

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД  
УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ  
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

### **ОРГАНІЗАЦІЯ БАЗ ДАНИХ**

Логічне проектування та робота з віддаленими базами даних

Методичні вказівки і завдання

до лабораторних робіт

для студентів 2-го курсу інженерно-технічного факультету спеціальності

123 – «Комп'ютерна інженерія».

**Ужгород – 2021**

## ЗМІСТ

### Вступ

Лабораторна робота №1. Тема: Побудова концептуальної моделі даних	4
Лабораторна робота №2. Тема: Побудова моделі даних «сутність-зв'язок».	8
Лабораторна робота №3. Тема: Проектування баз даних засобами <i>ERwin</i> .	26
Лабораторна робота №4. Тема: Система управління базами даних <i>MySQL</i> . Проектування бази даних і забезпечення прав доступу до неї.	56
Лабораторна робота №5. Тема: Використання ключів та індексів. Об'єднання таблиць.	81
Лабораторна робота №6. Тема: Знайомство з роботою програми <i>MySQL-Front</i> ( <i>HeidiSQL</i> ).	90
Список використаних джерел	100

## ВСТУП

Трудова діяльність людини постійно пов'язана зі сприйняттям і накопиченням інформації про навколишнє середовище, відбором та обробкою інформації при розв'язанні різних задач, обміном нею з іншими людьми. З часом комплекс цих операцій, методи і засоби їхньої реалізації послужили основою для створення інформаційних систем, основне призначення яких – інформаційне забезпечення користувача, тобто надання йому необхідних даних із визначеної предметної області. Завдяки появі електронної обчислювальної машини стало можливим створення автоматизованих інформаційних систем (АІС).

У розвитку АІС виділяють два покоління:

- перше покоління – інформаційні системи, які базуються на автономних файлах. Це системи з простою архітектурою й обмеженим набором можливостей. Вони складаються із набору автономних файлів і комплексу прикладних програм, призначених для обробки цих файлів і видачі документів. Такі системи мають ряд серйозних недоліків, що обмежують їхнє широке застосування: високу надлишковість даних, складність ведення і спільної обробки файлів, залежність програм від даних;

- друге покоління – банки даних. Це системи з високим ступенем інтеграції даних і автоматизації керування ними. Вони орієнтовані на колективне користування й в основному позбавлені недоліків, властивих АІС першого покоління.

Банки даних містять наступні складові:

1. Обчислювальну систему.
2. Систему управління базами даних (СУБД).
3. Одну або декілька баз даних (БД).
4. Набір прикладних програм (додатків БД).

Бази даних забезпечують зберігання інформації, а також зручний та швидкий доступ до даних. Це сукупність даних різного характеру, які організовані за певними правилами.

СУБД являє собою сукупність мовних та програмних засобів, які призначені для побудови, ведення та використання БД. Це програмне забезпечення для ефективного, зручного і безпечного збереження даних у БД, організації пошуку в ній та виведення даних на вимогу користувачів.

Дані методичні вказівки містять лабораторні роботи, які пропонуються для виконання при вивченні курсу «Організація баз даних».

Мета даної розробки – навчити студентів основам проектування реляційних баз даних (від англ. *relation* – зв’язок) та особливостям їх експлуатації, використовуючи персональні та багатокористувацькі СУБД; навчити їх мові визначення і маніпулювання даними, які знаходяться в БД з урахуванням сучасного стану та прогнозу розвитку інформаційних систем.

Методичні вказівки містять шість лабораторних робіт, які охоплюють основні питання курсу «Організація баз даних». Перші три роботи присвячені логічному проектуванню баз даних, решта – основам роботи з системою керування базами даних *MySQL*.

В роботах розглянуті питання логічного проектування баз даних, побудови моделі даних «сутність-зв’язок» засобами графічної мови *ERWin*, створення бази даних за допомогою програми *MySQL-Front*. Також розглядаються основи роботи з системою керування базами даних *MySQL*: операції з таблицями, встановлення прав доступу, привілеїв, створення ключів та індексів, виконання транзакцій.

У кожній роботі наведено достатню кількість теоретичних відомостей, які дозволяють студенту повністю виконати поставлені завдання.

## Лабораторна робота №1

**Тема:** Побудова концептуальної моделі даних.

**Мета роботи:** проаналізувати обрану предметну область, визначити основні сутності та атрибути сутностей

### Теоретичні відомості

На концептуальному рівні здійснюється інтегрований опис предметної області, для якої розробляється база даних, незалежно від її сприйняття окремими користувачами та способів реалізації в комп'ютерній системі. Дано означення основних понять, що використовуються на концептуальному рівні.

**Предметна область (ПО)** — частина реального світу, для якої здійснюється концептуальне моделювання.

**Концептуальна модель ПО** — формальне зображення сукупності думок, які характеризують можливі стани ПО, а також переходи з одного стану в інший (включно з класифікацією наявних у ПО сутностей, чинних правил, законів, обмежень тощо).

**Концептуальне моделювання ПО** — процес побудови концептуальної моделі ПО, яка б відображала ПО з урахуванням вимог, висунутих до цього процесу.

**Концептуальна схема** — фіксація концептуальної моделі ПО засобами конкретних мов моделей даних. У системах управління базами даних (СУБД) концептуальна модель подається у вигляді концептуальної схеми.

Опишемо властивості концептуальної моделі (схеми) й характерні особливості концептуального моделювання:

1. Спільне та однозначне тлумачення предметної області всіма зацікавленими особами. До розробки складної бази даних залучається великий колектив: експерти, системні аналітики, проектувальники, розробники, ті, хто займається впровадженням і супроводом. Усі вони повинні однозначно розуміти, чим є ПО, в чому зміст використаних понять, як вони взаємопов'язані між собою, які обмеження висуваються до моделі ПО тощо. Спільність понять має забезпечувати концептуальна модель.

2. Концептуальна схема відображує лише концептуально важливі аспекти ПО, виключаючи будь-які аспекти зовнішнього або внутрішнього відображення даних. Ця модель не повинна відображувати конкретні потреби окремих користувачів або застосувань. Вона має фіксувати, чим є ПО в цілому, а не з точки зору інтересів або потреб користувачів. Для отримання цілісного уявлення про ПО її модель має інтегрувати думки, погляди та інтереси окремих користувачів, але саме інтегрувати, а не виражати їхні конкретні побажання.

3. Визначення допустимих меж еволюції бази даних. У процесі експлуатації база даних може розвиватися, проте цей розвиток може відбуватися тільки в межах, допустимих для концептуальної схеми.

4. Відображення зовнішніх схем на внутрішню. Саме через концептуальну схему зовнішні дані відображуються на внутрішні, й навпаки. У такий спосіб створюється єдина основа для опису даних і підтримки цих відображень.

5. Забезпечення незалежності даних. Наявність відображень «концептуальний-зовнішній» і «концептуальний-внутрішній» дає змогу вирішувати проблему логічної та фізичної незалежності даних. Будь-які зміни в тій чи іншій зовнішній моделі не повинні спричиняти зміни в концептуальній або внутрішній моделях. У цьому випадку має змінитися тільки відповідне відображення «концептуальний-зовнішній». Аналогічно, будь-які зміни у внутрішній моделі не зачіпають концептуальну модель і моделі зовнішнього рівня, а тільки приводять до змін відображення «концептуальний-внутрішній».

6. Централізоване адміністрування. Саме через концептуальну схему здійснюється адміністрування баз даних.

7. Стійкість. Концептуальна схема не має підлаштовуватися до вимог тих чи інших користувачів (зовнішній рівень) або до вимог зберігання даних (внутрішній рівень). Будучи моделлю ПО, вона має змінюватися тільки тоді, коли входить у суперечність із нею.

Існує багато мов, які претендують на роль мов концептуального моделювання ПО. Найпопулярнішими і широкоживаними є мови, що належать до класу так званих графічних мов, які оперують поняттями «сутність-зв'язок» (*Entity-Relationship language*).

### **Порядок виконання роботи**

Згідно індивідуального завдання, виконати наступні дії:

1. Обрати предметну область, в якій виникає необхідність створення бази даних.
2. Ознайомитись з концептуальним рівнем моделювання предметної області.
3. Визначити основні сутності предметної області.
4. Визначити атрибути сутностей.
5. Структурувати інформацію про сутності та відповідні їм атрибути.
6. Зробити висновки, що до отриманих результатів, визначити основні етапи і принципи отримання інформації, охарактеризувати предметну область в термінах сутностей та атрибутів.
7. Оформити короткий звіт про виконану роботу, внести в нього основні теоретичні відомості.

### **Оформлення звіту:**

1. Титульний лист.
2. Мета роботи.
3. Порядок виконання роботи.
4. Таблиця сутностей та атрибутів.
5. Висновки.

### **Варіанти індивідуальних завдань**

**Варіант 1.** Спроекувати структуру бази даних про студентів, для їхнього розподілу по місцях практики: прізвище, рік народження, група, факультет, середній бал, місце роботи, місто.

**Варіант 2.** Спроекувати структуру бази даних про автомобілі: номер, рік випуску, марка, кольори, стан, прізвище власника, адреса.

**Варіант 3.** Спроекувати структуру бази даних про квартири, призначені для продажу: район, поверх, площа, кількість кімнат, відомості про власника, ціна.

**Варіант 4.** Спроекувати структуру бази даних про книги, куплені бібліотекою: назва, автор, рік видання, адреса автора, адреса видавництва, ціна, книготорговельна фірма.

**Варіант 5.** Спроекувати структуру бази даних про співробітників, що мають комп'ютер: прізвище, номер кімнати, назва відділу, дані про комп'ютери.

**Варіант 6.** Спроекувати структуру бази даних про замовлення, отриманих співробітниками фірми: прізвище, сума замовлення, найменування товару, назва фірми - клієнта, прізвище замовника.

**Варіант 7.** Спроекувати структуру бази даних про оцінки, отримані студентами на іспитах: прізвище, група, предмет, номер квитка, оцінка, викладач.

**Варіант 8.** Спроекувати структуру бази даних про викладачів кафедри: прізвище, посада, ступінь, номер кімнати, курси, що читають.

**Варіант 9.** Спроекувати структуру бази даних про авторів web-додатків та їхніх статей: ім'я, адреса, обліковий запис, пароль, тема, заголовок, текст статті, ілюстрації.

**Варіант 10.** Спроекувати структуру бази даних про список розсилок та передплатників: тема й зміст листа, дата відправлення, імена й адреси передплатників, їхні облікові записи й паролі.

**Варіант 11.** Спроекувати структуру бази даних «Мережа магазинів комп'ютерної техніки», яка містить інформацію про наступні об'єкти: магазини ( назва, адреса), наявність товару, кількість, ціна, продавці (ППП, адреса, дата народження, номер паспорту, магазин, відділ).

**Варіант 12.** Спроекувати структуру бази даних про працівників поліклініки: ППП, спеціальність, розклад прийому (день тижня, початок прийому, кінець прийому, кабінет, ділянка).



**Варіант 13.** Спроекувати структуру бази даних «Картотека бібліотеки», яка містить наступну інформацію: картка книги, видавництво, місто, тема книги, дисципліна, список вибірки книги по дисципліні, позиція вибірки.

**Варіант 14.** Спроекувати структуру бази даних для гуртожитку, яка містить наступну інформацію: корпус, кімнати, студенти, які там проживають, оплата, умови.

**Варіант 15.** Спроекувати структуру бази даних про відвідування студентами занять. База даних містить наступну інформацію: кафедра, група, студент, дисципліна, викладач, заняття, відвідування заняття.

### **Контрольні питання**

1. Моделі даних.
2. Архітектурні рівні бази даних.
3. Концептуальний рівень.
4. Поняття сутностей та атрибутів.
5. Основні поняття концептуального рівня.
6. Концептуальна модель даних.

## Лабораторна робота № 2

**Тема:** Побудова моделі даних «сутність-зв'язок».

**Мета роботи:** побудувати модель даних «сутність-зв'язок» та реалізувати спроектовану модель у вигляді **ER**-діаграми.

### Теоретичні відомості

**Інфологічний етап проектування баз даних.** Мета інфологічного етапу проектування полягає в одержанні семантичних (концептуальних) моделей, що відображають предметну область і інформаційні потреби користувачів, поняття про предмети, факти і події, якими буде оперувати дана інформаційна система. Як інструмент для побудови семантичних моделей даних та уніфікованого представлення даних, незалежного від програмного забезпечення, яке його реалізує, на етапі інфологічного проектування застосовується неформальна модель «сутність-зв'язок» (*entity-relationship model*, **ER-model**), яка була запропонована в 1976 р. Питером Пин-Шэн Ченом. Модель «сутність-зв'язок» ґрунтується на опорній семантичній інформації про реальний світ і призначена для логічного представлення даних у контексті їхнього взаємозв'язку з іншими даними. З моделі «сутність-зв'язок» можуть бути породжені всі існуючі моделі даних (ієрархічна, мережна, реляційна, об'єктно-орієнтовна), тому вона є найбільш загальною.

Моделювання предметної області базується на використанні графічних діаграм, що включають невелике число різномірних компонентів. Модель «сутність-зв'язок» не визначає операцій над даними й обмежується описом тільки їхньої логічної структури.

**Основні елементи інфологічних моделей.** Основними поняттями **ER**-моделі є сутність, зв'язок і атрибут. При інфологічному проектуванні бази даних довільний фрагмент предметної області представляють як множину сутностей, між якими існує деяка множина зв'язків.

**Сутність** (*entity*) – це об'єкт, що може бути ідентифікований деяким способом, що відрізняє його від інших об'єктів, це реальний чи уявний об'єкт

предметної області, інформація про який повинна зберігатися у базі даних і бути доступною. Приклади: конкретна людина, підприємство, подія і т.д.

Розрізняють такі поняття, як тип (клас) сутності й екземпляр сутності. **Набір (тип) сутностей** (*entity set*) – множина сутностей одного типу (тобто сутностей, що мають однакові властивості). Поняття тип сутності відноситься до набору однорідних предметів, подій, осіб, що виступають як єдине ціле. Приклади: усі люди, підприємства, свята і т.д. Набори сутностей не обов'язково повинні бути такими, що не перетинаються. Наприклад, сутність, що належить до набору ЧОЛОВІК, також належить набору ЛЮДИ. **Екземпляр сутності** відноситься до конкретного об'єкту в наборі. Представлення сутності у діаграмах моделей баз даних залежить від обраної нотації. У діаграмах **ER**-моделі сутність представляється у вигляді прямокутника (у нотації Баркера), що містить ім'я сутності. При застосуванні нотацій П. Чена та нотації типу «пташина лапка» застосовується як представлення сутностей у вигляді прямокутників (на етапі інфологічного, чи концептуального проектування), так і табличне представлення сутностей (на етапі логічного проектування у разі обрання реляційного типу бази даних). Сутність фактично задається множиною атрибутів, що описують властивості всіх членів даного набору сутностей і складають кортеж, що задає екземпляр сутності.

**Атрибут** – це поійменована характеристика сутності, що визначає її властивості і приймає значення з деякої множини значень (**домену**). П. Чен визначає атрибут як функцію, що відбиває набір сутностей у набір значень або у декартовий добуток наборів значень. Кожен атрибут забезпечується ім'ям, унікальним у межах сутності. При табличному представленні найменування атрибутів утворюють назви стовпців таблиці. Набір можливих значень атрибуту називають **доменом**.

Множина з одного чи декількох атрибутів, значення яких однозначно визначають кожен екземпляр сутності, називається **ідентифікатором (ключем)**. З визначення атрибуту за П. Ченом **ключем сутності** є така група

атрибутів, що відображення набору сутностей у відповідну групу наборів значень є взаємно однозначним відображенням. Іншими словами, ключ сутності – це один чи більше атрибутів, які унікально визначають дану сутність. Кожен екземпляр сутності повинен мати хоча б один **ідентифікатор** (ключ). Якщо ідентифікаторів кілька, один з них вибирається як привілейований, чи первинний, інші вважаються потенційними ключами.

Атрибути поділяються на прості та складені. **Складений** (*composite*) **атрибут** – це атрибут, який у подальшому можна розділити на кілька додаткових атрибутів. Наприклад, атрибут АДРЕСА можна розділити на атрибути ПОШТОВИЙ\_КОД, МІСТО, ВУЛИЦЯ, ДІМ, КВАРТИРА тощо. **Простий атрибут** такого розділення не допускає.

За кількістю значень, які можуть приймати атрибути, їх підрозділяють на однозначні та багатозначні. **Однозначні атрибути** приймають лише одне значення, наприклад, кожна людина може мати лише один код ДПА. При цьому однозначні атрибути можуть бути складеними, наприклад АДРЕСА\_ПРОПИСКИ. **Багатозначні атрибути** можуть приймати декілька значень, наприклад, людина може закінчити декілька учбових закладів або мати декілька машин різних марок. У моделі П. Чена багатозначні атрибути з'єднуються з сутністю подвійною лінією.

Атрибути можуть класифікуватися за належністю до одного з трьох різних типів: описові, вказівні, допоміжні. **Описові атрибути** надають факти, внутрішньо притаманні кожному екземпляру сутності. Атрибути, що вказують (**вказівні атрибути**), використовуються для присвоєння імені чи позначення екземплярів сутності. **Допоміжні атрибути** використовуються для зв'язку екземпляра однієї сутності з екземпляром іншої. Нарешті, можуть існувати **похідні** (*derived*) **атрибути**, тобто атрибути, які не треба зберігати у базі даних, а можна отримати за допомогою певного алгоритму. Наприклад, вік співробітника СПІВРОБІТНИК\_ВІК можна отримати як різницю між біжучою датою та датою народження (СПІВРОБІТНИК\_ДАТАНАР).

Атрибути підкоряються строго визначеним правилам (**правилам атрибутів**), дотримання яких забезпечується при переході від концептуальної до логічної моделі.

**Зв'язок** (*relationship*) – це асоціація, установлена між кількома сутностями, яка являє собою абстракцію набору відносин, що систематично виникають між різними видами предметів у реальному світі. Більшість зв'язків відносяться до категорії бінарних і має місце між двома сутностями. У **ER**-діаграмах ця асоціація є пойменованою і зображеною графічно у вигляді лінії, що поєднує зв'язані сутності.

Серед бінарних зв'язків існують три фундаментальних **види зв'язку** (**типи зв'язності, connectivity**): **«один-до-одного» ( $1:1$ )**, **«один-до-багатьох» ( $1:M$ )**, **«багато-до-багатьох» ( $M:M$ )**. Зв'язок **«один-до-одного» ( $1:1$ )** існує, коли один екземпляр однієї сутності зв'язаний з єдиним екземпляром іншої сутності. Зв'язок **«один-до-багатьох» ( $1:M$ )** має місце, коли один екземпляр однієї сутності зв'язаний з одним чи більше екземпляром іншої сутності, а кожен екземпляр другої сутності зв'язаний тільки з одним екземпляром першої сутності. **«багато-до-багатьох» ( $M:N$ )** існує, коли один екземпляр однієї сутності зв'язаний з одним чи більше екземпляром іншої сутності і кожен екземпляр другої сутності зв'язаний з одним чи більше екземпляром першої сутності. Окрім зв'язності, зв'язок характеризується **кардинальністю** (або **ступенем** чи **потужністю зв'язку**). Ту кількість сутностей, що може бути асоційована через набір зв'язків з іншою сутністю, називають **ступенем, або потужністю зв'язку**. Цей параметр визначає обов'язковість наявності хоча б одного екземпляру сутності, що приймає участь у зв'язку, а також чисельну кількість цих учасників. Кожний зв'язок має дві кардинальності. Кардинальність, проставлена з боку певної сутності, задає діапазон потужностей множини екземплярів сутності, з якими зв'язаний кожний екземпляр даної сутності.

Участь сутності у зв'язку може бути **обов'язковою** або ні. Участь сутності у зв'язку **необов'язкова**, якщо один екземпляр сутності не вимагає

наявності відповідного екземпляру сутності у окремому зв'язку. Такі зв'язки називаються **умовними**, або **необов'язковими** (*optional*). В умовних зв'язках, на відміну від **безумовних**, можуть існувати екземпляри сутності, які у зв'язку не приймають участі. Якщо зв'язок умовний по обидва боки, він називається **біумовним**. Існування необов'язковості (*optionality*) вказує на те, що для необов'язкової сутності мінімальне значення потужності зв'язку дорівнює 0. Як у діаграмах П. Чена, так і у діаграмах «пташина лапка», необов'язковий зв'язок показується колом з боку необов'язкової сутності.

Усі зв'язки вимагають описання. Опис повинен містити:

- ідентифікатор зв'язку;
- формулювання імен зв'язку з погляду кожної сутності, що бере участь у зв'язку;
- вид зв'язку (множинність (потужність та зв'язність) і умовність);
- формулювання того, як зв'язок був формалізований.

**Роль сутності в зв'язку** – функція, що виконує сутність у даному зв'язку. Наприклад, у зв'язку БАТЬКО-НАЩАДОК сутності ЛЮДИНА можуть мати ролі «батько» і «нащадок». Вказування ролей у моделі «сутність-зв'язок» не є обов'язковим і служить для уточнення семантики зв'язку.

**Набір зв'язків** (*relationship set*) – це відношення між  $n$  (причому  $n$  не менше 2) сутностями, кожна з яких відноситься до деякого набору сутностей.

**Приклад:**

сутності		набори сутностей
-----		-----
$e_1$	належить	$E_1$
$e_2$	належить	$E_2$
. . .		
$e_n$	належить	$E_n$

Тоді  $[e_1, e_2, \dots, e_n]$  – набір зв'язків  $R$ .

Хоча поняття «зв'язок» і «набір зв'язків» різні (перший є елементом другого), їх, проте, дуже часто змішують. Тому, будемо користатися термінами «зв'язок», маючи на увазі «набір зв'язків» і «сутність», маючи на увазі «набір сутностей».

У випадку  $n=2$ , тобто коли зв'язок поєднує дві сутності, він називається **бінарним**. Доведено, що  **$n$ -арний** набір зв'язків ( $n>2$ ) завжди можна замінити множиною бінарних, однак перші краще відображають семантику предметної області.

**Метою формалізації сутності** є визначення її через набір атрибутів та виявлення серед них ключа, що унікально визначає сутність.

Усі сутності відносяться до одного з чотирьох класів:

- стрижневі;
- асоціативні;
- характеристичні;
- такі, що позначають.

**Стрижнева сутність** (стрижень) являє собою незалежну сутність.

**Асоціативна сутність** (асоціація) – це сутність, що формалізує зв'язок виду  $M:N$  між двома чи більше сутностями чи зв'язок виду  $1:1$  між екземплярами сутностей.

**Характеристична сутність** (характеристика) являє собою сутність, що формалізує зв'язок виду  $1:M$  чи  $1:1$ . Єдина мета характеристики в рамках розглянутої предметної області полягає в описі чи уточненні деякої іншої сутності.

**Сутність, що позначає** (позначення) – це сутність, що також формалізує зв'язок виду  $1:M$  чи  $1:1$  між двома сутностями, але відрізняється від характеристики тим, що не залежить від сутності, яка позначається.

Якщо сутність залежить від існування одної чи більше інших сутностей, то говорять, що вона залежна від існування (*existence-dependent*). Наприклад, сутність СПІВРОБІТНИК\_ДІТИ у базі даних СПІВРОБІТНИКИ, що фіксує дітей співробітника для обрахування податків, подарунків тощо, та

описується атрибутами **КІЛЬКІСТЬ** та **ВІК**, є залежною від сутності **СПІВРОБІТНИК**. Очевидно, що залежні сутності можуть розглядатися як характеристичні. Якщо ж сутність існує незалежно від існування іншої, то вона є незалежною. Незалежні сутності можуть бути як стрижневими, так і такі, що позначаються.

Якщо одна сутність незалежна від існування іншої, то зв'язок між ними називається **слабким** (*weak*), або **неідентифікуючим** (*non-identifying*). З погляду проектування баз даних слабкі зв'язки мають місце тоді, коли первинний ключ зв'язаної сутності не містить первинних компонентів породжуючої сутності. **Сильний** (*strong*), або **ідентифікуючий** (*identifying*) зв'язок має місце між зв'язаними сутностями, залежними від існування. З погляду проектування баз даних сильні зв'язки мають місце тоді, коли первинний ключ зв'язаної сутності містить компоненти первинного ключа породжуючої сутності.

До числа більш складних елементів **ER**-моделі відносяться **підтипи** і **супертипи** сутностей. Сутність може бути розщеплена на два чи більш підтипи, що взаємно виключають один одного, кожний з яких має спільні атрибути та/або зв'язки. Ці спільні атрибути та/або зв'язки явно визначаються один раз на більш високому рівні. У підтипах можуть визначатися власні атрибути та/або зв'язки. Сутність, на основі якої визначаються підтипи, називається **супертипом**. Підтипи повинні утворювати повну множину, тобто будь-який екземпляр супертипу повинен відноситися до деякого підтипу. Аналогічно мовам об'єктно-орієнтованого програмування вводиться можливість успадковування типу сутності, виходячи з одного чи декількох супертипів.

**Категоризація сутності.** Сутність може бути розділена і представлена у вигляді двох або більше **сутностей-категорій**, кожна з яких має загальні атрибути та/або відносини, які визначаються одноразово на верхньому рівні і успадковуються на нижньому. Сутності-категорії можуть мати і свої власні атрибути та/або відносини, а також, у свою чергу, можуть бути



декомпоновані своїми сутностями-категоріями на наступному рівні. Розщеплена на категорії сутність отримала назву **загальної сутності** (зауважимо, що на проміжних рівнях декомпозиції одна і та ж сутність може бути як загальною сутністю, так і сутністю-категорією).

Для демонстрації декомпозиції сутностей на категорії використовуються діаграми категоризації. Така діаграма містить загальну сутність, дві і більше сутностей-категорій і спеціальний **вузол-дискримінатор**, який описує способи декомпозиції сутності (Рисунок 1).



Рисунок 1. Діаграма категоризації

Існує чотири можливі типи дискримінаторів :

1. **Повне і обов'язкове входження Е/М** (*exclusive/mandatory*) – сутність повинна бути однією і лише однією з категорій, що з неї слідує.
2. **Повне і необов'язкове входження Е/О** (*exclusive/optional*) – сутність може бути однією і лише однією з категорій, що з неї слідує.
3. **Неповне і обов'язкове входження І/М** (*inclusive/mandatory*) – сутність повинна бути принаймні однією з категорій, що з неї слідує.
4. **Неповне і необов'язкове входження І/О** (*inclusive/optional*) – сутність може бути принаймні однією з категорій, що з неї слідує.

У випадку дуже великої кількості сутностей і зв'язків між ними застосовується менш наочна, ніж мова **ER**-діаграм, але більш змістовна мова інфологічного моделювання, у якій сутності і зв'язки представляються реченнями вигляду:

**СУТНІСТЬ** (атрибут 1, атрибут 2, ..., атрибут n)

**ЗВ'ЯЗОК [СУТНІСТЬ  $S_1$ , СУТНІСТЬ  $S_2$ , ...] (атрибут 1, атрибут 2, ..., атрибут  $n$ ).**

**Приклад.**

Пацієнт, маючи одного лікаря, може мати також декілька лікарів-консультантів; лікар може бути лікарем декількох пацієнтів і може одночасно консультувати кілька інших пацієнтів.

Так, розглянутий вище приклад безлічі зв'язків між сутностями, може бути описаний у такий спосіб:

**Лікар** (Номер\_лікаря, Прізвище, Ім'я, По батькові, Спеціальність)

**Пацієнт** (Реєстраційний\_номер, Номер ліжка, Прізвище, Ім'я, По батькові, Адреса, Дата народження, Стать)

**Лікуючий\_лікар** [Лікар  $I$ , Пацієнт  $M$ ]

(Номер\_лікаря, Реєстраційний\_номер)

**Консультант** [Лікар  $M$ , Пацієнт  $N$ ]

(Номер\_лікаря, Реєстраційний\_номер).

**Зображення інфологічних моделей у вигляді ER-діаграм. Надання даних за допомогою моделі «сутність-зв'язок».** Перш, ніж перейти до створення системи автоматизованої обробки інформації, розроблювач повинен сформулювати поняття про предмети, факти і події, якими буде оперувати дана система. Для того, щоб привести ці поняття до тої чи іншої моделі даних, необхідно замінити їх інформаційними представленнями. Одним з найбільш зручних інструментів уніфікованого представлення даних, незалежного від програмного забезпечення, яке його реалізує, є модель «сутність-зв'язок» (*ER-model*).

Модель «сутність-зв'язок» не є моделлю даних у строгому розумінні, оскільки не визначає операцій над даними й обмежується описом тільки їхньої логічної структури.

