ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
МІНІСТЕРСТВА ОСВІТИ І НАУКИ УКРАЇНИ

С.В. Чопоров, І. В. Синєпольський

БАЗИ ДАНИХ

Навчально-методичний посібник до лабораторних занять
для студентів освітньо-кваліфікаційного рівня «бакалавр»
напряму підготовки «Програмна інженерія»

Затверджено
Вченою радою ЗНУ
Протокол № від р.

Запоріжжя
2015

УДК 000 (000.0)
ББК 0000.0а00
Ч000

Чопоров С.В. Бази даних: навчально-методичний посібник до лабораторних занять для студентів освітньо-кваліфікаційного рівня «бакалавр» напряму підготовки «Програмна інженерія» / С.В. Чопоров, І. В. Синєпольський. – Запоріжжя: ЗНУ, 2015. – ## с.

Навчально-методичний посібник до виконання лабораторних робіт з дисципліни «Бази даних» пропонується студентам, що навчаються за програмою освітньо-кваліфікаційного рівня «бакалавр» за напрямом підготовки «Програмна інженерія» та рекомендується для закріплення навчального матеріалу з дисципліни циклу професійної та практичної підготовки.

Посібник включає теоретичний матеріал з кожної теми дисципліни та зміст основних лабораторних робіт, що пропонуються при вивченні даного курсу, а також контрольні питання. Увага студентів акцентується на розширенні теоретичних знань та набутті практичних навичок створення, управління та взаємодією з базами даних.

Навчально-методичний посібник призначений для студентів, викладачів, а також усіх, хто цікавиться питаннями зазначеної тематики.

Рецензент *С.Ю. Борю*

Відповідальний за випуск *С.І. Гоменюк*

Зміст

Вступ.....5
Загальні вказівки.....6
Лабораторна робота № 1. Створення однотабличної бази даних.....7
Лабораторна робота № 2. Нормальні форми. Заповнення бази даних.....10
Лабораторна робота № 3. Відношення між таблицями “один до багатьох”.....12
Лабораторна робота № 4. Відношення між таблицями “багато до багатьох”.....16
Лабораторна робота № 5. Створення форм для введення даних.....19
Лабораторна робота № 6. Створення звітів.....28
Лабораторна робота № 7. Знайомство з РСКБД MySQL. Створення бази даних 29
Лабораторна робота № 8. Мова опису даних SQL. Створення структури даних в MySQL.....35
Лабораторна робота № 9. Маніпулювання даними: вставка, оновлення, видалення.....43
Лабораторна робота № 10. Маніпулювання даними: вибірка, групові операції, імпорт/експорт.....47
Індивідуальне завдання.....51
Глосарій.....53
Рекомендована література.....54
Додаток А. Назва додатку.....55

Вступ

База даних (БД) – це організований набір даних. Якщо дані розміщені на ЕОМ, то для взаємодії з базою зазвичай використовується система керування базами даних (СКБД) – комп'ютерна програма чи комплекс програм для забезпечення доступу, створення, редагування та видалення даних користувачами.

Мета курсу: набуття теоретичних знань, необхідних для використання основних підходів організації і функціонування СКБД, та формування практичних навичок з їх реалізації засобами мови визначення даних, мови маніпулювання даних та засобами програмних інтерфейсів у студентів напряму підготовки «Програмна інженерія», які можуть бути використані при подальшому навчанні, професійній, виробничій та науковій діяльності випускника вузу. Дисципліна «Бази даних» базується на знаннях дисциплін «Методи та засоби комп'ютерних інформаційних технологій», «Основи програмування та інформаційна культура студентів» та інших, що викладаються за програмою підготовки бакалавра галузі знань 0501 – «Інформатика та обчислювальна техніка».

Навчально-методичний посібник містить 12 лабораторних робіт, у яких потрібно розробити структуру баз даних, налаштувати відповідним чином СКБД та розробити прикладні програмні засоби для маніпулювання даними користувачем.

Завдання навчальної дисципліни: оволодіти основними принципами та методами організації і функціонування сучасних СКБД. Дати необхідну практичну підготовку та знання для подальшого їх застосування при вивченні спеціальних дисциплін та професійній підготовці.

Загальні вказівки

1. Лабораторні роботи виконуються в години, зазначені в розкладі і для кожного студента присутність на занятті є обов'язковою. Студенту, що пропустив лабораторне заняття без поважних причин і не захистив лабораторну роботу на наступному занятті, виставляється незадовільна оцінка з відповідної лабораторної роботи.

2. У комп'ютерний клас дозволяється входити тільки після дзвінка на заняття й у присутності викладача.

3. Вхід у комп'ютерний клас дозволяється тільки за наявністю документа, завіреного відповідальним органом ЗНУ, що засвідчує особистість .

4. Лабораторну роботу припиняють виконувати за дзвінком.

5. Забороняється виходити з аудиторії без дозволу викладача.

Вимоги до виконання лабораторних робіт

1. Ознайомитися з загальними теоретичними відомостями.
2. За номером варіанта[1] обрати завдання.
3. Уважно прочитати завдання.
4. Кожну роботу виконувати відповідно до завдання.
5. Оформити друкований звіт з виконання лабораторної роботи, який повинен містити:

– титульний лист;
– тему роботи;
– схеми основних алгоритмів;
– послідовність виконання роботи;
– при необхідності записуються теоретичні відомості, результати втілення та аналіз отриманих результатів;
– висновки.

6. При необхідності захистити результати, відповісти на додаткові питання та пред'явити виконану роботу.

[1] Номер варіанта визначає викладач.

Лабораторна робота № 1. Створення однотабличної бази даних

Теоретичні відомості

Система керування базами даних (СКБД) – комп’ютерна програма чи комплекс програм для забезпечення доступу, створення, редагування та видалення даних користувачами.

Основні характеристики СКБД

1. Несуперечливість даних.
2. Підтримка цілісності бази даних.
3. Цілісність описується за допомогою обмежень.
4. Незалежність прикладних програм від даних.
5. Спільне використання даних.
6. Підвищений рівень безпеки.

Більшість баз даних призначені для моделювання певної частини реального світу, яку називають предметної областю (ПрО). На логічному рівні об'єкти в ПрО є сутностями (entities) і пов'язані між собою через відношення (relationships). На фізичному рівні СКБД подає сутності у вигляді таблиць, а відношення – у вигляді обмежень зовнішнього ключа.

Таблиця — це набір елементів даних (значень), які організовані з використанням моделі вертикальних стовпців (з різними іменами) і горизонтальних рядків. Таблиця має визначену кількість стовпців на відміну від рядків. Кожен рядок ідентифікується особливим набором колонок який називається потенційним ключем.

Первинний ключ — атрибут, або набір атрибутів (стовпчиків), що однозначно ідентифікує рядок таблиці. Первинний ключ обов'язково унікальний, він єдиний і найголовніший із унікальних ключів. Якщо з ключем однієї таблиці пов’язані ключі інших таблиць, то такий ключ називається зовнішнім.

Існує багато різноманітного програмного забезпечення, яке тим чи іншим чином реалізують ідею системи керування бази даних. Одним із таких є компонент Libreoffice Base — безкоштовний аналог Microsoft Access. LibreOffice — вільний та крос-платформовий офісний пакет, який працює на операційних системах Microsoft Windows, Gnu/Linux та Mac OS X і є одним з провідних вільних аналогів Microsoft Office.

Основним форматом файлу LibreOffice є відкритий формат офісних документів OpenDocument, версія 1.1 якого була затверджена як міжнародний стандарт ISO/IEC 26300:2006/Amd 1:2012; окрім того, LibreOffice підтримує формати Microsoft Office та інших офісних пакетів для досягнення максимальної сумісності.

Хід роботи

Завантаження

Завантажити останню стабільну версію LibreOffice можливо за посиланням http://download.documentfoundation.org/libreoffice/stable/ або придбати завантажувач на загальнодоступному сервері кафедри.

Створення бази даних в СКБД LibreOffice Base

Під час першого запуску LibreOffice Base пропонується створити нову базу даних. Можна використати вбудовану СКБД HSQLDB.

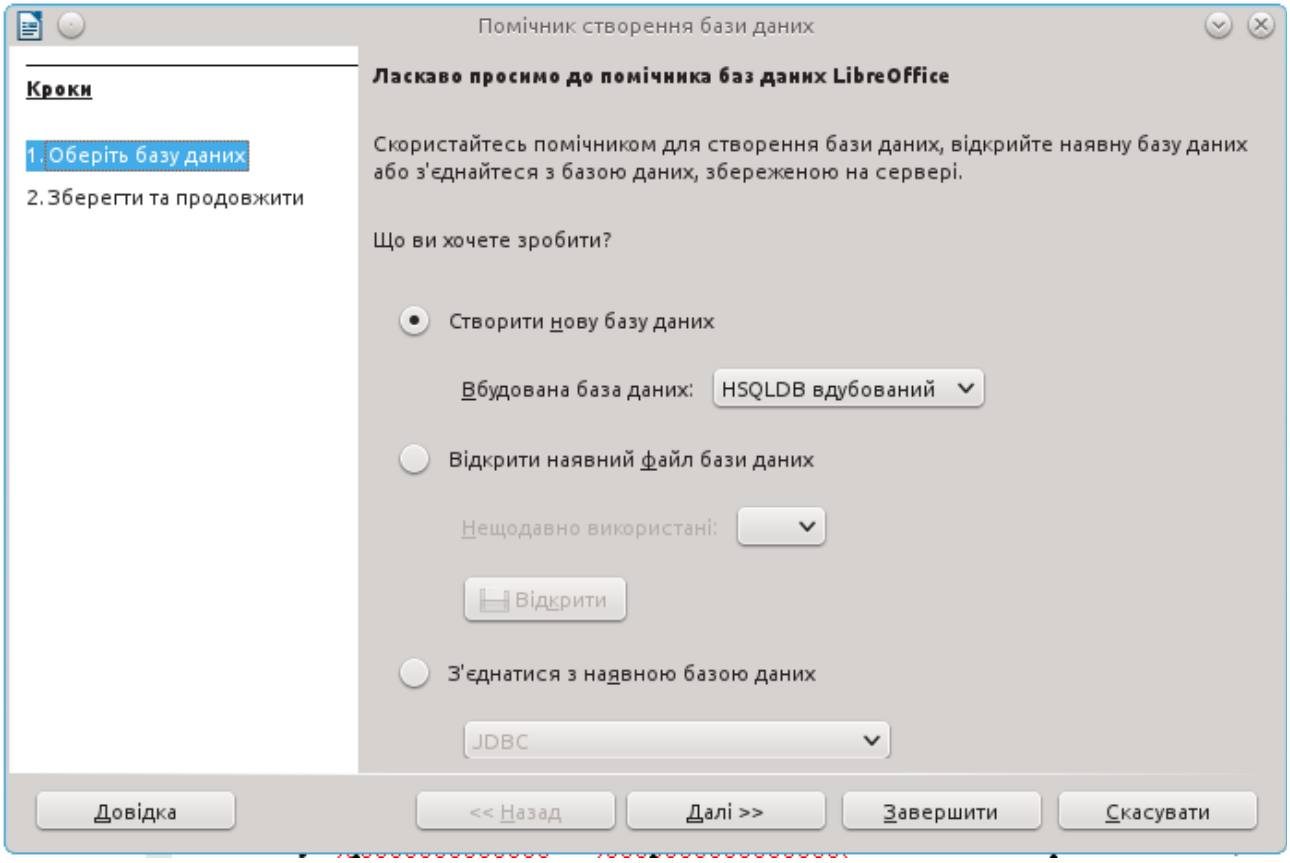
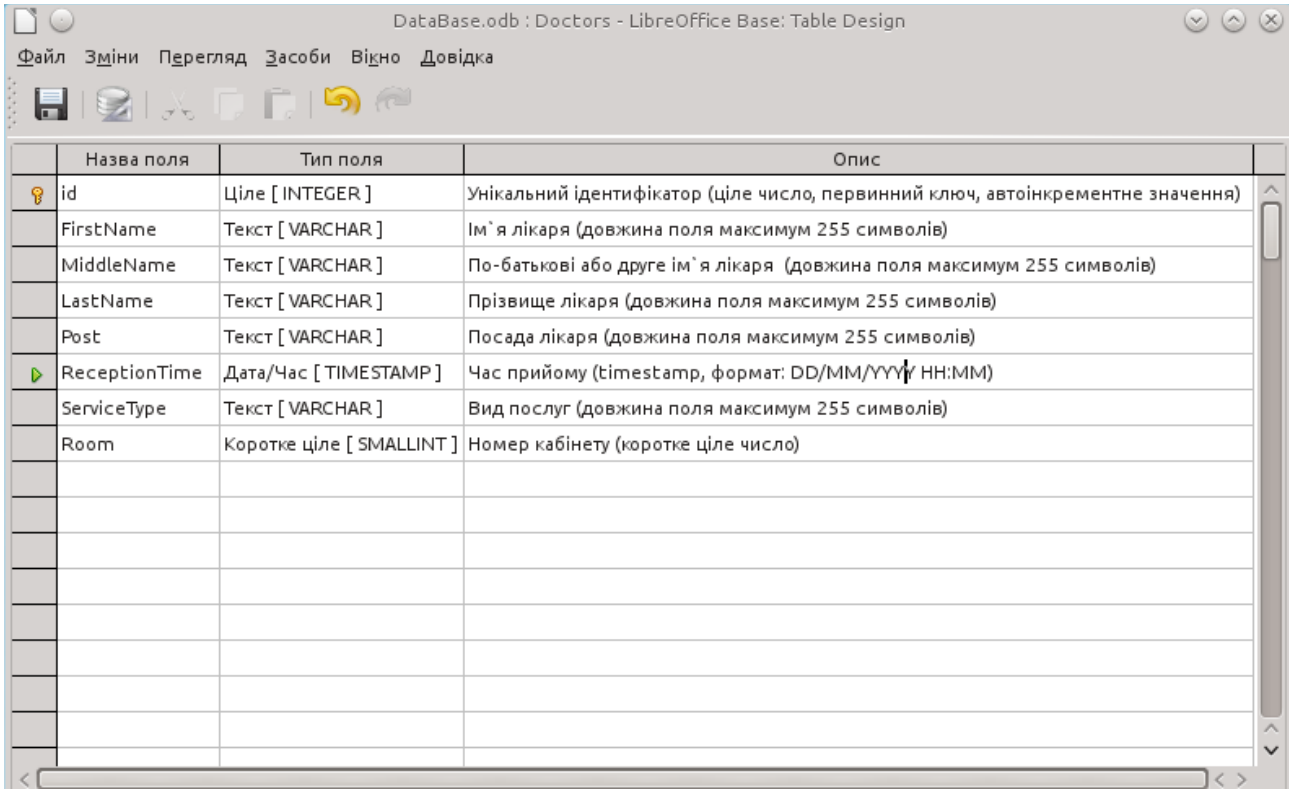


Рис. 1. Початковий діалог створення нової БД

Створення таблиці

Створимо нову таблицю в режимі дизайну з полями, зазначеними на рисунку 2, зберігаємо з назвою Doctors.



