МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

С. В. Чопоров, О. В. Чопорова, О. М. Мильцев, А. В. Столярова

БАЗИ ДАНИХ

Навчальний посібник
для здобувачів ступеня вищої освіти бакалавра
спеціальності «Інженерія програмного забезпечення»
освітньо-професійної програми «Програмна інженерія»

Затверджено
Вченою радою ЗНУ
Протокол №9 від 22.02.2022р.

Запоріжжя 2022

УДК: 004.65(075.8)
Б179

Бази даних : навчальний посібник для здобувачів ступеня вищої освіти бакалавра спеціальності «Інженерія програмного забезпечення» освітньо-професійної програми «Програмна інженерія» / С. В. Чопоров, О. В. Чопорова, О. М. Мильцев, А. В. Столярова. Запоріжжя: ЗНУ, 2022. – 60 с.

Навчальний посібник пропонується здобувачам ступеня вищої освіти бакалавра спеціальності «Інженерія програмного забезпечення» освітньо-професійної програми «Програмна інженерія». Посібник включає теоретичні відомості з кожної теми лабораторної роботи, практичні приклади та контрольні запитання. Увага студентів акцентується на формуванні практичних навичок з проектування і створення баз даних, з обробки даних при створенні інформаційних систем.

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

Рецензент *С. М. Гребенюк*

Відповідальний за випуск *С. В. Чопоров*

Зміст

Вступ

Комп’ютери зберігають дані у файлах, що являють собою набір записів, присвячених спільній тематиці. Записи часто складаються з полів. У традиційних системах окремі програми створюють і обробляють власні файли. У системах з базами даних інакше. Файли та записи у них організовані таким чином, щоб їх могли обробляти різні прикладні програми. При традиційній файловій обробці велика ймовірність неузгодженості, неоднозначності та неефективної праці розробників програмного забезпечення. Бази даних допомагають запобігти цих проблем.

До базових концепцій баз даних відносяться такі категорії «дані» і «модель даних». Дані – це набір конкретних значень параметрів, що характеризують певний об’єкт або процес. Дані самі собою не мають структури. Вони перетворюються на інформацію коли їх фіксують на носії інформації надав їм певну структуру. Відповідно щоб дані могли бути інтерпретовані іншими їх структура повинна бути зафіксована у доступному для інших вигляді – моделі даних. Модель даних – результат систематизації інформації, відображення її властивостей, структури, зв’язків між її елементами.

Сьогодні в результаті багатьох теоретичних і практичних досліджень розроблено досить ефективні моделі даних, що знайшли свою реалізацію у базах даних. Це класичні моделі даних: ієрархічна, мережева, реляційна і об’єктно-орієнтована. А також новітні такі як хеш-таблиці, документо-орієнтовані тощо.

Метою посібника є формування практичних навичок з проектування та розробки баз даних різних типів. Посібник складається з лабораторних робіт, що містять теоретичні відомості, Практичне завдання і контрольні запитання. Також у кінці посібника сформульовані індивідуальне завдання та індивідуальні теми для створення баз даних.

Практичним результатом кожної лабораторної роботи повинні бути робоча програма (скрипти) для створення або обробки бази даних, звіт, що висвітлює хід виконання роботи.

Тема 1. Моделювання сутностей та відношень предметної області

Теоретичні відомості

Основним призначенням баз даних та інформаційних систем є забезпечення користувачів інформацією. Частина реального або уявного світу утворює *предметну область*. Інформація подається у формі даних щодо предметної області. Підмножина даних, що виокремлено для створення бази даних або інформаційної системи, утворює *інформаційну* та *функціональну моделі* предметної області, які дозволяють моделювати предметну область з певною точністю.

Інформаційна та функціональна моделі предметної області створюють за результатами за результатами дослідження вимог щодо бази даних і зазвичай не залежать від обраної технології реалізації баз даних. Ці моделі є основою для проєктування бази даних. Такі моделі зазвичай не залежать від СУБД[1], програмної й апаратної платформ.

Сукупність об'єктів або процесів, що входять до моделей предметної області, утворює *ядро предметної області* та мають онтологічний статус. *Сутності предметної області* є результатом абстрагування шляхом і фіксації важливих властивостей об'єктів або процесів предметної.

Сутності предметної області та відношення між ними часто моделюють за допомогою ER-діаграм. Кожну сутність позначають прямокутником, який може містити перелік важливих властивостей. Наприклад, при проєктуванні багатьох інформаційних систем виникає сутність Персона (person, рис. 1), яка може включати такі властивості: ім'я (first_name), по батькові (second_name), прізвище (last_name), дата народження (birthday), адреса (address), контакти (contacts).

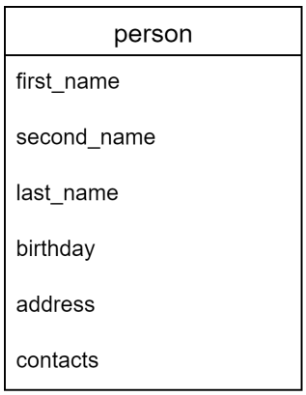

Рисунок 1. Сутність person

Для сутностей обирають *ключову властивість* (*атрибут*). Це властивість або група властивостей, що дозволяють відрізняти екземпляри сутності один від

[1] СУБД – система управління базами даних, англ. Database Management System, DBMS

одного. На діаграмах такі властивості позначають підкресленням. Якщо ключові властивості не визначено, то такими вважають усі властивості сутності.

Сутності предметної області можуть мати певні відношення, що характеризують зв'язки та/або процеси предметної області. Відношення між сутностями позначають можливі асоціації між екземплярами сутностей.

Одним з найбільш простих типів асоціації між екземплярами сутностей є зв'язок «суперклас-підклас». Таке відношення застосовують коли існують декілька самостійних сутностей, що мають велику кількість спільних атрибутів (інакше кажучи, відрізняються невеликою кількістю властивостей). Наприклад, при моделюванні відношень у типовому навчальному закладі студентів (student) і викладачі (professor) можна вважати підкласами класу "персона" (person). Водночас для студентів специфічною буде інформація щодо студентського квитка, а для викладачів – інформація про науковий ступінь і вчене звання (рис. 2).

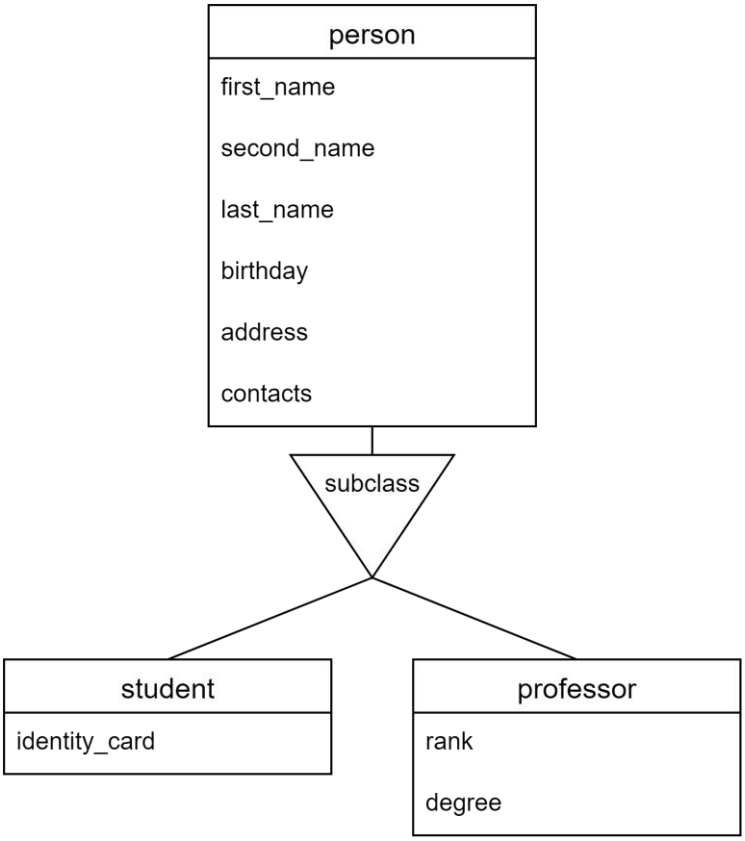

Рисунок 2. Зв'язок «суперклас-підклас» між сутностями person-student і person-professor

Більш складні асоціації відображають як сутності відносяться одна до одної. Найбільш поширеними є три типи відношень:

- *один до багатьох*;
- *багато до багатьох;*
- *один до одного*.