Довільний фрагмент предметної області може бути представлений як **множина сутностей**, між якими існує деяка **множина зв'язків**. Графічне відбиття цього представлення й утворює *ER*-діаграму. У процесі цього

графічного представлення сутності різного типу зображуються прямокутниками різної форми з відповідним обрамленням; атрибути – зв'язаними відрізками ліній з прямокутниками сутностей овалами або впровадженими всередину прямокутників записами; зв'язки – відрізками ліній, всередині яких може бути введений графічний елемент з текстом, що описує тип зв'язку, а на кінцях – графічне або текстове позначення кардинальності зв'язку. **ER-діаграма задає графічне представлення концептуальної схеми бази даних**, конкретний вигляд якого залежить як від предметної галузі та мети створення бази даних, так і від обраної нотації, тобто стандартизованого способу графічного представлення елементів **ER-діаграми**. Найчастіше застосовують нотації Чена, Мартіна («пташина лапка»), **IDEFIX**, Баркера.

**Зауваження.** Слід зазначити, що в методиці проектування даних є правило, відповідно до якого сутності позначаються за допомогою іменників, а зв'язки – дієслівними формами. Дане правило, однак, не є обов'язковим.

Дуже важливою властивістю моделі «сутність-зв'язок» є те, що вона може бути представлена у вигляді графічної схеми. Це значно полегшує аналіз предметної області. Існує кілька стандартизованих варіантів позначення (нотацій) елементів діаграми «сутність-зв'язок», кожний з яких має свої позитивні риси. Атрибути із сутностями і сутності зі зв'язками з'єднуються лініями, переважно прямими. Позначення елементів залежать від обраної нотації.

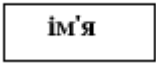
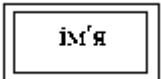
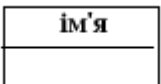
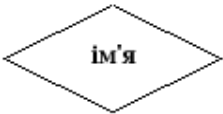
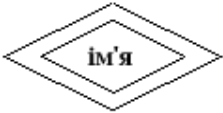
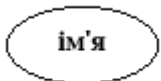
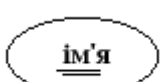
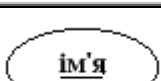
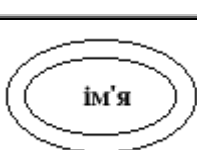
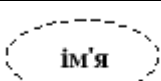
### **Нотації, що застосовуються при побудові моделей сутність зв'язок (ER-діаграм).**

#### **1. Нотація П. Чена.**

Це історично перша і найпростіша нотація, яка широко застосовується при інфологічному проектуванні (Таблиця 1). Основний її недолік – «перевантаженість» поля діаграми атрибутами для складних баз даних, перевага – ідентифікація типів атрибутів, простота сприйняття.

Таблиця 1. Позначення елементів при побудові **ER**-діаграм у нотації П.

Чена

Елемент діаграми	Позначає
	незалежна сутність
	залежна сутність
	батьківська сутність в ієрархічному зв'язку
	зв'язок
	ідентифікуючий зв'язок
	атрибут
	первинний ключ
	зовнішній ключ (поняття зовнішнього ключа вводиться у реляційній моделі даних)
	багатозначний атрибут
	отримуваний (успадкований) атрибут у ієрархічних зв'язках

Зв'язок з'єднується з поєднуваними сутностями лініями. Біля кожної сутності на лінії, що з'єднує її зі зв'язком, цифрами вказується кардинальність, як це показано нижче (Рисунок 2).

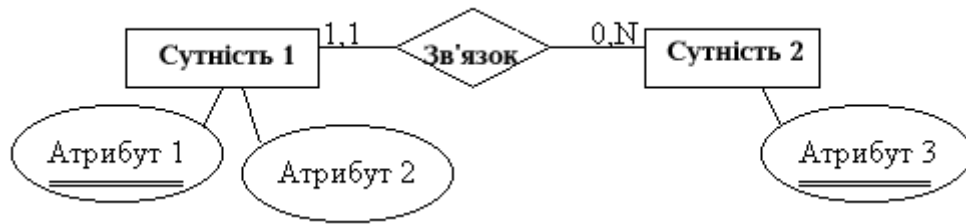


Рисунок 2. Позначення елементів **ER**-діаграми у нотації П. Чена

Представлення фрагменту бази даних СПІБРОБІТНИКИ для ситуації, коли один співробітник працює лише у одному підрозділі, наведене нижче. Зверху показана зв'язність, нижче – кардинальність зв'язку (Рисунок 3).

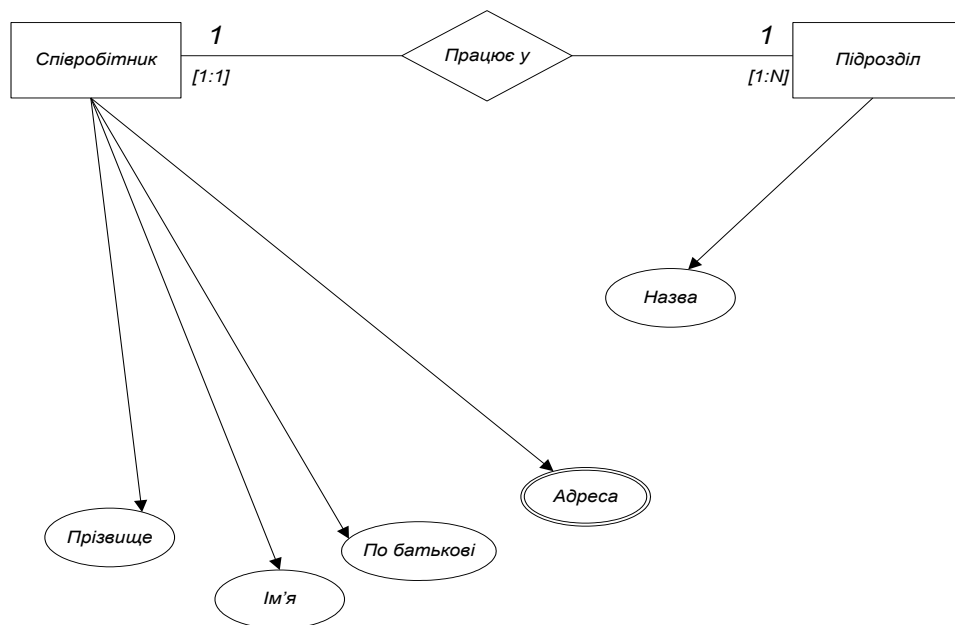


Рисунок 3. Фрагмент **ER**-діаграми у нотації П. Чена

## 2. Нотація Мартіна («пташина лапка», «crow's feet»)


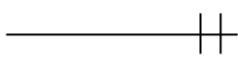
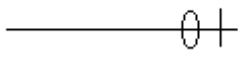
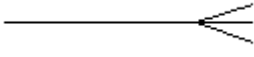
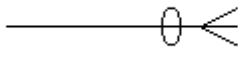
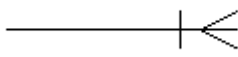
Методологія **IE** (*Information Engineering*), розроблена Мартіном, Финкельштейном і іншими авторами, використовується переважно в промисловості (Таблиця 2). Перевага – графічне позначення кардинальності, недолік – необхідність його пам'ятати при читанні діаграми, а також необхідність перебудови діаграми при внесенні коректив у кардинальність.

Таблиця 2. Позначення елементів при побудові **ER**-діаграм у нотації Мартіна («пташина лапка»)

Елемент діаграми	Позначає
	незалежна сутність
	залежна сутність
	батьківська сутність в ієрархічному зв'язку

Перелік атрибутів приводиться усередині прямокутника, що позначає сутність. Ключові атрибути підкреслюються. Зв'язки зображуються лініями, що з'єднують сутності, вигляд лінії в місці з'єднання із сутністю визначає кардинальність зв'язку (Таблиця 3).

Таблиця 3. Позначення кардинальності зв'язку при побудові **ER**-діаграм у нотації Мартіна («пташина лапка»)

Позначення	Кардинальність
	немає
	1,1
	0,1
	M,N
	0,N
	1,N

Ім'я зв'язку вказується на лінії, що його позначає (Рисунок 4).



Рисунок 4. Відбиття зв'язків у нотації Мартіна

### 3. Нотація *IDEFIX*.

Методологія *IDEFIX* (Таблиця 4) була розроблена для армії США і широко використовується в державних установах США, фінансових і промислових корпораціях.

Таблиця 4. Позначення елементів при побудові *ER*-діаграм у нотації *IDEFIX*

Елемент діаграми	Позначає
	незалежна сутність
	залежна сутність

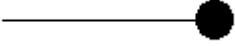
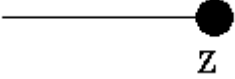
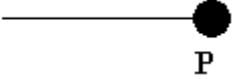
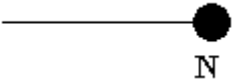
Список атрибутів наводиться усередині прямокутника, що позначає сутність. Атрибути, що складають ключ сутності, групуються у верхній частині прямокутника і відокремлюються горизонтальною рисою (Таблиці 5, 6, рисунок 5).

Таблиця 5. Позначення зв'язків при побудові *ER*-діаграм у нотації *IDEFIX*

Елемент діаграми	Позначає
	ідентифікуюча зв'язок
	неідентифікуючий зв'язок

Таблиця 6. Позначення кардинальності зв'язку при побудові *ER*-діаграм у нотації *IDEFIX*

Елемент діаграми	Позначає
	1,1

	0,M
	0,1
	1,M
	точно $N$ ( $N$ – довільне число)

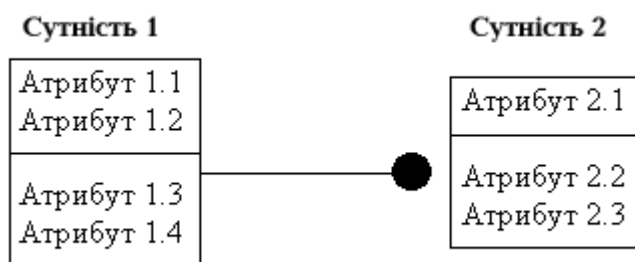


Рисунок 5. Відбиття зв'язків у нотації **IDEFIX**

Крім того, у **IDEFIX** вводится поняття «**відношення категоризації**», за змістом еквівалентне розглянутому нами ієрархічному зв'язку. Позначення відношення повної категоризації (сутності-категорії складають повну множину нащадків батьківської сутності) наведене на рисунку 6.

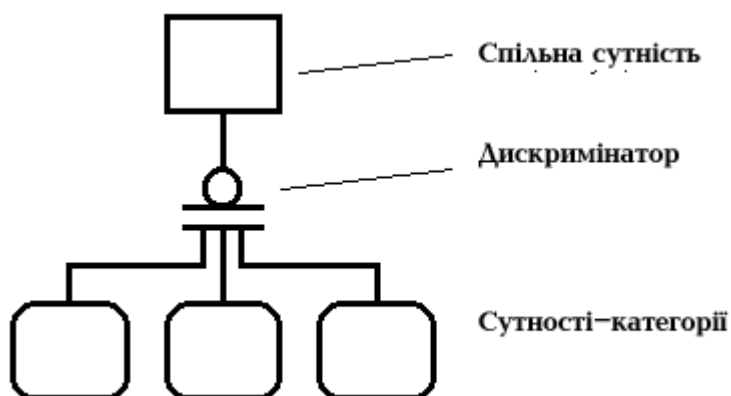


Рисунок 6. Відношення повної категоризації у нотації **IDEFIX**

Також може існувати відношення неповної категоризації (сутності-категорії складають неповну множину нащадків спільної сутності) (Рисунок 7)



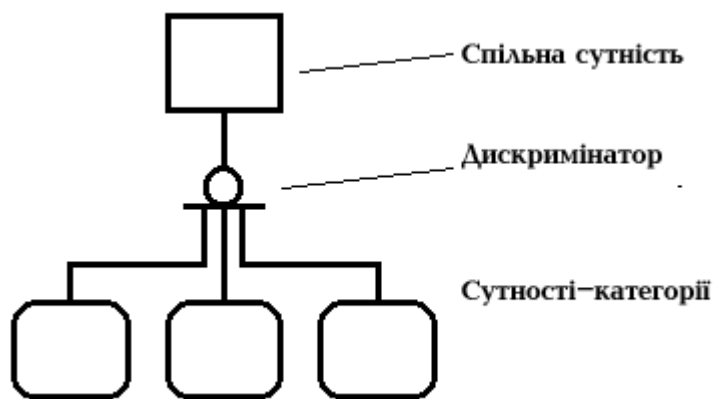


Рисунок 7. Відношення неповної категоризації у нотації **IDEFIX**

#### 4. Нотація Баркера.

Сутності позначаються прямокутниками, усередині яких наводиться список атрибутів. Ключові атрибути відзначаються символом # (решітка). Зв'язки позначаються лініями з іменами, місце з'єднання зв'язку і сутності визначає кардинальність зв'язку (Таблиця 7, Рисунок 8).

Таблиця 7. Позначення елементів при побудові **ER**-діаграм у нотації Баркера

Позначення	Кардинальність
-----	0,1
————	1,1
----->{	0,N
————>{	1,N

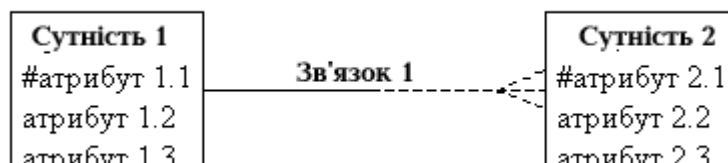


Рисунок 8. Відбиття зв'язків у нотації Баркера

Для позначення відношення категоризації уводиться елемент «дуга» (Рисунок 9).

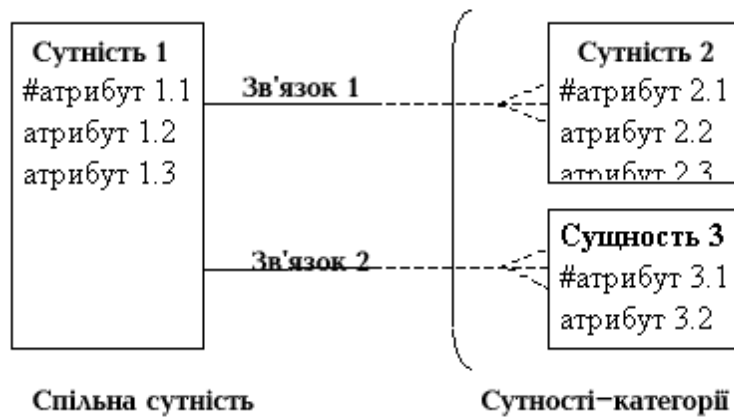


Рисунок 9. Відбиття відношення категоризації у нотації Баркера

У процесі побудови діаграми можна виділити кілька етапів:

- 1) ідентифікація сутностей, що становлять інтерес, і зв'язків;
- 2) присвоєння назв сутностям;
- 3) ідентифікація семантичної інформації в наборах зв'язків (наприклад, чи є деякий набір зв'язків відображенням  $1:N$ );
- 4) визначення типів сутностей й атрибутів, визначення зв'язностей, кардинальностей і обов'язковості (умовності) зв'язків;
- 5) визначення атрибутів і наборів їхніх значень (доменів);
- 6) організація даних у вигляді відносин «сутність-зв'язок»;
- 7) формулювання зв'язків з погляду кожної сутності, що бере в них участь;
- 8) виділення багатозначних атрибутів та вибір способу заміни їх однозначними, введення додаткових сутностей, якщо це потрібно;
- 9) виділення типів та супертипів сутностей, їх дискримінація;
- 10) формалізація зв'язків вигляду  $1:1$ ,  $1:M$ ,  $M:N$ ;
- 11) виділення стрижневих, асоціативних, характеристичних сутностей та сутностей, що позначають;
- 12) вибір нотації для представлення моделі;
- 13) нанесення на діаграму сутностей, під'єднання до них атрибутів з урахуванням типів;
- 14) позначення зв'язків та їх параметрів, остаточна побудова **ER**-діаграми.

### **Порядок виконання роботи**

1. На основі попередньої лабораторної роботи реалізувати діаграму.
2. Визначити типи зв'язків між сутностями.
3. Зобразити у вигляді реляційних схем відповідні сутності та атрибути.
4. Визначити первинні та зовнішні ключі в реляційних відношеннях.
5. Оформити короткий звіт про виконану роботу, внести в нього основні теоретичні відомості.

### **Оформлення звіту:**

1. Титульний лист.
2. Мета роботи.
3. Порядок виконання.
4. Представлення основних результатів по виконаній лабораторній роботі.
5. Висновки.

### **Контрольні питання**

1. Характеристика типів зв'язків між сутностями.
2. Поняття кортежу, домену, атрибута.
3. Кардинальність реляційного відношення.
4. Степінь відношення.
5. Поняття про ключ.
6. Зовнішній ключ, його властивості.
7. Нотації представлення моделей.

### Лабораторна робота № 3

**Тема:** Проектування баз даних засобами *ERwin*.

**Мета роботи:** здобуття студентами практичних навичок побудови логічних та фізичних моделей даних за допомогою *CASE (Computer Aided Software Engineering)* – засобів розробки інформаційних систем.

#### Теоретичні відомості

*CASE*-засоби – це програмні засоби, що підтримують процеси створення і супроводу інформаційних систем, включаючи аналіз і формулювання вимог, проектування прикладного програмного забезпечення (додатків) і баз даних, генерацію коду, тестування, документування, забезпечення якості. *ERwin* – *CASE*-засіб проектування баз даних від фірми *Computer Associates*. *ERwin* сполучає графічний інтерфейс *Windows*, інструменти для побудови *ER*-діаграм, редактори для створення логічного та фізичного опису моделі даних та прозору підтримку ведучих реляційних систем управління базами даних (СУБД). *ERwin* не прив'язаний до технології, будь-якої конкретної фірми, яка поставляє СУБД або засоби розробки. Він підтримує різні сервери баз даних та настільні СУБД, а також може звертатися до бази через інтерфейс *ODBC (Open Database Connectivity)* — програмний інтерфейс (*API – Application Programming Interface*) доступу до баз даних, який розроблений фірмою *Microsoft*).

Система *ERwin* підтримує пряме та зворотне моделювання баз даних. При прямому моделюванні схема бази даних описується в прямому вигляді з використанням діаграми «сутність-зв'язок». Сутності на діаграмі зображуються прямокутниками. Кожен прямокутник може мати різні візуальні атрибути. Кожній сутності повинно бути присвоєне унікальне ім'я. Імена сутностей необхідно задавати в однині. Це визначається тим, що система завжди оперує окремими екземплярами сутності. При цьому окремі екземпляри сутності розглядаються як об'єкти, а сутності – як клас об'єктів. Якщо сутності вже були визначені при моделюванні в *ERwin*, то є

можливість їх просто імпортувати в *ERwin*. Приклад діаграми із створеними сутностями наведено нижче (Рисунок 1).

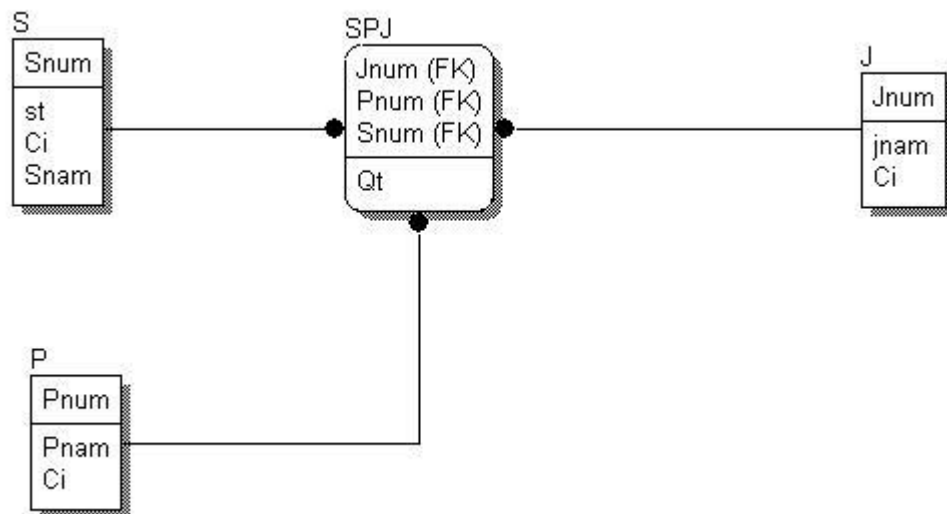


Рисунок 1. Приклад діаграми із створеними сутностями

Процес моделювання в *ERwin* базується на методології проектування реляційних баз даних *IDEF1X*, яка використовує умовний синтаксис, спеціально розроблений для зручної побудови концептуальної схеми. **Концептуальною схемою** ми називаємо універсальне уявлення структури даних в рамках комерційного підприємства, незалежне від кінцевої реалізації бази даних і апаратної платформи. Будучи статичним методом розробки, *IDEF1X* спочатку не призначений для динамічного аналізу за принципом "AS IS", тим не менш, він іноді застосовується в цій якості, як альтернатива методу *IDEF1 (Information Modeling)* – методологія проектування складних систем. Застосовується для побудови інформаційної моделі, яка представляє структуру інформації, необхідної для підтримки функцій виробничої системи або середовища.). Використання методу *IDEF1X* найбільш доцільно для побудови логічної структури бази даних після того, як всі інформаційні ресурси досліджені (скажімо за допомогою методу *IDEF1*) і рішення про впровадження реляційної бази даних, як частини корпоративної інформаційної системи, було прийнято. Однак не варто забувати, що засоби моделювання *IDEF1X* спеціально розроблені для побудови реляційних інформаційних систем, і якщо існує необхідність проектування іншої

системи, скажімо об'єктно-орієнтованої, то краще обрати інші методи моделювання.

Існує кілька очевидних причин, за якими *IDEFIX* не слід застосовувати у разі побудови не реляційних систем. По-перше, *IDEFIX* вимагає від проектувальника визначити ключові атрибути, для того щоб відрізнити одну сутність від іншої, в той час, як об'єктно-орієнтовані системи не вимагають завдання ключів, з метою ідентифікації об'єктів. По-друге, в тих випадках, коли більш, ніж один атрибут однозначно ідентифікує сутність, проектувальник повинен визначити один з цих атрибутів первинним ключем, а всі інші вторинними. І, таким чином, побудована проектувальником *IDEFIX*-модель і передана для остаточної реалізації програмісту, є некоректною для застосування методів об'єктно-орієнтованої реалізації, і призначена для побудови реляційної системи.

**Побудова моделей в ERWin.** Можливі дві точки зору на інформаційну модель і, відповідно, два рівня моделі. Перший – **логічний рівень** (точка зору користувача) означає пряме відображення фактів з реального життя. Наприклад, люди, столи, відділи, собаки та комп'ютери є реальними об'єктами. Вони іменуються на природній мові, з довільними роздільниками слів (пропуски, коми і т.д.). На **фізичному рівні** моделі розглядається використання конкретної СУБД, визначаються типи даних (наприклад, ціле або дійсне число), індекси для таблиць.

Для переключення між логічною та фізичною моделями даних призначений список вибору в лівій частині панелі інструментів *ERwin* (Рисунок 2).

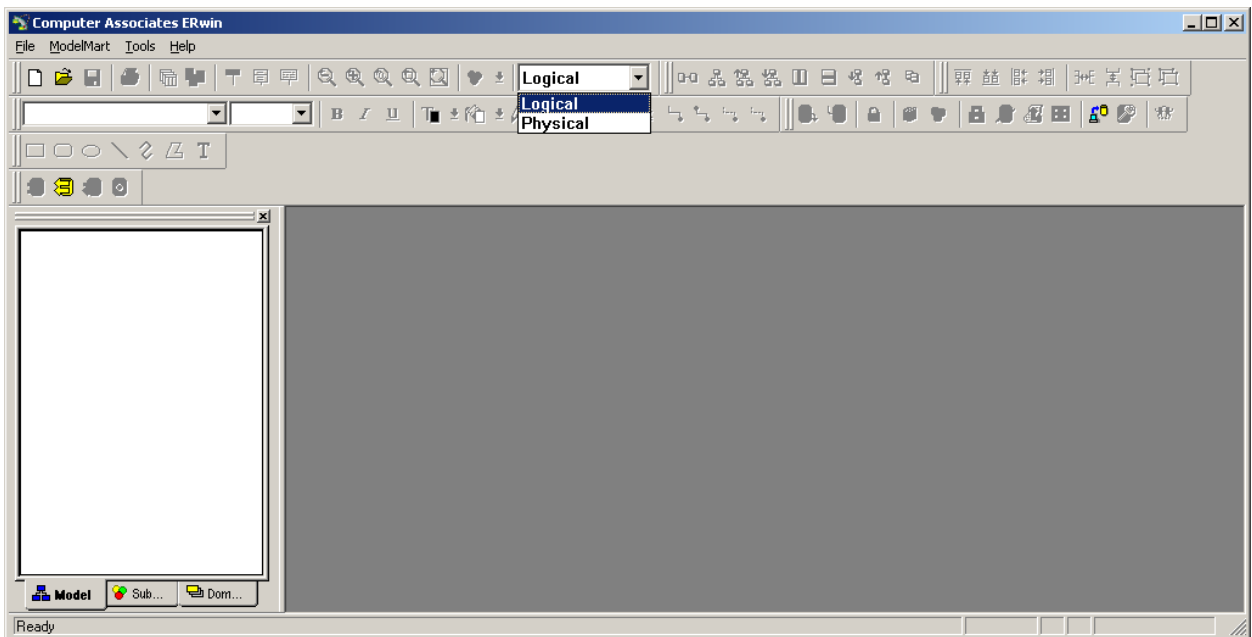


Рисунок 2. Переключення між фізичною та логічною моделями

При переключенні, якщо фізичної моделі не існує, вона буде створюватися автоматично.

**ERWin** надає можливості створювати і керувати цими двома різними рівнями представлення однієї діаграми (моделі), а також можливості мати багато варіантів відображення на кожному рівні. Термін «логічний рівень» в **ERWin** відповідає концептуальній моделі.

### **Етапи побудови інформаційної моделі:**

- визначення сутностей;
- визначення залежностей між сутностями;
- завдання первинних та альтернативних ключів;
- визначення атрибутів сутностей;
- зведення моделі до необхідного рівня нормальної форми;
- перехід до фізичного визначення моделі: призначення відповідностей: Ім'я\_сутності – Ім'я\_таблиці, Атрибут\_сутності – Атрибут\_таблиці;
- визначення тригерів, процедур і обмежень;
- генерація бази даних.

**ERWin** створює візуальне представлення (модель даних) для задачі, що розглядається. Це подання може використовуватися для детального аналізу, уточнення і поширення документації, необхідної в процесі розробки. Однак

**ERWin** далеко не тільки інструмент для малювання. **ERwin** автоматично створює базу даних, а саме таблиці, індекси, збережені процедури, тригери для забезпечення посилальної цілісності та інші об'єкти, необхідні для управління даними.

### Теоретичні відомості

Для запуску програми **ERwin** необхідно скористатися командами **Пуск/ERwin 4.0** В результаті відкриється діалогове вікно **Computer Associates ERWin** (Рисунок 3). В цьому вікні треба обрати опцію **Create a new model** (Створення нової моделі) і натиснути по кнопці **OK** (Створити).

**Зауваження.** Вікно **Computer Associates ERWin** дозволяє відкрити вже існуючу модель даних за допомогою вибору опції **Open an existing file** (Відкриття існуючого файлу).

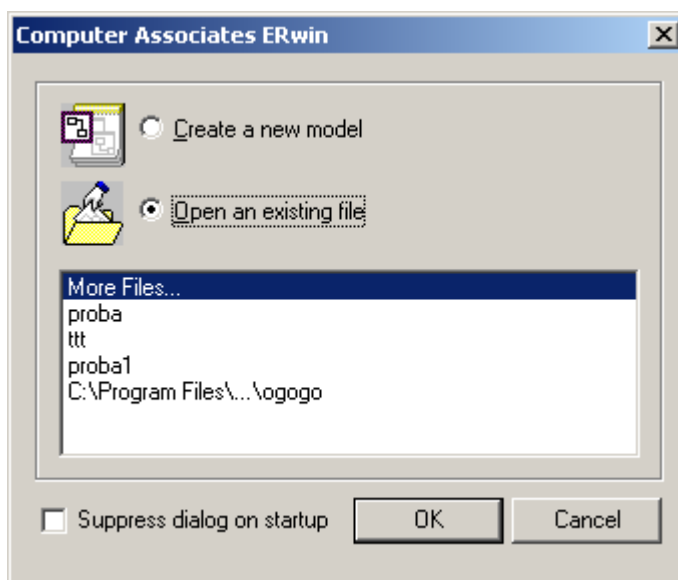


Рисунок 3. Вибір типу моделі

Після вказаних дій відкривається вікно **Create Model – Select Template** (Створення моделі – Вибір шаблону) (Рисунок 4).



