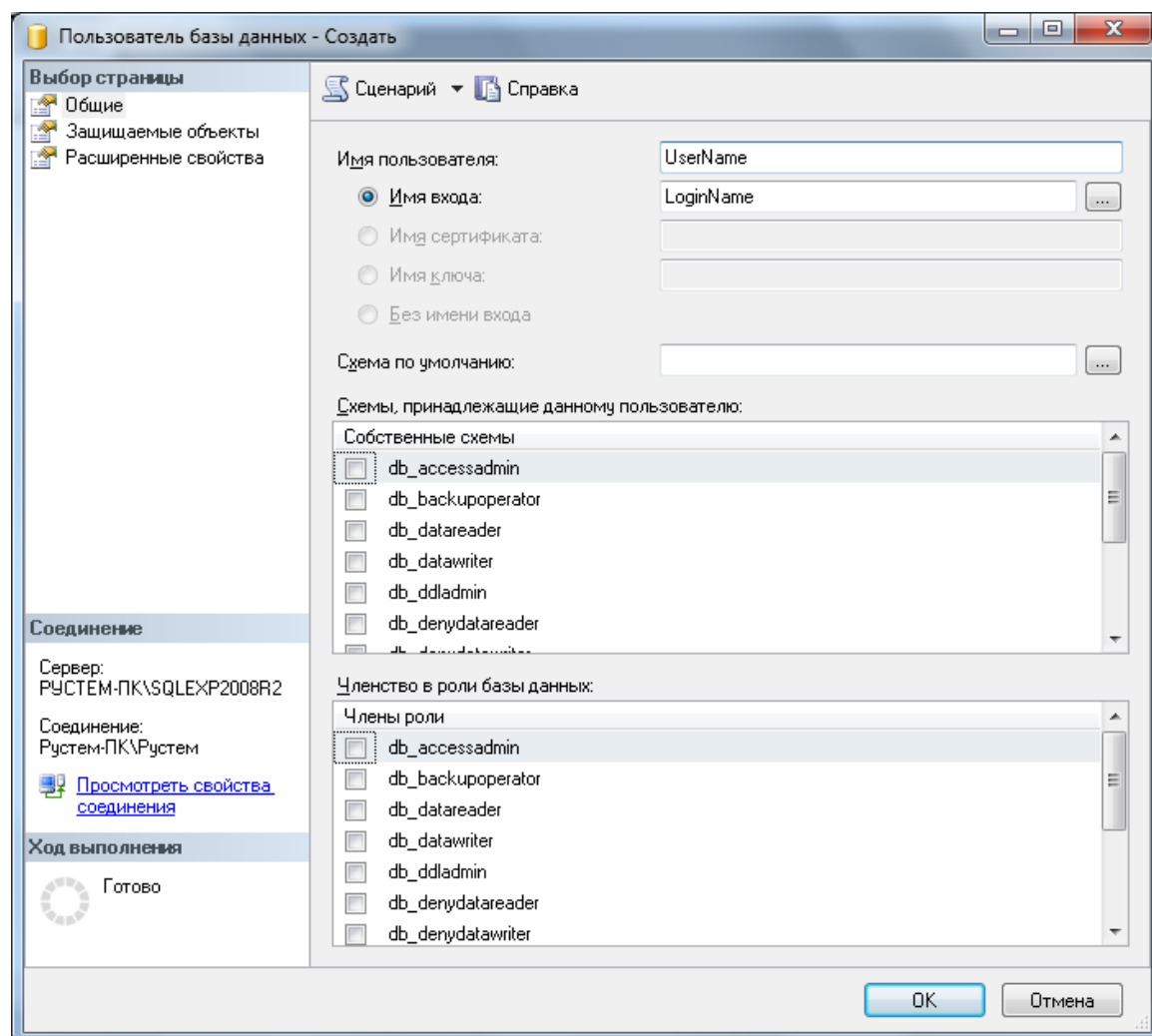
Після створення логінів наступне завдання - спуститися на рівень бази даних і створити користувачів бази даних. Користувачі баз даних - це спеціальні об'єкти, які створюються на рівні бази даних і використовуються для надання дозволів у базі даних (на таблиці, подання, збережені процедури й т.д.).

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).

За допомогою команди CREATE USER.