Назва поля	Тип поля	Опис
id	Ціле [INTEGER]	Унікальний ідентифікатор (ціле число, первинний ключ, автоінкрементне значення)
FirstName	Текст [VARCHAR]	Ім`я лікаря (довжина поля максимум 255 символів)
MiddleName	Текст [VARCHAR]	По-батькові або друге ім`я лікаря (довжина поля максимум 255 символів)
LastName	Текст [VARCHAR]	Прізвище лікаря (довжина поля максимум 255 символів)
Post	Текст [VARCHAR]	Посада лікаря (довжина поля максимум 255 символів)
ReceptionTime	Дата/Час [TIMESTAMP]	Час прийому (timestamp, формат: DD/MM/YYYY HH:MM)
ServiceType	Текст [VARCHAR]	Вид послуг (довжина поля максимум 255 символів)
Room	Коротке ціле [SMALLINT]	Номер кабінету (коротке ціле число)

Рис. 2. Створення нової таблиці БД

✍ *Завдання для лабораторної роботи*

Створити БД і таблицю в ній на довільну тему з обов'язковими атрибутами:

– цілочисельне поле — первинний автоінкрементний ключ;

– декілька полів строкового типу;

– хоча б одне цілочисельне поле;

– хоча б одне поле у форматі дата/час;

– для кожного поля має бути опис його значення в таблиці і параметри типу;

– імена полів мають бути на латиниці.

? *Контрольні запитання*

1. Дайте визначення, що таке система керування базами даних, наведіть приклади.
2. Які існують моделі даних?
3. Які основні принципи використовуються у реляційній моделі даних?
4. Дайте основну характеристику LibreOffice Base.
5. Які типи даних можна задати при створенні таблиці в LibreOffice Base?

Лабораторна робота № 2. Нормальні форми. Заповнення бази даних

Теоретичні відомості

Нормальна форма — властивість таблиці, що характеризує її з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти таблиця.

Перша нормальна форма (1НФ, 1NF) утворює ґрунт для структурованої схеми баз даних. Таблиця знаходиться у першій нормальній формі, якщо виконуються наступні вимоги.

– Кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис.

– Уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість раз в різних записах) правильно визначаючи неключові атрибути.

– Атомарність: кожна колонка повинна мати лише одне значення, а не множину значень.

Друга нормальна форма визначається на основі першої з додатковою умовою: усі неключові колонки повинні залежати від усіх ключових колонок, а не лише від частини ключових. Зазвичай сурогатний ключ, тобто стовпчик таблиці, який виконує лише роль унікального ідентифікатора рядка і є первинним ключем, повністю задовольняє цю вимогу.

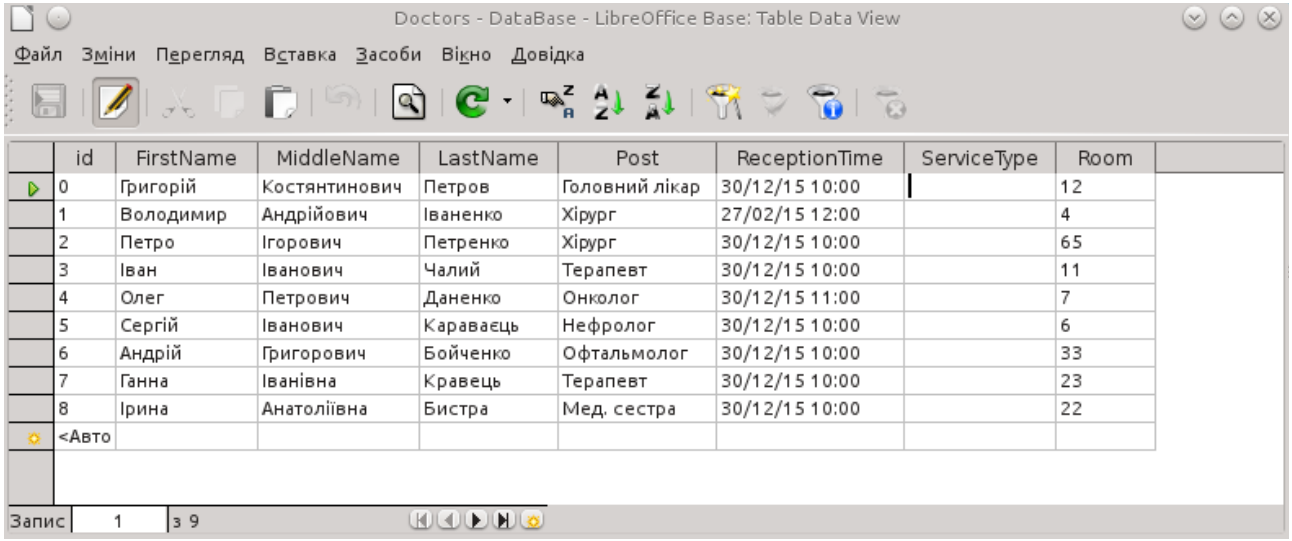
Третя нормальна форма також визначається на основі попередньої і також має додаткову умову у своєму визначенні: будь-яка колонка (атрибут) таблиці функціонально залежить лише від первинного ключа. Простіше кажучи, усі неключові атрибути, зміст яких може відноситись до декількох записів необхідно виносити в інші таблиці.

Інші нормальні форми: Бойса — Кодда, четверта, п’ята, НФ “домен-ключ”, шоста накладають більш суворі вимоги до структури таблиць і відношень між ними, але для нормального функціонування в більшості випадків достатньо реалізувати перші три.

Хід роботи

Заповнення даними

При заповненні даними можуть бути поміченими закономірності, які разом з попереднім аналізом предметної області дають представлення для побудови остаточної структури бази даних.



id	FirstName	MiddleName	LastName	Post	ReceptionTime	ServiceType	Room
0	Григорій	Костянтинович	Петров	Головний лікар	30/12/15 10:00		12
1	Володимир	Андрійович	Іваненко	Хірург	27/02/15 12:00		4
2	Петро	Ігорович	Петренко	Хірург	30/12/15 10:00		65
3	Іван	Іванович	Чалий	Терапевт	30/12/15 10:00		11
4	Олег	Петрович	Даненко	Онколог	30/12/15 11:00		7
5	Сергій	Іванович	Караваєць	Нефролог	30/12/15 10:00		6
6	Андрій	Григорович	Бойченко	Офтальмолог	30/12/15 10:00		33
7	Ганна	Іванівна	Кравець	Терапевт	30/12/15 10:00		23
8	Ірина	Анатоліївна	Бистра	Мед. сестра	30/12/15 10:00		22
<Авто							

Рис. 3. Введення даних

✍ *Завдання для лабораторної роботи*

Заповніть даними таблицю (не менше 50 записів), занотуйте назви полів, де строкові дані повторюються або може порушитися їх логічна атомарність. Перевірте виконання умов перших трьох нормальних форм.

❓ *Контрольні запитання*

1. Які особливості має атрибут із статусом первинного ключа?
2. Що таке атомарність даних і які наслідки її порушення?
3. Що таке нормальна форма? Назвіть умови побудови структури таблиць для перших трьох нормальних форм.

Лабораторна робота № 3. Відношення між таблицями “один до багатьох”

Теоретичні відомості

Існує три типи відношень: один-до-одного, при якому кожний рядок таблиці пов'язаний з нулем або одним рядком іншої таблиці; один-до-багатьох, при якому кожний рядок таблиці пов'язаний з нулем, однією або декількома рядками іншої таблиці, і багато-до-багатьох, при якому кожний рядок першої таблиці пов'язаний з нулем, однією або декількома рядками другої таблиці, а кожен рядок другої таблиці може бути пов'язаний з нулем, однією або декількома рядками першої таблиці

Відношення один-до-одного зустрічаються досить рідко. Як правило, вони використовуються у разі, коли набір атрибутів застосовується до невеликої кількості примірників сутності.

Відношення один-до-багатьох досить широко поширені. При побудові інтерфейсів це відношення також називають «головний-детальний» (master-detailed) (головний – «один», детальний – «багато»).

Хід роботи

Очевидно, що існує зв'язок між лікарськими посадами і самим лікарями, при чому посада являє собою окрему сутність, незалежну від персональних даних лікаря. Між сутностями “Посада” і “Лікар” виникає залежність “один до багатьох”, тобто до однієї посади можуть відноситись багато лікарів, але для одного лікаря притаманна лише одна посада. Тому створимо таблицю Posts, в якій будуть зберігатися дані лікарських професій.

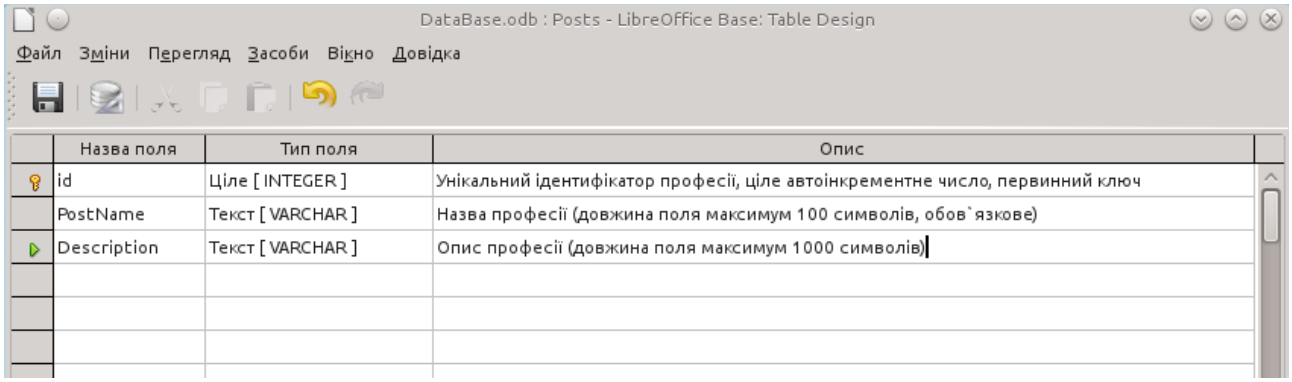


Рис. 4. Окрема таблиця Posts

Тепер таблицю треба заповнити даними. При чому обов'язково треба вказати вже наявні професії із даних попередньої таблиці.

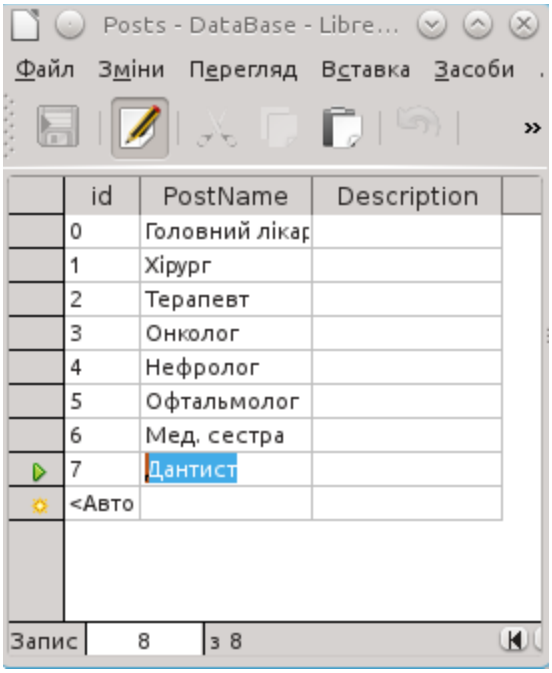


Рис. 5. Заповнення даними таблиці Posts

Далі доведеться замінити атрибут Post із таблиці Doctors на цілочисельний, при чому його тип має бути в точності такий, як у первинного ключа таблиці Posts і вставити замість назви професії відповідне число — унікальний ідентифікатор Posts (поле id).