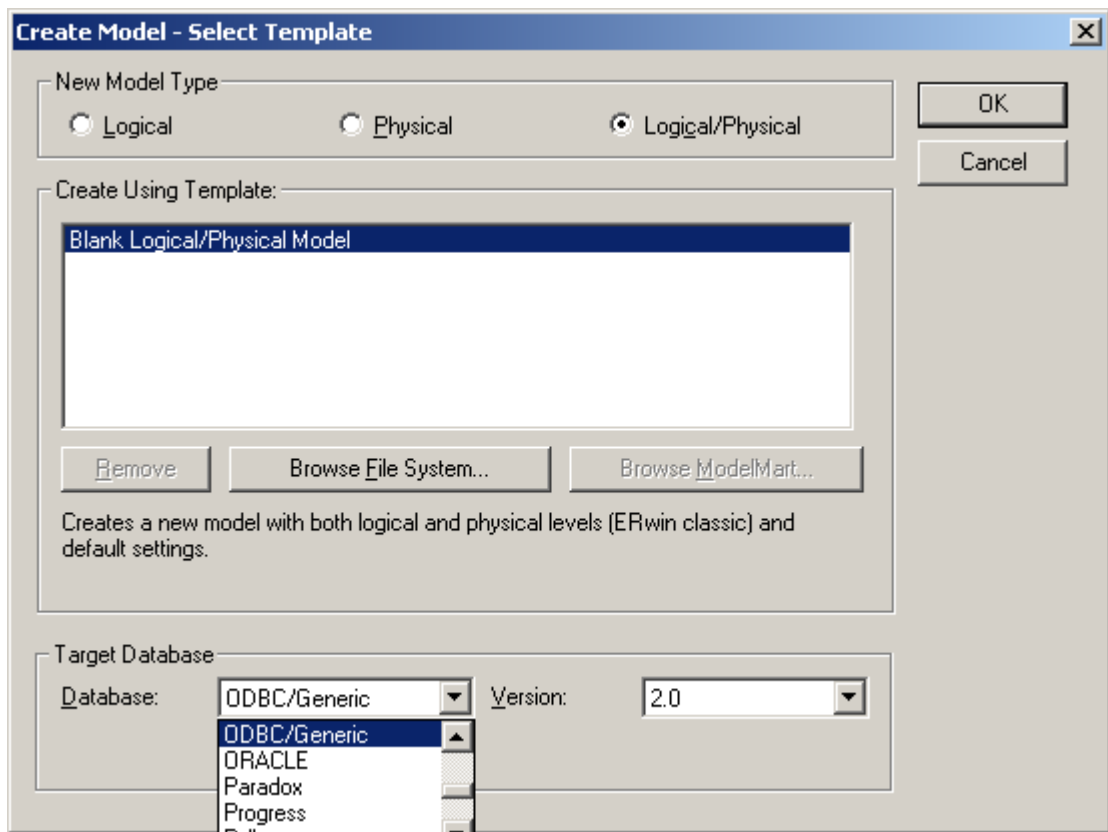


Рисунок 4 . Вибір шаблону моделі

У фреймі *New Model Type* необхідно встановити опцію *Logical/Physical*, а у фреймі *Target Database* (Цільовий сервер бази даних) обрати із списку, який випадає, наприклад *ODBC/Generic* (для СУБД *MySQL*). В результаті автоматично створюється незаповнена модель даних *ERwin* (Рисунок 5) і отримано доступ до інтерфейсу середовища *ERwin*.

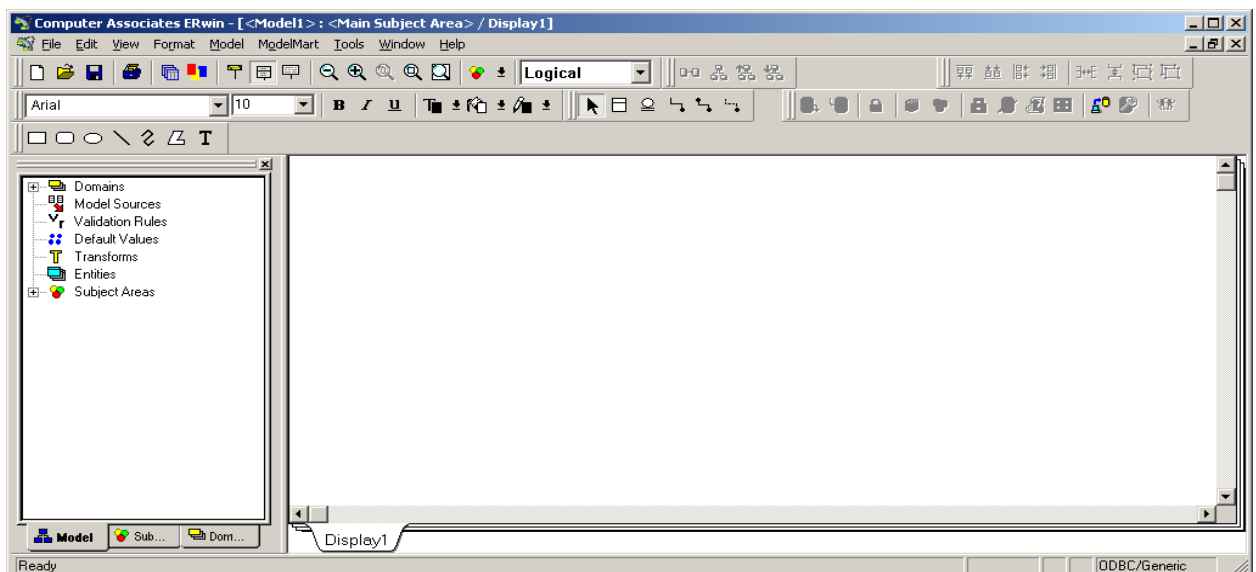


Рисунок 5. Незаповнена модель даних *ERwin*

Робочий простір **ERwin** містить «Вікно діаграми» (**Diagram Window**) та «Браузер (провідник) моделі» (**Model Explorer**) (Рисунок 6).

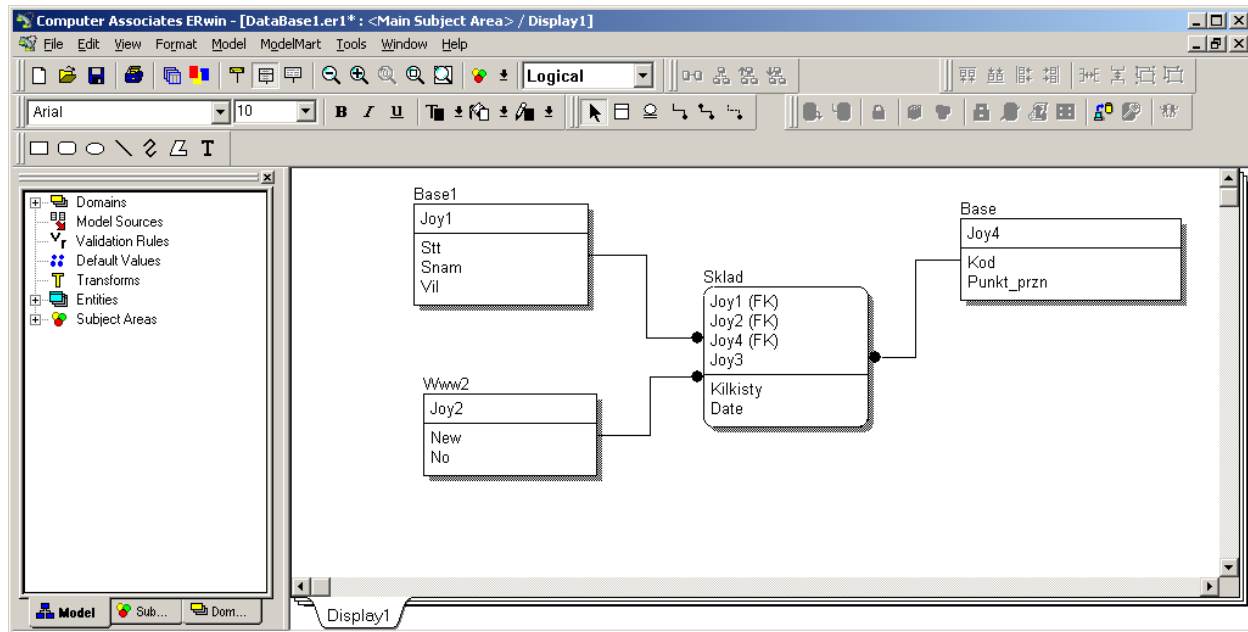



Рисунок 6. Робочий простір **ERwin**

**Створення сутності.** Для внесення сутності в модель необхідно натиснути на кнопку сутності на панелі інструментів (**Erwin Toolbox**) , потім – по тому місцю на діаграмі, де необхідно розташувати нову сутність. Натиснувши правою кнопкою миші по сутності і вибравши з контекстного меню пункт **Entity Editor**, можна викликати діалог **Entity Editor**, в якому визначаються ім'я, опис та коментарі сутності.

Кожна сутність повинна бути повністю визначена за допомогою текстів опису в закладці **Definition**. Ці визначення корисні як на логічному рівні, оскільки дозволяють зрозуміти, що це за об'єкт, так і на фізичному рівні, оскільки їх можна експортувати як частину схеми і використовувати в реальній базі даних (**CREATE COMMENT on entity\_name**). Дані **Note**, **Note2**, **Note3**, **UDP** (**User Defined Properties** – властивості, визначені користувачем) служать для внесення додаткових коментарів і визначень до сутності.

У закладці **Icon** кожної сутності можна поставити у відповідність зображення, яке буде відображатися в режимі перегляду моделі на рівні іконок і зображення, яке буде відображатися на всіх інших рівнях.

Закладка **UDP** діалогу **Entity Editor** служить для визначення властивостей, які задаються користувачем (**User – Defined Properties**). При натисканні на кнопку [...] цієї закладки викликається діалог **User - Defined Property Editor** (також викликається з меню **Edit / UDP**). У ньому необхідно вказати вид об'єкту, для якого створюється **UDP** (діаграма в цілому, сутність, атрибут і т.д.) і тип даних. Для внесення нової властивості слід в таблиці натиснути на кнопку [+] і внести ім'я, тип даних, значення за замовчуванням і визначення.

**Створення атрибутів.** Наступний етап створення моделі полягає у визначенні атрибутів для кожної сутності. При визначенні типу атрибуту є можливість використовувати домени. **Домен** – це абстрактний користувацький тип, який присвоюється будь-якому фізичному типу даних. При цьому кожен домен може мати свої значення за замовчуванням і правила перевірки введених даних. **ERwin** надає можливість документувати всі дії для створення власних типів даних (Рисунки 7 – 10). Використання концепції домену дає можливість перенесення бази даних на різні апаратні платформи.

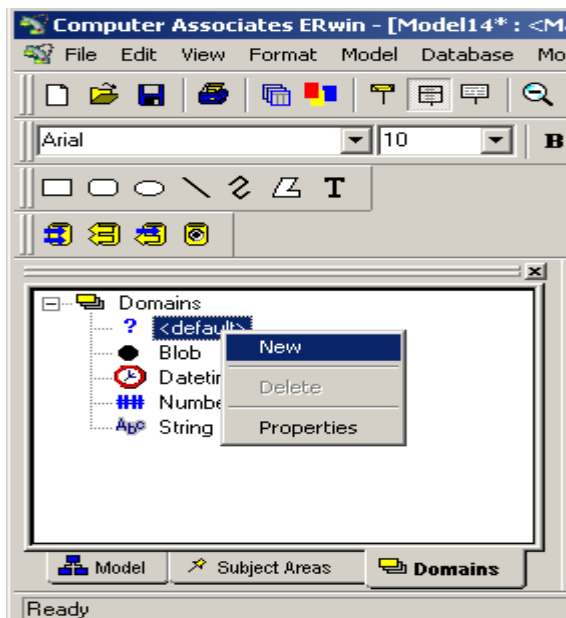


Рисунок 7. Створення нового домену

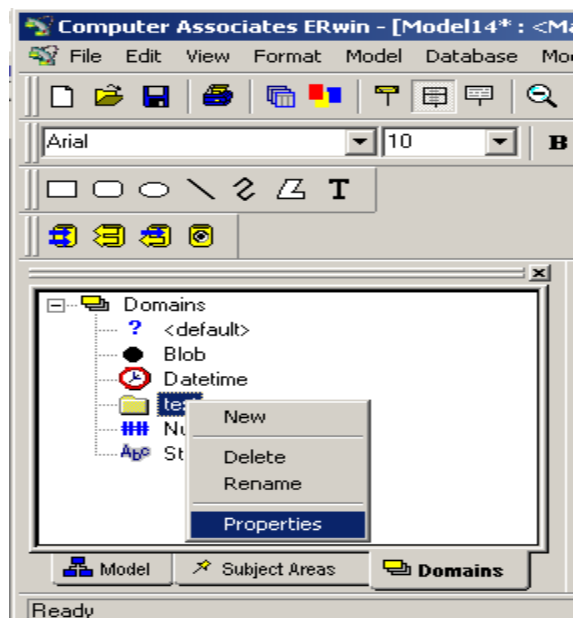


Рисунок 8. Визначення властивостей нового домену

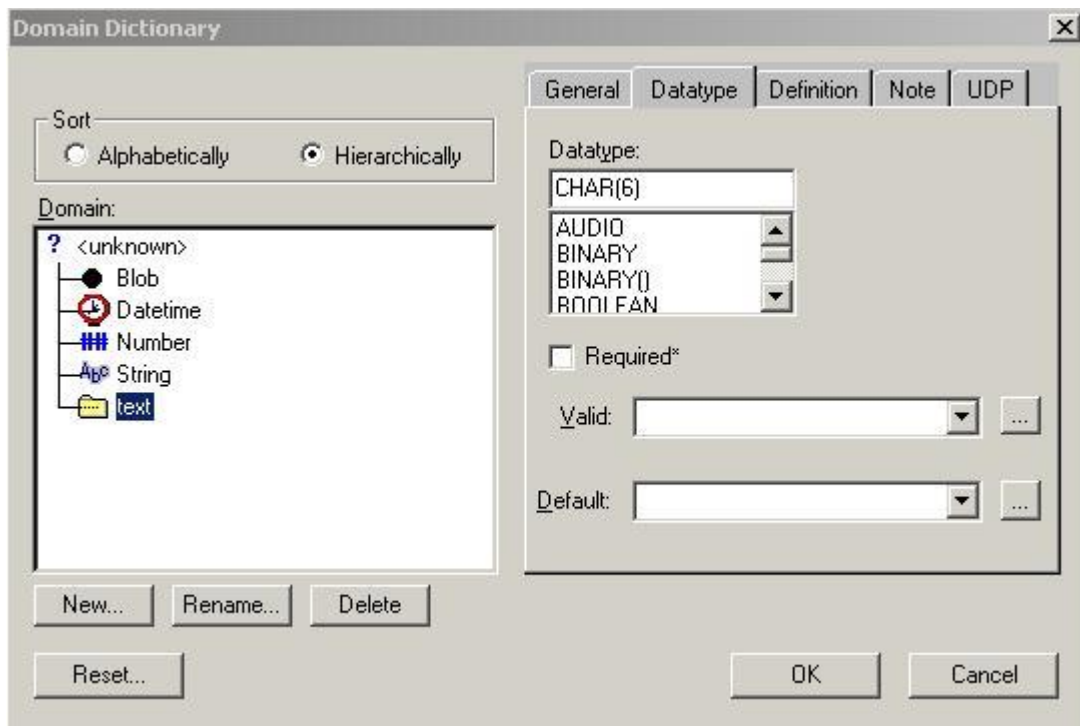


Рисунок 9. Значення по замовченню для нового домену

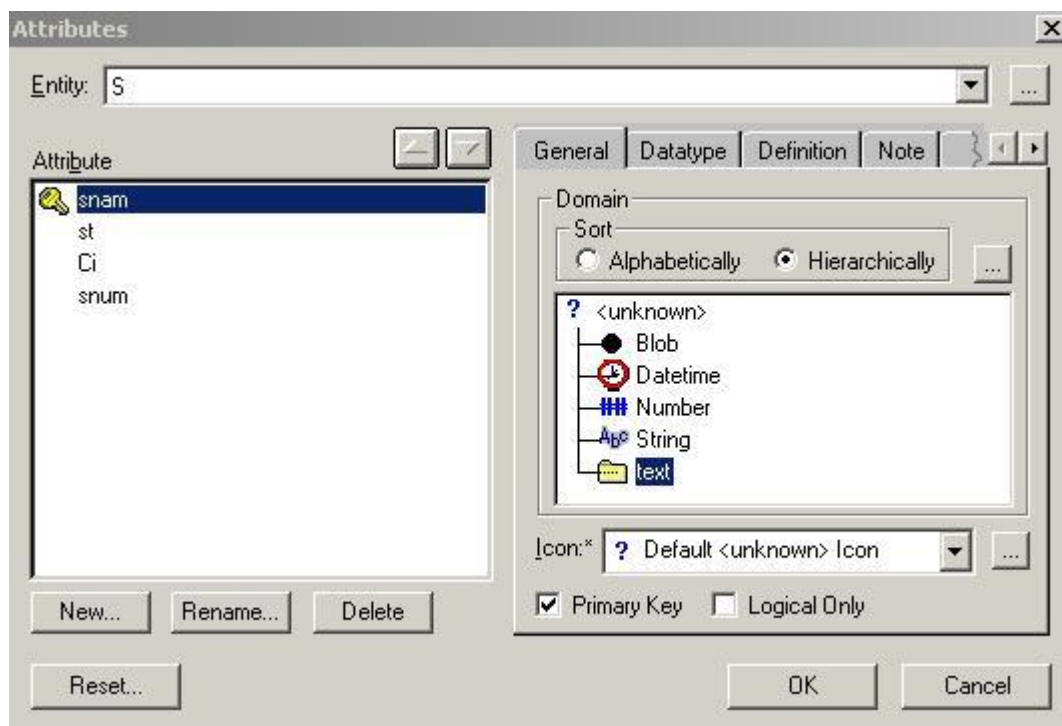


Рисунок 10. Використання домену для вказівки типу даних атрибуту

Для опису атрибутів необхідно натиснути правою кнопкою по сутності і обрати в меню пункт **Attribute Editor**. З'явиться діалог **Attribute Editor**.

Якщо натиснути на кнопку **New**, то в діалозі **New Attribute** можна вказати ім'я атрибута, ім'я відповідної йому у фізичній моделі колонки і домен. Домен


атрибути буде використовуватися при визначенні типу колонки на рівні фізичної моделі.




Для атрибутів первинного ключа в закладці **General** діалогу **Attribute Editor** необхідно зробити позначку у вікні вибору **Primary Key**. Дані **Definition**, **Note** і **UDP** несуть ті ж функції, що і при визначенні сутності, але на рівні атрибутів.

Для більшої наочності діаграми кожен атрибут можна з'єднати з іконкою. Це можна зробити за допомогою списку вибору **Icon** в закладці **General**.

Дуже важливо дати атрибуту правильне ім'я. Атрибути повинні іменуватися в однині і мати чітке смислове значення.

Згідно синтаксису **IDEFIX**, ім'я атрибута повинно бути унікальним в межах моделі (а не тільки в межах сутності!). За замовченням, при спробі внесення вже існуючого імені атрибута, **ERwin** надає йому нове ім'я. Наприклад, якщо атрибут **Коментар** вже існує в моделі, інший атрибут (в іншій сутності) буде названий **Коментар / 2**, потім **Коментар / 3** і т.д.

При перенесенні атрибутів всередині і між сутностями можна скористатися технікою «**drag & drop**», вибравши кнопку  на панелі інструментів.

Для створення нового зв'язку слід вибрати ідентифікуючий або не ідентифікуючий зв'язки в палітрі інструментів (**ERwin Toolbox**), натиснути спочатку по батьківській, а потім по дочірньої сутності. У палітрі інструментів кнопка  відповідає ідентифікуючому зв'язку, кнопка  – зв'язку «багато-до-багатьох», а кнопка  відповідає неідентифікуючому зв'язку. Для редагування властивостей зв'язку слід натиснути правою кнопкою миші по зв'язку і обрати в контекстному меню пункт **Relationship Editor**.

У закладці **General** з'явиться діалог, де можна задати потужність, ім'я та тип зв'язку.

Зв'язки на діаграмі зображуються лініями, що йдуть від однієї сутності (таблиці) до іншої. Кожному зв'язку присвоюється унікальне ім'я. З'єднані таблиці поділяють на батьківські та дочірні. Батьківські таблиці зображуються прямокутниками з прямими кутами, дочірні – із заокругленими.

**Потужність зв'язку (*Cardinality*)** – використовують для позначення відношення числа екземплярів батьківської сутності до числа екземплярів дочірньої.

Розрізняють чотири типи потужності:

- загальний випадок, коли одному екземпляру батьківської сутності відповідають 0, 1 або багато екземплярів дочірньої сутності, не позначається ні яким символом;
- символом **R** позначається випадок, коли одному екземпляру батьківської сутності відповідають 1 або багато екземплярів дочірньої сутності (виключено нульове значення);
- символом **Z** позначається випадок, коли одному екземпляру батьківської сутності відповідають 0 або 1 екземпляр дочірньої сутності (виключені множинні значення);
- цифрою позначається випадок, коли одному екземпляру батьківської сутності відповідає заздалегідь задане число екземплярів дочірньої сутності.

За замовченням символ, що позначає потужність зв'язку, на діаграмі не зображується. Для відображення імені слід в контекстному меню, яке з'являється, якщо натиснути правою кнопкою миші в будь-якому місці діаграми, не зайнятого об'єктами моделі, обрати пункт ***Display Options / Relationship*** і потім включити опцію ***Cardinality***.

**Тип зв'язку (ідентифікуючого / неідентифікуючого).** У ***IDEF1X*** розрізняють залежні і незалежні сутності. Тип сутності визначається її зв'язком з іншими сутностями. **Ідентифікуючий** зв'язок встановлюється між незалежною (батьківський кінець зв'язку) і залежною (дочірній кінець зв'язку) сутностями. Коли малюється ідентифікуючий зв'язок, ***ERwin***

автоматично перетворює дочірній зв'язок в залежний. Залежна сутність зображується прямокутником із заокругленими кутами.

Екземпляр залежної сутності визначається тільки через ставлення до батьківської сутності. При встановленні ідентифікуючого зв'язку атрибути первинного ключа батьківської сутності автоматично переносяться до складу первинного ключа дочірньої сутності. Ця операція доповнення атрибутів дочірньої сутності при створенні зв'язку називається **міграцією атрибутів**. У дочірньої сутності нові атрибути позначаються як зовнішні ключі – **FK**.

При встановленні неідентифікуючого зв'язку дочірня сутність залишається незалежною, а атрибути первинного ключа батьківської сутності мігрують до складу неключових компонентів дочірньої. Неідентифікуючий зв'язок служить для зв'язку незалежних сутностей. Ідентифікуючий зв'язок зображується на діаграмі суцільною лінією з жирною крапкою на дочірньому кінці зв'язку, неідентифікуючий – пунктирною.


Для неідентифікуючого зв'язку можна вказати обов'язковість (**Nulls** в закладці **General** діалогу **Relationship Editor**). У разі обов'язкової зв'язку (**No Nulls**) при генерації схеми бази даних атрибут зовнішнього ключа отримає ознаку **NOT NULL**, незважаючи на те, що зовнішній ключ не увійде до складу первинного ключа дочірньої сутності. У разі необов'язкового зв'язку (**Nulls Allowed**) зовнішній ключ може приймати значення **NULL**. Необов'язковий неідентифікуючий зв'язок позначається прозорим ромбом з боку батьківської сутності

**Ім'я зв'язку (Verb Phrase)** – фраза, яка характеризує відношення між батьківською і дочірньою сутностями. Для зв'язку «один-до-багатьох» ідентифікуючого або неідентифікуючого досить вказати ім'я, яке характеризує відношення батьківської до дочірньої сутності (**Parent-to-Child**). Для зв'язку «багато-до-багатьох» слід вказувати імена як **Parent-to-Child**, так і **Child-to-Parent**. Для відображення імені слід в контекстному меню, яке з'являється, якщо натиснути правою кнопкою миші в будь-якому

місці діаграми, не зайнятого об'єктами моделі, обрати пункт ***Relationship Display*** і потім включити опцію ***Verb Phrase***.

Ім'я ролі або функціональне ім'я (***Rolename***) – це синонім атрибуту зовнішнього ключа, що показує, яку роль відіграє атрибут в дочірній сутності. Задати ім'я ролі можна в закладці ***Rolename / RI Actions*** діалогу ***Relationship Editor***.

**Створення ключів.** Кожен екземпляр сутності повинен бути унікальний та відрізнятися від інших атрибутів.

**Первинний ключ (*primary key*)** – це атрибут або група атрибутів, які однозначно ідентифікують екземпляр сутності. Атрибути первинного ключа на діаграмі не вимагають спеціального позначення – це ті атрибути, які перебувають у списку атрибутів вище горизонтальної лінії. При внесенні нового атрибуту в діалозі ***Attribute Editor*** для того, щоб зробити його атрибутом первинного ключа, потрібно включити прапорець ***Primary Key*** в нижній частині закладки ***General***. На діаграмі ключовий атрибут можна внести до складу первинного ключа, скориставшись режимом перенесення атрибутів (кнопка  на панелі інструментів).

В одній сутності може виявитися кілька атрибутів або наборів атрибутів, що претендують на роль первинного ключа. Такі претенденти називаються **потенційними ключами (*candidate key*)**.

Ключі можуть бути **складними**, тобто містити кілька атрибутів. Складні первинні ключі не вимагають спеціального позначення – це список атрибутів вище горизонтальної лінії. При виборі первинного ключа перевага повинна віддаватися більш простим ключам, тобто ключам, які містять меншу кількість атрибутів.

Більшість сутностей мають тільки один потенційний ключ. Такий ключ стає первинним. Деякі сутності можуть мати більше одного можливого ключа. Тоді один з них стає первинним, а решта – альтернативними ключами.



**Альтернативний ключ (*Alternative Key*)** – це потенційний ключ, що не став первинним.

Кожному ключу відповідає індекс, ім'я якому також присвоюється автоматично. Імена ключа та індексу при бажанні можна змінити вручну.

На діаграмі атрибути альтернативних ключів позначаються як (*Akn.m.*), де *n* – порядковий номер ключа, *m* – порядковий номер атрибуту в ключі. Коли альтернативний ключ містить кілька атрибутів, (*Akn.m.*) ставиться після кожного.

**Зовнішні ключі (*Foreign Key*)** створюються автоматично, коли зв'язок з'єднує сутності: зв'язки утворюють посилення на атрибути первинного ключа в дочірній сутності і ці атрибути утворюють зовнішній ключ в дочірній сутності (міграція ключа). Атрибути зовнішнього ключа позначаються символом **FK** після свого імені.

Залежна сутність може мати один і той самий ключ з декількох батьківських сутностей. Сутність може також отримати один і той самий зовнішній ключ кілька разів від одного і того ж батька через кілька різних зв'язків. Коли **ERwin** виявляє одну з цих подій, він розпізнає, що два атрибути однакові, і розміщує атрибути зовнішнього ключа в залежній сутності тільки один раз. Це комбінування або об'єднання ідентичних атрибутів називається **уніфікацією**.

Є випадки, коли уніфікація небажана. Наприклад, коли два атрибути мають однакові імена, але насправді вони відрізняються за змістом, і необхідно, щоб ця відмінність відбивалася в діаграмі. В цьому випадку необхідно використовувати імена ролей зовнішнього ключа.

Зв'язки на діаграмі зображуються лініями, що йдуть від однієї сутності (таблиці) до іншої. Кожному зв'язку присвоюється унікальне ім'я. З'єднані таблиці поділяють на батьківські та дочірні. Батьківські таблиці відображаються прямокутниками з прямими кутами, дочірні – із заокругленими.

Після вказівки всім атрибутам формату даних, необхідно створену логічну модель перетворити в фізичну. Для цього необхідно в *Tools* обрати *Derive New Model*, де як *Target Databases* вибрати *ODBC / Generic* (для використання в *СУБД MySQL*) (Рисунок 11). Наша модель (Рисунок 6) буде перетворена до вигляду (Рисунок 12).

Далі, обравши в меню *Tools / Forward Engineer / Shema Generation* і задаючи необхідні настройки, отримаємо в меню *Preview* код на мові *SQL* для реалізації схеми бази даних в *СУБД MySQL* (Рисунок 13).