Нехай прикладом предметної області, на якій далі буде продемонстровано застосування відношень між сутностями, буде заклад освіти, де кожен студент може навчатися тільки в одній групі. Група готується за освітньою програмою певної спеціальності та галузі знань, водночас може бути декілька груп на одній освітній програмі. Один зі студентів групи є старостою. Кожен студент може обрати собі довільну кількість дисциплін. Одну дисципліну забезпечує один викладач, але викладач може вести декілька дисциплін. Викладач може працювати тільки на одному департаменті (кафедрі), один з викладачів департаменту є головою департаменту. Викладач може бути куратором декількох груп.

Відношення “один до багатьох” виникає коли до одного екземпляра сутності бази можуть відповідати декілька екземплярів іншої сутності. Приклади таких відношень зображено на рис. 3).

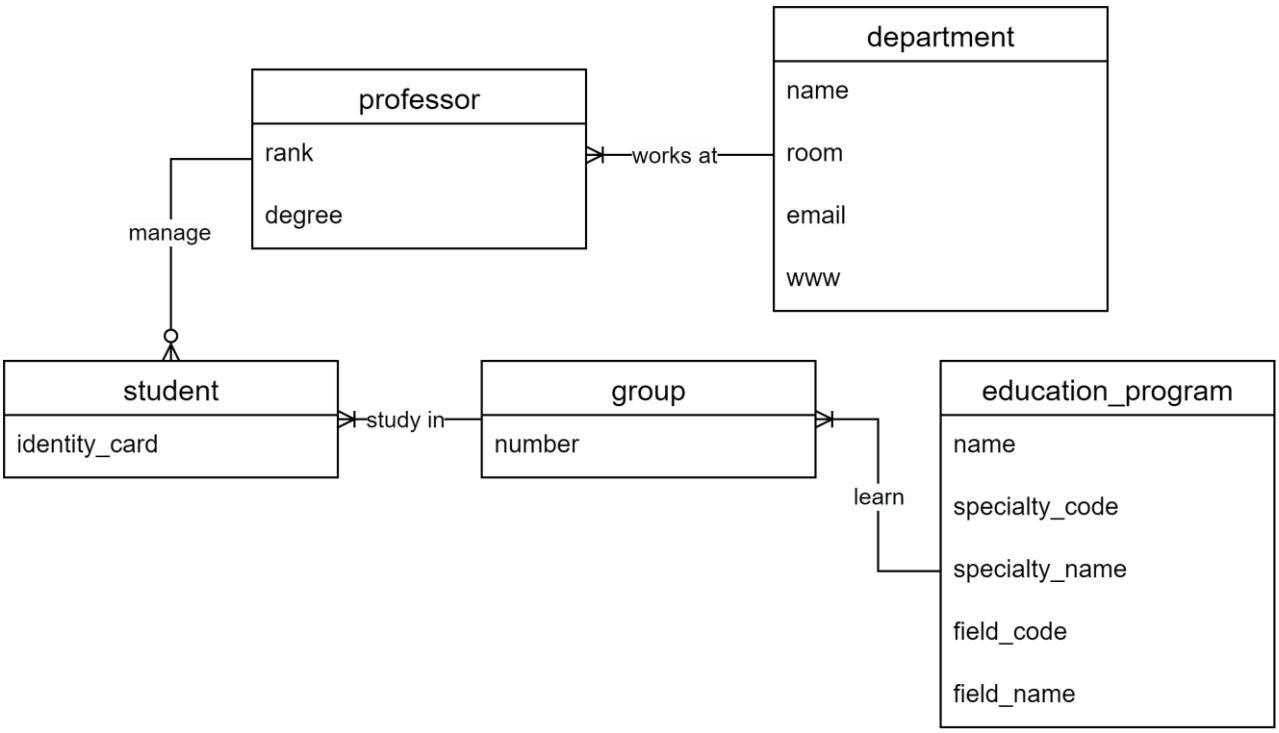

Рисунок 3 – Приклади відношень “один до багатьох”

Прикладом відношення “багато до багатьох” у нашому прикладі є відношення між студентами та обраними курсами: кожен курс може вибрати довільна кількість студентів, водночас студент може вибрати певну кількість курсів (рис. 4).

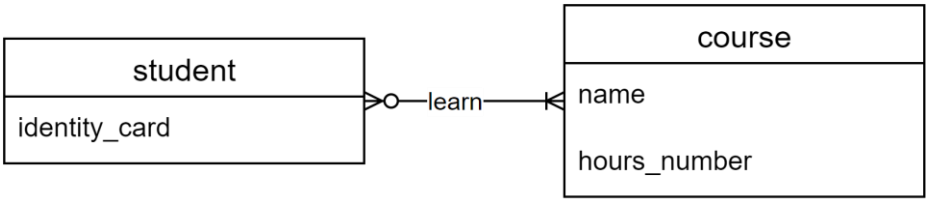

Рисунок 4 – Приклад відношення “багато до багатьох”

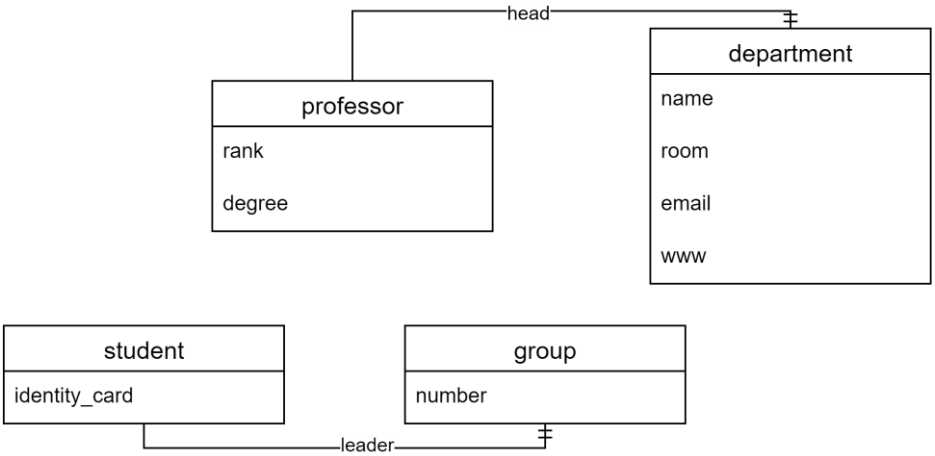

Рисунок 5 – Приклади відношень “один до одного”