Назва поля	Тип поля	Опис
id	Ціле [INTEGER]	Унікальний ідентифікатор (ціле число, первинний ключ, автоінкрементне значення)
FirstName	Текст [VARCHAR]	Ім`я лікаря (довжина поля максимум 255 символів)
MiddleName	Текст [VARCHAR]	По-батькові або друге ім`я лікаря (довжина поля максимум 255 символів)
LastName	Текст [VARCHAR]	Прізвище лікаря (довжина поля максимум 255 символів)
ReceptionTime	Дата/Час [TIMESTAMP]	Час прийому (timestamp, формат: DD/MM/YYYY HH:MM)
ServiceType	Текст [VARCHAR]	Вид послуг (довжина поля максимум 255 символів)
Room	Коротке ціле [SMALLINT]	Номер кабінету (коротке ціле число)
Post	Ціле [INTEGER]	Посада лікаря (ідентифікатор з таблиці Posts)

Рис. 6. Зміна типу стовпчика Post в таблиці Doctors

id	FirstName	MiddleName	LastName	ReceptionTime	ServiceType	Room	Post
0	Григорій	Констянтинович	Петров	30/12/15 10:00		12	0
1	Володимир	Андрійович	Іваненко	27/02/15 12:00		4	1
2	Петро	Ігорович	Петренко	30/12/15 10:00		65	1
3	Іван	Іванович	Чалий	30/12/15 10:00		11	2
4	Олег	Петрович	Даненко	30/12/15 11:00		7	3
5	Сергій	Іванович	Караваєць	30/12/15 10:00		6	4
6	Андрій	Григорович	Бойченко	30/12/15 10:00		33	5
7	Ганна	Іванівна	Кравець	30/12/15 10:00		23	2
8	Ірина	Анатоліївна	Бистра	30/12/15 10:00		22	6
<Авто							

Рис. 7. Заповнення даними таблиці Doctors з урахуванням зв'язку з таблицею Posts

Після цього засобами Libreoffice Base треба створити зв'язок між таблицями встановленням залежності між атрибутом Post таблиці Doctors і id таблиці Posts (меню Засоби / Зв’язки) .

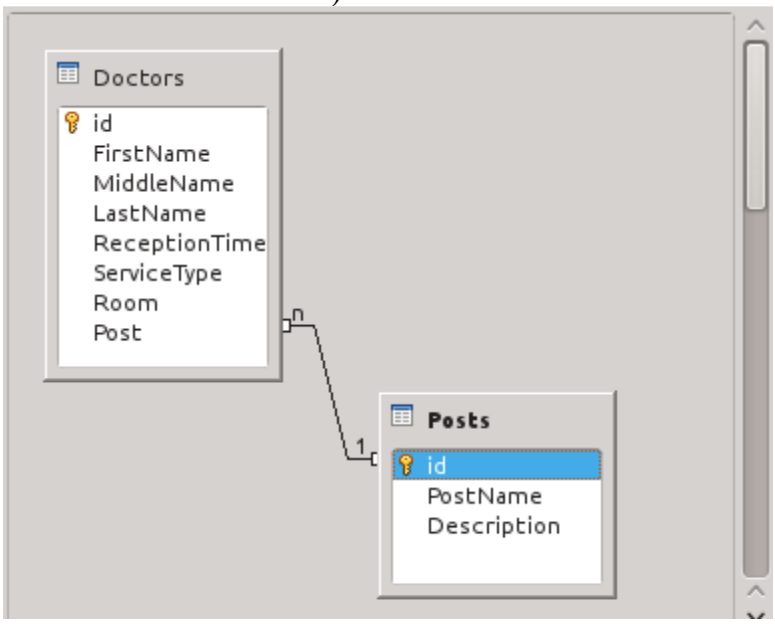


Рис. 8. Встановлення зв'язку між таблицями Doctors і Posts

Якщо намагатися оновити значення атрибуту Post таблиці Doctors на не існуючий в таблиці Posts, то середовище поверне помилку.

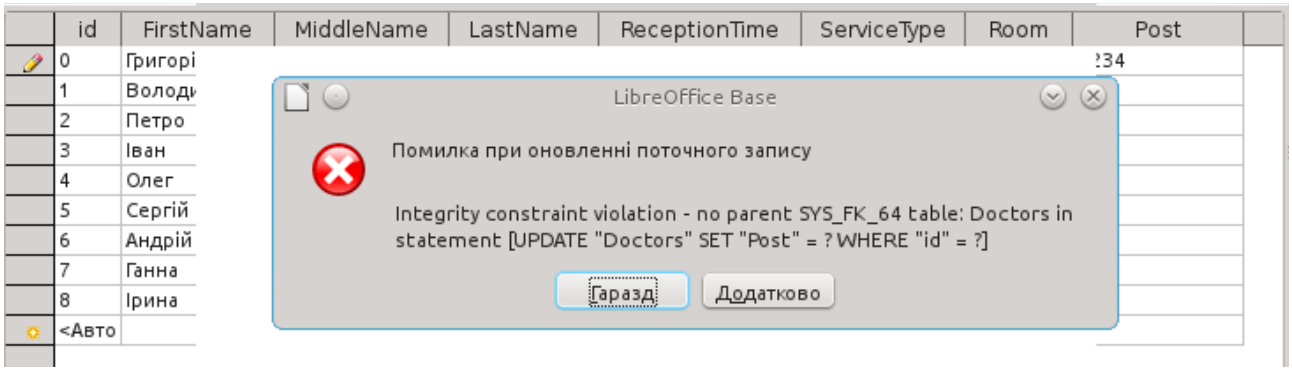


Рис. 9. Помилка при введенні невірного значення атрибуту Post

Також не можна змінювати або видаляти пов'язані записи із таблиці Posts.

✍ *Завдання для лабораторної роботи*

Створити тематичні допоміжні таблиці (довідники), заповнити їх, зв'язати з основною таблицею відношенням “один до багатьох”. Результати оформити у вигляді звіту.

? *Контрольні запитання*

1. Які реляційні відношення вам відомі?
2. Які особливості має відношення “один до багатьох”?

Лабораторна робота № 4. Відношення між таблицями “багато до багатьох”

Теоретичні відомості

Для реалізації відношення багато-до-багатьох використовується таблиця-посередник (вузлова таблиця). Вузлова таблиця складається з первинних ключів таблиць з кожної зі сторін відносини. Відношення один-до-багатьох встановлюється між вузловою таблицею і кожною з оригінальних таблиць. Тож відношення багато-до-багатьох реалізується з використанням вузлової таблиці.

Хід роботи

При більш детальному дослідженні предметної області додалась сутність “Пацієнт”, при чому декілька лікарів лікують одного пацієнта і декілька пацієнтів можуть лікуватися в одного лікаря, тобто виникає відношення “багато до багатьох” між сутностями “Пацієнт” і “Лікар”. Таке ж відношення виникає між сутностями “Пацієнт” і “Хвороба”.

Щоб коректно встановити взаємовідношення “багато до багатьох”, необхідно додатково створити так звані слабкі сутності: сутність “Лікар — Пацієнт” і сутність “Пацієнт — Хвороба”.

Створюємо для вище вказаних сутностей таблиці Patients, Diseases, DoctorPatient, PatientDesease.

	Назва поля	Тип поля	Опис
	id	Ціле [INTEGER]	Унікальний ідентифікатор, ціле автоінкрементне число, первинний ключ
	FullName	Текст [VARCHAR]	ПІБ пацієнта (максимальна довжина: 255, обов`язкове поле)
	BornDate	Дата [DATE]	Дата народження пацієнта (DD/MM/YYYY, обов`язкове поле)
	Comment	Текст [VARCHAR]	Додаткові відмітки (максимальна довжина: 1000)

Рис. 10. Структура таблиці Patients

Назва поля	Тип поля	Опис
id	Ціле [INTEGER]	Унікальний ідентифікатор, ціле автоінкрементне число, первинний ключ
DiseaseName	Текст [VARCHAR]	Назва хвороби (максимальна довжина: 255, обов`язкове поле)
Description	Текст [VARCHAR]	Опис хвороби (максимальна довжина: 1000)

Рис. 11. Структура таблиці Diseases

Назва поля	Тип поля	Опис
DoctorID	Ціле [INTEGER]	Ідентифікатор лікаря
PatientID	Ціле [INTEGER]	Ідентифікатор пацієнта

Рис. 12. Структура таблиці DoctorPatient

Назва поля	Тип поля	Опис
PatientID	Ціле [INTEGER]	Ідентифікатор пацієнта
DiseaseID	Ціле [INTEGER]	Ідентифікатор хвороби

Рис. 13. Структура таблиці PatientDisease

Зверніть увагу: в таблицях DoctorPatient і PatientDisease створюється складний первинний ключ, складений із двох атрибутів — ідентифікаторів із інших відповідних таблиць.

Тепер створюємо зв’язки між таблицями.

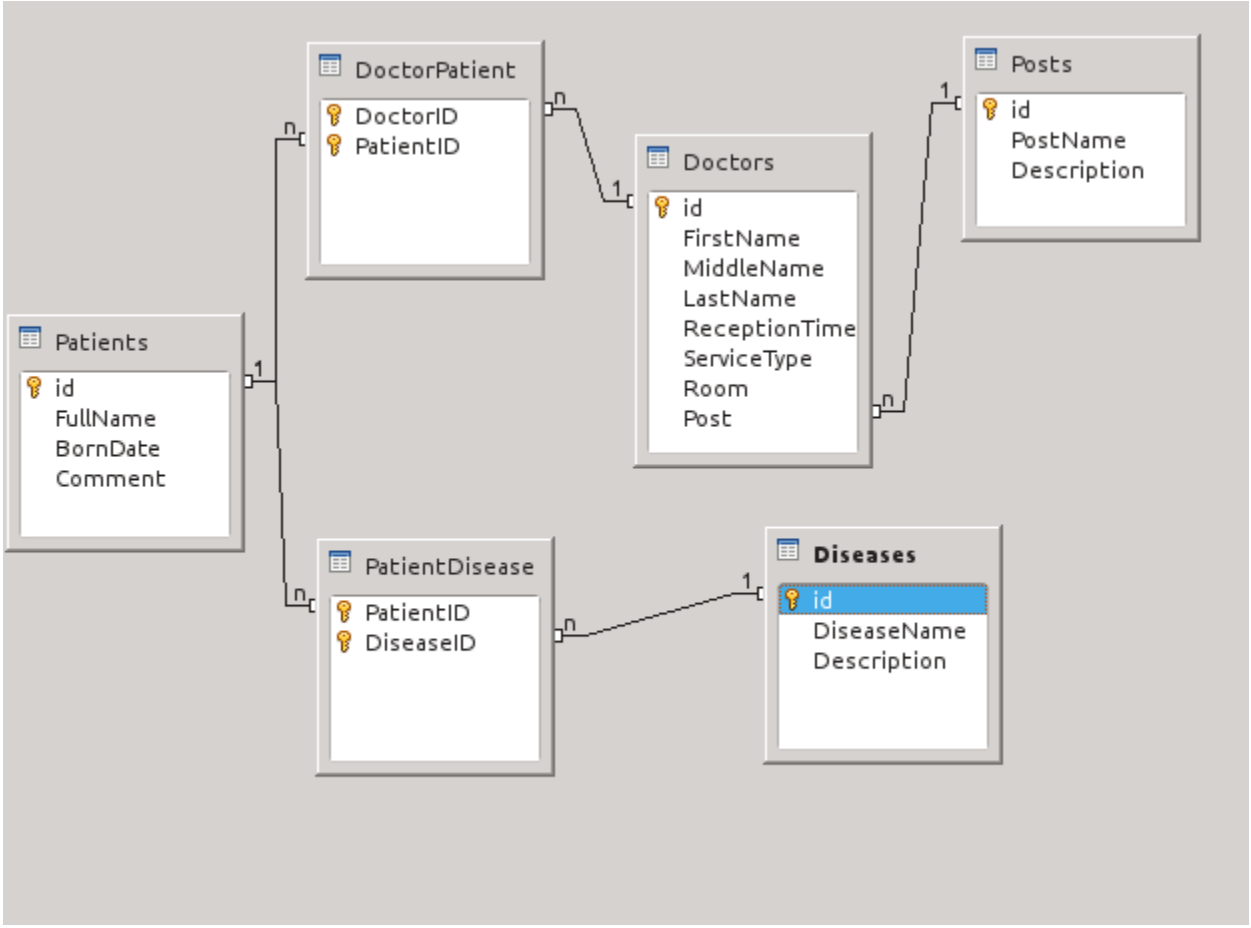


Рис. 14. Встановлення зв'язків

✍ *Завдання для лабораторної роботи*

Створити додатково 2 таблиці, заповнити їх, зв'язати між собою відношенням “багато до багатьох” з використанням слабкої сутності. Заповнити даними. Результати оформити у вигляді звіту.

❓ *Контрольні запитання*

1. Які реляційні відношення вам відомі?
2. Які особливості має відношення “багато до багатьох”?
3. Що таке слабка сутність? Яка її роль у реляційних відношеннях?

Лабораторна робота № 5. Створення форм для введення даних

Теоретичні відомості

Інтерфейс користувача (англ. User Interface, UI) — засіб зручної взаємодії користувача з інформаційною системою.

Інтерфейс користувача — сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача; у графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон, гнучким налаштовуванням як самих вікон, так і окремих їхніх елементів (файли, теки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Форма являє собою деякий електронний бланк, у якому є поля для введення даних. Складач вводить дані в ці поля, і дані автоматично заносяться в таблиці бази.

Дані в таблицю можна вносити і без допомоги яких-небудь форм, але існують принаймні чотири причини, що роблять форми незамінним засобом введення даних у базу.

По-перше, малокваліфікованому персоналу не можна давати доступ до таблиць (найціннішому з того, що є в базі). Уявіть, що буде, якщо новачок «наведе порядок» у банковій таблиці, що містить розрахункові рахунки клієнтів.