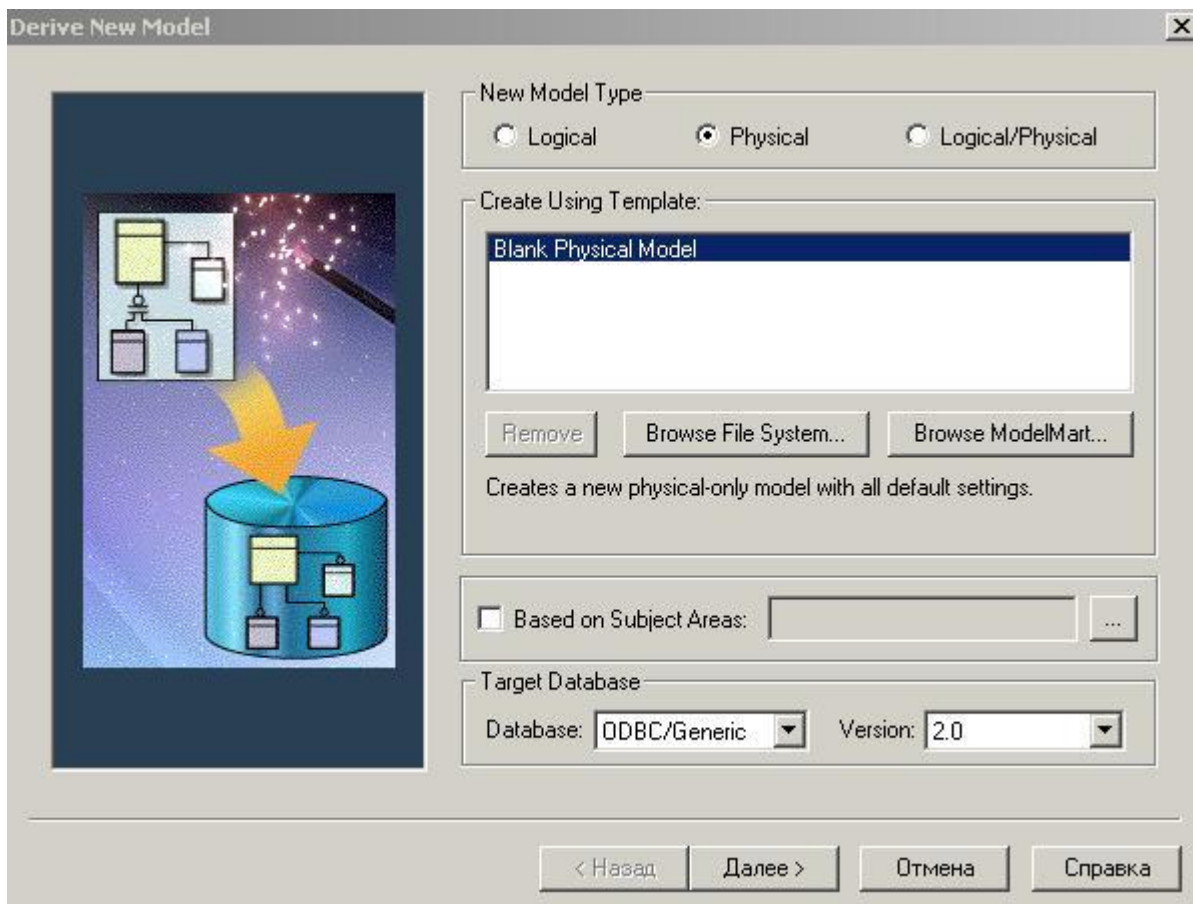


Рисунок 11. Перетворення логічної моделі в фізичну

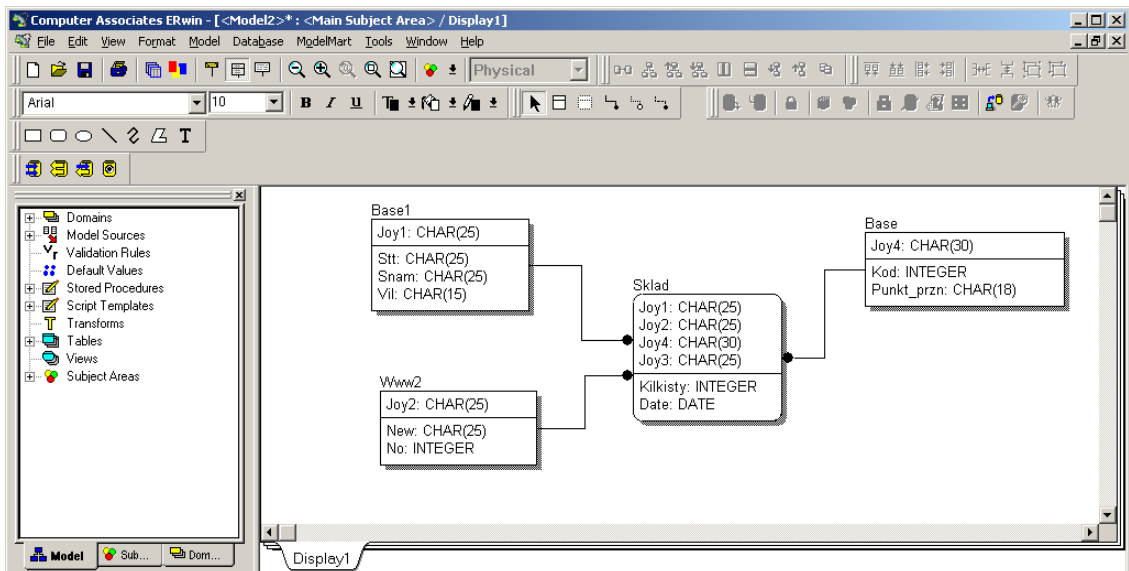


Рисунок 12. Фізична модель з визначенням формату даних

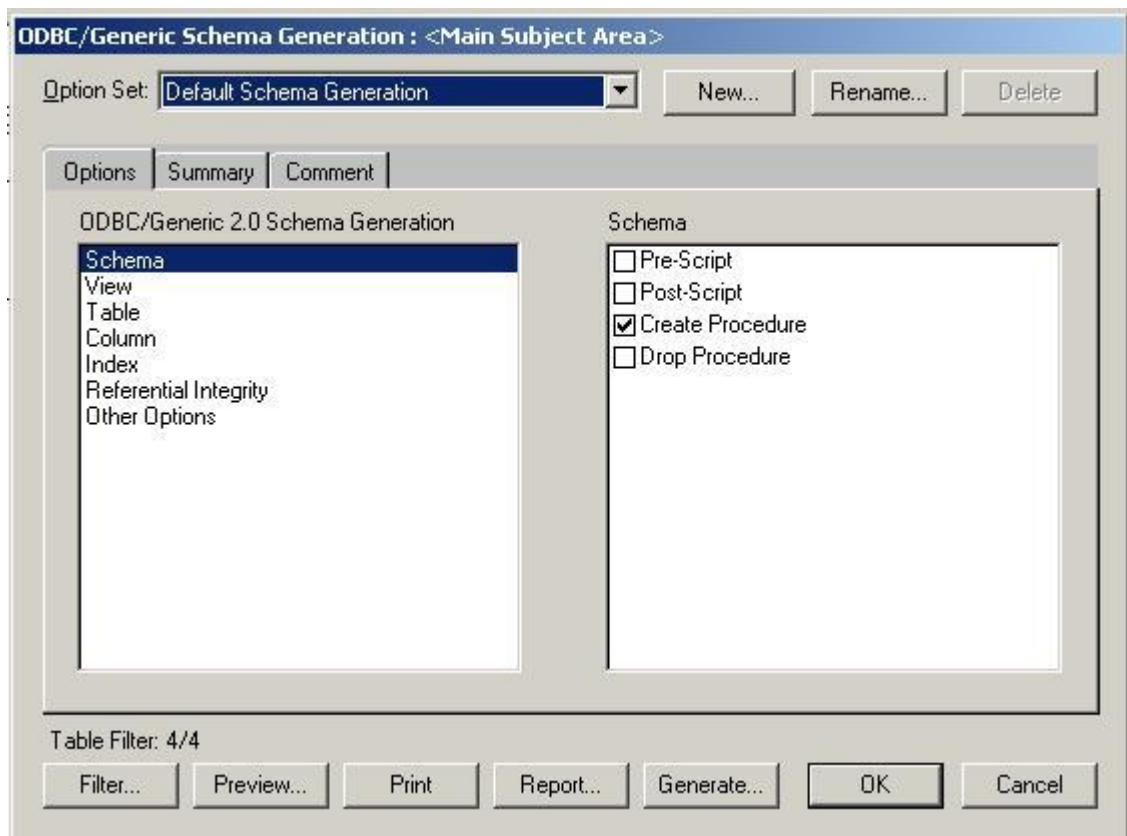


Рисунок 13. Генерація коду *SQL*

### Порядок виконання роботи

1. Запустити візуальне середовище проектування інформаційних систем *ERwin*. Перед тим, як почати роботу, потрібно відкрити навчальну модель, але спочатку необхідно закрити всі раніше відкриті моделі, скориставшись командою *Close* з меню *File*.

2. В меню **File** обрати опцію **Open**. З'явиться діалогове вікно відкриття моделі (Рисунок 14). Вказати шлях та ім'я моделі: **C:\Program Files\Computer Associates\ERWin 4.0\Samplies\Standard\Emovies** і натиснути **OK**.

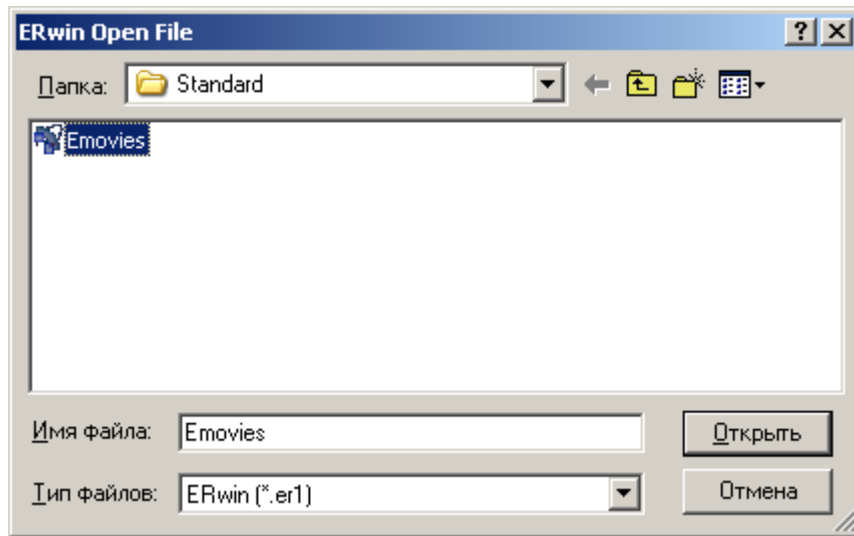


Рисунок 14. Діалогове вікно для відкриття файлу

3. Повторно натиснути **OK**, коли з'явиться діалогове вікно з повідомленням про те, що цей файл має атрибут «**read only**» (тільки для читання). У вікні редагування має відкритися модель **Emovies.er1**. Перемикач на панелі інструментів має бути встановлений в положення **Logical**. Зберегти модель під новим ім'ям (Рисунок 15).

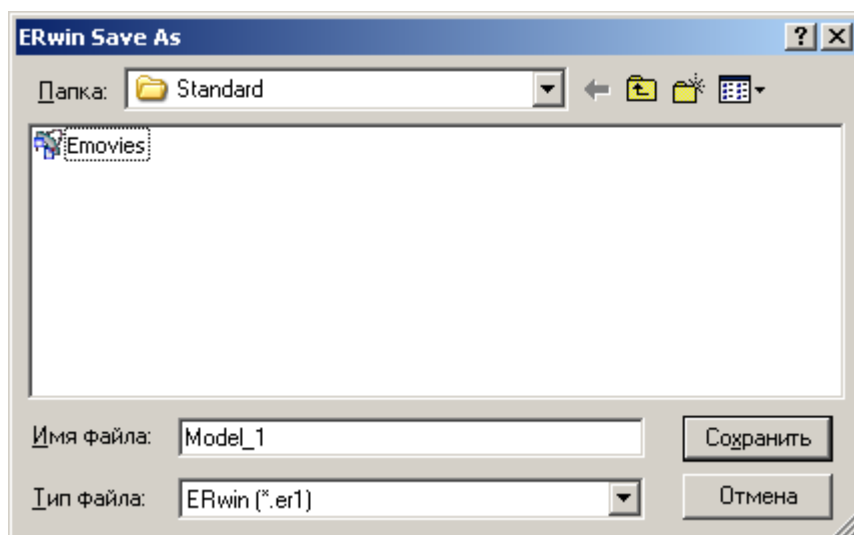
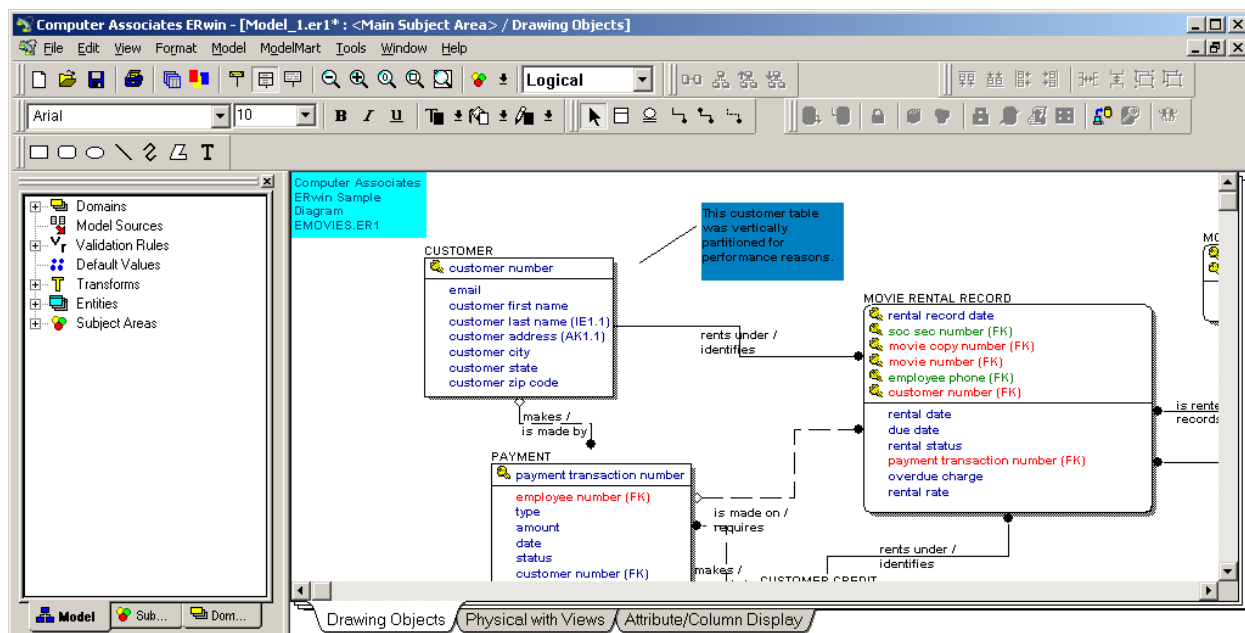


Рисунок 15. Вікно збереження файлу під новим ім'ям **Model\_1**

4. Обрати команди головного меню **View/Toolbars**. Звернути увагу на те, як реагує на цю дію інструментальна панель (має з'явитися меню, яке

випадає). Деякі опції в ньому є активними. Змінити перелік активних опцій і подивитися, як це впливає на інтерфейс середовища **ERwin**.

5. Переключитися по чергово спочатку на фізичну модель, а потім на логічну (опції **Physical** та **Logical** інструментальної панелі **ERwin**) і звернути увагу на зміни у вигляді моделі. Логічні назви (імена) замінюються фізичними назвами (іменами) і браузер незалежних атрибутів переключачється на браузер незалежних стовпців (якщо браузер не відображається, натиснути **Ctrl+B**) (Рисунок 16). Зберегти результати своєї роботи.



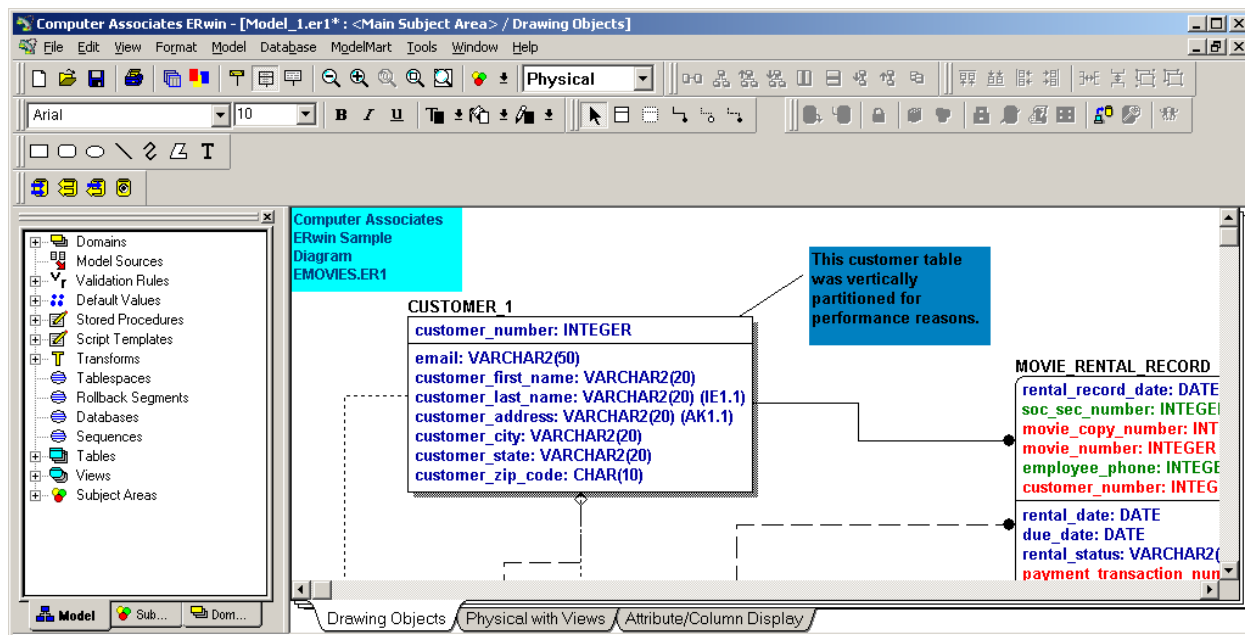


Рисунок 16. Зміни у вікні редагування

6. Закрити інструментальне середовище створення моделі процесів. Для цього, використовуючи головне меню **ERwin**, виконати **File/Exit**. Для закриття **ERwin** без збереження результатів моделювання, скористатися опцією **Close without saving** і натиснути **OK** (Рисунок 17).

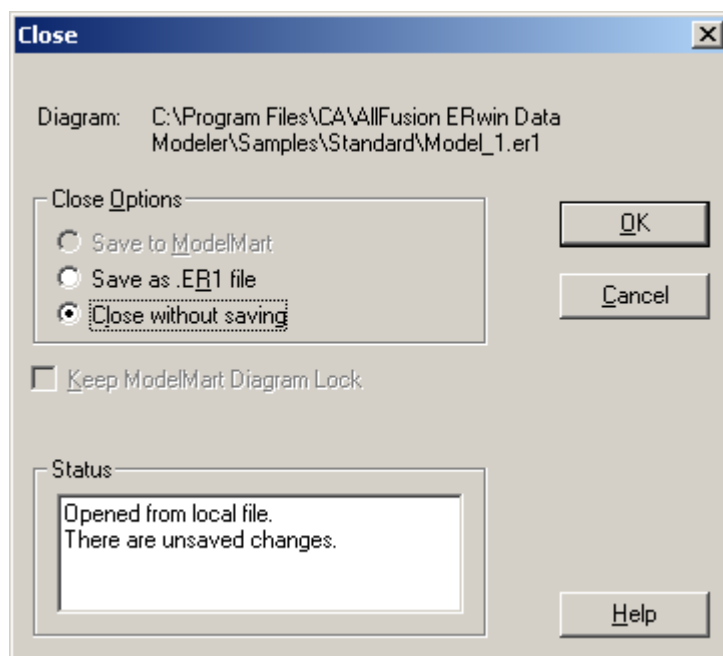


Рисунок 17. Діалогове вікно **Close**.

Згідно індивідуального завдання до Лабораторної роботи №1 виконати наступне:

1. Побудувати діаграму із заданими сутностями (пряме моделювання) для заданої предметної області.
2. Задати атрибути для кожної визначеної сутності. При визначенні атрибутів, використати домени.
3. Ввести зв'язки між сутностями. Дати зв'язкам унікальні імена.
4. Побудувати концептуальну та фізичну моделі бази даних в **СУБД MySQL**.
4. Використовуючи **СУБД MySQL**, виконати пряму генерацію бази даних для проекрованої інформаційної.
5. Оформити короткий звіт про виконану роботу, внести в нього основні теоретичні відомості.

### Оформлення звіту:

1. Титульний лист.
2. Мета роботи.
3. Порядок виконання.
4. Представлення основних результатів по виконаній лабораторній роботі.
5. Висновки.

### Контрольні питання

1. Що таке **ERwin** та для чого його використовують?
2. Що таке фізична та логічна моделі даних?
3. У чому полягає відмінність логічного та фізичного рівнів представлення моделей даних за допомогою **ERwin**?
4. У чому відмінність між моделями даних, представлених у формі діаграми «сутність-зв'язок», на основі ключів і у вигляді повної атрибутивної моделі?
5. Які основні компоненти містять моделі даних, які представлені за методологією **IDEFIX**?
6. Які властивості та загальна характеристика системи меню **ERwin**?
7. Дайте характеристику меню: **File, Edit, View, Format, Model, Model Mart, Tools, Window, Help**.
8. Яке призначення кнопок стандартної панелі інструментів?

9. Які функції палітри інструментів?
10. Що таке сутність?
11. Домен, його призначення.
12. Що таке атрибут? Як створюється атрибут в **ERwin**?
13. Тип зв'язку (ідентифікуючого / неідентифікуючого). Потужність зв'язку.
14. Що таке ключ (первинний, альтернативний, зовнішній)? Його призначення.



Таблиця 1.1. Зміст пунктів головного меню

Назва пункту меню	Призначення пункту меню
File	Відкриття, закриття, збереження моделі
Edit	Редагування моделі
View	Вигляд моделі
Format	Форматування моделі
Model	Властивості моделі
Model Mart	Сховище моделей
Tools	Інструментарій
Window	Вікна
Help	Довідка

Таблиця 1.2. Зміст меню *File*

Назва підпункту меню	Призначення підпункту меню
New	Викликає вікно ERWin Template Selection, в якому можна обрати шаблон, щоби використати як основу для створення нової моделі.
Open	Показує вікно ERWin Open File, в якому можна обрати існуючу модель, щоби відкрити її.
Close	Викликає вікно Close dialog, в якому є можливість зберегти або відмовитися від збереження змін, які були внесені в модель.
Save	Викликає вікно Save As, в якому є можливість зберегти модель під старим або новим ім'ям (команда доступна тільки після того, як модель буде хоча б один раз збережена).

Save As	Викликає вікно Save As, в якому є можливість зберегти модель під новим ім'ям(команда доступна завжди).
Save As New Model	Викликає вікно Save As, в якому є можливість зберегти модель під новим ідентифікаційним номером (команда доступна завжди).
Import	Імпорт
BPwin	Імпорт моделі із BPWin в ERWin або експорт моделі BPWin в ERWin.
Desiner2000	Імпорт моделі із Oracle Designer/2000 в ERWin або експорт моделі ERWin і в сховище Oracle Designer/2000.
Export	Експорт.
BPwin	Експорт моделі ERWin в BPWin
Desiner2000	Експорт моделі ERWin в сховище Oracle Designer/2000.
Print	Викликає вікно Print dialog, в якому можна обрати виведення на принтер активної моделі .
Print Setup	Викликає вікно Print Setup, в якому можна обрати установки принтера.
Recent File	Список з чотирьох останніх моделей ERWin, які доступні для відкриття.
Exit	Вихід з ERWin.

Таблиця 1.3. Зміст меню *Edit*

Назва підпункту меню	Призначення підпункту меню
Cut	Вирізання виділених об'єктів із моделі
Copy	Копіювання виділених об'єктів моделі
Paste	Вставка вмісту буферу обміну в модель
Select All	Виділення всіх об'єктів активної (поточної) моделі
Go To	Перехід до вказаного об'єкту моделі

Таблиця 1.4. Зміст меню *View*

Назва підпункту меню	Призначення підпункту меню
Redraw Diagram	Перемалювати діаграму моделі
Toolbars	Відображення або укриття панелі інструментів
Model Explorer	Відображення або укриття провідника моделі
Stored Display Tabs	Відображення або укриття вкладеної моделі
Status Bar	Відображення або укриття рядку стану моделі
Zoom	Керування масштабом відображення моделі

Таблиця 1.5. Зміст меню **Format**

Назва підпункту меню	Призначення підпункту меню
Display Level	Рівень відображення моделі
Entity Display	Рівень відображення сутностей моделі
Relationship Display	Рівень відображення відношень між сутностями моделей
Stored Display Tabs	Виклик діалогового вікна Stored Display для завдання даних про автора моделі та установки параметрів відображення моделі
Preferences	Виклик діалогового вікна Format Preferences для завдання привілейних режимів відображення сутностей та інших параметрів моделі
Default Fonts & Colors	Виклик діалогового вікна Default Fonts & Colors для завдання параметрів шрифту та його кольору при відображенні різних параметрів моделі
Align or Space Evenly	Вирівнювання або переміщення виділених об'єктів моделі в межах вікна моделі
Show Shadows	Відображення або укриття тіней для сутності моделі
Show Page Grid	Відображення або укриття координатної сітки для спрощення розміщення сутностей моделі

Таблиця 1.6. Зміст меню *Model*

Назва підпункту меню	Призначення підпункту меню
Subject Areas	Виклик діалогового вікна Subject Areas (підмножина моделі) для вибору тематично загальних сутностей моделі
Entities	Виклик діалогового вікна Entities для завдання параметрів виділеної сутності моделі
Attributes	Виклик діалогового вікна Attributes для завдання параметрів атрибутів виділеної сутності моделі
Relationships	Виклик діалогового вікна Relationships для завдання параметрів відношення виділеного зв'язку між сутностями моделі
Key Groups	Виклик діалогового вікна Key Groups для завдання параметрів ключових атрибутів виділеної сутності моделі
Domain Dictionary	Виклик діалогового вікна Domain Dictionary для завдання параметрів доменів сутностей моделі
Validation Rules	Виклик діалогового вікна Validation Rules для завдання параметрів перевірки коректності завдання сутностей моделі
Default Values	Виклик діалогового вікна Default Values для завдання параметрів моделі за замовченням
UDP Dictionary	Виклик діалогового вікна User Defined Property Dictionary для завдання словника параметрів моделі, які визначаються користувачем
Model Sources	Виклик діалогового вікна Model Sources для завдання джерел моделі
Model Properties	Виклик діалогового вікна Model Properties для завдання

	даних про модель
Logical Model	Відображення логічної моделі даних
Physical Model	Відображення фізичної моделі даних

Таблиця 1.7. Зміст меню *ModelMart*

Назва підпункту меню	Призначення підпункту меню
Open	Відкрити сховище моделей
Close	Закрити сховище моделей
Save	Зберегти модель в сховищі моделей
Save As	Зберегти модель в сховищі моделей під вказаним ім'ям
Lock	Закрити модель, яка збережена в сховищі моделей, для редагування
Merge	Розбити модель
Version	Версія моделі
Change Control	Змінити параметри контролю стану моделі
Refresh	Перемалювати модель
Library	Виклик бібліотеки моделей
Subject Areas	Робота з підмножиною моделі
ModelMart Manager	Виклик менеджера сховища моделей
Connection	Завдання параметрів з'єднання із сховищем моделей
Session	Параметри сесії з'єднання із сховищем моделей
Security	Завдання параметрів захисту моделі
BP/ER Synchronizer	Завдання параметрів синхронізації моделей в середовищах BPWin та ERWin

Таблиця 1.8. Зміст меню *Tools*

Назва підпункту меню	Призначення підпункту меню
Forward Engineer/Schema Generation	Виклик діалогового вікна <Database> Server Schema Generation для вибору параметрів генерування системного каталогу для вказаного цільового серверу СУБД та (або) збереження сценарію генерації системного каталогу в форматі сценарію DDL або ASCII текстовому файлі (тільки для фізичної моделі)
Reverse Engineer	Виклик діалогового вікна Reverse Engineer – Set Options для завдання параметрів генерації ERWin моделі на основі існуючої SQL DDL або DL сценарію або системного каталогу СУБД
Add Model Source	Виклик діалогового вікна майстра Source Model, яке дозволяє завдати відомості про джерела (зручно при великій кількості джерел)
Sync with Model Source	Виклик діалогового вікна Sync with Model Source, яке дозволяє синхронізувати існуючу модель ERWin з джерелами моделі при груповій роботі над проектом
Derive New Model	Виклик діалогового вікна Derive New Model, яке дозволяє створити новий варіант моделі на основі вже існуючої ERWin моделі
Split L/P Model	Розбиття логічної та фізичної моделі
Report Builder	Виклик діалогового вікна майстра генерування звіту про модель
Data Browser	Виклик редактору звітів про модель
Volumetric	Виклик діалогового вікна Volumetric для визначення кількісних характеристик параметрів моделі (розмір таблиць та інше)

Names	Виклик діалогового вікна Model Naming Options для редагування параметрів імен, що використовуються в моделі (максимальна кількість символів та інше)
Datatypes	Виклик діалогового вікна Model Datatype Options для використання стандартних типів файлів даних в моделі
Add Ins	Виклик діалогового вікна Add-In Manager, яке дозволяє підключити до ERWin додаткове програмне забезпечення, наприклад, ERWin Examiner та інше.