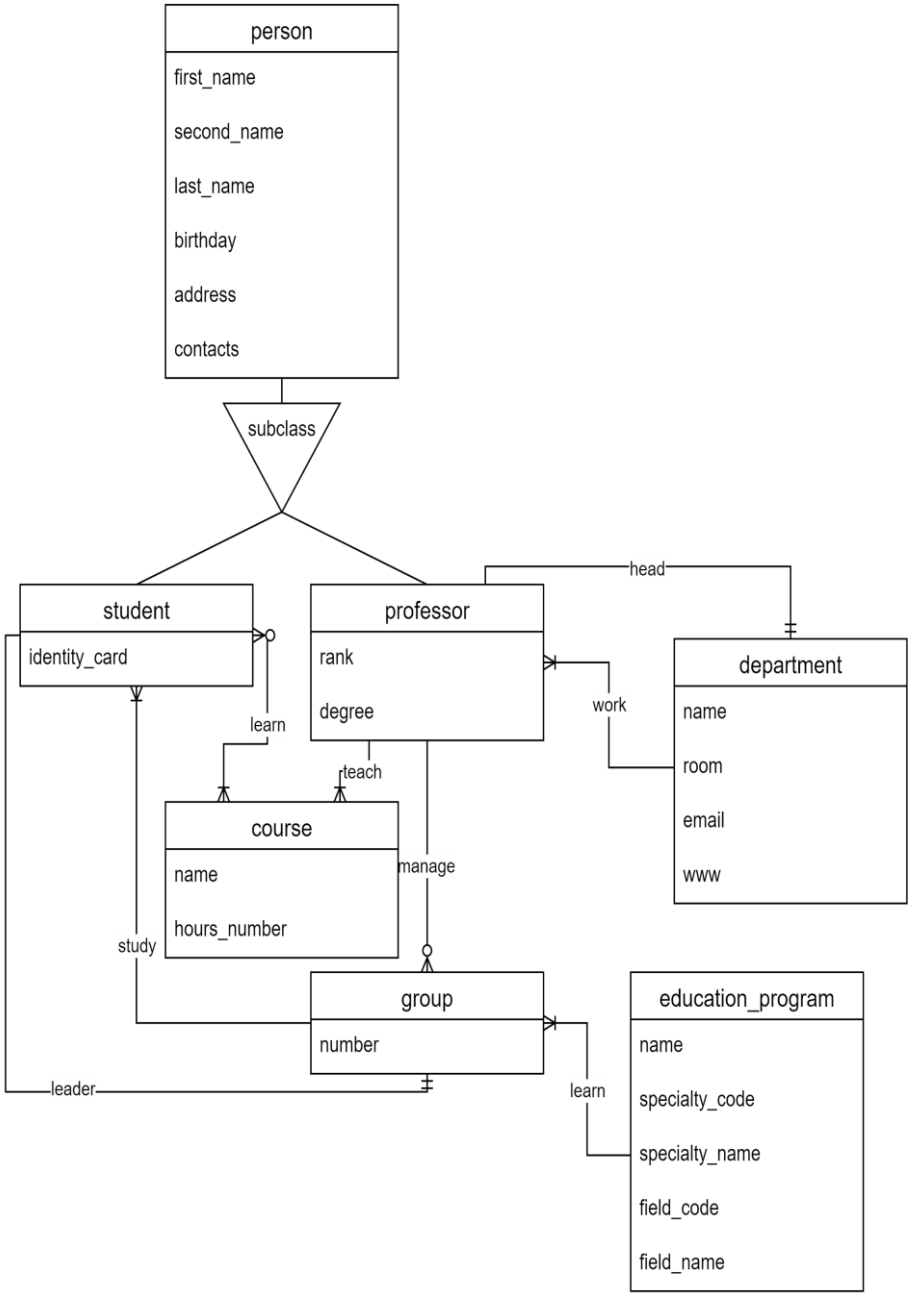

Рисунок 6 – Діаграма предметної області

Відношення “один до одного” можна показати за допомогою сутностей “студент” і “група” (один студент староста однієї групи, а у групи тільки доин староста) та “викладач” з “департамент” (один з викладачів – голова департаменту), рис. 5.

Підсумкова діаграма предметної області матиме вигляд рис. 6.

💻 Практичне завдання

Згідно з описом предметної області (додаток А) розробити модель сутностей та відношень предметної області. Модель необхідно подати у вигляді діаграми предметної області з описом сутностей та обґрунтування зв’язків між ними.

? Контрольні запитання

1. Що таке дані?
2. Дайте визначення моделі даних.
3. Назвіть блоки моделі сутностей та відношень.
4. Що таке атрибут даних?
5. Назвіть типів відношень, наведіть приклади.

Тема 2. Розробка реляційної моделі

Теоретичні відомості

Реляційна модель даних – логічна модель даних, що ґрунтується на використанні математичного поняття відношення[2]. Реляційна модель зазвичай включає такі складові частини:

- *структурна частина* – n-арне відношення, яке часто представляють у формі таблиць, де кожен рядок є *кортеж*, а кожен стовпець – *атрибут*[3], визначений на деякому *домені*;
- *маніпуляційна частина*, що включає реляційну алгебру і реляційне числення;
- *цілісна частина*, що забезпечує *цілісність сутностей*[4] та *цілісність посилань*[5].

Алгоритм перетворення ER-моделі в реляційну модель такий:

1. Для кожної сильної сутності ER-моделі створити базове відношення. Для кожної простої властивості сутності створити атрибут відношення. Ключову властивість сутності перетворити у ключовий атрибут відношення.
2. Для кожної слабкої сутності ER-моделі створити відношення, що включає усі прості властивості цієї сутності як атрибути та додаткові атрибути – зовнішні ключі, що посилаються на первинні ключі первинних сутностей, від яких залежить існування слабкої сутності.
3. Якщо дві сутності беруть участь у відношенні “один до одного”, то відношення, що представляє будь яку з цих двох сутностей, повинне включати атрибут зовнішнього ключа, який реалізує зв’язок з іншим відношенням.
4. Якщо дві сутності беруть участь у зв’язку “один до багатьох”, то відношення, що представляє сутність з боку кардинальності “багато” повинне включати атрибут зовнішнього ключа, який реалізує зв’язок з іншим відношенням.
5. Якщо дві сутності беруть участь у зв’язку “багато до багатьох”, то необхідно створити додаткове відношення, атрибути якого є зовнішніми ключами, що посилаються на первинні ключі відношень, що подають сутності зі зв’язку.

---

[2] Відношення – математична структура, що формально визначає властивості різних об'єктів і їхні взаємозв’язки. Термін відношення часто замінюють синонімічним терміном таблиця.

[3] Термін атрибут часто замінюють синонімічним терміном стовпець.

[4] Цілісність сутностей – вимога щодо первинного ключа, який дозволяє відрізнити кортежи відношення один від одного.

[5] Цілісність посилань – вимога щодо зовнішніх ключів (посилань): кожне значення зовнішнього ключа повинне бути або невизначеним (нульовим), або дорівнювати значенню первинного ключа певного відношення.