По-друге, різні люди можуть мати різні права доступу до інформації, що зберігається в таблицях. Наприклад, один має право вводити тільки імена й адреси клієнтів, іншій - тільки номера їхніх розрахункових рахунків, а третій - тільки грошові суми, що зберігаються на цих рахунках. Змова між цими людьми повинна бути виключена. Для введення даних їм дають різні форми, хоча дані з форм можуть надходити в одну таблицю.

По-третє, введення даних у таблицю - надзвичайно стомливе заняття. Вже після декількох годин роботи люди роблять помилки. Введення даних у форму простіше. Тут багато чого можна автоматизувати, до того ж елементи керування форм настроюють таким чином, щоб при введенні даних виконувалася їхня первинна перевірка.

І нарешті, по-четверте, треба згадати, звідки береться інформація для баз даних. Як правило, її беруть із паперових бланків (анкет заяв, рахунків, відомостей, довідок і т п.). Екранні форми можна зробити точною копією паперових бланків, із яких відбувається введення даних. Завдяки цьому набагато зменшується кількість помилок при введенні і значно знижується стомлюваність персоналу.

Форма — це спеціальний об'єкт бази даних, створений розробником і призначений для зручного введення даних до таблиці чи відображення даних. У формах можна виводити дані з кількох зв'язаних таблиць. Можна також

створювати форми, які забезпечують різне подання одних і тих самих даних. Такі різні види подання даних називають типами форм.

Типи форм:

1. Повноекранна (у стовпчик) — дані полів розташовано у стовпчик, а підписи — збоку. Залежно від розташування підписів (у цьому офісі) автоформа має назву “Стовпчик-підписи ліворуч” або “Стовпчик-підписи праворуч”.

2. Стрічкова — дані полів розташовано в рядок, а блоки-підписи — згори. Автоформа (в даному офісному пакеті) має відповідну назву: “Блоки-підписи згори”. У цій формі кожний запис займає окремий рядок-стрічку. Такі форми дають можливість одночасно і зручно переглядати кілька записів. Для переміщення по списку записів, виведених на екран, використовують вертикальну смугу прокрутки, поле номера запису або кнопки переходу, розташовані в лівому нижньому куті форми.

3. Таблична (як аркуш даних) — поля розташовано у вигляді звичайної таблиці бази даних. Як і попередній тип, дає можливість переглядати одночасно кілька записів. Різниця полягає в тому, що табличні подають лише вміст бази даних без растрових зображень, розмітки тощо

4. Головна (основна) — з якої відкривають інші форми.

5. Підпорядкована — яку відкривають з головної форми.

6. Вільна таблиця — таблиця, не обтяжена формою для заповнення.

7. Діаграма — подає числові дані діаграмою.

8. Монопольна форма — форма, вікно якої є монопольним. Інакше кажучи, поки вікно діалогу цієї форми відкрите, неможливо активувати жодне інше з виведених на екран вікон СКБД.

Використання форм для введення та редагування даних має ряд переваг в порівнянні з використанням таблиць. При перегляді даних в режимі таблиці часто важко побачити всі поля окремого запису. Крім цього, форми дозволяють вибрати зручне розташування даних полів і забезпечити перегляд графічних об'єктів, які зберігаються у полях OLE — об'єктів.

Елементи керування — графічні елементи (поле, кнопка тощо) — розташовано у формі або звіті для керування поданням даних.

Хід роботи

Розглянемо приклад створення простої форми введення даних з урахуванням зв'язку “багато до багатьох”.

Спочатку в розділі “Форми” необхідно використати помічника для створення форми.

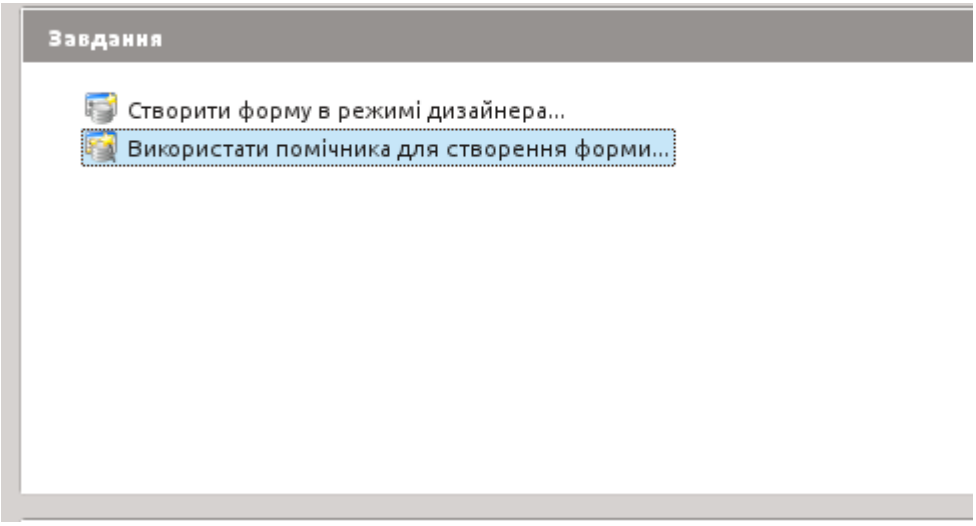


Рис. 15. Використання помічника для створення форм

Далі обираємо таблицю (Doctors) і необхідні атрибути для відображення на формі.

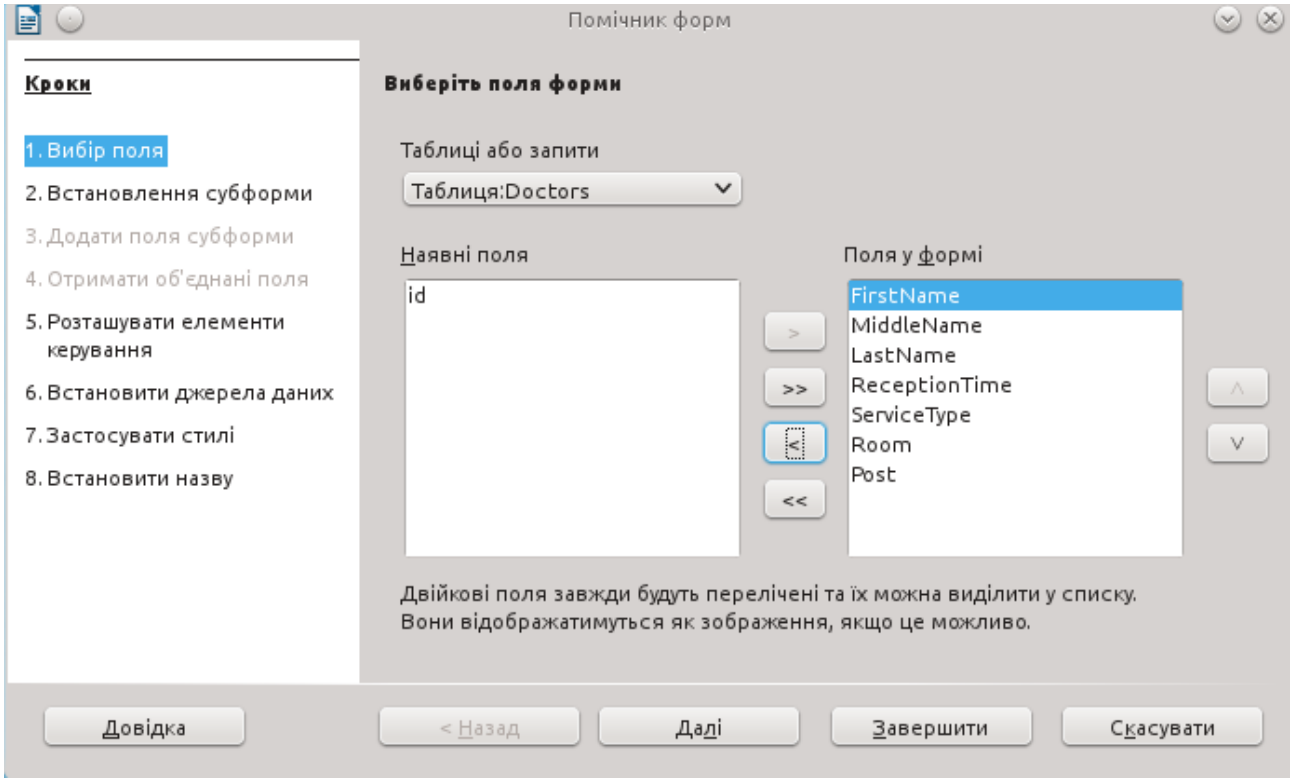


Рис. 16. Вибір поля

Пропускаємо кроки для встановлення субформ. Розташування має бути у вигляді стовпчиків з підписами зверху.

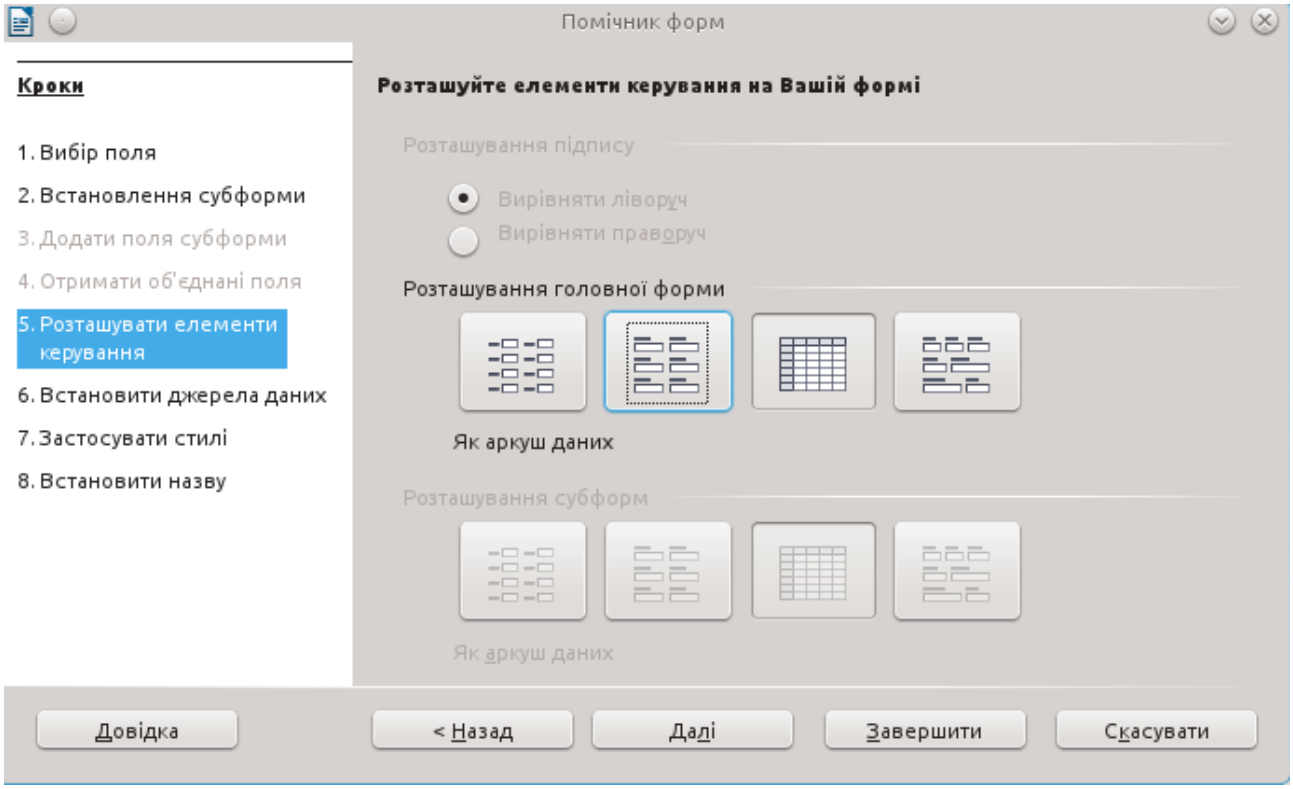


Рис. 17. Розташування елементів керування

Потім для джерела даних необхідно встановити лише значення “Форма для відображення усіх даних”, назначити стилі і назву форми. Після усіх кроків отримуємо найпростішу форму введення/редагування даних таблиці Doctors.