Таблиця 1.9. Зміст меню **Window**

Назва підпункту меню	Призначення підпункту меню
Cascade	Вікна моделі в робочому просторі ERWin орієнтовані каскадом.
Tile Horizontal	Вікна моделі в робочому просторі ERWin орієнтовані з горизонтальним розбиттям.
Tile Vertical	Вікна моделі в робочому просторі ERWin орієнтовані з вертикальним розбиттям.



Таблиця 1.10. Зміст меню *Help*

Назва підпункту меню	Призначення підпункту меню
Help Topics	Виклик вікна ERWin Online довідкової системи.
Tutorial	Виклик діалогового вікна підручника по роботі з ERWin.
What's New	Виклик огляду новинок, що включені в поточну версію ERWin порівняно з попередньою версією.
How to Use Help	Виклик діалогового вікна майстра How to Use Help, в якому можна знайти рекомендації по використанню довідкової системи.
About ERWin	Виклик діалогового вікна довідки про поточну версію ERWin.

## Лабораторна робота №4

**Тема:** Система управління базами даних *MySQL*. Проектування бази даних і забезпечення прав доступу до неї.

**Мета роботи:** Засвоїти основні засоби для проектування баз даних в середовищі *MySQL*. Вивчити команди мови *SQL* для роботи з базами даних *MySQL*.

### Теоретичні відомості

**Особливості бази даних *MySQL*.** База даних являє собою структуровану сукупність даних. Для запису, вибірки й обробки даних, що зберігаються в комп'ютерній базі даних, необхідна система керування базою даних, якою і є програмне забезпечення *MySQL*. Оскільки комп'ютери чудово справляються з обробкою великих об'ємів даних, керування базами даних відіграє центральну роль в обчисленнях. Реалізовано таке керування може бути по-різному – як у вигляді окремих утиліт, так й у вигляді коду, що входить до складу інших додатків.

*MySQL* – це система управління реляційними базами даних (СУБД). У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається вииграш у швидкості й гнучкості. Таблиці зв'язуються між собою за допомогою залежностей, завдяки чому забезпечується можливість поєднувати при виконанні запиту дані з декількох таблиць. *SQL*, як частину системи *MySQL*, можна охарактеризувати як мову структурованих запитів плюс як найпоширенішу стандартну мову для доступу до баз даних.

Припустимо, що необхідно оформити дану адресну книгу (Рисунок 1) у вигляді таблиці з рядками і колонками. Кожен рядок (називається також записом) буде відповідати певній особистості, а кожна колонка (поле) буде містити значення для кожного типу даних: прізвище, ім'я, по-батькові; телефонні номери і адреси, що представляють у кожному рядку.

Адресна книга могла б виглядати в такий спосіб:

П І П	Телефон	Адреса
Іванко І.І.	(044)365-8775	03056 Київ

Іванов П.П. (055)874-3553 02056 Севастополь

Іванчук Б.Б. (0 66)976-3665 01056 Львів

Рисунок 1. Адресна книга

Даний двовимірний масив даних є основою реляційної бази даних.

Однак, реляційні бази даних рідко складаються з однієї таблиці. Часто для баз даних малого розміру однієї таблиці достатньо, але якщо кількість записів починає сягати сотень тисяч, то для мінімізації ресурсів необхідних для збереження всіх даних, для зменшення кількості помилок в записах, завжди доцільно створювати кілька таблиць із зовнішніми зв'язками.

**Встановлення і початок роботи з базою даних *MySQL*.** База даних *MySQL* не є комерційним продуктом і розповсюджується безкоштовно. Свіжу версію дистрибутиву *MySQL* для різних операційних систем завжди можна завантажити з сайту розробника [www.mysql.org](http://www.mysql.org).

СУБД *MySQL* містить, окрім сервера баз даних, потужну клієнтську частину, яка дозволяє успішно адмініструвати бази даних, що знаходяться на сервері. Реалізації клієнтської частини розвивається у двох напрямках. Перший і більш потужний засіб – це формування клієнтської частини програмно за допомогою стандартних функцій СУБД. Другий – це консольна клієнтська програма *mysql*, яка містить підтримку усіх можливостей клієнтського доступу, який надає СУБД *MySQL*.

Дана програма надається разом із пакетом встановлення сервера СУБД *MySQL* і називається *MySQL Command Line Client* (Рисунок 2).

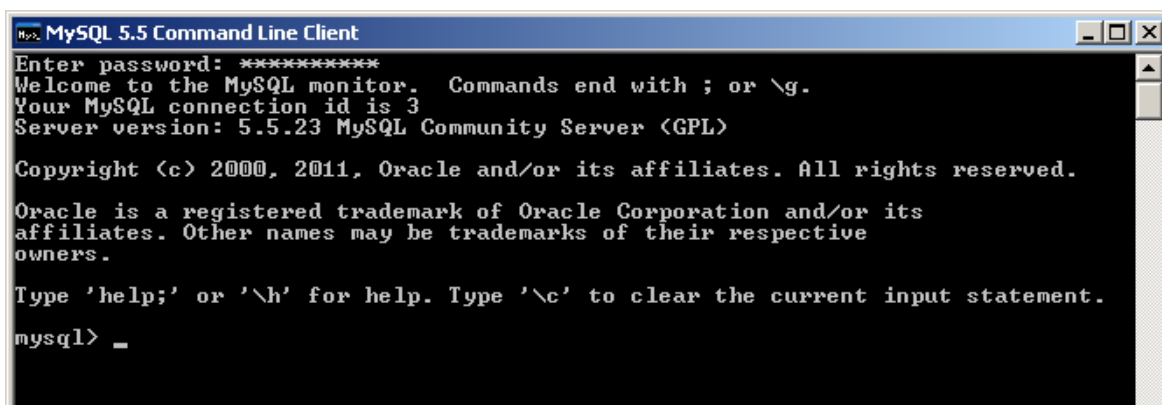


Рисунок 2. Клієнтська програма *MySQL Command Line Client*

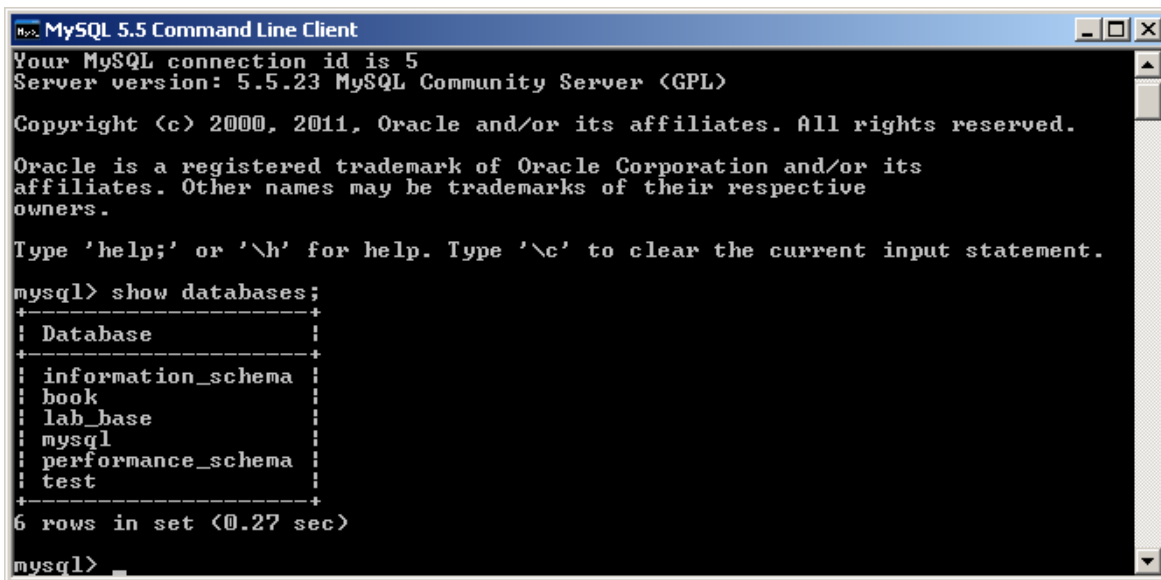
Дана програма, окрім власне функціоналу *MySQL*, включає ряд внутрішніх команд для більш зручної роботи, список команд і їх опис можна викликати командою *help* (*\h*).

За замовченням, на сервері баз даних уже є дві створені бази даних, які можна переглянути командою *show databases*, яку будемо використовувати і надалі для перегляду наявних баз даних (Рисунок 3).

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
+-----+
2 rows in set (0.00 sec)
```

Рисунок 3. Команда *show databases* для перегляду баз даних

Після створення потрібних баз даних інформація про існуючі бази зміниться (Рисунок 4).



```
MySQL 5.5 Command Line Client
Your MySQL connection id is 5
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| book      |
| lab_base  |
| mysql     |
| performance_schema |
| test      |
+-----+
6 rows in set (0.27 sec)

mysql>
```

Рисунок 4. Перегляд існуючих баз даних

Для відображення стовпців потрібної таблиці можна скористатися командами *show columns from* *<ім'я таблиці>*; або *describe* *< ім'я таблиці>* (Рисунок 5).

```

mysql> use book;
Database changed
mysql> show columns from book;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11) | NO | PRI | NULL | auto_increment |
| autor | char(20) | NO |     | goffman |  |
| nazva | char(25) | YES |     | Delphi6 |  |
| rik_vidannya | int(11) | YES |     | 2008 |  |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> describe book;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11) | NO | PRI | NULL | auto_increment |
| autor | char(20) | NO |     | goffman |  |
| nazva | char(25) | YES |     | Delphi6 |  |
| rik_vidannya | int(11) | YES |     | 2008 |  |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)

mysql>

```

Рисунок 5. Перегляд полів таблиці *book*

Для перегляду індексів використовують команду *show index from* *<ім'я таблиці>*.

**Основні команди керування базою даних *MySQL* за допомогою мови *SQL*.** Керування базою даних відбувається в діалоговому режимі: введення команди – повернення повідомлення або про її успішне виконання або про помилки, наприклад, синтаксису мови *SQL*.

В додатку А наведені основні команди і ключові слова для керування базою даних *MySQL* з прикладами.

Для створення нової бази даних використовують команду *create database [if not exist] <ім'я бази даних>*, а для видалення – *drop database <db\_name>* (Рисунок 6). Команду *use <db\_name>* використовують для переходу до потрібної бази даних та її відкриття.

```

MySQL 5.5 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database firstDB;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| book |
| firstdb |
| lab_base |
| mysql |
| performance_schema |
| test |
+-----+
7 rows in set (0.00 sec)

mysql> drop database firstDB;
Query OK, 0 rows affected (0.30 sec)

mysql> use book;
Database changed
mysql> show tables;
+-----+
| Tables_in_book |
+-----+
| biblioteka |
| book |
+-----+
2 rows in set (0.19 sec)

mysql> select * from book;
+----+-----+-----+-----+
| id | autor | nazva | rik_vidannya |
+----+-----+-----+-----+
| 1 | ddd | ggg | 1995 |
| 2 | rrr | hhh | 2001 |
| 3 | goffman | Delphi | 2004 |
+----+-----+-----+-----+
3 rows in set (0.09 sec)

mysql> _

```

Рисунок 6. Робота з базами даних в діалоговому режимі

СУБД *MySQL* дозволяє переглянути інформацію про поля вже створеної таблиці. Це можна здійснити командою *describe <db\_name>* (Рисунок 7).

```

MySQL 5.5 Command Line Client
Your MySQL connection id is 8
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

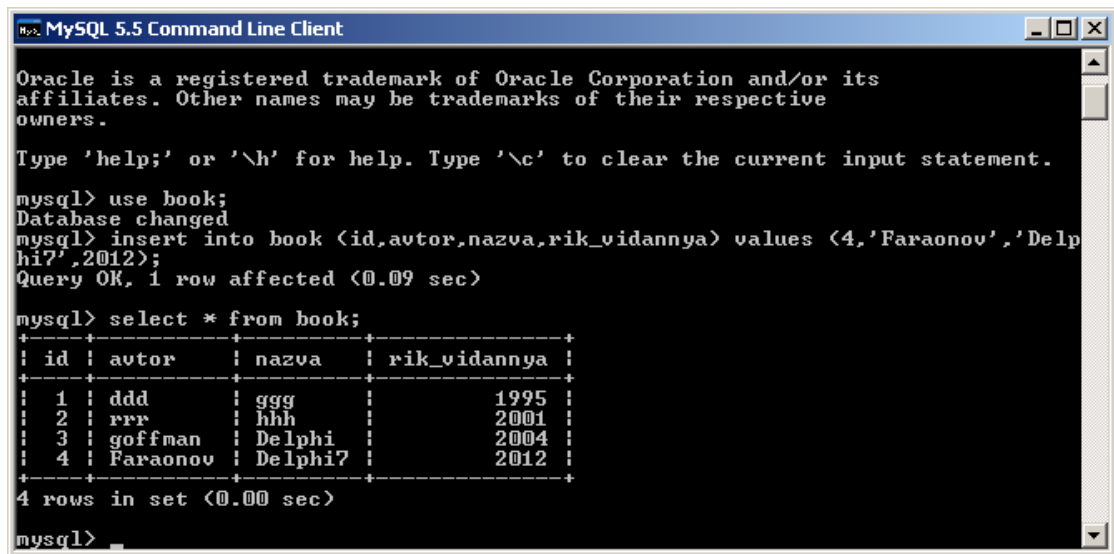
mysql> use book;
Database changed
mysql> describe book;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment |
| autor | char(20) | NO | | goffman | |
| nazva | char(25) | YES | | Delphi6 | |
| rik_vidannya | int(11) | YES | | 2008 | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.09 sec)

mysql> _

```

Рисунок 7. Перегляд даних про поля таблиці

Внесення записів в таблицю здійснюється командою *insert into* (Рисунок 8).



```
MySQL 5.5 Command Line Client

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use book;
Database changed
mysql> insert into book (id,avtor,nazva,rik_vidannya) values (4,'Faraonov','Delphi7',2012);
Query OK, 1 row affected (0.09 sec)

mysql> select * from book;
+----+-----+-----+-----+
| id | avtor  | nazva  | rik_vidannya |
+----+-----+-----+-----+
| 1  | ddd    | ggg    | 1995         |
| 2  | rrr    | hhh    | 2001         |
| 3  | goffman | Delphi | 2004         |
| 4  | Faraonov | Delphi7 | 2012         |
+----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Рисунок 8. Внесення нового запису в базу даних *book*

**Безпека даних в MySQL.** Для забезпечення захисту даних в базі можна встановити пароль для користувача *root*. Для цього, після запуску утиліти *mysql*, за допомогою оператора *set password* необхідно присвоїти пароль для облікового запису користувача *root*:

```
mysql> set password for 'root'@'localhost'=password ('<пароль>');
```

Надалі необхідно виконати наступну команду

```
mysql> set password for 'root'@'%'=password ('<пароль>');
```

і за допомогою команди

```
mysql> flush privileges;
```

перезавантажити тільки що внесені зміни. Для перевірки отриманих результатів необхідно виконати наступні дії:

```
mysql> use mysql;
```

```
mysql> select host, user, password from user;
```

Після виходу з утиліти та наступного її запуску з'явиться запрошення на введення паролю. Цей пароль надалі зберігається в таблиці *user* в зашифрованому вигляді і має вводитися користувачем при кожному підключенні до *mysql*.

Для забезпечення безпеки зберігання даних і контролю прав доступу до них у СУБД *MySQL* існує спеціальна база даних *mysql*, у якій є п'ять таблиць: *user*, *host*, *db*, *tables\_priv* й *columns\_priv*. Таблиця *user* призначена для визначення, чи може користувач підключатися до сервера *MySQL* і чи володіє він привілеями адміністратора. У ній зберігається ім'я хоста, з якого можна підключатися, ім'я користувача й пароль, а також які привілеї він має. Таблиця *db* визначає, які користувачі до яких таблиць і з яких хостів можуть одержати доступ. Таблиця *host* доповнює таблицю *db*. Якщо користувач повинен підключатися до бази даних з декількох хостів, то в таблиці *db* для нього не буде зазначено жодного хоста – всі вони будуть перераховані в таблиці *host*. Таблиці *tables\_priv* й *columns\_priv* призначені для зберігання привілеїв на рівні таблиць і рівні стовпців. Вони працюють подібно таблиці *db*, але забезпечують привілею для таблиць у конкретній базі даних і стовпців у конкретній таблиці. Основних привілеїв десять, що дають право на виконання операцій *select*, *insert*, *update*, *delete*, *create*, *drop*, *grant*, *references*, *index* й *alter*. У таблиці *user* їх на 4 більше: *reload*, *shutdown*, *process* й *file*, які потрібні для адміністрування СУБД.

Для установки прав доступу застосовується команда ***grant***:

***grant select, insert, delete, update on {<ім'я таблиці>/ \*/ \*.\*/ <ім'я бази даних>.\*} to 'user'@'host' identified by 'password' ;***

Для оператора опції використовують наступним чином.

Глобальний рівень. Привілеї належать до сервера *MySQL* та всім його базам даних та таблиць. Використовують синтаксис *.\**. Також можна використовувати *\**, якщо на момент виконання оператора ***grant*** немає відкритих баз даних.

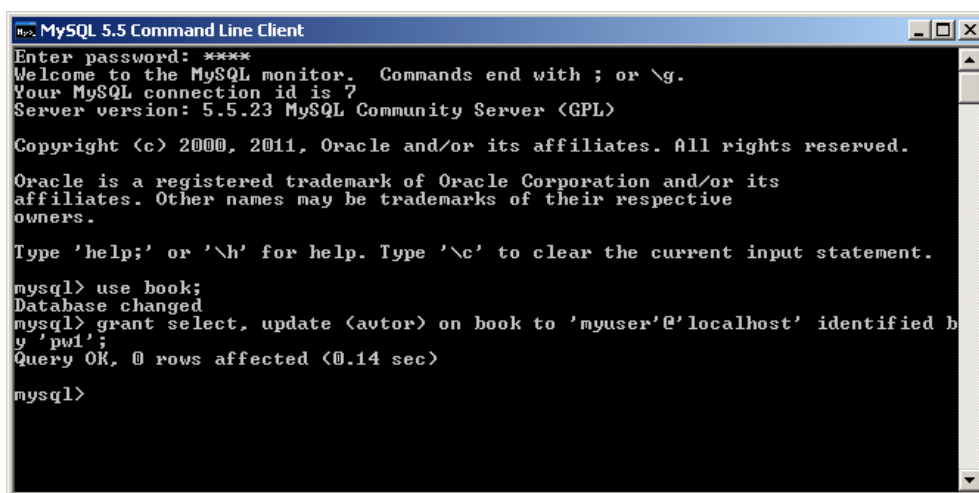
Рівень бази даних. Використовують синтаксис *<ім'я бази даних>.\**. Якщо дана база даних відкрита, то достатньо вказати *\**.

Рівень таблиці. Використовують синтаксис *<ім'я бази даних>.<ім'я таблиці>*.



Рівень стовпців. Разом з іменем таблиці в операторі **grant** необхідно вказати імена потрібних стовпців.

В наступному прикладі (Рисунок 9) показано, як додати в базу даних користувачів і призначити їм привілеї.



```
MySQL 5.5 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

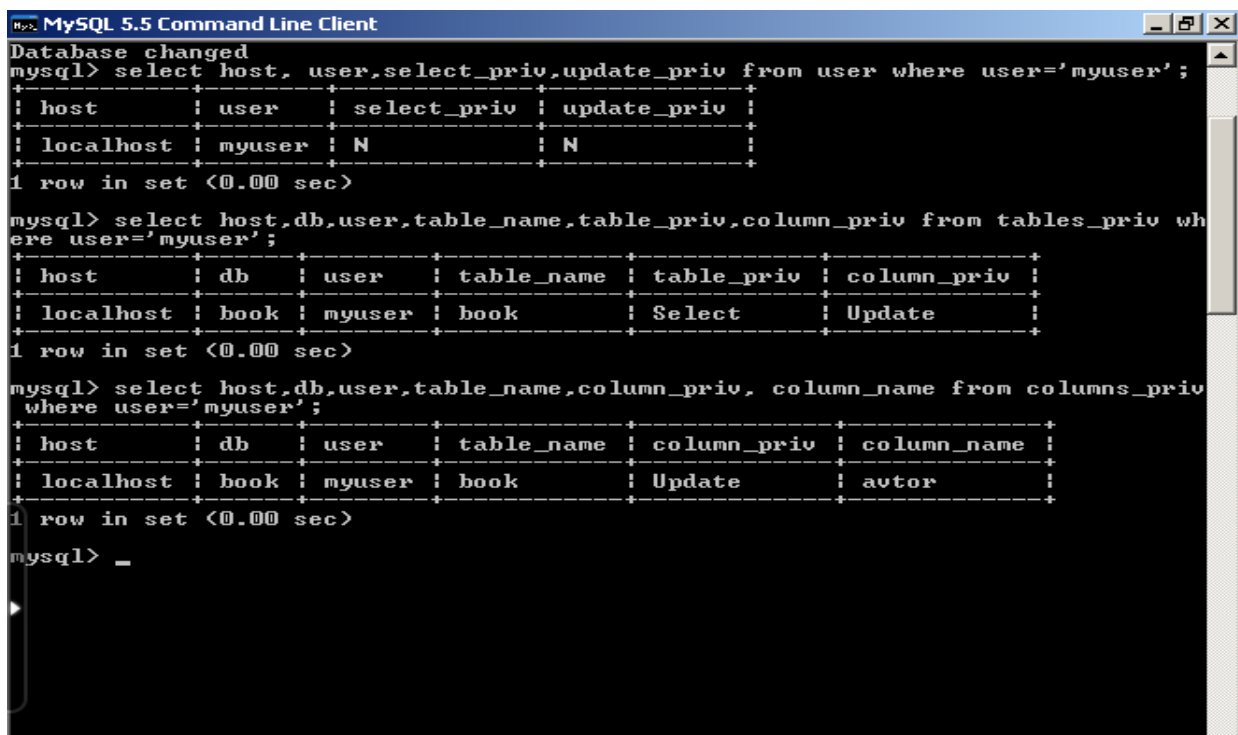
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use book;
Database changed
mysql> grant select, update (avtor) on book to 'myuser'@'localhost' identified by 'pw1';
Query OK, 0 rows affected (0.14 sec)

mysql>
```

Рисунок 9. Додавання в базу даних користувачів і призначення їм привілеїв

Після створення користувача **myuser**, він має бути внесений до таблиць **user**, **tables\_priv** і **columns\_priv**. Щоби це перевірити, треба вивести за допомогою **SQL**-операторів дані цих таблиць (Рисунок 10).



```
MySQL 5.5 Command Line Client
Database changed
mysql> select host, user, select_priv, update_priv from user where user='myuser';
+-----+-----+-----+-----+
| host      | user      | select_priv | update_priv |
+-----+-----+-----+-----+
| localhost | myuser    | N           | N           |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

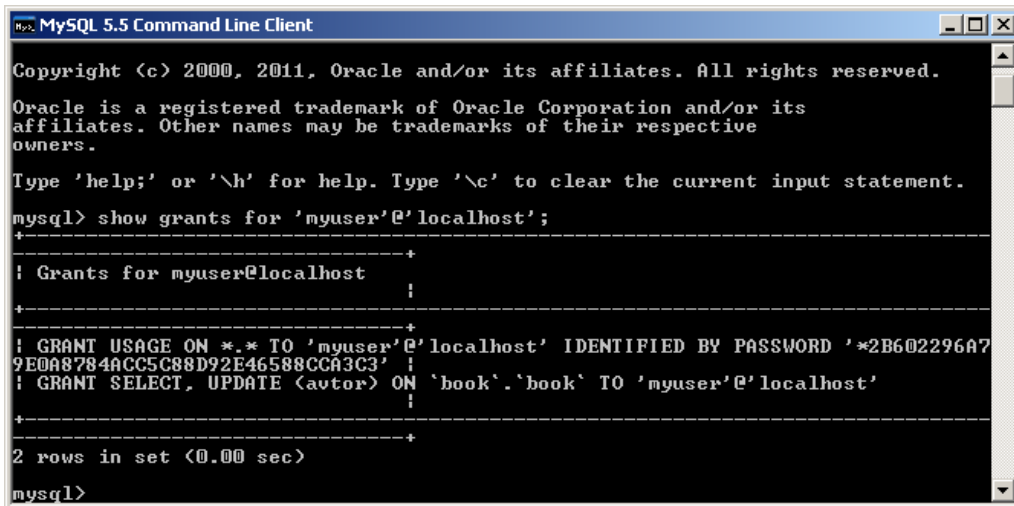
mysql> select host, db, user, table_name, table_priv, column_priv from tables_priv where user='myuser';
+-----+-----+-----+-----+-----+-----+
| host      | db      | user      | table_name | table_priv | column_priv |
+-----+-----+-----+-----+-----+-----+
| localhost | book    | myuser    | book       | Select     | Update      |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select host, db, user, table_name, column_priv, column_name from columns_priv where user='myuser';
+-----+-----+-----+-----+-----+-----+
| host      | db      | user      | table_name | column_priv | column_name |
+-----+-----+-----+-----+-----+-----+
| localhost | book    | myuser    | book       | Update      | avtor        |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

Рисунок 10. Вибірка з таблиць **user**, **tables\_priv** і **columns\_priv**

За допомогою оператора *show grants* можна відобразити на екрані привілеї, які призначені користувачеві *myuser* (Рисунок 11).