6. Якщо сутність має багатозначний атрибут, то для його представлення необхідно створити окреме відношення. Один атрибут цього відношення буде зовнішнім ключем вихідного відношення, а інший подавати багатозначний атрибут. Первинним ключем у такому випадку часто є пара цих атрибутів.
7. Якщо у зв'язку бере участь більше двох сутностей, то необхідно створити додаткове відношення, атрибути якого є зовнішніми ключами, що посилаються на первинні ключі відношень, що подають сутності зі зв'язку.
8. Якщо сутності беруть участь у зв'язку «суперклас-підклас», то для сутності суперкласу необхідно створити окреме відношення, що включає спільні властивості всіх відношень, а для підкласів створити відношення, що включають окремі властивості як атрибути та атрибут – зовнішній ключ, що посилається на первинний ключ відношення суперкласу.

Відношення знаходиться у *першій нормальній формі* (*1НФ*), якщо всі його атрибути є простими, всі домени мають тільки скалярні значення (рис. 7).

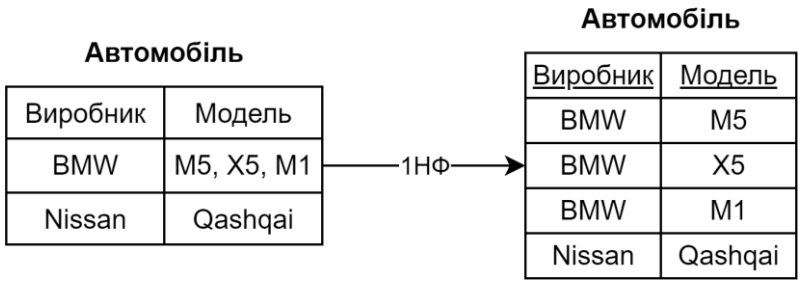

Рисунок 7 – Приклад 1НФ

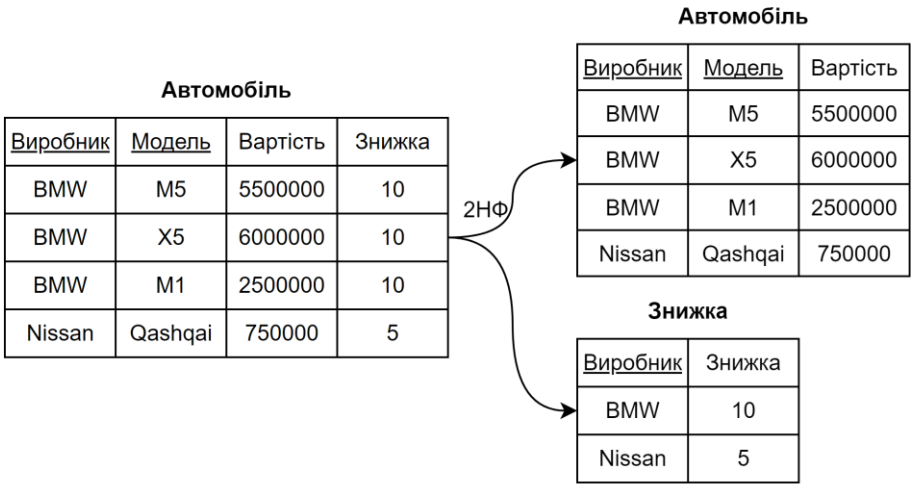

Рисунок 8 – Приклад 2НФ

Відношення знаходиться у *другій нормальній формі* (*2НФ*), якщо воно знаходиться у 1НФ, а кожен не ключовий атрибут незмінно залежить від первинного ключа (рис. 8). 2НФ вимагає, аби дані, що зберігаються у таблицях з композитним ключем, не мали залежності лише від певної частини ключа.

Функціональна залежність між атрибутами *X* і *Y* визначає те, що для будь-якого можливого набору кортежів у цьому відношенні виконується така умова: якщо два кортежи збігаються за значенням по *X*, то вони збігаються і за значенням по *Y*.

Відношення знаходиться у *третій нормальній формі* (*3НФ*), якщо воно знаходиться у 2НФ, а кожен неключовий атрибут нетранзитивно залежить від первинного ключа (рис. 9).

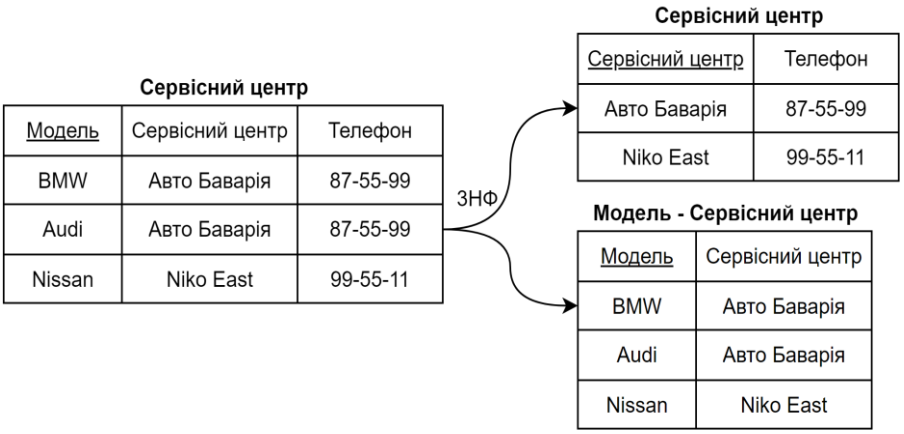

Рисунок 9 – Приклад 3НФ

Відношення знаходиться у *нормальній формі Бойса-Кодда* (*НФБК*), якщо воно знаходиться у 3НФ, а також коли кожна нетривіальна та неприводима зліва функціональна залежність має потенційний ключ як детермінант (рис. 10).

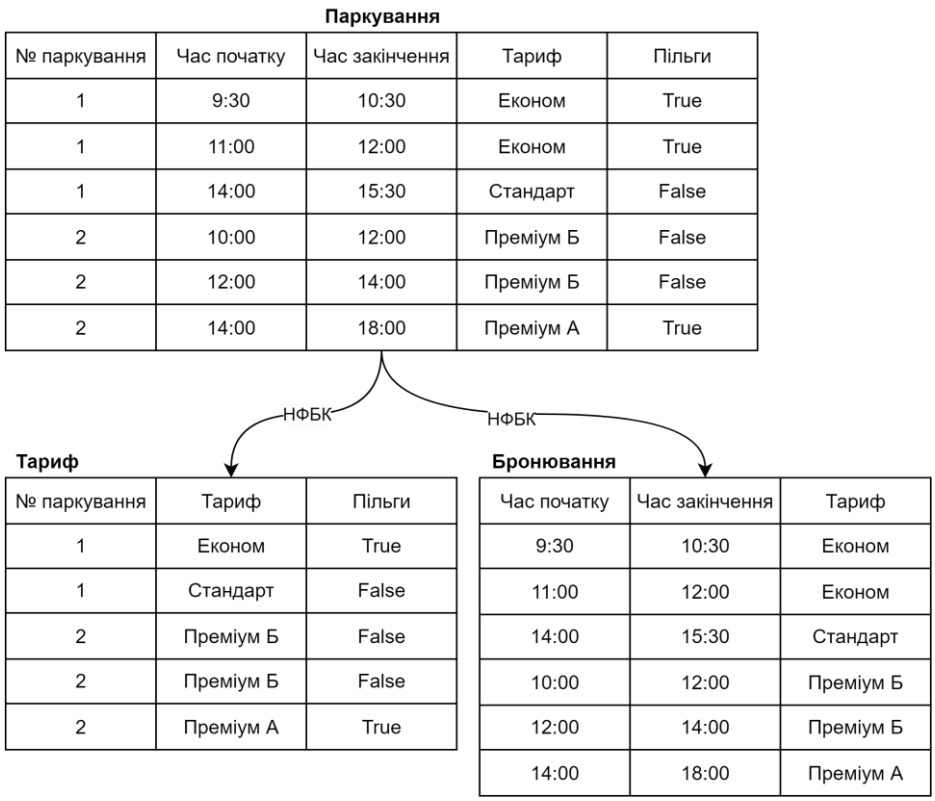

Рисунок 10 – Приклад НФБК

Відношення знаходиться у *четвертій нормальній формі* (*4НФ*), якщо воно знаходиться у НФБК, а всі нетривіальні багатозначні залежності є функціональними залежностями від потенційних ключів відношення.