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;

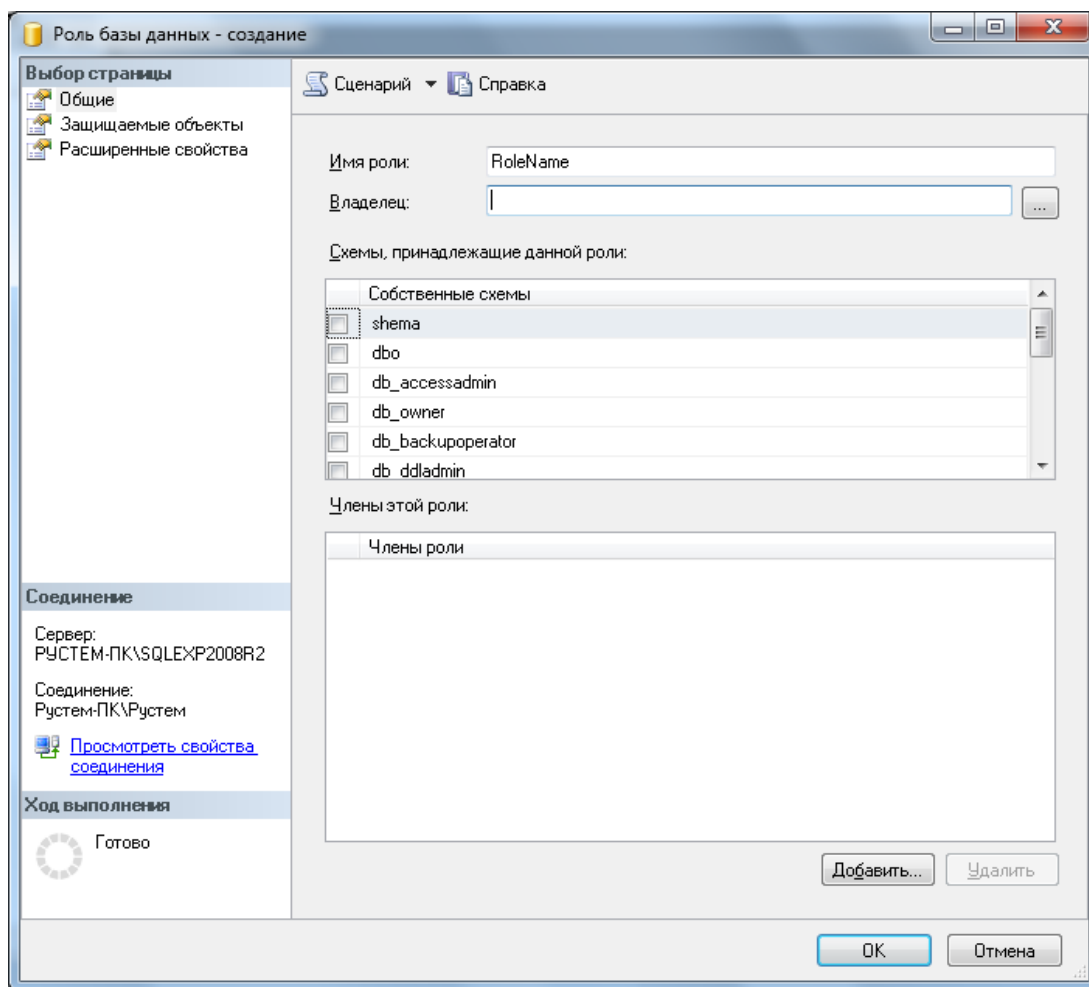
`db_backupoperator` - ця роль надає право виконувати резервне копіювання бази даних;

`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 або за допомогою команди CREATE ROLE.



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.

Вбудованим 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. Властник 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 сотня 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.

ЗМІСТ І ПОСЛІДОВНІСТЬ ВИКОНАННЯ ЗАВДАНЬ

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

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?

ТЕСТОВІ ПИТАННЯ

- 1 Привілей GRANT OPTION дозволяє користувачеві
 - а) завантажувати дані з файлу;
 - б) передавати свої привілеї іншим користувачам;
 - в) зареєструватися в системі;
 - г) оновляти привілею.
- 2 Привілей USAGE дозволяє користувачеві
 - а) завантажувати дані з файлу;
 - б) передавати свої привілеї іншим користувачам;
 - в) зареєструватися в системі;
 - г) оновляти привілею.
- 3 Привілей RELOAD дозволяє користувачеві
 - а) завантажувати дані з файлу;
 - б) передавати свої привілеї іншим користувачам;
 - в) зареєструватися в системі;
 - г) оновляти привілею.

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