```
MySQL 5.5 Command Line Client
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show grants for 'myuser'@'localhost';
+-----+
| Grants for myuser@localhost |
+-----+
| GRANT USAGE ON *.* TO 'myuser'@'localhost' IDENTIFIED BY PASSWORD '*2B602296A79E0A8784ACC5C88D92E46588CCA3C3' |
| GRANT SELECT, UPDATE (avtor) ON `book`.`book` TO 'myuser'@'localhost' |
+-----+
2 rows in set (0.00 sec)
mysql>
```

Рисунок 11. Результат роботи оператора *show grants*

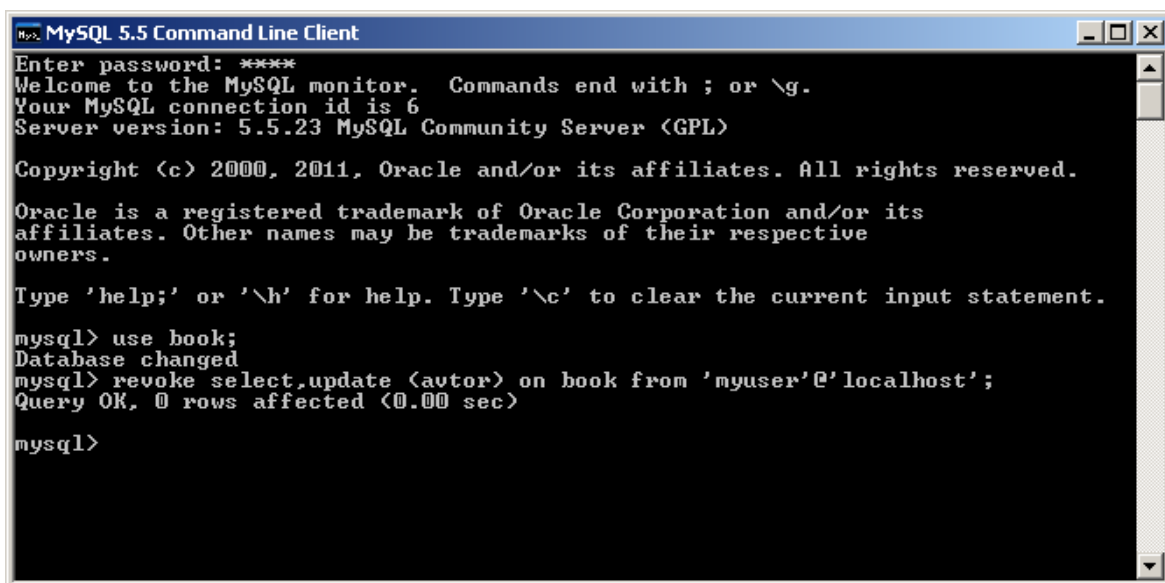
Для шифрування пароля застосовується функція *password*:

*identified by password ('mypass')*

Щоб відібрати привілею, застосовують команду *revoke*:

*revoke all on {<ім'я таблиці>/ \*/ \*.\* /<ім'я бази даних>.\*} from user*

В наступному прикладі (Рисунок 12) показано, як позбавити користувача *myuser* всіх наданих йому привілеїв:



```
MySQL 5.5 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use book;
Database changed
mysql> revoke select,update (avtor) on book from 'myuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql>
```

Рисунок 12. Позбавлення користувача *myuser* всіх наданих йому привілеїв

Видалити користувача із системи можна за допомогою команди

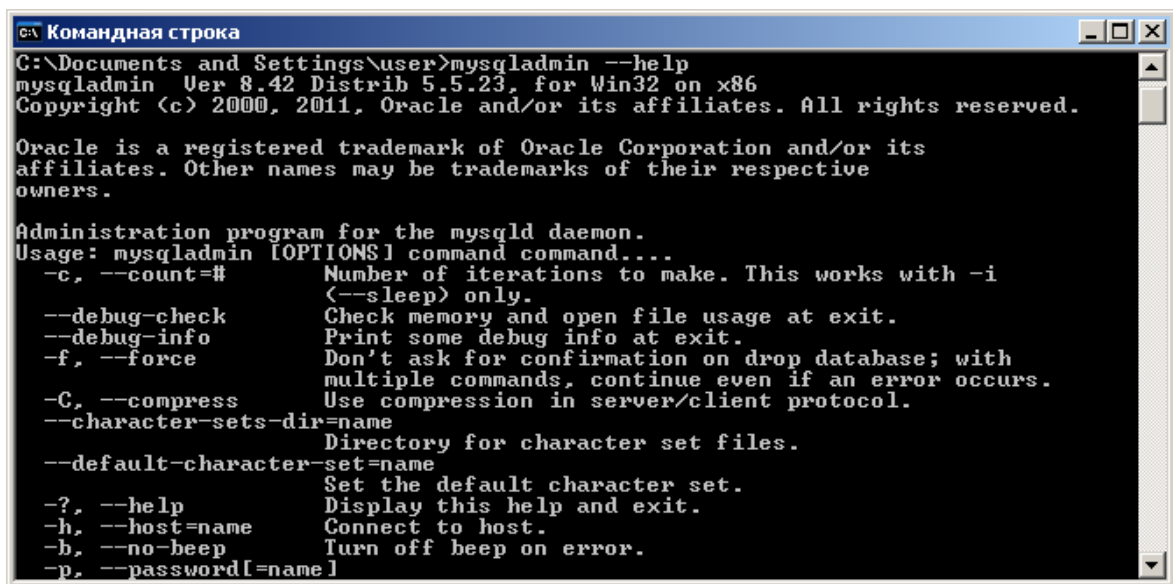
*drop user 'myuser'@'localhost'*

Якщо при встановленні *MySQL* були створені анонімні користувачі, то їх можна видалити з таблиці *db* за допомогою команди

*delete from db where user=' '*

Використання утиліти *mysqladmin* для адміністрування *MySQL*.

Утиліта *mysqladmin* викликається з командного рядку. Опція *--help* виводить та описує всі допустимі команди *mysqladmin* (Рисунок 13).



```
Командная строка
C:\Documents and Settings\user>mysqladmin --help
mysqladmin Ver 8.42 Distrib 5.5.23, for Win32 on x86
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Administration program for the mysqld daemon.
Usage: mysqladmin [OPTIONS] command command....
-c, --count=#          Number of iterations to make. This works with -i
                        <--sleep> only.
--debug-check          Check memory and open file usage at exit.
--debug-info           Print some debug info at exit.
-f, --force            Don't ask for confirmation on drop database; with
                        multiple commands, continue even if an error occurs.
-C, --compress         Use compression in server/client protocol.
--character-sets-dir=name
                        Directory for character set files.
--default-character-set=name
                        Set the default character set.
-?, --help            Display this help and exit.
-h, --host=name        Connect to host.
-b, --no-beep         Turn off beep on error.
-p, --password[=name]

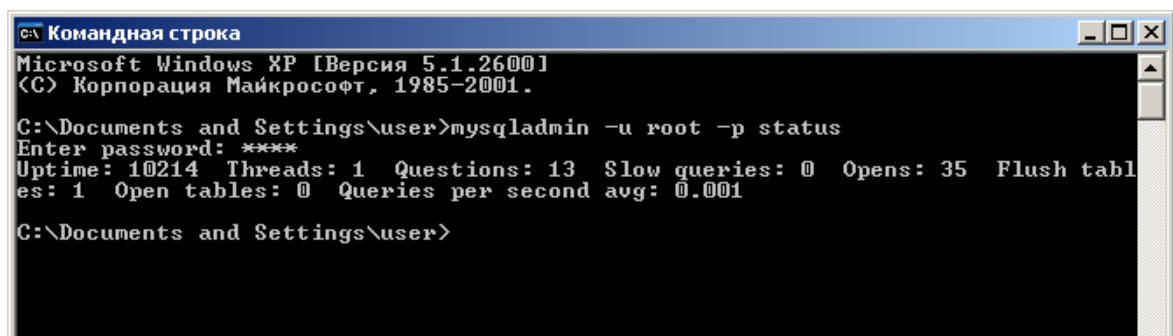
```

Рисунок 13. Опція *--help*

Наступні команди використовують для виконання деяких адміністративних функцій.

1. Інформація про поточний стан сервера *MySQL* (Рисунок 14):

*mysqladmin -u root -p status*



```
Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\Documents and Settings\user>mysqladmin -u root -p status
Enter password: ****
Uptime: 10214 Threads: 1 Questions: 13 Slow queries: 0 Opens: 35 Flush tabl
es: 1 Open tables: 0 Queries per second avg: 0.001

C:\Documents and Settings\user>

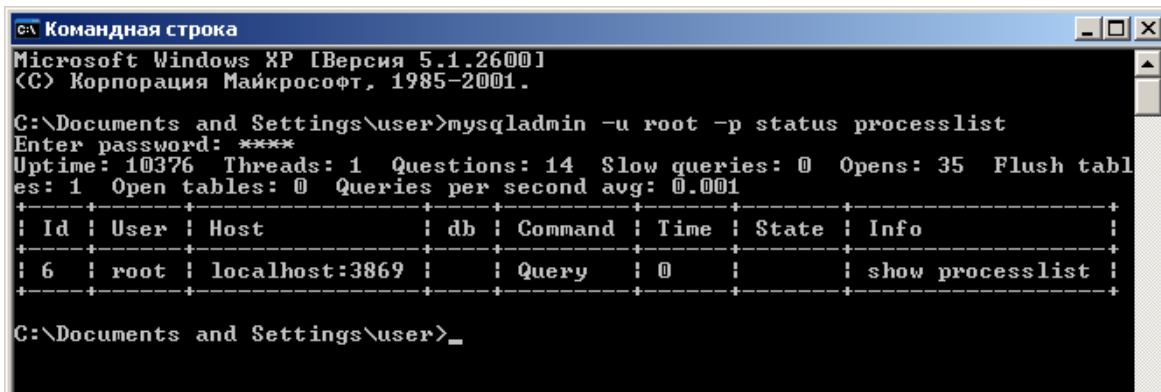
```

Рисунок 14. Інформація про поточний стан сервера *MySQL*

2. Інформація про стан та список потоків сервера *MySQL*

(Рисунок 15).

*mysqladmin -u root -p status processlist*



```
C:\ Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

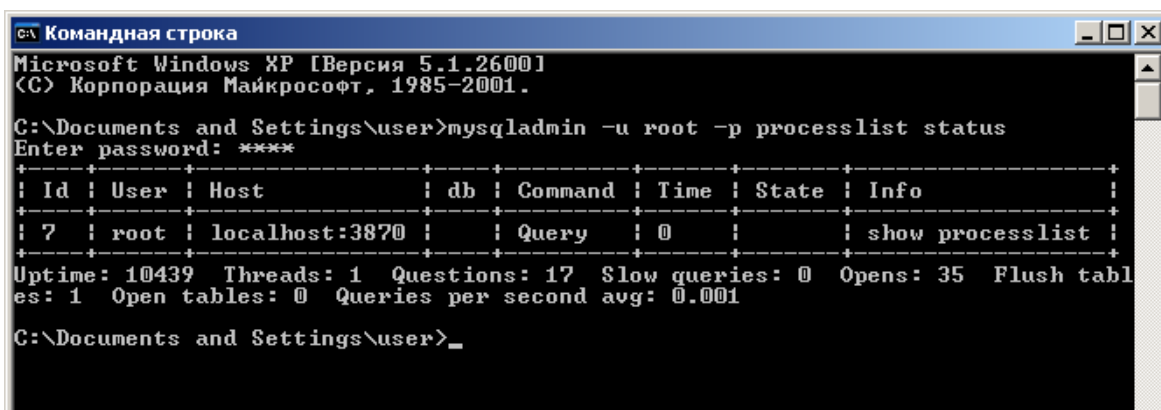
C:\Documents and Settings\user>mysqladmin -u root -p status processlist
Enter password: ****
Uptime: 10376 Threads: 1 Questions: 14 Slow queries: 0 Opens: 35 Flush tabl
es: 1 Open tables: 0 Queries per second avg: 0.001
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host          | db | Command | Time | State | Info          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 6  | root | localhost:3869 |    | Query   | 0    |      | show processlist |
+-----+-----+-----+-----+-----+-----+-----+-----+

C:\Documents and Settings\user>_
```

Рисунок 15. Інформація про стан та список потоків сервера *MySQL*

3. Наступна команда подібна попередній (Рисунок 16):

*mysqladmin -u root -p processlist status*



```
C:\ Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

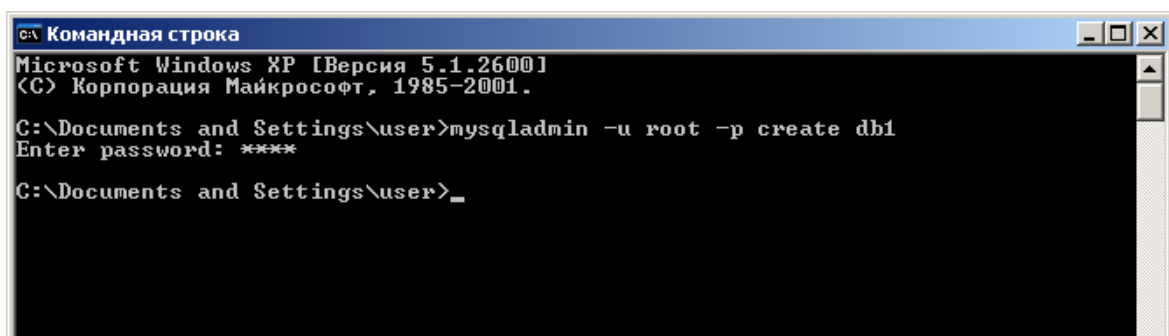
C:\Documents and Settings\user>mysqladmin -u root -p processlist status
Enter password: ****
+-----+-----+-----+-----+-----+-----+-----+-----+
| Id | User | Host          | db | Command | Time | State | Info          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7  | root | localhost:3870 |    | Query   | 0    |      | show processlist |
+-----+-----+-----+-----+-----+-----+-----+-----+
Uptime: 10439 Threads: 1 Questions: 17 Slow queries: 0 Opens: 35 Flush tabl
es: 1 Open tables: 0 Queries per second avg: 0.001

C:\Documents and Settings\user>_
```

Рисунок 16. Інформація про стан та список потоків сервера *MySQL*

4. Створення нової бази даних (Рисунок 17):

*mysqladmin -u root -p create db1*



```
C:\ Командная строка
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

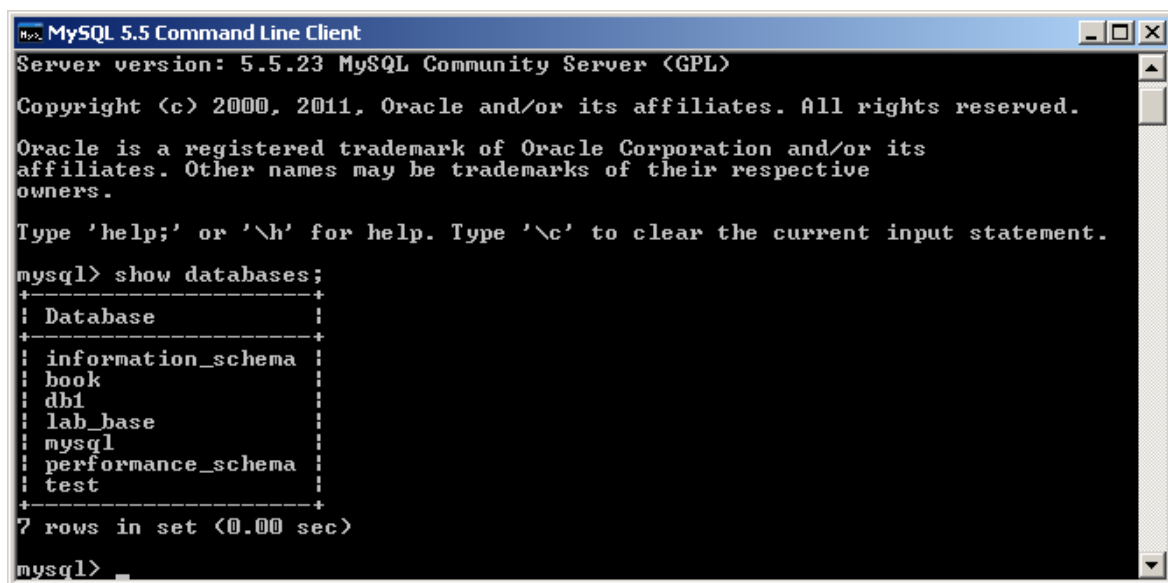
C:\Documents and Settings\user>mysqladmin -u root -p create db1
Enter password: ****

C:\Documents and Settings\user>_
```

Рисунок 17. Створення нової бази даних

Для перевірки результату (Рисунок 18) необхідно запустити утиліту *mysql* і набрати команду

```
mysql> show databases;
```



```
MySQL 5.5.23 Command Line Client
Server version: 5.5.23 MySQL Community Server (GPL)
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| book |
| db1 |
| lab_base |
| mysql |
| performance_schema |
| test |
+-----+
7 rows in set (0.00 sec)

mysql>
```

Рисунок 18. Перевірка результату створення нової бази даних *db1*

## 6. Видалення бази даних

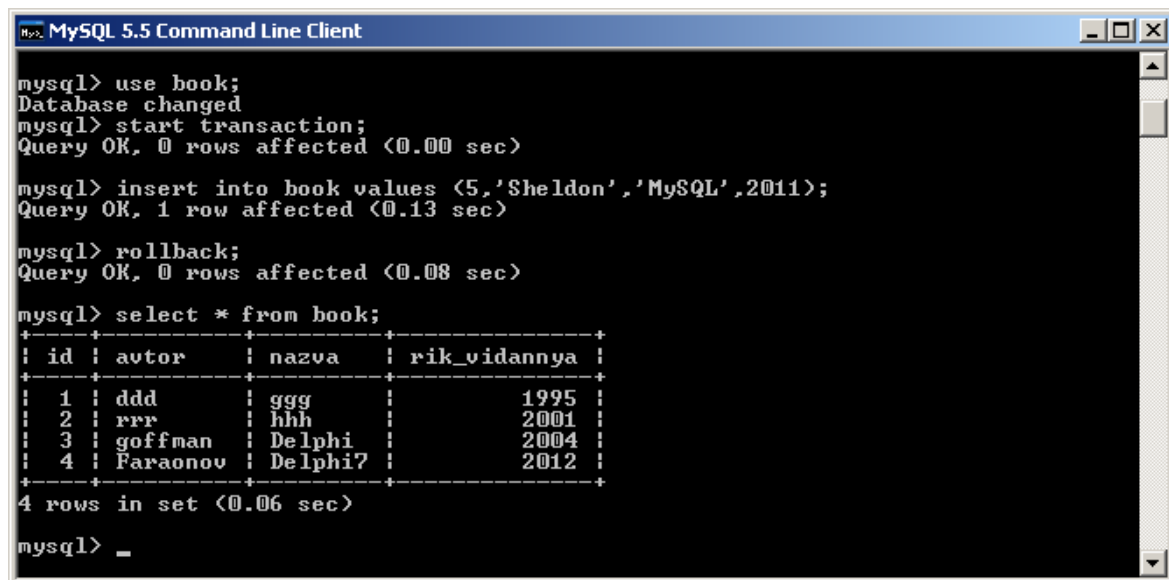
```
mysqladmin -u root -p drop db1
```

Тут в команді включені опція *-u* (*--user=name*) разом із наступним за нею іменем користувача *root* та опція *-p* (*--password[=name]*), яка на відміну від інших опцій, вимагає вказувати значення пароля окремо від утиліти *mysqladmin*.

**Виконання транзакцій.** Оператор *start transaction* ізолює всі наступні оператори в одиницю транзакції. Оператори розглядаються як частина транзакції до її фіксації в базі даних або до її відміни, і якщо щось, то в базу даних не вноситься ніяких змін.

Оператор *commit* використовується для завершення транзакції і збереження всіх змін, виконаних транзакцією в базі даних.

Для завершення транзакції можна також використовувати оператор *rollback*. На відміну від *commit*, *rollback* не фіксує зміни в базі даних, а виконує відміну транзакції. Наступний приклад демонструє, як оператор *rollback* відмінює додавання п'ятого запису до таблиці *book* (Рисунок 19).



```
mysql> use book;
Database changed
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into book values (5,'Sheldon','MySQL',2011);
Query OK, 1 row affected (0.13 sec)

mysql> rollback;
Query OK, 0 rows affected (0.08 sec)

mysql> select * from book;
+----+-----+-----+-----+
| id | autor  | nazva  | rik_vidannya |
+----+-----+-----+-----+
| 1  | ddd    | ggg    | 1995         |
| 2  | rrr    | hhh    | 2001         |
| 3  | goffman | Delphi | 2004         |
| 4  | Faraonov | Delphi? | 2012         |
+----+-----+-----+-----+
4 rows in set (0.06 sec)

mysql> _
```

Рисунок 19. Виконання транзакції

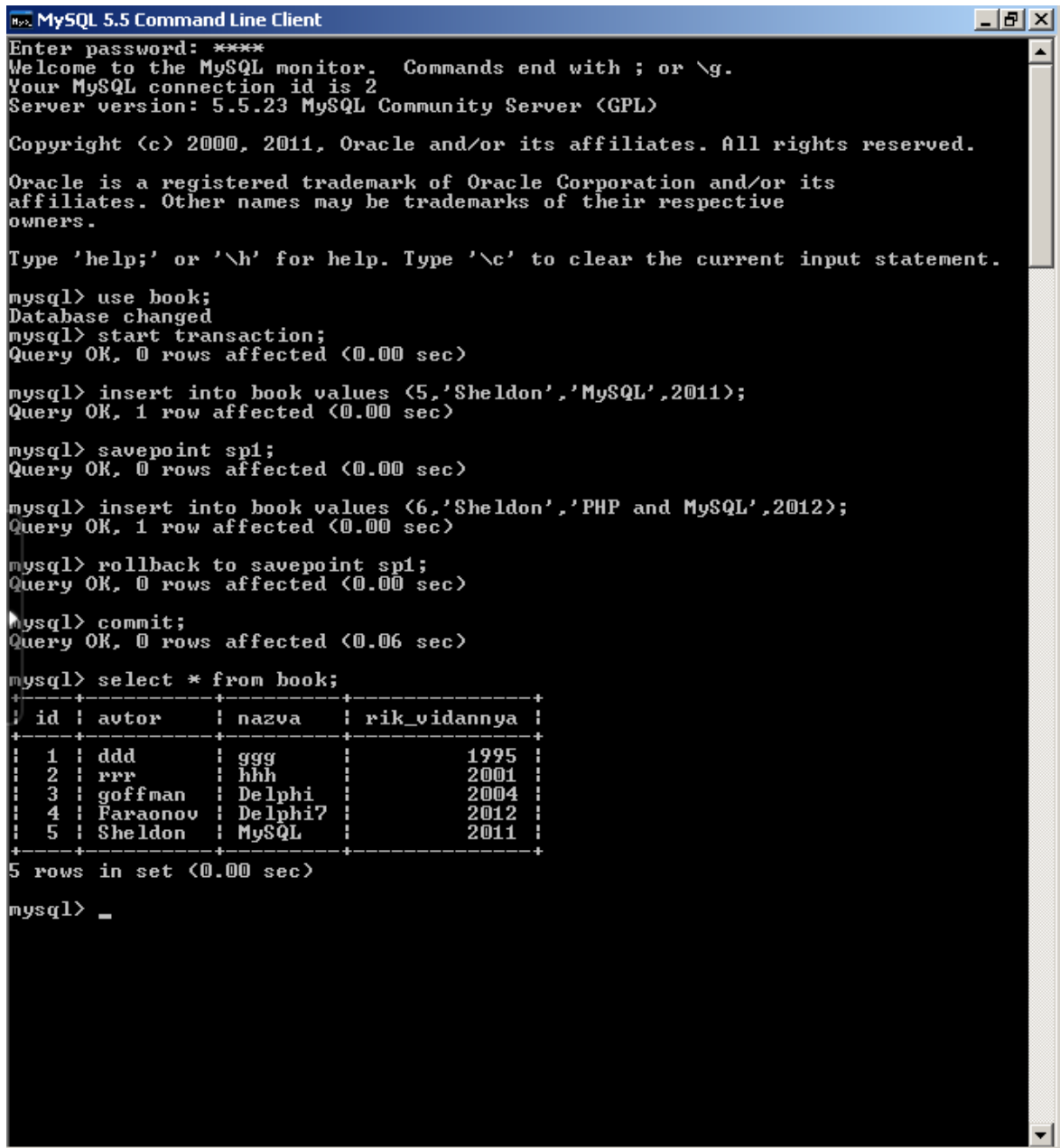
*MySQL* містить ряд операторів, які не мають включатися в транзакцію. При їх видачі в транзакції вона автоматично фіксується, після чого фіксується сам оператор. Крім того, ні один з цих операторів не можна відмінити.

Наступні оператори, які не слід включати в транзакцію.

- ***ALTER TABLE***. Модифікує таблицю.
- ***CREATE INDEX***. Створює індекс в таблиці.
- ***DROP DATABASE***. Видаляє базу даних на сервері *MySQL*.
- ***DROP INDEX***. Видаляє індекс з таблиці.
- ***DROP TABLE***. Видаляє таблицю з бази даних.
- ***LOCK TABLES***. Запобігає паралельний доступ до таблиць.
- ***RENAME TABLES***. Перейменує таблицю.
- ***SET AUTOCOMMIT = 1***. Включає режим автоматичної фіксації.
- ***START TRANSACTION***. Починає транзакцію.
- ***TRUNCATE TABLE***. Знищує дані в таблиці.
- ***UNLOCK TABLES***. Відмінняє блокування таблиць.

Для ізоляції частин транзакції використовують оператори *savepoint* та *rollback to savepoint*. *Savepoint* дозволяє визначити точки зберігання в

транзакції, а *rollback to savepoint* – відмінити транзакцію до визначеної точки зберігання (Рисунок 20).



```
MySQL 5.5 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use book;
Database changed
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into book values (5,'Sheldon','MySQL',2011);
Query OK, 1 row affected (0.00 sec)

mysql> savepoint sp1;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into book values (6,'Sheldon','PHP and MySQL',2012);
Query OK, 1 row affected (0.00 sec)

mysql> rollback to savepoint sp1;
Query OK, 0 rows affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.06 sec)