У відношенні $R(A, B, C)$ існує багатозначна залежність $R.B \rightarrow\rightarrow R.A$, якщо існують значення B, що відповідають (A, B) і водночас залежать лише від A.

У практиці проєктування баз даних поширеним є прийом введення штучних первинних ключів. Такі штучні первинні ключі дозволяють спростити створення та підтримку баз даних, у певних випадках заощадити пам’ять, що використовується для збереження таблиць. Наприклад діаграма предметної області з рис. 5 після введення штучних ідентифікаторів як первинних ключів, а також нормалізації може бути перетворена до діаграми з рис. 11.

💻 Практичне завдання

Використовуючи розроблену в попередній лабораторній роботі ER-модель запропонувати відповідну реляційну. Перевірити відповідність реляційної моделі нормальним формам (1-3, Бойса-Кодда).. Якщо реляційна модель не відповідає одній з нормальних форм, то необхідно перетворити до форми, що задовольнить. У звіті необхідно довести відповідність нормальним формам шляхом обґрунтування та перерахування функціональних залежностей, що виявлено у предметній області

? Контрольні запитання

1. Що таке реляційна модель?
2. Наведіть складові частини реляційної моделі.
3. Що таке цілісність посилань?
4. Що таке цілісність сутностей?
5. Наведіть ознаки нормальних форм реляційної моделі.
6. Які переваги та недоліки нормалізації реляційної моделі бази даних?

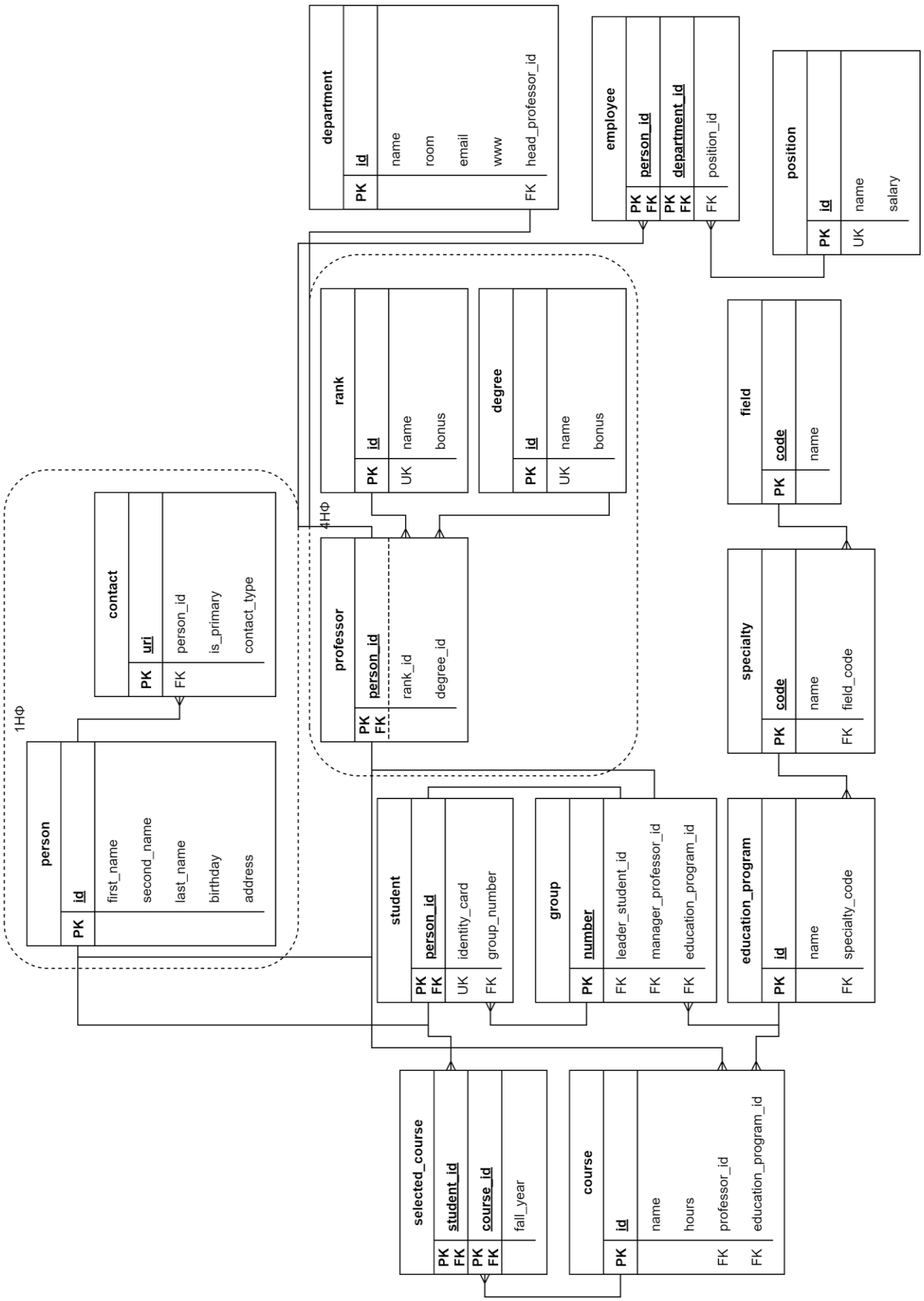

Рисунок 11 – Приклад нормалізованої реляційної моделі

Тема 3. Розробка бази даних на базі серверу MySQL

Теоретичні відомості

MySQL підтримує типи даних SQL з таких категорій:

- числові типи;
- дата і час;
- текстові типи;
- просторові типи;
- JSON.

MySQL забезпечує можливість використання стандартних числових типів SQL (табл. 1). Зокрема типи для подання точних значень (`INTEGER`, `SMALLINT`, `DECIMAL` і `NUMERIC`), а також типи для представлення дійсних чисел (`FLOAT`, `REAL` і `DOUBLE PRECISION`). Ключове слово `INT` – синонім для типу `INTEGER`, а також `DEC` і `FIXED` – синоніми `DECIMAL`. MySQL інтерпретує `DOUBLE` як синонім для `DOUBLE PRECISION`. MySQL також дозволяє використовувати `REAL` як синонім для `DOUBLE PRECISION`, якщо включено модуль `REAL_AS_FLOAT`.

Тип `BIT` зберігає бітові значення. Він підтримується MyISAM, MEMORY, InnoDB і NDB типами таблиць.