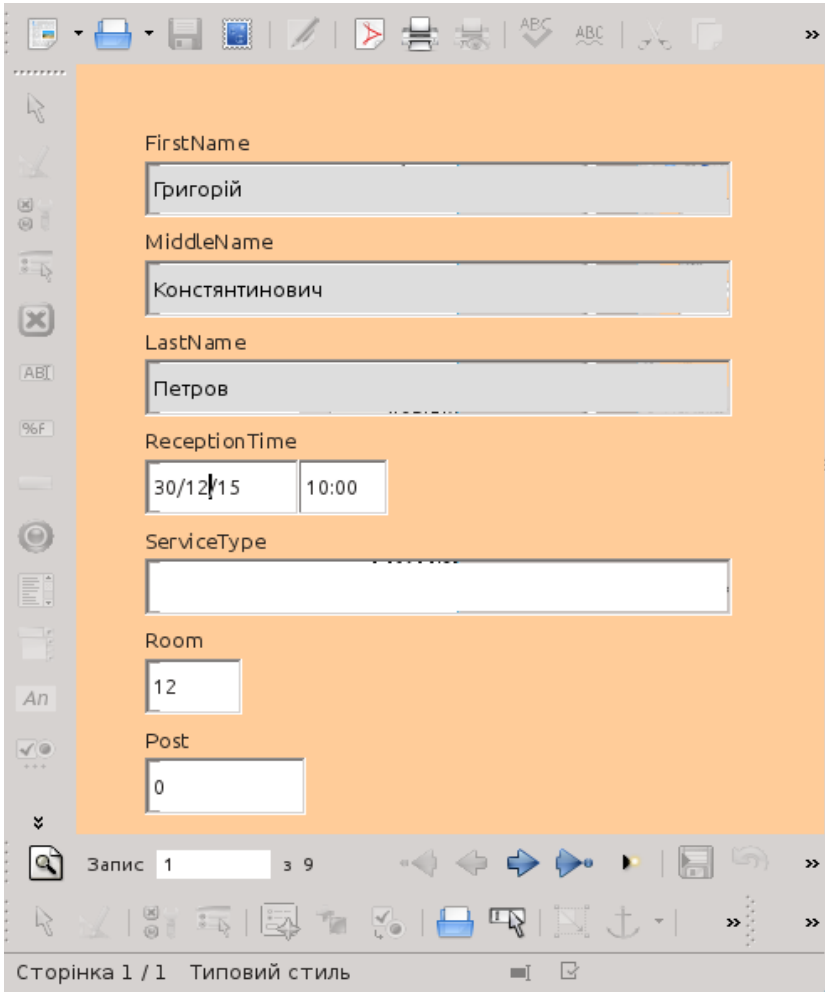


Рис. 18. Вигляд форми

Тепер необхідно змінити форму: в навігаторі форм додаємо субформу.

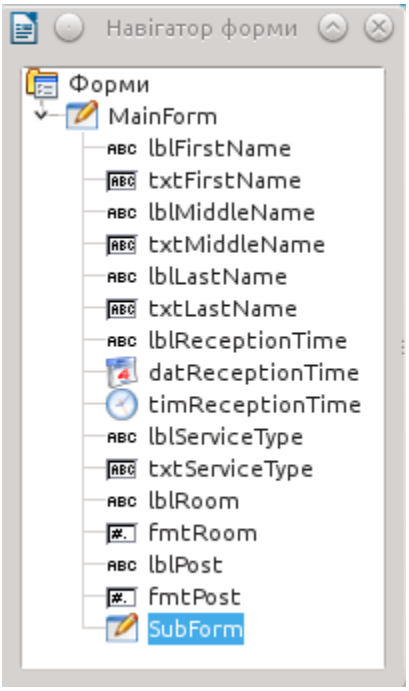


Рис. 19. Навігатор форми: додана субформа.

Змінюємо дані субформи: зміст та зв'язки з головним та підлеглими полями з урахуванням того, що зв'язок буде між таблицями Doctors та Patients через таблицю (слабку сутність) DoctorPatient.

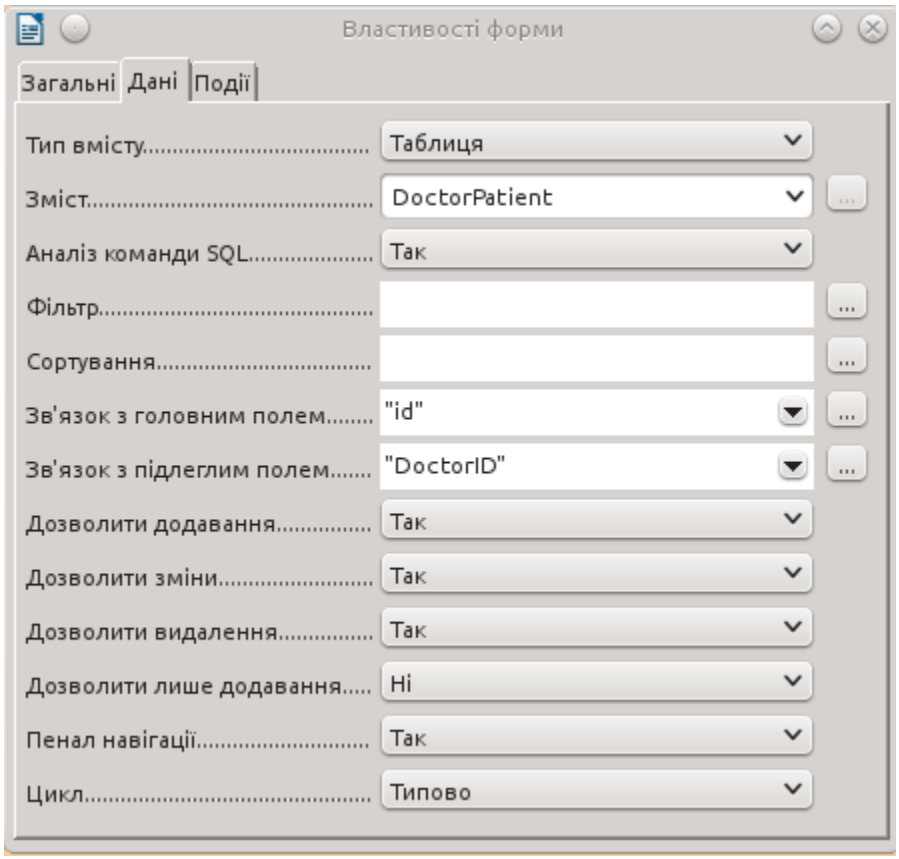


Рис. 20. Дані субформи.

Вмикаємо показ панелі інструментів “Додаткове керування” і обираємо на панелі вставку таблиці . В таблиці необхідно вручну додати стовпчик у цю таблицю у вигляді списку, змінити дані списку: кожний елемент буде являти собою об’єднання ПІБ і дати народження пацієнта..

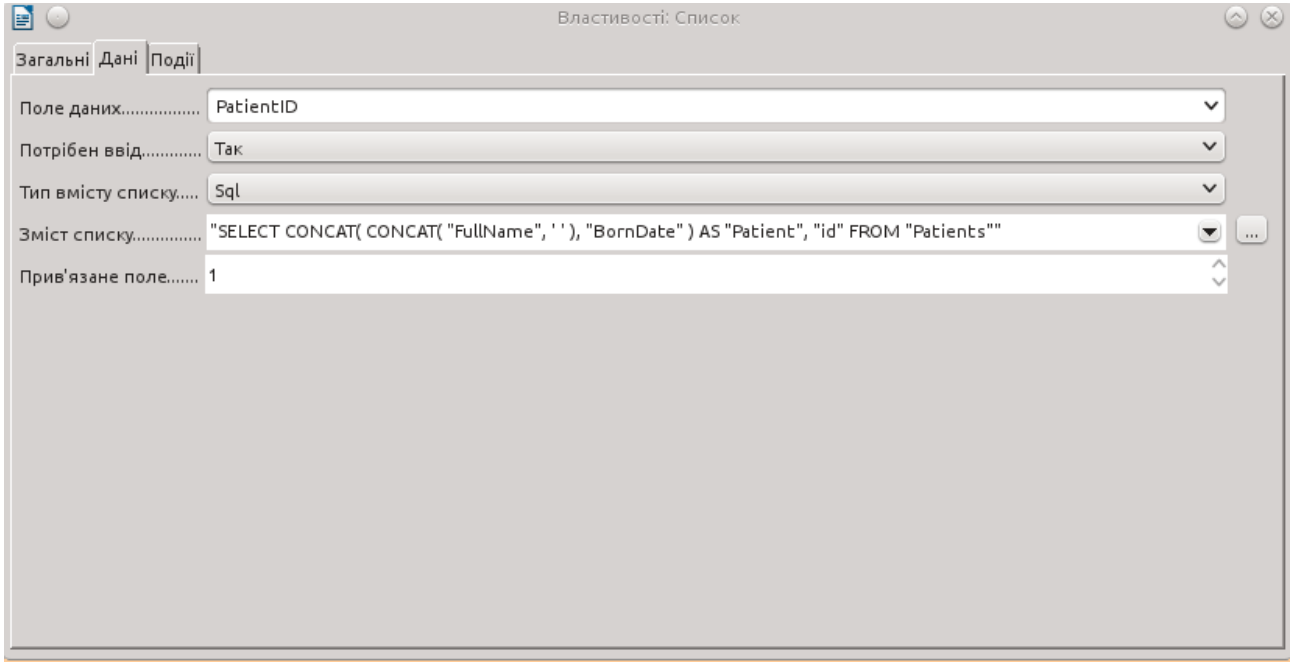


Рис. 21. Властивості даних списку.

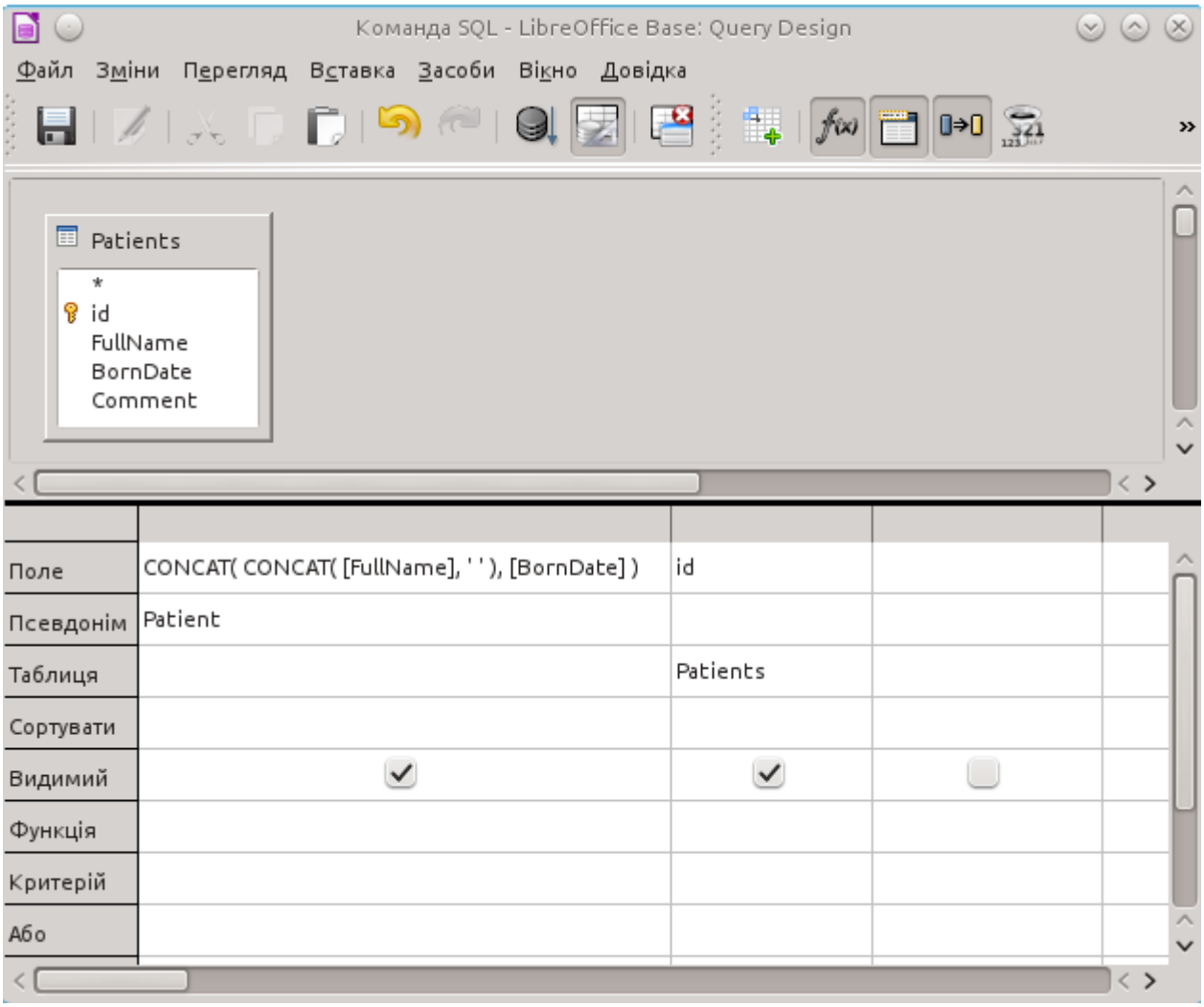


Рис. 22. Зміст списку.

Після маніпуляцій, вказаних вище, отримуємо остаточний макет форми.

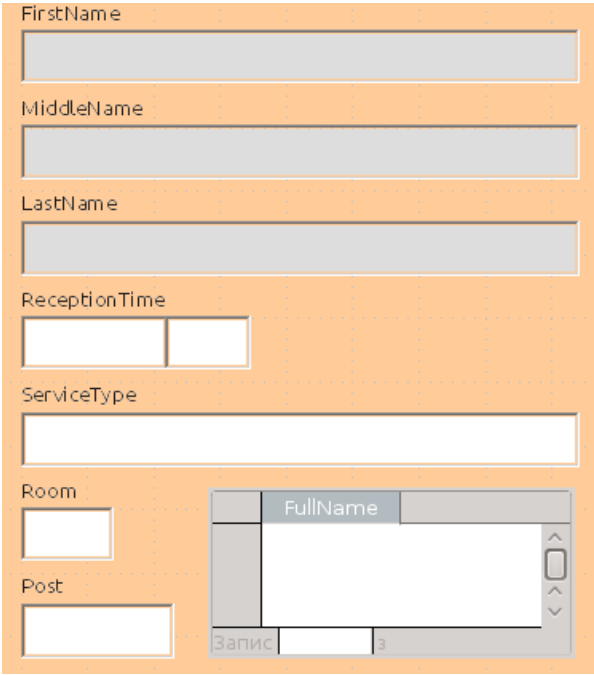


Рис. 23. Остаточний макет форми.