mysql> select * from book;
+----+-----+-----+-----+
| id | autor | nazva | rik_vidannya |
+----+-----+-----+-----+
| 1  | ddd   | ggg   | 1995         |
| 2  | rrr   | hhh   | 2001         |
| 3  | goffman | Delphi | 2004         |
| 4  | Faraonov | Delphi7 | 2012         |
| 5  | Sheldon | MySQL | 2011         |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> _
```

Рисунок 20. Використання команд *savepoint* та *rollback to savepoint*

### Методичні вказівки

1. Підключитися до сервера *MySQL*.
2. Після реєстрації на сервері змінити командою *password* свій пароль.
3. Бази даних користувачів і права доступу до них створюються адміністратором. Необхідно лише попросити його задати пароль для доступу

до своєї бази даних.

4. При роботі з монітором *mysql* в кінці команд ставити крапку з комою.

5. Довгі команди (створення таблиць, введення даних у таблицю) краще вводити в текстовий файл, переданий монітору через командний рядок.

6. Монітор *mysql* пам'ятає в буфері раніше уведені команди. Для їхнього виклику використовують стрілку нагору.

7. Для перегляду вмісту таблиць у кодуванні, що відрізняється від кодування сервера *MySQL*, зручно використати утиліту *netadmin.exe*.

8. Як правило, користувачі не мають прав доступу до бази даних *mysql*. Для перевірки призначених Вами прав доступу до Вашої бази даних потрібно звернутися до адміністратора (викладача).

9. Після закінчення роботи з базою даних вивантажити її структуру й дані в текстовий файл і зберегти його. Він може знадобитися для відновлення зіпсованої або знищеної бази даних.

### **Порядок виконання роботи**

1. Спроекувати структуру бази даних відповідно до варіанта.
2. Підключитися до *MySQL*-сервера, створити таблиці, освоїти операції додавання, отримання, зміни й видалення записів.
3. Організувати виконання транзакції з її фіксацією та її відміною.
4. Виконати додавання користувачів до бази даних та їх видалення.
5. Налаштувати права доступу до бази даних, забезпечивши право на отримання даних заданому користувачеві без пароля з будь-якого домену, і права на вставку, зміну, додавання, видалення записів – користувачеві із заданим обліковим записом і паролем, що заходить із заданого домену.
6. Оформити короткий звіт про виконану роботу, внести в нього основні теоретичні відомості.



## **Оформлення звіту:**

1. Титульний лист.
2. Мета роботи.
3. Порядок виконання.
4. Представлення основних результатів по виконаній лабораторній роботі.
5. Висновки.

## **Варіанти завдань**

**Варіант 1.** Спроекувати базу даних про студентів, для їхнього розподілу по місцях практики: прізвище, рік народження, група, факультет, середній бал, місце роботи, місто.

**Варіант 2.** Спроекувати базу даних про автомобілі: номер, рік випуску, марка, кольори, стан, прізвище власника, адреса.

**Варіант 3.** Спроекувати базу даних про квартири, призначені для продажу: район, поверх, площа, кількість кімнат, відомості про власника, ціна.

**Варіант 4.** Спроекувати базу даних про книги, які були куплені бібліотекою: назва, автор, рік видання, адреса автора, адреса видавництва, ціна, книготорговельна фірма.

**Варіант 5.** Спроекувати базу даних про співробітників, що мають комп'ютер: прізвище, номер кімнати, назва відділу, дані про комп'ютери.

**Варіант 6.** Спроекувати базу даних про замовлення, отриманих співробітниками фірми: прізвище, сума замовлення, найменування товару, назва фірми - клієнта, прізвище замовника.

**Варіант 7.** Спроекувати базу даних про оцінки, отриманих студентами на іспитах: прізвище, група, предмет, номер квитка, оцінка, викладач.

**Варіант 8.** Спроекувати базу даних про викладачів кафедри: прізвище, посада, ступінь, номер кімнати, курси, що читають.

**Варіант 9.** Спроекувати базу даних про авторів *web*-додатку та їхніх статей: ім'я, адреса, обліковий запис, пароль, тема, заголовок, текст статті, ілюстрації.

**Варіант 10.** Спроекувати базу даних про список розсилання й передплатників: тема й зміст листа, дата відправлення, імена й адреси передплатників, їхні облікові записи й паролі.

**Варіант 11.** Спроекувати базу даних «Мережа магазинів комп'ютерної техніки», яка містить інформацію про наступні об'єкти: магазини ( назва, адреса), наявність товару, кількість, ціна, продавці (ППП, адреса, дата народження, номер паспорту, магазин, відділ).

**Варіант 12.** Спроекувати базу даних про працівників поліклініки: ППП, спеціальність, розклад прийому (день тижня, початок прийому, кінець прийому, кабінет, ділянка).

**Варіант 13.** Спроекувати базу даних «Картотека бібліотеки», яка містить наступну інформацію: картка книги, видавництво, місто, тема книги, дисципліна, список вибірки книги по дисципліні, позиція вибірки.

**Варіант 14.** Спроекувати базу даних для гуртожитку, яка містить наступну інформацію: корпус, кімнати, студенти, які там проживають, оплата, умови.

**Варіант 15.** Спроекувати базу даних про відвідування студентами занять. База даних містить наступну інформацію: кафедра, група, студент, дисципліна, викладач, заняття, відвідування заняття.

### **Контрольні питання**

1. Варіанти підключення до бази даних за допомогою монітора *MySQL*.
2. Таблиці системи привілеїв *MySQL*.
3. Послідовність контролю доступу до даних в *MySQL*.
4. Визначення прав доступу до даних в *MySQL*.
5. Транзакції.
6. Архітектура мови *SQL*.

7. Створення й видалення баз даних і таблиць.
8. Типи даних в *MySQL*,
9. Індокси, їхнє призначення й створення.
10. Послідовності й автоінкрементування в *MySQL*.
11. Додавання даних у таблицю.
12. Зміна й видалення даних.
13. Запити на добування даних.
14. Об'єднання даних.
15. Групування й упорядкування даних.

## Додаток А

### Основні функції *MySQL*

#### Команда довідки по синтаксису і опису команд і ключових слів

#### SQL:

*help* команда;     або ? команда;

наприклад:

*mysql> help SELECT;*   або *mysql>? CREATE;*

#### Вибір бази даних:

*mysql> USE database;*

#### Виведення списку раніше створених БД:

*mysql> SHOW DATABASES;*

#### Виведення списку раніше створених таблиць в БД:

*mysql> SHOW TABLES;*

#### Проглянути опис формату таблиці:

*mysql> DESCRIBE table;*

#### Створення нової БД:

*mysql> CREATE DATABASE db\_name;*

#### Створення нової таблиці в БД:

*mysql> CREATE TABLE table\_name (field1\_name TYPE(SIZE),  
field2\_name TYPE (SIZE));*

Приклад: *mysql> CREATE TABLE personnel (name VARCHAR(20),  
sex CHAR(1), birth DATE);*

**Додавання записів в таблицю:**

*mysql> INSERT INTO table\_name (column\_name1 VARCHAR(20),  
column\_name2 TEXT, column\_name3 DATE) VALUES ('MyName', 'MyOwner',  
'2002-08-31');*

**Модифікація записів в таблиці:**

*mysql> UPDATE table SET column\_name = "new\_value" WHERE  
record\_name = "value";*

**Вибірка даних з таблиці БД:**

*mysql> SELECT from\_columns FROM table WHERE умова;*

**Вибірка всіх даних з таблиці:**

*mysql> SELECT \* FROM table;*

**Вибірка всіх даних з таблиці, які задовольняють умові:**

*mysql> SELECT \* FROM table WHERE rec\_name = "value";*

**Вибірка всіх даних з таблиці, які задовольняють декільком  
умовам:**

*mysql> SELECT \* FROM TABLE WHERE reel = "value1" AND  
rec2 = "value2";*

**Вибірка певних полів з даними з таблиці:**

*mysql> SELECT column\_name FROM table;*

**Вибірка унікальних записів з таблиці:**

*mysql> SELECT DISTINCT column\_name FROM table;*

**Сортування:**

*mysql> SELECT col1, col2 FROM table ORDER BY col1 ASC;*

**Сортування в зворотному порядку:**

*mysql> SELECT col1, col2 FROM table ORDER BY col1 DESC;*

### **Пошук інформації по заданому критерію:**

*mysql> SELECT \* FROM table WHERE rec LIKE "blah%";* (% - заміняє довільне число символів, \_ - заміняє один символ)

### **Пошук інформації по заданому критерію з регулярними виразами:**

*mysql> SELECT \* FROM table WHERE rec '^[a-f]';*

Можливі варіанти:

<значення> – значення, що перевіряється, має містити вказане значення.

<^> – значення, що перевіряється, не має містити вказане значення.

[<символи>] – значення, що перевіряється має містити принаймні один із символів, які перераховані в квадратних дужках.

[<діапазон>] – значення, перевіряється має містити принаймні один із символів, вказаних в квадратних дужках діапазоні значень.

^ – значення, що перевіряється, має починатися із значення, якому передуює знак вставки (^).

\$ – значення, що перевіряється, має закінчуватися значенням, за яким слідує знак долара (\$).

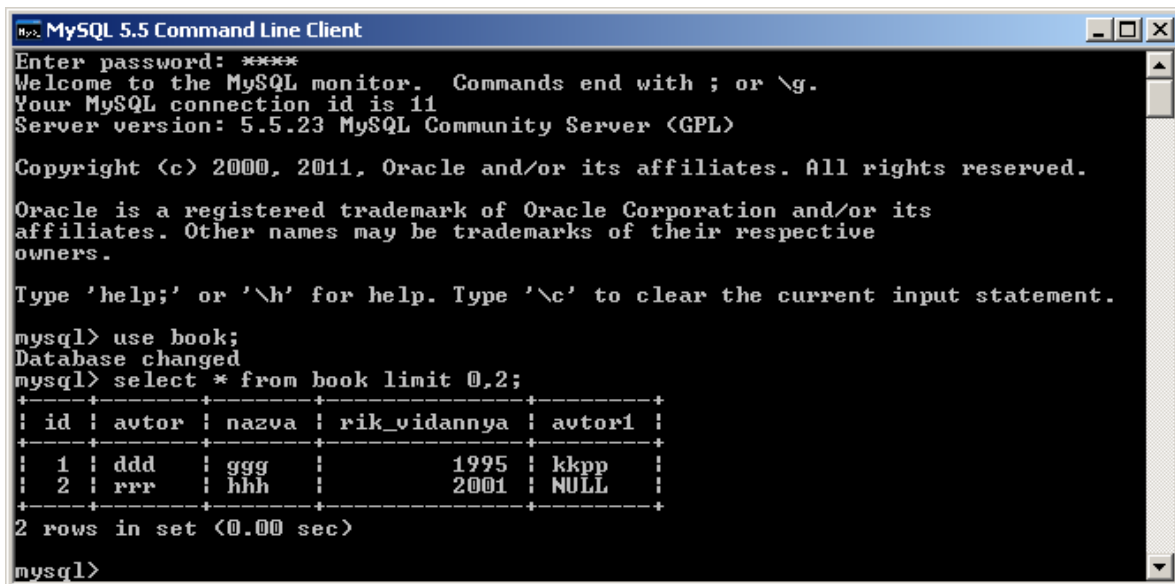
\* – значення, що перевіряється, має містити нуль або більше символів, які розміщені перед знаком зірочки (\*).

. – значення, що перевіряється, може містити будь-який окремий символ, який представлений крапкою.

### **Лічильник кількості однакових записів в таблиці:**

*mysql> SELECT COUNT(\*) FROM table;*

Ще одним оператором запиту *SELECT* є обмеження *LIMIT* <перше значення>, <кількість записів вибірки> (Рисунок 22).



```
MySQL 5.5 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.5.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use book;
Database changed
mysql> select * from book limit 0,2;
+----+-----+-----+-----+-----+
| id | avtor | nazva | rik_vidannya | avtor1 |
+----+-----+-----+-----+-----+
| 1  | ddd   | ggg   | 1995         | kkpp   |
| 2  | rrr   | hhh   | 2001         | NULL   |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Рисунок 22. Вибірка з обмеженням

Групування записів з визначенням кількості записів в кожній групі:

```
mysql> SELECT owner, COUNT(*) FROM table GROUP BY owner;
```

Вибірка записів одночасно з декількох таблиць:

```
mysql> SELECT pet.name, comment FROM pet, event WHERE pet.name = event.name;
```

(Ви можете об'єднувати таблиці назначаючи для себе нові назви колонок за допомогою ключового слова 'AS')

Показує поточно вибрану БД:

```
mysql> SELECT DATABASE ( );
```

Вибірка максимального значення

```
mysql> SELECT MAX(col_name) AS label FROM table;
```

Колонки з автоматичним інкрементуванням чисел в кожному наступному записі:

```
mysql> CREATE TABLE table (number INT NOT NULL AUTO_INCREMENT, name CHAR(10) NOT NULL);
```

```
mysql> INSERT INTO table (name) VALUES ('torn', 'dick', 'harry');
```

Додавання поля до вже раніше створеної таблиці:

*mysql> ALTER TABLE tbl ADD COLUMN format SMALLINT  
AFTER col\_name;*

**Видалення поля з таблиці:**

*mysql> ALTER TABLE tbl DROP COLUMN format;*

**Вибірка з використанням оператора CASE**

*CASE WHEN [умова1] THEN результат, що повертається1 [WHEN  
[умова2] THEN результат, що повертається2 ... ] [ELSE результат, що  
повертається при невиконанні жодної з умов]*

*END;*

*CASE вираз WHEN значення виразу THEN результат, що повертається  
[WHEN [значення виразу] THEN результат, що повертається...]*

*[ELSE результат, що повертається за замовченням при невиконанні жодної  
з умов]*

*END;*

*SELECT (CASE (paid) WHEN 1 THEN 'платна' WHEN 0 THEN 'державна'  
ELSE 'невідома' END) AS form FROM students ;*

Більш детальну інформація по синтаксису *SQL* можна знайти на сайті  
[mysql.com](http://mysql.com).

**Додаток Б**

**Типи даних MySQL**

Тип поля може бути:

- Цілочисловим;
- Дійсним (реальним);
- Строковим;
- Бінарним;
- Дата й час;
- Переліки й множини.

Можливі типи даних, діапазони й описи представлені в наступних  
таблицях:

### Цілочислові типи даних

Тип	Діапазон
TINYINT	-128 ... +127
SMALLINT	-32768 ... +32768
MEDIUMINT	-8 388 608 ... +8 388 608
INT	-2 147 483 648 ... +2 147 483 647
BIGINT	-9 223 327 036 854 775 808... +9223 327 036 854 775 808

### Дійсні (реальні) числа

Дійсні (реальні) типи записуються у вигляді:

**ТИП (ДОВЖИНА, ЗНАКИ) [UNSIGNED]**

Довжина – це кількість знакомісць, у яких буде розміщене все число при його передачі, а ЗНАКИ – це кількість знаків після десяткової крапки, які будуть враховуватися. Якщо зазначено модифікатор **UNSIGNED**, знак числа враховуватися не буде.

Тип	Опис
FLOAT	Невелика точність
DOUBLE	Подвійна точність
REAL	Те ж, що і DOUBLE
DECIMAL	Дробове число, що зберігається у вигляді рядка
NUMERIC	Те ж, що і DECIMAL

### Рядкові типи

Тип	Опис
-----	------



TINYTEXT	Максимальна довжина 255 символів
TEXT	Максимальна довжина 65535 (64 Кб) символів
CHAR ( )	Максимальна довжина до 255 символів
VARCHAR ( )	Максимальна довжина до 255 символів
MEDIUMTEXT	Максимальна довжина 16777215 символів
LONGTEXT	Максимальна довжина 4294967295 символів

У більшості випадків застосовується тип **VARCHAR** або просто **CHAR**, що дозволяє зберігати рядки, що містять до 255 символів. У дужках після типу вказується довжина рядка: **VARCHAR(48); CHAR(73);**

Якщо 255 символів для вашого завдання недостатньо, можна використати інші типи, наприклад, **TEXT**.

#### Бінарні типи даних

Тип	Опис
TINYBLOB	Максимальна довжина 255 символів
BLOB	Максимальна довжина 65535 символів
MEDIUMBLOB	Максимальна довжина 16777215 символів
LONGBLOB	Максимальна довжина 4294967295 символів

#### Дата і час

Тип	Опис
DATE	Дата у форматі РРРР-ММ-ДД діапазон 1000-01-01-до 9999
TIME	Час у форматі ГГ:ХХ:СС
TIMESTAMP	Дата і час, виводиться у вигляді РРРР-ММ-ДД ГГ:ХХ:СС діапазон 1970-01-01 00:00:00 частково до 2037
DATETIME	Дата і час, виводиться у вигляді РРРР-ММ-ДД ГГ:ХХ:СС діапазон 1000-001-01 00:00:00 до 9999
YEAR	РРРР діапазон від 1901 до 2155 (і 0000)

## Лабораторна робота №5

**Тема:** Використання ключів та індексів. Об'єднання таблиць.

**Мета:** Навчитися створювати індекси, дослідити вплив індексів на швидкість сортування даних під час виконання запитів. Навчитися будувати складні конструкції **SELECT** для вибірки даних із кількох таблиць.

### Теоретичні відомості

**Об'єднання таблиць.** При нормалізації баз даних створюється багато таблиць. При вибірці даних з баз даних часто потрібні дані містяться в кількох таблицях, пов'язаних між собою за допомогою спільних полів і відношень між ними. Для того, щоб вибрати дані з кількох таблиць і мати можливість використовувати при цьому засоби **SQL** для обробки даних використовують об'єднання кількох таблиць в одну, згідно з певними критеріями. Результат об'єднання існує тільки в момент виконання запиту.

Об'єднання таблиць дозволяє робити конструкція **JOIN**. Вона може мати один з таких форматів:

*<таблиця> INNER JOIN <таблиця> [ON<умова>]*

*<таблиця> LEFT | RIGHT [OUTER] JOIN <таблиця> ON*

*<умова>*

Де під *<таблиця>* розуміють таку конструкцію:

*<ім'я таблиці> [ [ AS ] псевдонім ]*

Для розуміння об'єднання таблиць розглянемо конкретний приклад. Візьмемо навчальну базу даних **Університет**, яка містить дані про студентів університету. Інформація міститься в трьох таблицях: назви факультетів (**Факультет**), назви спеціальностей (**Спеціальність**), і власне інформація про студента (**Студенти**). До того ж, в таблиці **Студенти** містяться тільки посилання (по номеру поля) на назву факультету і спеціальності.

Припустимо, потрібно вивести список студентів, що складається з двох полів (імені студента і факультету, на якому він навчається):

**SELECT Студенти.Ім'я, Факультет.Ім'я**

**FROM** *Студенти* **LEFT JOIN** *Факультет* **ON**  
*Студенти.факультет\_id* = *Факультет.ID*;

Гаврилюк Юрій Романович	Інженерно-технічний
Грещук Олег Васильович	Інженерно-технічний
Даниленко Олег Володимирович	Інженерно-технічний
Іваськів Тарас Миронович	Інженерно-технічний
Кобель Володимир Сергійович	Інженерно-технічний
Коваль Юрій Васильович	Інженерно-технічний
...	...

У даному випадку до таблиці **Студенти** додається таблиця **Факультет** так, що до кожного рядка таблиці студентів приєднується справа рядок з таблиці факультетів за таким критерієм: *Студенти.факультет\_id* = *Факультет.ID*. Якщо рядка, який відповідає заданій умові не знайшлося, то у відповідні поля результату заноситься **NULL**.

Оскільки при об'єднанні використано **LEFT JOIN**, то з таблиці **Студенти** (лівої) будуть виведені всі рядки, а з таблиці **Факультет** ті, які відповідають умові об'єднання.

При створюванні зовнішніх об'єднань службове слово **OUTER** можна опускати.

Розглянемо ще один приклад. Припустимо потрібно зробити вибірку всіх довідкових служб з бази даних **Ужгород**, вивести їх назви, телефони і послуги, які вони надають. Для цього потрібно вивести поля *Назва* і *Телефон* з таблиці **Фірма** і поле *Послуги\_Назва* з таблиці **Послуги**.

При зовнішньому об'єднанні таблиць запит виглядатиме так:

**SELECT** *Фірма.Назва, Фірма.Телефон, Послуги.Послуги\_Назва*  
**FROM** *Фірма* **LEFT JOIN** *Послуги* **ON** *Фірма.фірм\_id* =  
*Послуги.фірм\_id*

**WHERE** Фірма.Назва **LIKE** '%довідк%';

При виконанні даного запиту отримаємо такий результат:

«Довідка Ужгородської міської ради»	«0312 080»	«Довідка Ужгородської міської ради»
«Довідка аптек»	«0312 0-67»	
«Довідка залізничної станції «Ужгород»	«0312 23-33-33»	
«Довідка залізничного вокзалу станції «Ужгород»»	«0312 005»	«Довідка залізничного вокзалу станції»
«Довідка міжміського автобусного вокзалу м.Ужгород»	«0312 004»	«Довідка міжміського автобусного вокзалу»
«Довідка приміського автобусного вокзалу м.Ужгород»	«0312 23-44-44»	«Довідка приміського автобусного вокзалу»
«Обласне адресне довідкове бюро УМВС України в Закарпатській області»	«0312 23-77-77»	«Адресне довідкове бюро»

При внутрішньому об'єднанні результат буде іншим:

**SELECT** Фірма.Назва, Фірма.Телефон, Послуги. Послуги\_Назва

**FROM** Фірма **INNER JOIN** Послуги **ON** Фірма.фірм\_id =  
Послуги.фірм\_id

**WHERE** Фірма.Назва **LIKE** '%довідк%';

«Довідка Ужгородської міської ради»	«0312 080»	«Довідка Ужгородської міської ради»
«Довідка залізничного вокзалу станції «Ужгород»»	«0312 005»	«Довідка залізничного вокзалу станції»
«Довідка міжміського автобусного вокзалу м.Ужгород»	«0312 004»	«Довідка міжміського автобусного вокзалу»

«Довідка приміського автобусного вокзалу м.Ужгород»	«0312 23-44-44»	«Довідка приміського автобусного вокзалу»
«Обласне адресне довідкове бюро УМВС України в Закарпатській області»	«0312 23-77-77»	«Адресне довідкове бюро»

Очевидно, що в другому випадку ми втратили два рядки. Оскільки при внутрішньому об'єднанні виводяться тільки ті рядки, які мають відповідник в обох таблицях, то «Довідка аптек» та «Довідка залізничної станції «Ужгород»» в другому запиті не показані, бо для цих організацій немає жодного відповідника в таблиці *Послуги*. Таким чином, вибір способу об'єднання таблиць може впливати на кінцевий результат, і в загальному випадку призвести до втрати окремих рядків.

**Використання індексів (ключів).** Будь-які поля в таблицях можуть бути індексовані, створення індексів є одним із способів пришвидшити виконання *SQL*-запитів.

Індекси використовують для того, щоб:

- Швидко знайти рядки, що відповідають виразу *WHERE*.
- Витягнути рядки з інших при виконанні об'єднань (оператор *JOIN*).
- Знайти величини *MIN( )* або *MAX( )* для заданого індексованого поля.
- Виконувати сортування або групування в таблиці, якщо ці операції робляться на крайньому ліворуч префіксі ключа, що використовується.

Якщо дана таблиця має індекс, який утворений за декількома стовпцями, то будь-який крайній ліворуч префікс цього індексу може бути оптимізатором для знаходження рядків. Наприклад, якщо є індекс за трьома полями (*col1,col2,col3*), то існує потенційна можливість індексованого пошуку по (*col1*), (*col1,col2*) і (*col1,col2,col3*).

У *MySQL* не можна використати частковий індекс, якщо поля не утворюють крайній ліворуч префікс цього індексу. Припустимо, що є команди *SELECT*, показані нижче:

```
SELECT * FROM tbl_name WHERE col1=val1;
```

```
SELECT * FROM tbl_name WHERE col2=val2;
```

```
SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;
```

Якщо індекс існує по  $(col1, col2, col3)$ , то тільки перший, показаний вище, запит використає цей індекс. Другий і третій запити дійсно включають індексовані стовпці, але  $(col2)$  і  $(col2, col3)$  не є крайньою ліворуч частиною префіксів  $(col1, col2, col3)$ .

*MySQL* застосовує індекси також для порівнянь *LIKE*, якщо аргумент у виразі *LIKE* є статичним рядком, що не починається із символу-шаблону. Наприклад, такі команди *SELECT* використають індекси:

```
SELECT * FROM book WHERE avtor LIKE 'Sh%';
```

```
SELECT * FROM book WHERE avtor LIKE 'Sh%_o%';
```

Наступні команди *SELECT* не будуть використовувати індекси:

```
SELECT * FROM book WHERE avtor LIKE '%She%';
```

```
SELECT * FROM book WHERE avtor LIKE avtor_col;
```

У першій команді параметр виразу *LIKE* починається із шаблонного символу. У другій команді цей *LIKE* параметр не є константою.

В таблиці може бути тільки один первинний ключ. Якщо поле визначене, як поле первинного ключа, то генерується індекс. Тоді вже немає ніякої необхідності визначити звичайний індекс по цьому полю.

Слід зауважити, що створення зайвих індексів спричинить сповільнення роботи з базою даних, оскільки при кожній операції *UPDATE* чи *INSERT*, крім того, що витрачається час на модифікацію таблиці, необхідно ще й модифікувати всі індекси, які містять поле, значення якого змінилося.

**Зовнішні ключі.** Коли всі значення одного поля таблиці представлені в полі іншої таблиці, кажуть, що перше поле посилається на друге. Це вказує

на прямий зв'язок між значеннями двох полів. Наприклад, кожен із студентів в таблиці *Студенти* має поле *факультет\_id*, яке вказує на факультет, описаний в таблиці *Факультет*.

Коли одне поле в таблиці посиляється на інше, воно називається **зовнішнім ключем**; а поле, на яке воно посиляється, називається **батьківським ключем**. Таким чином, поле *факультет\_id* з таблиці *Студенти* – це зовнішній ключ, а поле *ID*, на яке воно посиляється в таблиці *Факультет* – це батьківський ключ.

Імена зовнішнього ключа і батьківського ключа часто бувають однаковими (у нашому прикладі це не так), так роблять, щоб підкреслити зв'язок між таблицями і зробити його більш наглядним.

Обмеження (зовнішні ключі, «*foreign key*») підтримують таблиці *InnoDB*. Синтаксис визначення зовнішнього ключа такий:

*[CONSTRAINT<ім'я обмеження> ] FOREIGN KEY [<ім'я індексу>]*  
(ім'я поля дочірньої таблиці 1, ...)

*REFERENCE <ім'я батьківської таблиці> (ім'я поля батьківської таблиці 1,...)*

*[ON DELETE { RESTRICT / CASCADE / SET NULL / NO ACTION / SET DEFAULT}]*

*[ON UPDATE { RESTRICT / CASCADE / SET NULL / NO ACTION / SET DEFAULT}]*

Для визначення зовнішніх ключів повинні виконуватися такі умови:

- Обидві таблиці повинні мати тип *InnoDB*, жодна з таблиць не повинна бути тимчасовою (параметр *TEMPORARY*).
- У батьківській таблиці повинен бути індекс по батьківському ключу. Індекс по відповідному полю в дочірній таблиці створюється автоматично при визначенні обмеження.
- Зовнішні ключі по полях *BLOB* і *TEXT* не підтримуються.

- Якщо вказано [**CONSTRAINT**<ім'я обмеження>] (ім'я обмеження), то воно повинне бути унікальним в даній базі даних, якщо ні, то ім'я буде згенероване автоматично.

**InnoDB** відкидає запит **INSERT** або **UPDATE**, що намагається створити значення в полі зовнішнього ключа дочірньої таблиці, якому немає відповідного значення в полі батьківської таблиці. Щодо **DELETE** і **UPDATE**, які стосуються батьківської таблиці, і впливають на записи, на які посилаються записи з дочірньої таблиці, то виконується дія, що була вказана при створенні зовнішнього ключа. **InnoDB** підтримує такі варіанти дій:

- **CASCADE** – знищення або модифікація рядків у батьківській таблиці викликає знищення або модифікацію відповідних рядків дочірньої таблиці.
- **SET NULL** – видалення або модифікація рядків батьківської таблиці, встановлює значення **NULL** для відповідних рядків дочірньої таблиці. Це можливо лише в тому випадку, якщо поле зовнішнього ключа в дочірній таблиці приймає порожні значення.
- **NO ACTION** – коли з батьківської таблиці видаляються записи або коли поновлюються значення в полях батьківського ключа, в дочірній таблиці ніяких операцій не виконується.
- **RESTRICT** – забороняє знищення або модифікацію рядків батьківської таблиці, які мають відповідники в дочірній таблиці.
- **SET DEFAULT** – коли з батьківської таблиці видаляються рядки або коли в батьківській таблиці поновлюються поля батьківського ключа, значення в полях зовнішнього ключа дочірньої таблиці перетворюються в значення за замовченням.

Зовнішні ключі можуть визначатися при створенні таблиці (в директиві **CREATE TABLE**), або при модифікації таблиці в директиві **ALTER TABLE**.

```
CREATE TABLE Студенти ( ID INT(11) NOT NULL  
AUTO_INCREMENT,  
Ім'я varchar(50) default NULL,
```



*факультет\_id int(11) default '0',*  
*Адреса varchar(120),*  
**PRIMARY KEY (ID),**  
**KEY зовн.кл (факультет\_id),**  
**CONSTRAINT зовн.кл FOREIGN KEY (факультет\_id) REFERENCES**  
**Факультет (ID))**  
**ENGINE=InnoDB;**

Якщо таблиця вже існує, то для визначення зовнішнього ключа можна використати конструкцію **ALTER TABLE**, наприклад:

**ALTER TABLE Студенти ADD CONSTRAINT зовн.кл FOREIGN KEY**  
**(факультет\_id) REFERENCES Факультет (ID);**

Знищити зовнішній ключ можна, наприклад, так:

**ALTER TABLE Студенти**  
**DROP FOREIGN KEY зовн.кл;**

### Порядок виконання роботи

В створеній, згідно індивідуального завдання, базі даних:

1. Виконати два запити пошуку з цієї таблиці:  
пошук за числовим полем;  
пошук за текстовим полем.
2. Виконати такі ж запити, але створивши попередньо індекси для поля (полів), за яким (яких) здійснюється пошук. Порівняти час виконання запитів.
3. Створити підлеглу таблицю до даної і виконати об'єднання **RIGHT JOIN** та **LEFT JOIN**. Порівняти отримані результати, міняючи таблиці місцями.
4. Для даних таблиць виконати внутрішнє та зовнішнє об'єднання та порівняти отримані результати.
5. Оформити короткий звіт про виконану роботу, внести в нього основні теоретичні відомості.

### **Оформлення звіту:**

1. Титульний лист.
2. Мета роботи.
3. Порядок виконання.
4. Представлення основних результатів по виконаній лабораторній роботі.
5. Висновки.

### **Контрольні питання**

1. Внутрішнє та зовнішнє об'єднання таблиць. Різниця між ними.
2. Індокси, їх використання.
3. Зовнішні ключі, синтаксис визначення. Видалення зовнішнього ключа.

## Лабораторна робота № 6

**Тема:** знайомство з роботою програми *MySQL-Front*.

**Мета:** Ознайомитися з інтерфейсом програми *MySQL-Front*. Навчитися з'єднуватися з сервером *MySQL*, створювати таблиці та заносити в них дані за допомогою даної програми.

### Теоретичні відомості

*MySQL-Front* є типовою *Windows*-програмою. Запустити її можна з Головного меню *Пуск/ Програми/ MySQL-Front/ MySQL-Front*. При першому запуску програми необхідно створити профіль (сесію) для під'єднання до сервера (Рисунок 1)

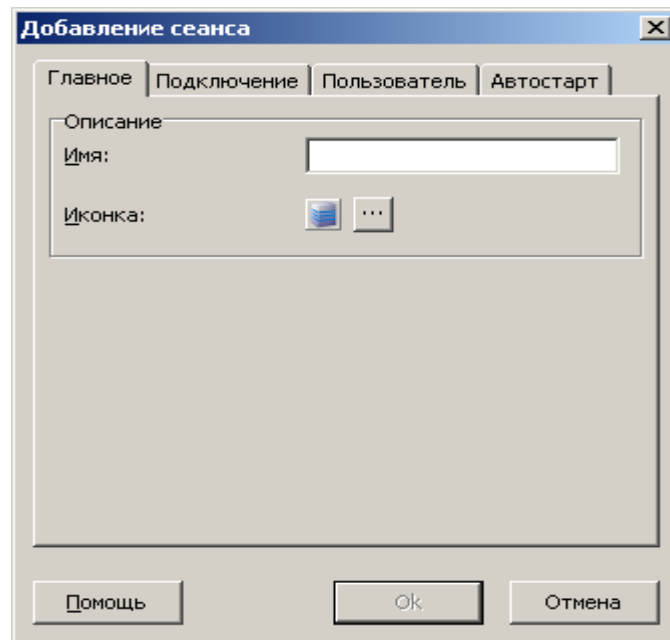


Рисунок 1. Створення профілю

На закладці *Головне* ввести назву профілю (наприклад, Ваше ім'я або прізвище), на закладці *Підключення* (Рисунок 2) – параметри *MySQL*-сервера, до якого треба під'єднатися, а саме ім'я або *IP*-адресу комп'ютера (поле *Сервер*), на якому розміщений *MySQL*-сервер, наприклад, 127.0.0.1. Решту параметрів на цій закладці залишити без змін.

На закладці *Користувач* (Рисунок 3) ввести ім'я користувача і пароль і натиснути кнопку *ОК*. З'явиться вікно вибору профілю, в якому треба обрати щойно створений профіль і знову натискаємо *ОК* (Рисунок 4).

Якщо всі параметри задано правильно, то програма повинна успішно з'єднатися з сервером і показати вікно, розділене на дві частини, зі списком баз даних сервера (Рисунок 5).