Таблиця 1. Типи даних для зберігання цілих чисел

Тип даних	Розмір сховища в байтах	Мінімальне значення зі знаком	Мінімальне значення без знаку	Максимальне значення зі знаком	Максимальне значення без знаку
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

MySQL використовує *чотири байта* для дійсних типів *одинарної точнос*ті та *вісім байт* у випадку *подвійної точності*. Точність від 0 до 23 значущих цифр для значень типу `FLOAT`. Водночас тип `DOUBLE` дозволяє зберігати до 53 значущих цифр.

`DATE`, `TIME`, `DATETIME`, `TIMESTAMP` і `YEAR` – типи даних, що представляють часові значення (табл. 2).

MySQL дозволяє для секунд використовувати дійсні значення з максимум 6 значущими цифрами після дійсної крапки для типів `TIME`, `DATETIME` і `TIMESTAMP`. Цифри після дійсної крапки подають мікросекунди. Для визначення стовпців , що зберігають значення з мікросекундами потрібно використовувати такий синтаксис:

```
type_name(fsp),
```

де `type_name` – це `TIME`, `DATETIME` або `TIMESTAMP`, а `fsp` – це кількість значущих цифр.

Таблиця 2. Типи даних для зберігання значень часу

Синтаксис	Опис
`DATE`	Дата. Дозволений діапазон значень – від `'1000-01-01'` до `'9999-12-31'`. MySQL відображає значення `DATE` у форматі `'YYYY-MM-DD'`, але дозволяє привласнення значень записам типу `DATE` з використанням рядків.
`DATETIME``[(fsp)]`	Комбінація дати і часу. Дозволений діапазон значень – від `'1000-01-01 00:00:00.000000'` до `'9999-12-31 23:59:59.999999'`. MySQL відображає значення `DATETIME` у форматі `'YYYY-MM-DD hh:mm:ss[.fraction]'`.
`TIMESTAMP``[(fsp)]`	Штамп часу. Дозволений діапазон значень – від `'1970-01-01 00:00:01.000000' UTC` до `'2038-01-19 03:14:07.999999' UTC`. Значення `TIMESTAMP` зберігаються ік кількість секунд від початку епохи Unix[6]. Тип `TIMESTAMP` не може зберегти значення `'1970-01-01 00:00:00'` тому що воно відповідає `0` секунд з початку епохи Unix, яке зарезервовано для значення `'0000-00-00 00:00:00'`, тобто для “нуля” `TIMESTAMP`.
`TIME``[(fsp)]`	Час. Дозволений діапазон значень – від `'-838:59:59.000000'` до `'838:59:59.000000'`. MySQL відображає значення `TIME` у форматі `'hh:mm:ss[.fraction]'`.
`YEAR``[(4)]`	Рік у форматі чотирьох цифр. MySQL відображає значення `YEAR` у форматі `YYYY`.

Рядкові типи даних представлено такими типами (табл. 3):

- `CHAR;`
- `VARCHAR;`
- `BINARY;`
- `VARBINARY;`

[6] Початком епохи Unix вважають 1 січня 1970 року.

- BLOB;
- TEXT;
- ENUM;
- SET.

Таблиця 3. Типи даних для зберігання текстової інформації

Синтаксис	Опис
[NATIONAL] **CHAR**[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]	Рядок фіксованої довжини, для зберігання якого виділяється фіксований обсяг пам'яті. M – довжина рядку у символах. Діапазон значень M від 0 до 255. Якщо M не зазначено, то пам'ять резервється під 1 символ.
[NATIONAL] **VARCHAR**(M) [CHARACTER SET charset_name] [COLLATE collation_name]	Рядок змінної довжини. M – максимальна кількість символів у рядку. Діапазон значень M від 0 до 65535. Обсяг пам'яті також обмежено значенням у 65535 байтів, що встановлює додаткові обмеження на максимальну кількість символів, що може бути збережена. Наприклад, кодування символів utf8 може вимагати до трьох байт на один символ, що обмежує стовпець VARCHAR кількістю 21844 символів.
BINARY[(M)]	Тип BINARY аналогічний до типу CHAR, але зберігає інформацію побайтово. Опціональне M – довжина поля у байтах. M за замовчуванням дорівнює 1.
VARBINARY(M)	Тип VARBINARY аналогічний до типу VARCHAR, але зберігає інформацію побайтово. M - максимальна кількість байт у запису.
BLOB[(M)]	Стовпець BLOB дозволяє зберігати значення максимального обсягу 65535 ($2^{16} - 1$) **байтів. Кожне значення типу** BLOB збережено з використанням 2-байтового префіксу, що визначає кількість збережених байт. Опціональне M дозволяє визначити тип з кількістю байт на значення, що достатнє для збереження M байтів.
TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]	Стовпець TEXT для збереження максимум 65535 ($2^{16} - 1$) **символів. Фактичний розмір кожного запису залежить від кількості символів у запису. Кожне значення** TEXT збережено з використанням 2-байтового префіксу, що визначає кількість збережених байт. Опціональне M дозволяє визначити тип з кількістю байт на значення, що достатнє для збереження M символів.

Синтаксис	Опис
ENUM('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]	Перелічуваний тип даних. Рядок, що може зберігати одне зі списку значень 'value1', 'value2', ..., NULL або спеціальне значення '', що відповідає помилці. Значення ENUM внутрішньо представлені як цілі значення. Стовпець ENUM може мати максимум 65535 різних елементів. Кількість символів у одному елементі обмежно значення 255, а максимальний обсяг пам'яті на один елемент – 1020 байтів.
SET('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]	Множина. Рядковий об'єкт, що може нуль або більше різних значень зі списку 'value1', 'value2', ... Значення типу SET внутрішньо представлено як цілі значення. Стовпець SET може максимально мати 64 різні елементи. Кількість символів у одному елементі обмежно значення 255, а максимальний обсяг пам'яті на один елемент – 1020 байтів.

MySQL має типи даних, що відповідають класам OpenGIS для обробки просторових даних. Типи даних, що дозволяють зберігати одиночні об'єкти геометрії, такі:

- GEOMETRY;
- POINT;
- LINESTRING;
- POLYGON.

GEOMETRY може зберігати об'єкт довільної форми. Інші типи (POINT, LINESTRING і POLYGON) відповідають певним геометричним об'єктам.

Колекції значень можуть зберігати такі просторові типи даних:

- MULTIPOINT;
- MULTILINESTRING;
- MULTIPOLYGON;
- GEOMETRYCOLLECTION.

GEOMETRYCOLLECTION може зберігати колекцію довільних геометричних об'єктів. Інші типи колекцій (MULTIPOINT, MULTILINESTRING і MULTIPOLYGON) можуть зберігати геометричні об'єкти лише відповідного типи.

MySQL дозволяє використовувати тип даних JSON для збереження документів у форматі JSON[7] згідно зі стандартом RFC 7159[8]. Тип даних JSON зберігає формату JSON як рядки. Водночас присутня валідація документів

[7] The JavaScript Object Notation (JSON) Data Interchange Format

[8] https://datatracker.ietf.org/doc/html/rfc7159.html

формату JSON, які зберігаються у стовпець типу `JSON`. Для документів, що не пройшли валідацію, генерується помилка.