Форма в дії (після відкриття) буде мати наступний вигляд.

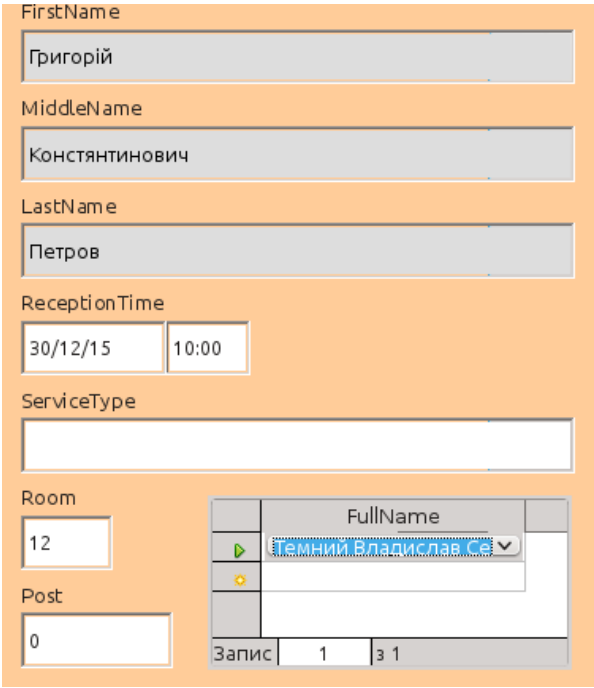


Рис. 24. Відкрита форма.

✍ *Завдання для лабораторної роботи*

Створити щонайменше дві форми: одну для введення / редагування / перегляду даних окремої таблиці без врахування зв'язків, а другу — з такими ж можливостями, але з врахуванням зв'язків "один до багатьох" або "багато до багатьох".

? *Контрольні запитання*

1. Що таке форма в термінології СКБД? Які є види форм?
2. Які особливості має побудова форми з урахування відношення "багато до багатьох"?

Лабораторна робота № 6. Створення звітів

Теоретичні відомості

Звіт призначено для подання інформації баз даних. При використанні звіту можна, крім відображення вмісту полів з таблиць і запитів, групувати дані, вставляти у звіт верхні і нижні колонтитули тощо.

Дуже простий звіт

FirstName	*MiddleName*	*LastName*
Андрій	Григорович	Бойченко
Володимир	Андрійович	Іваненко
Ганна	Іванівна	Кравець
Григорій	Констянтинович	Петров
Іван	Іванович	Чалий
Ірина	Анатоліївна	Бистра
Олег	Петрович	Даненко
Петро	Ігорович	Петренко
Сергій	Іванович	Караваєць

Рис. 25. Приклад простого звіту.

Аналогічно створенню форм, створення звітів можна виконувати як за допомогою помічника для створення звітів, так і у режимі дизайну. Зазвичай, помічник використовують для створення основної частини звіту, тобто структури, а режим дизайну — для удосконалення зовнішнього вигляду звіту.

Завдання для лабораторної роботи

Створити простий звіт на основі однієї таблиці із стандартним оформленням.

Контрольні запитання

1. Дайте визначення звіту в термінах СКБД.
2. Які є види звітів? На основі чого вони формуються?

Лабораторна робота № 7. Знайомство з РСКБД MySQL. Створення бази даних

Теоретичні відомості

Моделі даних

Модель даних – це формальна теорія представлення та обробки даних у СКБД, що включає щонайменше три аспекти:

- ***аспект структури*** – методи опису типів та логічних структур в БД;
- ***аспект маніпуляції*** – методи маніпулювання даними;
- ***аспект цілісності*** – методи опису та підтримки цілісності БД.

Усі СКБД за моделлю даних можна розділити на наступні типи:

- ***мережеві*** (Cerebrum, dbVista, Cache);
- ***ієрархічні*** (реєстр ОС Windows, Information Management System)
- ***реляційні*** (MySQL, MS SQL Server, Oracle Database, Литнер)
- ***об'єктно-реляційні*** (PostgreSQL, Informix, IBM DB2)
- ***об'єкто-орієнтовані*** та інші ***постреляційні*** (Jusmine, ObjectDB).

Найбільшу популярність здобули реляційні та об'єктно-реляційні СКБД. Тому слід відмітити, що усі сучасні СКБД з мережевою або ієрархічною моделлю даних здебільшого підтримують також і реляційні моделі без чіткого віднесення до типу.

Реляційні системи керування базами даних

Реляційна модель орієнтована на організацію у вигляді двовимірних таблиць. При чому

- кожний елемент таблиці є атомарним (неділимим),
- всі елементи в стовпці мають однаковий тип,
- кожний стовпець має унікальне ім'я,
- однакові рядки в таблиці відсутні,
- порядок наступності рядків і стовпців може бути довільним.

Рядок з вище вказаними умовами зазвичай називають кортежем, стовпець – атрибутом, тип даних – доменом.

MySQL

MySQL – вільна (ліцензія GNU General Public License v.2) реляційна система керування базами даних. СКБД має клієнт-серверну модель обміну даними і працює на багатьох різноманітних платформах.

Основні внутрішні характеристики MySQL:

- система написана на мовах програмування C та C++,
- протестовано на широкому спектрі компіляторів,

- багатопоточне виконання з використанням потоків ядра, може працювати у багатопроцесорних системах,
- забезпечує транзакційний та нетранзакційний механізми обробки даних,
- має підтримку багатьох підсистем зберігання даних (рушіїв): ***MyISAM***, ***MRG_MyISAM***, ***InnoDB***, ***CSV***, ***Memory*** та інших,
- швидкі з'єднання, які використовують оптимізовані однопрохідні мультиз'єднання,
- система заснована на паролях та привілеях і є гнучкою та безпечною, дозволяє проводити верифікацію засобами хосту, а весь трафік паролів під час з'єднання шифрується,
- доступні API (Application Programming Interface) для таких мов програмування, як C, C++, Eiffel, Java, Perl, PHP, Python, Ruby і Tcl.

Дистрибутив MySQL складається з багатьох програм, кожну з яких можна віднести до трьох груп:

1. Сервер MySQL і сценарії для його запуску.
2. Клієнтські програми для з'єднання з сервером, здатні працювати як локально, так і віддалено.
3. Програми, які працюють незалежно від сервера.

Клієнтські програми MySQL

До клієнтських програм, які забезпечують зв'язок із сервером, належать:

- ***mysql*** – консольний клієнт (термінальний монітор), дозволяє виконувати SQL-запити і здійснювати адміністрування сервера;
- ***mysqladmin*** – утиліта для виконання адміністративних функцій, таких як створення або видалення баз даних, отримання інформації про стан сервера, процесах та ін.;
- ***mysqldump*** – виводить вміст бази даних MySQL у вигляді файлу з sql-операторами або у вигляді текстового файлу з даними таблиць;
- ***mysqlhotcopy*** – утиліта для створення резервної копії таблиць без зупинки сервера;
- ***mysqlimport*** – виконує перенос інформації із текстового файлу в таблиці бази даних;
- ***mysqlshow*** – відображає інформацію про існуючі бази даних, таблиці, поля та індекси.

Термінальний монітор mysql

Для простого доступу до сервера MySQL через системну консоль (термінал) використовується наступна команда:

 mysql [**-u***username*] [**-p***password*] [**-h***hostname*] [*dbname*]

Для цієї команди усі параметри не є обов'язковими (використовують типові значення). Основні параметри і ключі:

- **-u***username* (**--user=***username*) – задається ім'я користувача *username*;
- **-p***password* (**--password=***password*) – задається парольна фраза *password* (якщо ключ без значення, то виникне запрошення для безпечного надання пароля);
- **-h***hostname* (**--host=***hostname*) – вказується назва мережевого ресурсу *hostname*, де знаходиться сервер або його IP-адреса;
- ***dbname*** (**--database=***dbname*) – вказується назва бази даних *dbname*, яка буде обрана при з'єднанні із сервером;
- **-?** (**--help**) – виведення усіх можливих ключів з інформацією про них та їхні типові значення.

Типовий приклад використання:

```
 m:\>mysql -h10.1.103.26 -ustud -p
```

Очевидно, що краще використовувати ключ пароля без значення, щоб він не зберігався в історії команд. Після введення пароля виникне запрошення для введення SQL-виразів (інструкцій) разом з інформацією про деякі параметри сервера і з'єднання з ним.

```
 Welcome to the MySQL monitor.  Commands end with ; or \g.
 Your MySQL connection id is 138266
 Server version: 5.5.37-0ubuntu0.12.04.1-log (Ubuntu)
 
 Copyright (c) 2000, 2014, 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>
```

Усі команди та SQL-інструкції крім деяких винятків закінчуються символом-розділювачем. Таким типовим символом є крапка з комою (`;`), проте, починаючи з версії 5.0, розділювач можна задати командою `DELIMITER`.

У середовищі MySQL існує два типи коментарів (рядків, які не сприймаються для виконання) :

1. Строковий – починається з двох знаків мінус і пробілу (-- коментар).
2. Блок-коментар – має такий же принцип, як і в мові C (/* блок-коментар */) .

Якщо перейти на наступний рядок без друку символа-розділювача, запрошення терміналу змінюється з `mysql>` на `->`. Щоб відмінити запит, треба ввести послідовність `\c` або команду `clear`. Повний список команд і їх коротких аналогів представлений у таблиці 1.

Таблиця 1 – Команди термінального монітору mysql

Команда	Короткий аналог	Дія команди
`clear`	`\c`	Відміна введеного виразу
`connect`	`\r`	Перепідключитися до сервера
`delimiter`	`\d`	Встановити розділювач між виразами
`ego`	`\G`	Виконати вираз і вивести результат вертикально
`exit`	`\q`	Вихід з термінального монітору
`go`	`\g`	Виконати вираз
`help`	`\h`	Отримати довідку
`notee`	`\t`	Відмінити виведення команд і їх результатів у файл
`print`	`\p`	Виведення поточної команди
`prompt`	`\R`	Змінити текст запрошення (`>mysql`) на власний
`quit`	`\q`	Синонім для exit
`rehash`	`\#`	Перезібрати хеши назв БД, таблиць і полів
`source`	`\.`	Виконати SQL-інструкції з файлу в пакетному режимі (приймає шлях до нього)
`status`	`\s`	Виведення поточної інформації про стан сервера

tee	\T	Підключити файл для запису в нього команд і їх результатів (приймає шлях)
use	\u	Змінити поточну базу даних
charset	\C	Змінити кодування
warnings	\W	Показувати повідомлення після виконання кожного виразу
nowarnings	\w	Не показувати повідомлення після виконання кожного виразу

У термінальному моніторі для коректного відображення символів кирилиці треба задати кодування UTF8 за домогою команди:

 mysql>**SET NAMES** utf8;

Операції над базами даних MySQL

Для створення нової бази даних використовується SQL-вираз CREATE DATABASE або CREATE SCHEMA, які є синонімами:

 mysql>**CREATE DATABASE** [IF NOT EXISTS] ***dbname*** [DEFAULT CHARACTER SET = *charset_name* COLLATE = *collation_name*];

У шаблоні SQL-виразу:

- *IF NOT EXISTS* – перевіряє на існування бази з подібною назвою і створює, якщо така не існує;
- *dbname*– назва бази даних, яку треба створити;
- *charset_name* – назва кодування символів для усіх символьних полів таблиць нової БД;
- *collate_name* – порівняння для сортування символів Юнікоду (для правильного сортування українських слів слід застосовувати порівняння utf8_unicode_ci);

Слід зауважити, що перевірка подібності назв БД виконуються засобами операційної системи, з якою взаємодіє сервер MySQL. Тому чутливість до регістру алфавітних символів при перевірці, набір допустимих символів взагалі та інші правила формування і порівняння назв залежить від ОС і є точно такими, як і правила для назв директорій.

Для перегляду існуючих баз даних використовується SQL-вираз SHOW DATABASES:

 mysql>**SHOW DATABASES;**

При виконанні інструкції у термінал виводиться таблиця усіх доступних баз даних.

Для того, щоб змінити назву БД (*dbname*) на нову (*new_dbname*), починаючи з версії 5.1.7, використовується SQL-вираз `RENAME DATABASE`:

 `mysql>`**`RENAME DATABASE`** ***`dbname`*** **`TO`** ***`new_dbname;`***

В більш ранніх версіях необхідно перейменувати назву каталогу відповідної БД у директорії даних MySQL.

Знищити базу даних з назвою *dbname* можливо за допомогою SQL-виразу `DROP DATABASE`:

 `mysql>`**`DROP DATABASE`** `[IF EXISTS]` ***`dbname;`***

Зміна параметрів БД відбувається виконанням SQL-виразу `ALTER DATABASE`. Наприклад, наступний вираз змінює кодування БД *mydb* на Windows-1251:

 `mysql>`**`ALTER DATABASE`** `mydb CHARACTER SET cp1251;`

Для будь-якої команди (*command*) SQL у термінальному моніторі доступна довідка:

 `mysql>`**`HELP`** ***`command;`***

✍ *Завдання для лабораторної роботи*

Підключитися до навчального серверу через термінальний монітор MySQL, створити базу даних з кодуванням utf8, порівнянням utf8_unicode_ci та назвою, що включає транслітерацію Вашого прізвища та ініціалів. Перевірити наявність БД. Вивести довідку по тим командам, що використовувались, а також список доступних кодувань і інформацію про з'єднання із сервером. Результат оформити у вигляді звіту.