The screenshot shows a dialog box titled 'Добавление сеанса' (Add Session) with a close button (X) in the top right corner. It has four tabs: 'Главное' (Main), 'Подключение' (Connection), 'Пользователь' (User), and 'Автозапуск' (Autostart). The 'Подключение' tab is selected. Inside the dialog, there are several input fields: 'Сервер' (Server) with the value '127.0.0.1', 'Порт' (Port) with the value '3306', 'Тип соединения' (Connection type) with a dropdown menu set to 'Прямое' (Direct), and 'Кодировка' (Encoding) with an empty dropdown menu. At the bottom, there are three buttons: 'Помощь' (Help), 'Ok', and 'Отмена' (Cancel).

Рисунок 2. Завдання параметрів *MySQL*-сервера

The screenshot shows the same dialog box 'Добавление сеанса' (Add Session) but with the 'Пользователь' (User) tab selected. The 'Основное' (Basic) section contains three input fields: 'Пользователь' (User) with the value 'root', 'Пароль' (Password) with masked characters 'xxxxxxxx', and 'База данных' (Database) with the value '"book"' and a browse button (three dots). The 'Подключение' (Connection) tab is also visible in the background. At the bottom, there are three buttons: 'Помощь' (Help), 'Ok', and 'Отмена' (Cancel).

Рисунок 3. Завдання імені користувача та паролю

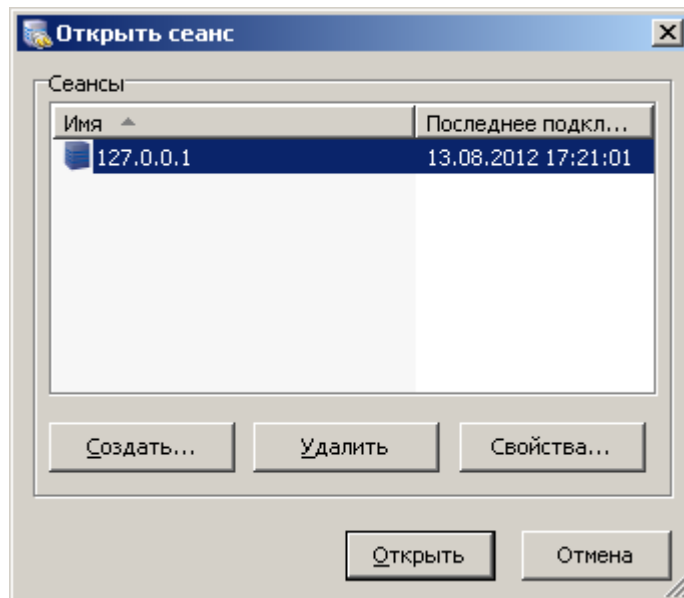


Рисунок 4. Вікно вибору профілю

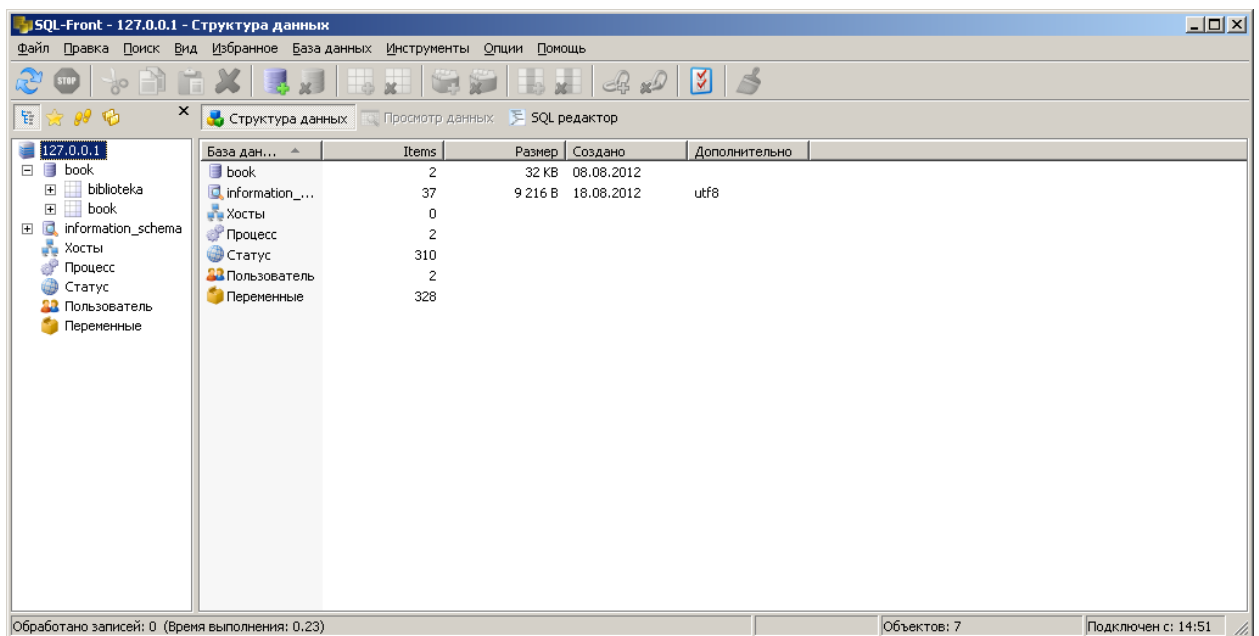


Рисунок 5.Вікно програми *MySQL-Front*

Список баз даних представлений у вигляді дерева. При розкритті вітки зі значком бази, з'явиться список таблиць, що містяться у цій базі (Рисунок 6).

Для перегляду списку полів а також для редагування структури таблиці, слід вибрати потрібну таблицю і клацнути по ній мишкою. Тоді в правій частині вікна програми буде відображено список полів та індексів таблиці (Рисунки 7 та 8). Права частина вікна може мати три режими роботи:

1. Перегляд структури (також є можливість редагування) (Рисунок 8).

2. Перегляд записів таблиці (також є можливість редагування) (Рисунок 9).
3. Виконання *SQL*-запитів.

Режими роботи перемикаються кнопками над вікном. У режимі перегляду даних записи можна редагувати, зробивши подвійний клік на клітинці. При перегляді структури, поля також можна редагувати, два рази

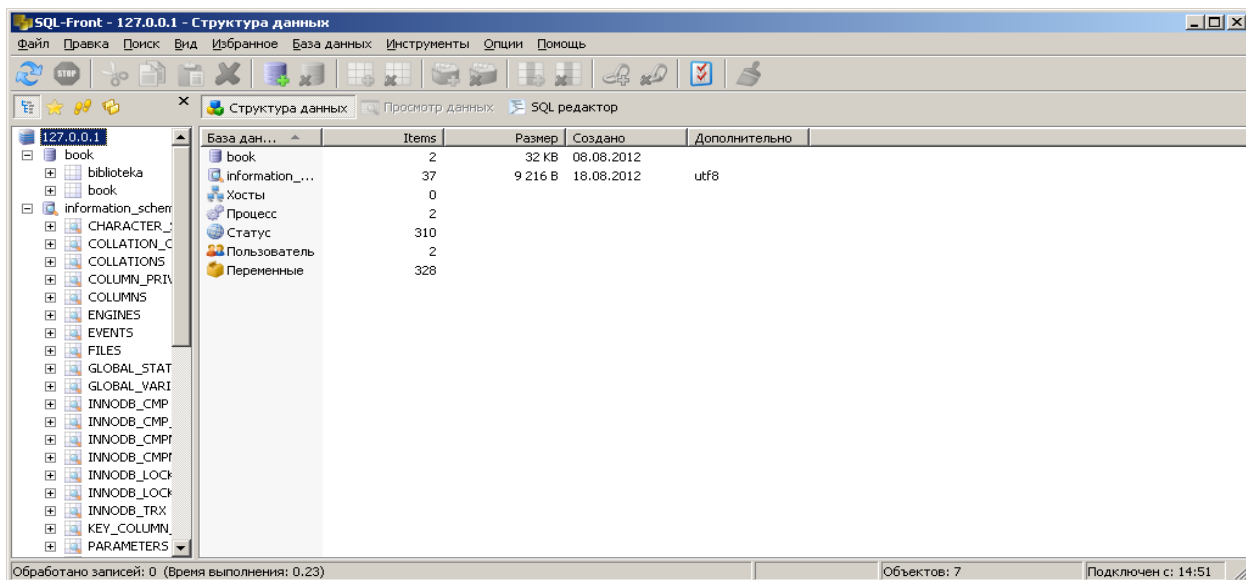


Рисунок 6. Список таблиць бази *information\_schema*

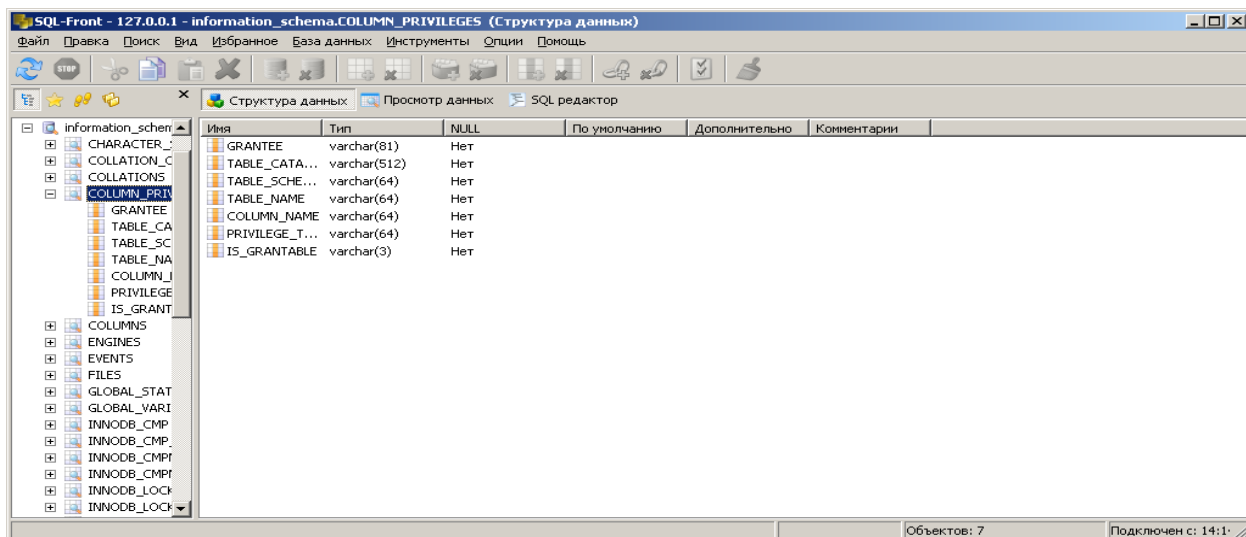


Рисунок 7. Перегляд списку полів таблиці *COLUMN\_PRIVILEGES*

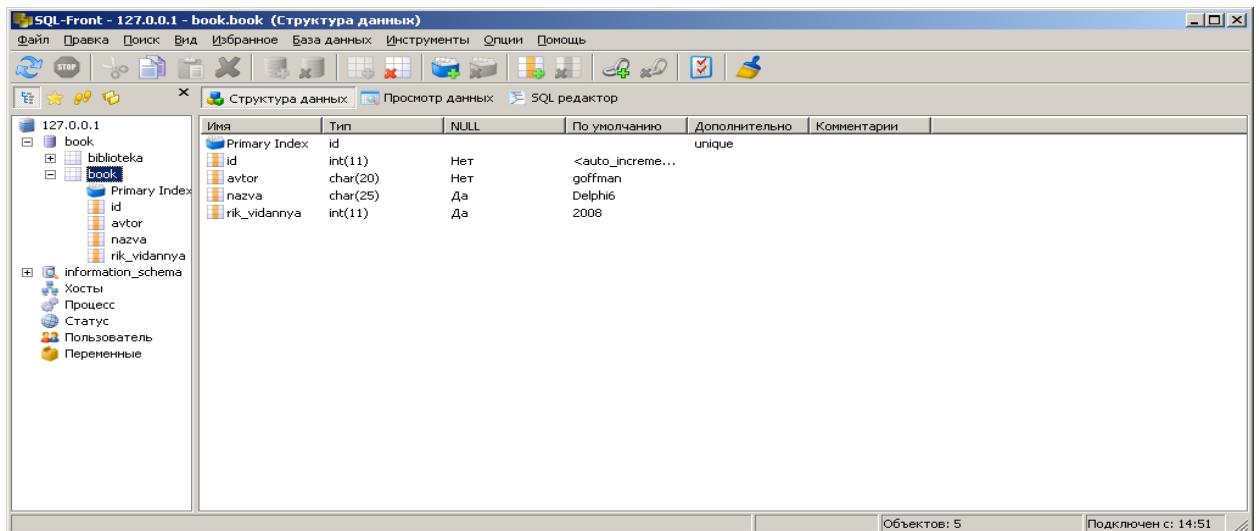


Рисунок 8. Перегляд списку полів таблиці *book* (структура даних)

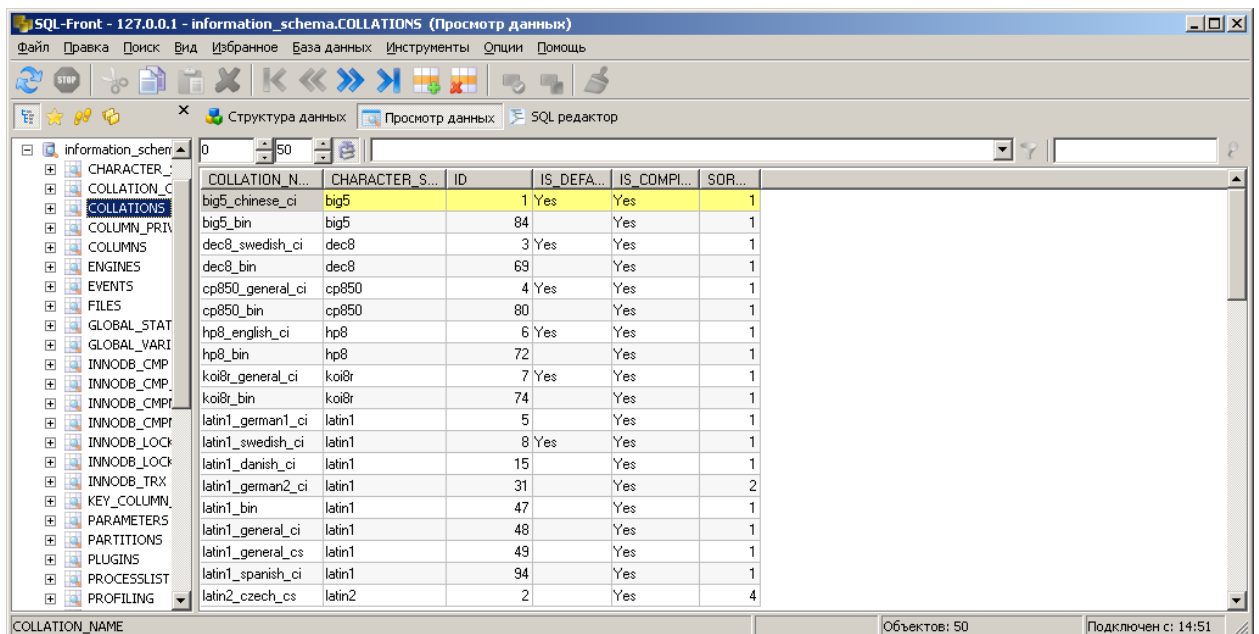


Рисунок 9. Перегляд даних таблиці *COLLATIONS*

клацнувши на них або в контекстному меню обрати *Властивості*.

Для створення нової таблиці зручно використовувати контекстне меню бази даних, до якої треба додати таблицю. В меню слід обрати *Створити / Таблиця*. При створенні таблиці необхідно задати ім'я таблиці та визначити її поля (Рисунки 10, 11 а-в).

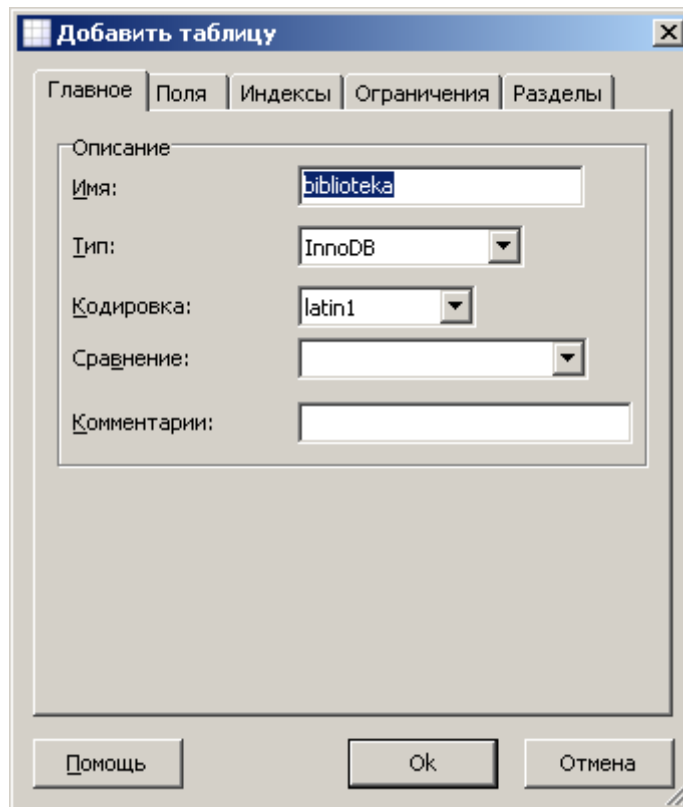


Рисунок 10. Створення нової таблиці

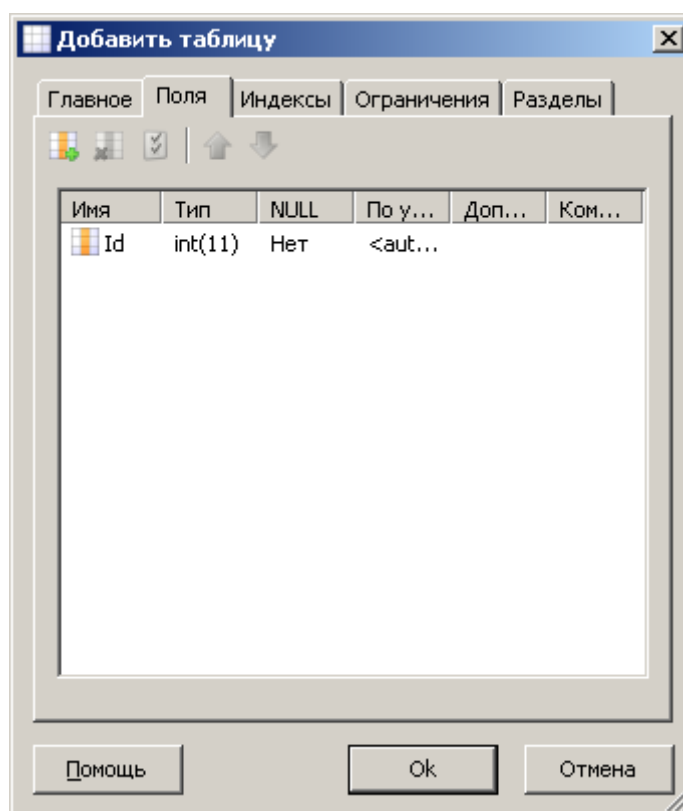


Рисунок 11(а). Визначення полів нової таблиці



**Добавить поле**

Описание

Позиция: После поля "Id"

Имя: Delphi7

Тип: VarChar

Длина: 255

По умолчанию: ☒ <NULL> ☐

Кодировка: latin1

Сравнение: latin1\_swedish\_ci

Комментарии:

Атрибуты

☒ NULL разрешен

Помощь Ok Отмена

Рисунок 11(б). Визначення полів нової таблиці

**Добавить таблицу**

Главное Поля Индексы Ограничения Разделы

Имя	Тип	NULL	По у...	Доп...	Ком...
Id	int(11)	Нет	<aut...		
Del...	varc...	Да	<NU...		

Помощь Ok Отмена

Рисунок 11(в). Визначення полів нової таблиці

Для створення ключів та індексів необхідно обрати в головному меню **База даних / Властивості / Таблица** і закладку **Индексы** (Рисунки 12 та 13).

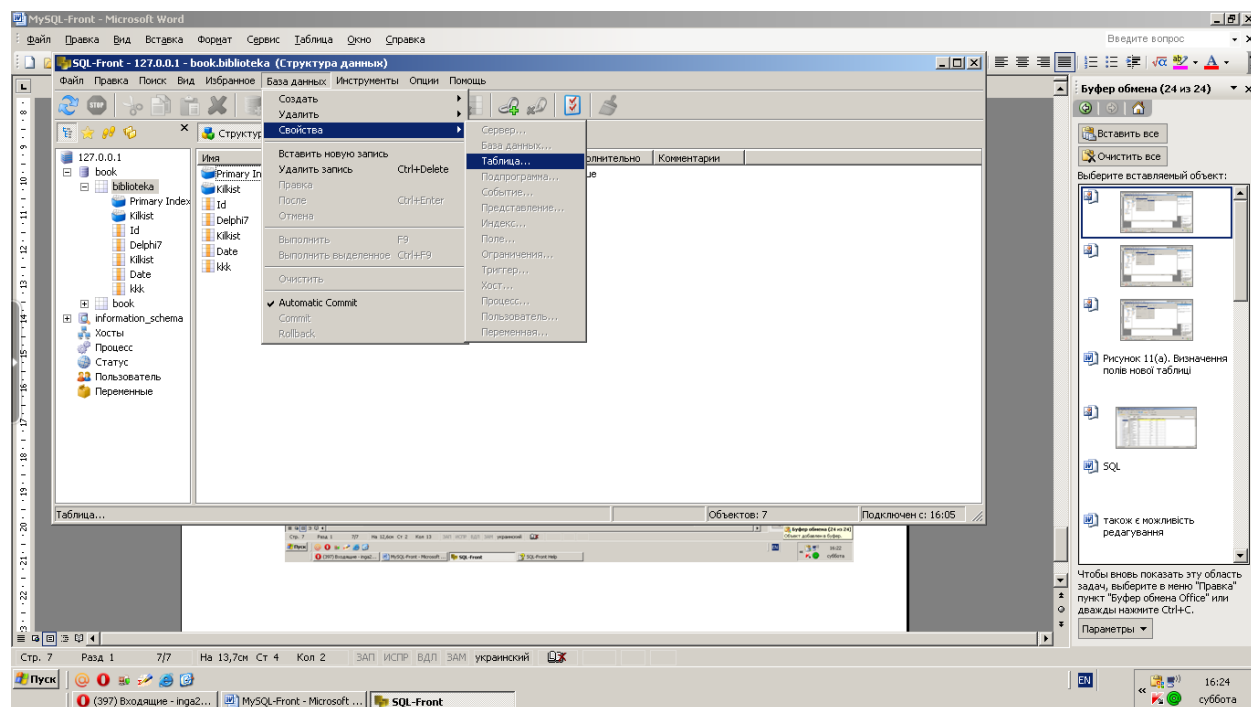


Рисунок 12. Команда **База даних** головного меню

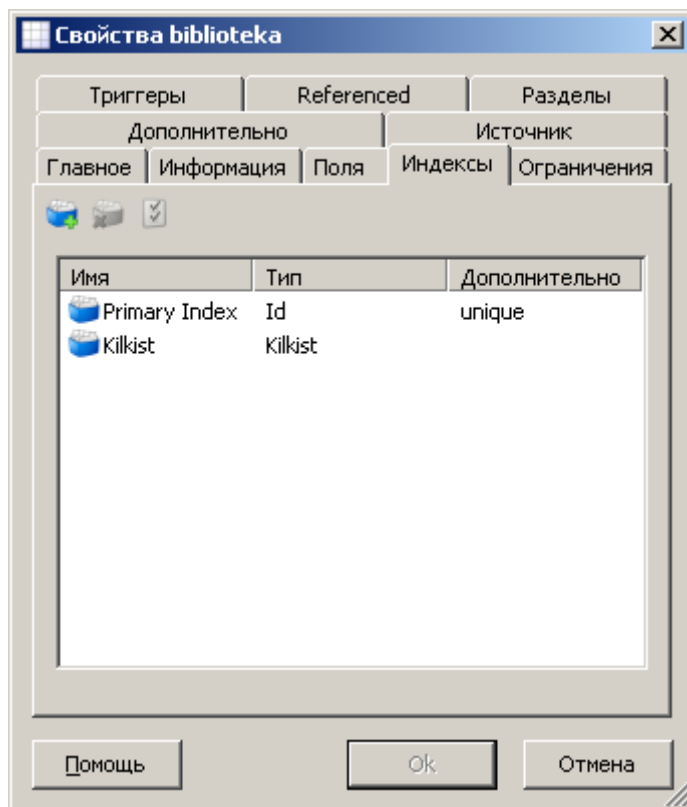


Рисунок 13. Створення ключів та індексів

## Порядок виконання роботи

1. За допомогою **MySQL-Front** створити у своїй базі даних таблицю **Group**, в таблиці повинні міститися дані про студентів Вашої групи. Поля таблиці придумати самостійно (наприклад *id*, *name*, *surname*, *father\_name*, *born\_year* тощо). Ввести у таблицю не менше десяти записів.
2. Створити таблицю **Предмети (Subjects)** з полями:

- 1) номер по порядку (*id*);
- 2) назва (*name*).

Внести в таблицю 3 – 4 навчальних предмети, з яких у Вас були екзамени минулого семестру.

3. Створити таблицю **Оцінки (Marks)**.

- 1) *id* – номер по порядку;
- 2) *student\_id* – номер студента з таблиці **Group**;
- 3) *subject\_id* – номер предмету з таблиці **Subjects**;
- 4) *mark* – оцінка, отримана студентом за вказаний предмет минулого семестру.

Заповнити таблицю **Marks**, внісши до неї оцінки студентів з таблиці **Group**, з предметів таблиці **Subjects**, і проставивши за них оцінки, які студенти отримали за минулий семестр.

4. За допомогою мови **SQL** створити таблицю **book** з такими полями:

- 1) *id* – номер по порядку;
- 2) *name* – назва книги;
- 3) *author* – автор книги;
- 4) *print\_year* – рік видання;
- 5) *publisher* – видавництво.

**SQL**-запити, які використовувалися при виконанні завдання, зберегти в окремому файлі.

5. За допомогою тільки мови **SQL** розділити відношення **book** так, щоб:

- 1) автори і видавництва були розміщені в окремих таблицях, а не в таблиці **book**;

2) база даних повинна підтримувати введення кількох авторів для однієї книги (враховуючи і попередній пункт).

*SQL*-запити, які використовувалися при виконанні завдання, зберегти в окремому файлі.

6. Для бази даних, створеної згідно індивідуального завдання, вміти використовувати основні можливості, які надає програма *MySQL-Front*: створення таблиці, редагування, перегляд структури, виконання *SQL*-запитів, створення ключів та індексів, додавання та знищення полів, вставка та знищення записів тощо.
7. Оформити короткий звіт про виконану роботу, внести в нього основні теоретичні відомості.

### **Оформлення звіту:**

1. Титульний лист.
2. Мета роботи.
3. Порядок виконання.
4. Представлення основних результатів по виконаній лабораторній роботі.
5. Висновки.

### **Контрольні питання**

1. Що таке *MySQL-Front*? Її призначення.
2. Як запустити *MySQL-Front*? Що для цього потрібно зробити?
3. Як виглядає вікно програми *MySQL-Front*?
4. Які режими роботи має права частина вікна програми *MySQL-Front*?
5. Як створити нову таблицю?
6. Як вказати поля нової таблиці?
7. Як створити ключі та індекси?

### СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пасічник В.В., Резніченко В. А. Організація баз даних та знань.—К. : Видавнича група «ВНУ», 2006. —384 с.
2. Базы даних в інформаційних системах : підруч. / В. І. Гайдаржи, І. В. Ізварін. - К. : Ун-т Україна, 2018. - 418 с.
3. Балик Н.Р., Мандзюк В.І. Базы даних MySQL: Навчальний посібник. — Тернопіль: «Навчальна книга – Богдан», 2010.— 160 с.
4. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 1. Організація баз даних та знань. – «Комп'ютинг», 2006. – 460с.
5. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Організація баз даних та знань. – «Комп'ютинг», 2006. – 590с.
6. Коннолли Т., Бегг К., Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : Пер. с англ. — М.: Издательский дом «Вильямс», 2016. — 1440с.

Гарнітура Times New Roman. Папір офсетний.  
Формат видання 60х84/16.  
Умов. друк. арк.4.54 Зам. № 21. Наклад 50 прим.

Видруковано ПП «АУТДОР - ШАРК»  
88000, м. Ужгород, пл. Жупанатська, 15/1. Тел.: 3-51-25. E-mail: [office@shark.com.ua](mailto:office@shark.com.ua)

*Свідоцтво про внесення суб'єкта видавничої справи  
до державного реєстру видавців,  
виготівників і розповсюджувачів  
видавничої продукції*  
Серія 3т № 40 від 29 жовтня 2012 року