Вираз `CREATE DATABASE` створює базу даних з певною назвою. Для використання цього виразу користувачу необхідно привілею `CREATE`. Вираз `CREATE SCHEMA` – синонім для `CREATE DATABASE`. Загальний синтаксис цього виразу такий:

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
  | COLLATE [=] collation_name
  | ENCRYPTION [=] {'Y' | 'N'}
}
```

Вираз `DROP DATABASE` видаляє всі таблиці з бази даних, а потім і саму базу даних. Для використання `DROP DATABASE` користувачу необхідна привілея `DROP`. Вираз `DROP SCHEMA` – синонім для `DROP DATABASE`. Загальний синтаксис виразу такий:

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

Вираз `CREATE TABLE` створює таблицю із заданим ім'ям. За замовчуванням таблиця створюється у поточній базі даних з використанням сховища InnoDB. Помилка може виникнути, якщо таблиця вже існує, немає поточної бази даних, або поточна база даних не існує.

MySQL не має обмеження на кількість таблиць. Проте файлова система, що використовується на сервері, може мати обмеження на кількість файлів. Окремі сховища можуть додавати власні обмеження на кількість таблиць. Наприклад, сховище InnoDB встановлює обмеження у 4 мільярди таблиць. Загальний синтаксис виразу для створення таблиць такий:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [partition_options]

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    [(create_definition,...)]
    [table_options]
    [partition_options]
    [IGNORE | REPLACE]
    [AS] query_expression

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    { LIKE old_tbl_name | (LIKE old_tbl_name) }

create_definition: {
    col_name column_definition
  | {INDEX | KEY} [index_name] [index_type] (key_part,...)
      [index_option] ...
  | {FULLTEXT | SPATIAL} [INDEX | KEY] [index_name] (key_part,...)
```

```
      [index_option] ...
  | [CONSTRAINT [symbol]] PRIMARY KEY
      [index_type] (key_part,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX | KEY]
      [index_name] [index_type] (key_part,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (col_name,...)
      reference_definition
  | check_constraint_definition
}

column_definition: {
    data_type [NOT NULL | NULL] [DEFAULT {literal | (expr)} ]
      [VISIBLE | INVISIBLE]
      [AUTO_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]
      [COMMENT 'string']
      [COLLATE collation_name]
      [COLUMN_FORMAT {FIXED | DYNAMIC | DEFAULT}]
      [ENGINE_ATTRIBUTE [=] 'string']
      [SECONDARY_ENGINE_ATTRIBUTE [=] 'string']
      [STORAGE {DISK | MEMORY}]
      [reference_definition]
      [check_constraint_definition]
  | data_type
      [COLLATE collation_name]
      [GENERATED ALWAYS] AS (expr)
      [VIRTUAL | STORED] [NOT NULL | NULL]
      [VISIBLE | INVISIBLE]
      [UNIQUE [KEY]] [[PRIMARY] KEY]
      [COMMENT 'string']
      [reference_definition]
      [check_constraint_definition]
}

data_type:
    (see Chapter 11, Data Types)

key_part: {col_name [(length)] | (expr)} [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option: {
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'
  | {VISIBLE | INVISIBLE}
  |ENGINE_ATTRIBUTE [=] 'string'
  |SECONDARY_ENGINE_ATTRIBUTE [=] 'string'
}

check_constraint_definition:
    [CONSTRAINT [symbol]] CHECK (expr) [[NOT] ENFORCED]

reference_definition:
    REFERENCES tbl_name (key_part,...)
      [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
      [ON DELETE reference_option]
      [ON UPDATE reference_option]

reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT
```

```
table_options:
    table_option [[,] table_option] ...

table_option: {
    AUTOEXTEND_SIZE [=] value
  | AUTO_INCREMENT [=] value
  | AVG_ROW_LENGTH [=] value
  | [DEFAULT] CHARACTER SET [=] charset_name
  | CHECKSUM [=] {0 | 1}
  | [DEFAULT] COLLATE [=] collation_name
  | COMMENT [=] 'string'
  | COMPRESSION [=] {'ZLIB' | 'LZ4' | 'NONE'}
  | CONNECTION [=] 'connect_string'
  | {DATA | INDEX} DIRECTORY [=] 'absolute path to directory'
  | DELAY_KEY_WRITE [=] {0 | 1}
  | ENCRYPTION [=] {'Y' | 'N'}
  | ENGINE [=] engine_name
  | ENGINE_ATTRIBUTE [=] 'string'
  | INSERT_METHOD [=] { NO | FIRST | LAST }
  | KEY_BLOCK_SIZE [=] value
  | MAX_ROWS [=] value
  | MIN_ROWS [=] value
  | PACK_KEYS [=] {0 | 1 | DEFAULT}
  | PASSWORD [=] 'string'
  | ROW_FORMAT [=] {DEFAULT | DYNAMIC | FIXED | COMPRESSED | REDUNDANT | COMPACT}
  | SECONDARY_ENGINE_ATTRIBUTE [=] 'string'
  | STATS_AUTO_RECALC [=] {DEFAULT | 0 | 1}
  | STATS_PERSISTENT [=] {DEFAULT | 0 | 1}
  | STATS_SAMPLE_PAGES [=] value
  | TABLESPACE tablespace_name [STORAGE {DISK | MEMORY}]
  | UNION [=] (tbl_name[,tbl_name]...)
}

partition_options:
    PARTITION BY
        { [LINEAR] HASH(expr)
        | [LINEAR] KEY [ALGORITHM={1 | 2}] (column_list)
        | RANGE{(expr) | COLUMNS(column_list)}
        | LIST{(expr) | COLUMNS(column_list)} }
    [PARTITIONS num]
    [SUBPARTITION BY
        { [LINEAR] HASH(expr)
        | [LINEAR] KEY [ALGORITHM={1 | 2}] (column_list) }
      [SUBPARTITIONS num]
    ]
    [(partition_definition [, partition_definition] ...)]

partition_definition:
    PARTITION partition_name
        [VALUES
            {LESS THAN {(expr | value_list) | MAXVALUE}
            |
            IN (value_list)}]
        [[STORAGE] ENGINE [=] engine_name]
        [COMMENT [=] 'string' ]
        [DATA DIRECTORY [=] 'data_dir']
        [INDEX DIRECTORY [=] 'index_dir']
        [MAX_ROWS [=] max_number_of_rows]
        [MIN_ROWS [=] min_number_of_rows]
        [TABLESPACE [=] tablespace_name]
        [(subpartition_definition [, subpartition_definition] ...)]

subpartition_definition:
    SUBPARTITION logical_name
        [[STORAGE] ENGINE [=] engine_name]
        [COMMENT [=] 'string' ]
```

```
        [DATA DIRECTORY [=] 'data_dir']
        [INDEX DIRECTORY [=] 'index_dir']
        [MAX_ROWS [=] max_number_of_rows]
        [MIN_ROWS [=] min_number_of_rows]
        [TABLESPACE [=] tablespace_name]

query_expression:
    SELECT ...   (Some valid select or union statement)
```

Вираз `DROP TABLE` видаляє одну або більше таблиць. Видалення таблиці зумовлює видалення тригерів, що прив'язано до цієї таблиці. Загальний синтаксис виразу такий:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

💻 Практичне завдання

Відповідно до запропонованої у попередньому практичному завданні моделі розробити запити для створення бази даних на сервері MySQL. Запроси повинні враховувати можливість існування бази даних з такою назвою (у випадку, якщо у базі даних таблиця вже існує запит повинен нічого не робити). Також необхідно розробити запити для коректного видалення усіх таблиць і самої бази даних. Запити повинні включати обмеження реляційної моделі (первинні та зовнішні ключи). Результат необхідно продемонструвати на сервері. Для створеної бази даних необхідно згенерувати ER-діаграму та порівняти її з діаграмами, що створено у попередніх завданнях.