? *Контрольні запитання*

1. Дайте визначення, що таке система керування базами даних, наведіть приклади.
2. Які існують моделі даних?
3. Які основні принципи використовуються у реляційній моделі даних?
4. Дайте основну характеристику СКБД MySQL.
5. Які команди використовуються для створення нових БД і для перегляду?
6. Що станеться, якщо ввести інструкцію в термінальному моніторі MySQL без крапки з комою?

Лабораторна робота № 8. Мова опису даних SQL. Створення структури даних в MySQL

Теоретичні відомості

Мова опису даних

Термін “мова опису даних” (data definition language, DDL) вперше застосовано у моделях баз даних CODASYL наприкінці 60-х рр. XX ст. Тоді схема даних описувалася за допомогою мови із специфічним синтаксисом для створення записів, полів і множин (користувацьких моделей даних). Пізніше, з появою реляційної алгебри, концепція окремої мови для опису і маніпулювання даними набула розвитку в мові структурних запитів.

В загальному сенсі ***мова опису даних*** — це набір синтаксичних одиниць (ключових слів та літералів) для опису структури баз даних. Функції таких мов визначаються дієсловом у команді на звичайній мові, наприклад, “вибрати”, “створити”, “редагувати”, “знищити”.

Мова структурних запитів

Мова структурованих запитів (англ. Structured query language, ***SQL***) – формальна непроцедурна інформаційно-логічна мова програмування, яка застосовується для опису і маніпулювання даними у реляційних системах керування базами даних. Мова заснована на підрозділі реляційної алгебри — численні кортежів.

У цій роботі розглядається застосування SQL як мова опису даних.

Створення, редагування і видалення баз даних і таблиць

Операції з базами даних на мові SQL розглядались у теоретичному матеріалі попередньої роботи, використовуючи інструкції `CREATE DATABASE`, `DROP DATABASE`, `ALTER DATABASE`.

Для створення структури таблиці з заданим ім’ям у поточній базі даних використовується команда `CREATE TABLE`. У загальному випадку ця команда має складний синтаксис, проте у спрощеному варіанті її можливо описати наступним шаблоном:

```
 mysql>CREATE TABLE [IF NOT EXISTS] table_name(
 -> column_name type [NOT NULL | NULL] [DEFAULT
   default_value] [AUTO_INCREMENT] [PRIMARY KEY] [comment]
   [,…]
 ->);
```

У шаблоні SQL-виразу:

- *IF NOT EXISTS* – перевіряє на існування таблиці з подібною назвою і створює, якщо така не існує;

- *table_name* – назва таблиці;
- *column_name* – назва атрибута таблиці (стовпця);
- *type* – тип атрибута – одне з ключових слів у таблиці 2;
- *NOT NULL або NULL* – заборона або дозвіл на можливе порожнє значення (необов'язково);
- *DEFAULT default_value* – встановлення типового значення атрибута у *default_value* (необов'язково);
- *AUTO_INCREMENT* – якщо атрибут має тип цілого числа, то з кожним новим записом значення буде збільшуватись на одиницю (необов'язково);
- *PRIMARY KEY* – надання атрибуту статусу первинного ключа – унікального значення для визначеного кортежу таблиці (рядка);
- *comment* – коментар атрибута: його роль у таблиці (необов'язково).

Таблиця 2 – Типи даних MySQL

Ключове слово	**Опис**
TINYINT(size)	Ціле число від -128 до 127 (від 0 до 255). Максимальне число цифр задається в дужках.
SMALLINT(size)	Ціле число від -32768 до 32767 (від 0 до 65535). Максимальне число цифр задається в дужках.
MEDIUMINT(size)	Ціле число від -8388608 до 8388607 (від 0 до 16777215). Максимальне число цифр задається в дужках.
INT(size)	Ціле число від -2147483648 до 2147483647 (від 0 до 4294967295). Максимальне число цифр задається в дужках.
BIGINT(size)	Ціле число від -9223372036854775808 до 9223372036854775807 (від 0 до 18446744073709551615). Максимальне число цифр задається в дужках.
FLOAT(size,d) DOUBLE (size,d)	Число з плаваючою комою. Максимальне число цифр задається в параметрі size. Максимальне число цифр після десяткової коми задається в параметрі d.
DECIMAL (size,d)	DOUBLE, що зберігається як рядок з фіксованою комою. Максимальне число цифр задається в параметрі size. Максимальне число цифр після десяткової коми задається в параметрі d.
CHAR(size)	Містить рядок фіксованої довжини (може містити букви, цифри, та інші символи). Фіксована довжина задається в дужках. Може зберігати до 255 символів.
VARCHAR(size)	Містить рядок змінної довжини. Найбільша довжина

	задається в дужках. Може зберігати до 255 символів.
TEXT	Містить рядок з найбільшою довжиною до 65535 символів.
LONGTEXT	Містить рядок з найбільшою довжиною до 4294967295 символів.
BLOB	Містить бінарне поле з найбільшою довжиною до 65535 символів.
LONGBLOB	Містить бінарне поле з найбільшою довжиною до 4294967295 символів.
ENUM	Перерахування можливих значень до 65535 елементів.
SET	Подібно до ENUM окрім того, що SET може містити до 64 значень списку, і не може зберігати більше одного вибору.
TIMESTAMP	Кількість секунд з початку епохи Unix ('1970-01-01 00:00:00' UTC). Формат: YYYY-MM-DD HH:MM:SS
DATE	Дата у діапазоні від '1000-01-01' до '9999-12-31'
DATETIME	Дата і час у діапазоні від '1000-01-01 00:00:00' до '9999-12-31 23:59:59'
TIME	Час у діапазоні від '-838:59:59' до '838:59:59'

Для видалення таблиці з назвою *table_name* існує команда DROP TABLE:

 mysql>**DROP TABLE** [IF EXISTS] ***table_name;***

Щоб змінити структуру таблиці, використовується команда **ALTER TABLE**. Ця команда у загальному випадку також має складний синтаксичний шаблон, тому далі будуть розглядатися конкретні приклади використання команди.

І нарешті, команда для перегляду детальної структуру таблиці з назвою *table_name*:

 mysql>**SHOW FULL COLUMNS FROM *table_name;***

Для перегляду списку імен усіх таблиць бази даних є наступна команда:

 mysql>**SHOW TABLES;**

Приклади

В якості прикладу можна взяти структуру даних музичного каталогу.

Таблиця 3 – Структура даних музичного каталогу

Назва атрибуту в таблиці	Тип даних	Коментар
id	Ціле число, автоінкрементне поле	Унікальний ідентифікатор в рамках таблиці
FileLocation	Рядок символів змінної величини	Фізичне розміщення файлу, повний шлях, включно з назвою
MimeType	Рядок символів змінної величини	Медіа тип, наприклад, audio/x-mpeg-3
Title	Рядок символів змінної величини	Назва треку
Artist	Рядок символів змінної величини	Виконавець
Album	Рядок символів змінної величини	Музичний альбом
Year	Ціле число	Рік виходу альбому
Comment	Текст	Коментар
Genre	Рядок символів змінної величини	Жанр
Duration	Час	Довжина треку

У середовищі MySQL запит на створення відповідної таблиці має наступний вигляд:

```
 mysql>CREATE TABLE music_catalog(
 -> id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 -> FileLocation VARCHAR(255) NOT NULL,
 -> MimeType VARCHAR(255),
 -> Title VARCHAR(255) NOT NULL,
 -> Artist VARCHAR(255),
 -> Album VARCHAR(255),
```

```
 -> Year SMALLINT(4) DEFAULT NULL,
 -> Comment TEXT DEFAULT NULL,
 -> Genre VARCHAR(64),
 -> Duration TIME
 ->) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE =
   utf8_unicode_ci;
```

Для перегляду результату запиту, як один із варіантів, можна виконати запит **SHOW FULL COLUMNS FROM music_catalog \G**
Тоді буде виведена повна інформація про структуру таблиці; нижче представлено виведення інформації в термінальному моніторі (перші чотири атрибути):

```
   *************************** 1. row ***************************
       Field: id
        Type: int(11)
   Collation: NULL
        Null: NO
         Key: PRI
     Default: NULL
       Extra: auto_increment
 Privileges: select,insert,update,references
     Comment:
*************************** 2. row ***************************
       Field: FileLocation
        Type: varchar(255)
   Collation: utf8_unicode_ci
        Null: NO
         Key:
     Default: NULL
       Extra:
 Privileges: select,insert,update,references
     Comment:
*************************** 3. row ***************************
       Field: MimeType
        Type: varchar(255)
   Collation: utf8_unicode_ci
        Null: YES
         Key:
     Default: NULL
       Extra:
 Privileges: select,insert,update,references
     Comment:
```

```
 *************************** 4. row ***************************
       Field: Title
        Type: varchar(255)
   Collation: utf8_unicode_ci
        Null: NO
         Key:
     Default: NULL
       Extra:
 Privileges: select,insert,update,references
     Comment:
 ...
```

Наприклад, постала задача додати декілька атрибутів у структурі таблиці: дату і час останньої модифікації рядка таблиці та інформацію про користувача, що вносить або редагує запис. Тоді найбільш правильно буде скористатися запитом ALTER TABLE:

```
 mysql>ALTER TABLE music_catalog
 -> ADD Modified TIMESTAMP NULL DEFAULT
   CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP AFTER
   Duration,
 -> ADD UserInfo VARCHAR(255) NULL DEFAULT "Anonymous"
   AFTER  Modified;
```

Наведений вище запит додає атрибут «Modified» після атрибуту «Duration» і атрибут «UserInfo» після останнього. При чому для атрибуту «Modified» при вставці або оновленні запису таблиці буде призначено поточні дату і час у UNIX-форматі, а для атрибуту «UserInfo» у випадку порожнього значення буде присвоєно строку “Anonymous”.

Зауваження: у таблицях MySQL може існувати лише **один** атрибут типу Timestamp з параметром DEFAULT CURRENT_TIMESTAMP, тобто з типовим значенням, що відповідає поточним даті і часу.

Далі може виникнути задача змінити тип атрибуту або перейменувати атрибут. Наприклад, задача: змінити максимальний розмір типу VARCHAR для атрибуту MimeType до 64 символів і перейменувати атрибут Year на AlbumYear. В обох випадках буде використовуватись команда ALTER TABLE.

```
 mysql>ALTER TABLE music_catalog
 -> -- Для зміни типу
 -> CHANGE MimeType MimeType VARCHAR(64),
 -> /* Для перейменування */
 -> CHANGE Year AlbumYear SMALLINT(4);
```

Щоб знищити непотрібний атрибут таблиці, як приклад, можна використати наступний запит:

```
 mysql>ALTER TABLE music_catalog
 -> -- Для знищення атрибуту Comment
 -> DROP Comment;
```

Зовнішні ключі

Зовнішні ключі в таблицях використовуються для реалізації відношень “один до одного” і “один до багатьох”. Відношення “багато до багатьох” реалізується через слабку сутність — таблицю з первинними ключами двох таблиць, що поєднуються цим відношенням.

Для демонстрації відношення “багато до багатьох” необхідно спочатку створити дві таблиці: таблицю користувачів і слабку сутність — зв’язок між користувачами і музичним каталогом — які записи каталогу сподобалися.

```
 mysql>CREATE TABLE users(
 -> id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 -> UserName VARCHAR(255) NOT NULL,
 -> UserInfo TEXT,
 -> UserCreated TIMESTAMP DEFAULT CURRENT_TIMESTAMP
 ->) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE =
 utf8_unicode_ci;
 mysql>CREATE TABLE user_vote(
 -> UserID INT NOT NULL,
 -> TrackID INT NOT NULL,
 -> VoteMark INT NOT NULL,
 -> VoteCreated TIMESTAMP DEFAULT CURRENT_TIMESTAMP
 ->) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE =
 utf8_unicode_ci;
```

Потім треба видалити атрибут UserInfo.

```
 mysql>ALTER TABLE music_catalog
 -> DROP UserInfo;
```

Далі назначаємо зовнішні ключі:

```
 mysql>ALTER TABLE user_vote
 -> ADD FOREIGN KEY (UserID)
 -> REFERENCES users(id);
 mysql>ALTER TABLE user_vote
```

```
 -> ADD FOREIGN KEY (TrackID)
 -> REFERENCES music_catalog(id);
```

Тепер таблиці пов’зані відношенням “багато до багатьох”.

✍ *Завдання для лабораторної роботи*

1. Підключити файл для виведення тексту із термінального монітора (tee-файл), наприклад:

```
 mysql>\T c:/mysql_out.txt
```

2. За допомогою команди CREATE TABLE створити таблицю (тематика довільна, можна скористатися структурами даних із попередніх робіт) з рушієм даних InnoDB та порівнянням utf8_unicode_ci. В таблиці мають бути сурогатний первинний ключ, строковий атрибут, числовий атрибут, атрибут з типом «дата/час». Показати структуру таблиці.

3. Додати текстові атрибути temp1, temp2. Показати структуру таблиці.

4. Видалити один із них. Показати структуру таблиці.

5. Змінити тип атрибуту, що залишився, на числовий, встановити типове значення 0 і перейменувати його на TempAttr. Показати структуру таблиці.

6. Створити ще одну таблицю і пов’язати з поточною відношенням “багато до багатьох”.

7. Вміст tee-файлу долучити до звіту.

❓ *Контрольні запитання*

1. Дайте визначення терміну «мова структурних запитів».
2. Які типи атрибутів таблиць MySQL вам відомі?
3. Чим відрізняються типи DATETIME і TIMESTAMP.
4. Які особливості має атрибут із статусом первинного ключа?

Лабораторна робота № 9. Маніпулювання даними: вставка, оновлення, видалення.

Теоретичні відомості

Вставка

Формат вставки у таблицю table_name рядка даних value1, value2, …,valueN для відповідних атрибутів attr1, attr2, …, attrN:

```
 mysql>INSERT [INTO] table_name (attr1,attr2,...,attrN)
  VALUES (value1,value2,... ,valueN);
```

Якщо необхідно вставити декілька рядків, то формат прийме наступний вигляд:

```
 mysql>INSERT [INTO] table_name (attr1,attr2,... ,attrN)
  VALUES
 ->(row1_value1,row1_value2,... ,row1_valueN),
 ->(row2_value1,row2_value2,... ,row2_valueN),
 ->...
 ->(rowN_value1,rowN_value2,... ,rowN_valueN);
```

Оновлення

Формат оновлення даних в таблиці table_name атрибутів attr1, attr2, …, attrN на відповідні значення value1, value2, …,valueN у вибраних кортежах, що відповідають умові вибірки Condition :

```
 mysql>UPDATE table_name SET attr1 = value1, attr2 =
  value2,..., attrN = valueN WHERE Condition;
```

Видалення

Формат видалення рядків даних , що відповідають умові вибірки Condition, в таблиці table_name:

```
 mysql>DELETE FROM table_name WHERE Condition;
```

Приклади

Вставка даних у музичний каталог (4 записи):

```
 mysql>INSERT INTO music_catalog
 ->  (FileLocation, MimeType, Title, Artist,
 ->  Album, AlbumYear, Genre, Duration)
 ->VALUES
 ->('/home/user/music/TheWho/TheWhoSellOut/Odorono.mp3',
 ->  'audio/mp3','Odorono','The Who',
```

```
 ->  'The Who Sell Out',1967,'Power Pop','00:02:16'),
 ->('/home/user/music/TheBeatles/Help/Yesterday.mp3',
 ->  'audio/mp3','Yesterday','The Beatles',
 ->  'Help!',1965,'Baroque Pop','00:02:03'),
 ->('/home/user/music/TheBeatles/RubberSoul/Wait.mp3',
 ->  'audio/mp3','Wait','The Beatles',
 ->  'Rubber Soul',1965,'Folk Rock','00:02:16'),
 ->('/home/user/music/MJackson/Thriller/Thriller.mp3',
 ->  'audio/mp3','Thriller','Michael Jackson',
 ->  'Thriller',1982,'Disco','00:05:59');
```

Для перегляду результату вставки можна скористатися запитом вибірки усіх даних :

```
 mysql> SELECT * FROM music_catalog \G
 *************************** 1. row ***************************
          id: 1
 FileLocation: /home/user/music/TheWho/TheWhoSellOut/Odorono.mp3
     MimeType: audio/mp3
        Title: Odorono
       Artist: The Who
        Album: The Who Sell Out
    AlbumYear: 1967
      Comment: NULL
        Genre: Power Pop
     Duration: 00:02:16
 *************************** 2. row ***************************
          id: 2
 FileLocation: /home/user/music/TheBeatles/Help/Yesterday.mp3
     MimeType: audio/mp3
        Title: Yesterday
       Artist: The Beatles
        Album: Help!
    AlbumYear: 1965
      Comment: NULL
        Genre: Baroque Pop
     Duration: 00:02:03
 *************************** 3. row ***************************
          id: 3
 FileLocation: /home/user/music/TheBeatles/RubberSoul/Wait.mp3
     MimeType: audio/mp3
        Title: Wait
```

```
        Artist: The Beatles
         Album: Rubber Soul
     AlbumYear: 1965
       Comment: NULL
         Genre: Folk Rock
      Duration: 00:02:16
*************************** 4. row ***************************
            id: 4
  FileLocation: /home/user/music/MJackson/Thriller/Thriller.mp3
      MimeType: audio/mp3
         Title: Thriller
        Artist: Michael Jackson
         Album: Thriller
     AlbumYear: 1982
       Comment: NULL
         Genre: Disco
      Duration: 00:05:59
```

Щоб змінити атрибут Comment у всіх кортежах, які відповідають певним умовам (рядок виконавця має у кінці рядка символи “Beatles”):

```
mysql>UPDATE music_catalog SET Comment='good music'
 -> WHERE Artist LIKE '%Beatles';
```

Тепер видалимо запис з id=4:

```
mysql>DELETE FROM music_catalog WHERE id = 4;
```

✍ *Завдання для лабораторної роботи*

1. Підключити файл для виведення тексту із термінального монітора (tee-файл), наприклад:

```
mysql>\T c:/mysql_out.txt
```

2. Вставити щонайменше 10 записів у ваші таблиці із попередньої роботи. Переглянути результат, вибравши усі записи.

3. Оновити щонайменше 3 записи, використовуючи різні умови для оновлення. Переглянути результат.

4. Видалити один запис за вказаних значенням первинного ключа і декілька записів за деякою умовою. Переглянути результат.

6. Вміст tee-файлу долучити до звіту.

❓ *Контрольні запитання*

1. Які є основні операції маніпулювання даними? Як вони реалізовані в СКБД MySQL?

2. Чим відрізняється результати запитів DELETE FROM *table_name* і TRUNCATE *table_name*?

3. Що вам відомо про зовнішні ключі і реляційні відношення? Як ці концепції реалізовані в СКБД MySQL?

4. Для чого потрібні індекси для атрибутів таблиць? Як їх створити в MySQL?

Лабораторна робота № 10. Маніпулювання даними: вибірка, групові операції, імпорт/експорт.

Теоретичні відомості

Вибірка

У загальному випадку формат команди вибірки є досить складним.

```
 mysql>SELECT
 ->     [ALL | DISTINCT | DISTINCTROW ]
 ->       [HIGH_PRIORITY]
 ->       [STRAIGHT_JOIN]
 ->       [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
 ->       [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
 ->     select_expr [, select_expr ...]
 ->     [FROM table_references
 ->     [WHERE where_condition]
 ->     [GROUP BY {col_name | expr | position}
 ->       [ASC | DESC], ... [WITH ROLLUP]]
 ->     [HAVING where_condition]
 ->     [ORDER BY {col_name | expr | position}
 ->       [ASC | DESC], ...]
 ->     [LIMIT {[offset,] row_count | row_count OFFSET offset}]
 ->     [INTO OUTFILE 'file_name' export_options
 ->       | INTO DUMPFILE 'file_name'
 ->       | INTO var_name [, var_name]]
 ->     [FOR UPDATE | LOCK IN SHARE MODE]]
```

Пояснення і приклади для основних ключових слів наведені в таблиці 4. Для більш детального ознайомлення можна скористатися офіційною документацією https://dev.mysql.com/doc/refman/5.0/en/select.html .

Таблиця 4 – Розбір ключових слів у шаблоні запитів на вибірку

Ключове слово	Пояснення і приклади

SELECT	Головне ключове слово для вибірки. Використовується для отримання рядків, вибраних з однієї або декількох таблиць, і може включати в себе вирази об’єднання (UNION), підзапити, операції та функції.
	`mysql> select 2+2;`
	`-> 4`

ALL | DISTINCT	Врахування повторень: ALL — не враховувати і показати усі результати (типове значення), DISTINCT — лише унікальні

	рядки.
HIGH_PRIORITY	Надає запитам високий пріоритет. Зазвичай запити на оновлення мають більший пріоритет, ніж вибірка.
STRAIGHT_JOIN	Директива оптимізатору здійснювати поєднання JOIN у тому порядку, в якому вони розміщені у виразі FROM.
SQL_SMALL_RESULT | SQL_BIG_RESULT	Директива оптимізатору, що очікується мала кількість рядків (SQL_SMALL_RESULT) або велика (SQL_BIG_RESULT).
SQL_BUFFER_RESULT	Директива оптимізатору про необхідність збереження результату в тимчасову таблицю.
SQL_CACHE | SQL_NO_CACHE	Необхідність кешувати результат запиту у загальному кеші.
SQL_CALC_FOUND_ROWS	Підрахунок кількості знайдених записів, ігноруючи обмеження в кількості (LIMIT).
select_expr	Вираз вибірки — те, що вибирається: назва атрибуту або композиція із атрибутів з використанням функцій, або підзапит. Є можливість іменувати кожен вираз за допомогою псевдонімів (ключове слово AS). `mysql> select CONCAT(Artist,' - ',Title) AS FullName FROM music_catalog;`
FROM ***table_references***	Вираз для визначення джерела даних: однієї таблиці або сукупності таблиць, пов’язаних операторами поєднання (LEFT JOIN, RIGHT JOIN, INNER JOIN) чи загальним декартовим добутком усіх значень атрибутів. Також в якості таблиці може бути запит з обов’язковим псевдонімом.
WHERE ***where_condition***	Ключове слово для визначення специфічних умов вибірки — критеріїв запиту, що являють собою сукупність логічних виразів. Псевдоніми, задані в розділі SELECT не використовуються, тому що при

	вибірці першими опрацьовуються критерії вибірки, а потім атрибути.
GROUP BY {***col_name*** | ***expr*** | ***position***} [ASC | DESC], … [WITH ROLLUP]	Групування отриманих даних запиту за вказаними атрибутами або порядковими номерами атрибутів у виразі SELECT, починаючи з 1. Також можна використовувати агрегатні функції з атрибутами в якості параметрів для групування. Доступне сортування за зростанням (ASC) і за спаданням (DESC). За допомогою ключового виразу WITH ROLL UP обчислюються загальні значення для агрегатних функцій. Детальніше https://dev.mysql.com/doc/refman/5.0/en/group-by-modifiers.html
LIMIT {[*offset*,] *row_count* | *row_count* OFFSET *offset*}	Обмеження кількості результатів вибірки: буде повернено щонайбільше row_count записів зі зміщенням відносно початку offset.
HAVING *where_condition*	Критерії запиту для отриманих результатів з умовами WHERE, GROUP. Можуть використовуватись псевдоніми атрибутів із розділу SELECT.
ORDER BY {*col_name* | *expr* | *position*} [ASC | DESC], ...	Умови сортування за зростанням (ASC) чи спаданням (DESC) атрибутів, виразів чи позицій у розділі SELECT.

Імпорт

Для імпорту даних існує клієнтська утиліта *mysqlimport* (https://dev.mysql.com/doc/refman/5.0/en/mysqlimport.html). Але зазвичай будь-які sql-вирази, записані у файл, можливо подати на вхід термінального монітору mysql.

```
 >mysql -h127.0.0.1 -ustud -p some_database < f.sql
```

Експорт

Для експорту структури таблиць і даних вказаної БД використовується аналогічна утиліта mysqldump. Формуються sql-інструкції для створення таблиць і заповнення їх даними.

```
 >mysqldump -h127.0.0.1 -ustud -p some_database > f.sql
```

✍ *Завдання для лабораторної роботи*

1. Підключити файл для виведення тексту із термінального монітора (tee-файл), наприклад:

```
 mysql>\T c:/mysql_out.txt
```

2. Вибрати 3 записи вашої таблиці з попередніх робіт із зміщенням у 6 записів.

3. Вибрати перші 10 записів двох поєднаних таблиць зовнішніми ключами з використанням *LEFT JOIN, RIGHT JOIN, INNER JOIN*.

4. Вибрати суми усіх числових атрибутів, згруповані по кожній даті атрибуту з типом DATE (якщо немає таких, то створити і заповнити даними) і загальні суми незалежно від групування, використовуючи інструкцію WITH ROLLUP.

5. Вибрати усі неунікальні значення тих атрибутів, що можуть їх містити.

6. Вміст tee-файлу долучити до звіту.

7. Експортувати вашу базу у sql-файл. Команду експорту долучити до звіту.

❓ *Контрольні запитання*

1. Як вони реалізовані команди вибірки в СКБД MySQL?

2. Чим відрізняється дія інструкцій *WHERE* і *HAVING*?

3. Які основні поєднання таблиць використовуються у виразі *FROM*? Які вони мають особливості?

4. Як зробити експорт лише структури таблиць без даних?

Індивідуальне завдання

Розробити конструктор запитів, використовуючи HTML / CSS, HTTP-сервер і сервер MySQL наступним чином: обираються таблиці; обираються атрибути таблиць або вписуються вирази; обираються типи і порядок з'єднань (LEFT JOIN, RIGHT JOIN, INNER JOIN), вписуються умови з'єднання (ON-вирази); вписуються умови вибірки (WHERE-вирази), групування і сортування. Результати відображаються у вигляді таблиці.

Для нотаток

Глосарій

Бази даних – визначення терміну 1.
Термін 2 – визнаяення терміну 2.

Рекомендована література

Основна:

1. Кузнецов М. MySQL 5 / Максим Кузнецов. – СПб. : БХВ-Петербург, 2010. – 1024 с.
2. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. / К. Дж. Дейт. – М. : Издательский дом «Вильяме», 2005. – 1328 с.
3.

Додаткова:

1.

Інформаційні ресурси:

1. http://www.mysql.ru/docs/man/
2. http://www.weblibrary.biz/mysql
3.

Додаток А. Назва додатку

Текст.

```
Код 1 ...
Код 2 ...
…
```

Для нотаток

Методичне видання
(українською мовою)

Чопоров Сергій Вікторович, Синєпольський Ігор Володимирович

БАЗИ ДАНИХ

Навчально-методичний посібник до лабораторних занять
для студентів освітньо-кваліфікаційного рівня «бакалавр»
напряму підготовки «Програмна інженерія»

Рецензент *С.Ю. Борю*
Відповідальний за випуск *С.І. Гоменюк*
Коректор *Н.В. Плюта*