? Контрольні запитання

1. Які категорії типів даних SQL відомі?
2. У чому полягає основна відмінність між CHAR і VARCHAR?
3. У чому полягає основна відмінність між BINARY і VARBINARY?
4. У чому полягає основна відмінність між CHAR і BINARY?
5. Яким чином додати до створеної таблиці обмеження зовнішнього ключа? Наведіть приклад.
6. Який порядок створення таблиць у базі даних?
7. Які привілеї необхідно мати для створення та видалення таблиць?

Тема 4. Розробка запитів для додавання, оновлення та видалення даних з таблиць

💾 Теоретичні відомості

Вирах `INSERT` вставляє нові рядки у таблицю. `INSERT ... VALUES`, `INSERT ... VALUES ROW()`, та `INSERT ... SET` такі форми використовують для вставки рядків на основі явно вказаних значень. `INSERT ... SELECT` дана форма вставляє рядки, що вибрані в іншій таблиці. Можна також використовувати `INSERT ... TABLE` у MySQL 8.0.19 та більш нових версіях для вставлення рядків з однієї таблиці. `INSERT` з виразом `ON DUPLICATE KEY UPDATE` дозволяє оновлювати наявні рядки, якщо рядок, який буде вставлено, зумовлює повторення значення первинного ключа або унікальних індексів. Загальний синтаксис додавання даних такий:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    { {VALUES | VALUE} (value_list) [, (value_list)] ...
      |
      VALUES row_constructor_list
    }
    [AS row_alias[(col_alias [, col_alias] ...)]]
    [ON DUPLICATE KEY UPDATE assignment_list]
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [AS row_alias[(col_alias [, col_alias] ...)]]
    SET assignment_list
    [ON DUPLICATE KEY UPDATE assignment_list]
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    [AS row_alias[(col_alias [, col_alias] ...)]]
    {SELECT ... | TABLE table_name}
    [ON DUPLICATE KEY UPDATE assignment_list]
value:
    {expr | DEFAULT}
value_list:
    value [, value] ...
row_constructor_list:
    ROW(value_list)[, ROW(value_list)][, ...]
assignment:
    col_name = [row_alias.]value
assignment_list:
    assignment [, assignment] ...
```

Вираз `INSERT VALUES` дозволяє додавати множну рядків. Для цього використовують список значень рядків. Значення кожного стовпця у рядку

розділяються комами та беруть у дужки. Список значень рядків також розділяють комами. Наприклад:

```
INSERT INTO tbl_name (a,b,c)
    VALUES(1,2,3), (4,5,6), (7,8,9);
```

Виразом `INSERT ... SELECT` можна швидко додати множину рядків у таблицю з результатів вибору даних з іншої таблиці. Наприклад:

```
INSERT INTO tbl_temp2 (fld_id)
  SELECT tbl_temp1.fld_order_id
  FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;
```

Якщо додати `ON DUPLICATE KEY UPDATE` то можна визначити поведінку щодо оновлення даних на випадок дублювання первинного ключа або унікального індексу. Наприклад:

```
INSERT INTO t1 (a,b,c) VALUES (1,2,3)
  ON DUPLICATE KEY UPDATE c=c+1;
UPDATE t1 SET c=c+1 WHERE a=1;
```

Вираз `UPDATE` дозволяє змінювати дані у таблиці. Синтаксис виразу такий:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET assignment_list
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
value:
    {expr | DEFAULT}
assignment:
    col_name = value
assignment_list:
    assignment [, assignment] ...
```

Вираз `UPDATE` оновлює дані зазначених стовпців. Фраза `SET` дані яких саме стовпців потрібно оновити. Кожне значення може бути визначене як вираз, ключове слово `DEFAULT` (для явного збереження значення за замовчуванням). Фраза `WHERE`, якщо задана, визначає умови для відбору рядків, що необхідно оновити. Якщо фразу `WHERE` не визначено, то усі рядки буде оновлено. Якщо зазначено фразу `ORDER BY`, то рядки буде оновлено у порядку, визначеному цією фразою. Фраза `LIMIT` дозволяє обмежити кількість рядків для оновлення.

Назва рядка може бути частиною виразу `UPDATE`. У такому випадку використовується поточне значення, наприклад, такий вираз збільшує значення рядку на 1:

```
UPDATE t1 SET col1 = col1 + 1;
```

Можливо застосовувати `UPDATE` для декількох таблиць. Проте у такому випадку не можна використовувати `ORDER BY` або `LIMIT`. Приклад застосування такий:

```
UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

`DELETE` це вираз, що дозволяє керувати видаленням записів із таблиці. Загальний синтаксис цього виразу такий:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

У виразах DELETE можна використовувати умови WHERE для специфікації умов відбирання рядків, що підлягають видаленню. Наприклад, такий запит видалить записи з реєстрацією до початку 2021 року:

```
DELETE FROM history WHERE registration_date  < '2021-01-01';
```

💻 Практичне завдання

Для запропонованої у попередній лабораторній роботі розробити запити:

1. Додавання тестових даних для кожної таблиці.
2. Оновлення тестових даних для кожної таблиці.
3. Видалення даних для кожної таблиці (повного та за деякою умовою).

? Контрольні запитання

1. Які особливості запитів Insert?
2. Наведіть приклада запиту Update із підзапитом.
3. Наведіть приклад створення таблиці із рядків, що вибрано з іншої таблиці.

Тема 5. Розробка запитів для пошуку даних у таблицях

Теоретичні відомості

Вираз `SELECT` використовують для отримання проєкції даних з однієї чи більше таблиць. Загальний синтаксис виразу такий:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr] ...
    [into_option]
    [FROM table_references
      [PARTITION partition_list]]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
    [HAVING where_condition]
    [WINDOW window_name AS (window_spec)
        [, window_name AS (window_spec)] ...]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [into_option]
    [FOR {UPDATE | SHARE}
        [OF tbl_name [, tbl_name] ...]
        [NOWAIT | SKIP LOCKED]
      | LOCK IN SHARE MODE]
    [into_option]

into_option: {
    INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
  | INTO DUMPFILE 'file_name'
  | INTO var_name [, var_name] ...
}
```

Для обробки результатів у виразі `SELECT` можна використовувати підсумкові функції (табл. 5.1). До таких функції часто застосовують вираз `GROUP BY` для визначення правил групування.

Таблиця 5.1. Підсумкові функції SQL

Назва функції	Призначення функції
`AVG()`	Обчислення середнього арифметичного значення
`BIT_AND()`	Повертає результат побітового “І”
`BIT_OR()`	Повертає результат побітового “АБО”

Назва функції	Призначення функції
`BIT_XOR()`	Повертає результат побітового “виключне АБО”
`COUNT()`	Обчислює кількість рядків (повторення можливі)
`COUNT(DISTINCT)`	Обчислює кількість різних рядків
`GROUP_CONCAT()`	Результат конкатенації рядків
`JSON_ARRAYAGG()`	Повертає результат як масив формату JSON
`JSON_OBJECTAGG()`	Повертає результат як єдиний об’єкт JSON
`MAX()`	Знаходить максимальне значення
`MIN()`	Знаходить мінімальне значення
`STD()`	Знаходить стандартне відхилення
`SUM()`	Знаходить суму

Наприклад, такий запит знаходить середній бал кожного студента:

```
mysql> SELECT student_name, AVG(test_score)
       FROM student
            GROUP BY student_name;
```

Такий запит може бути прикладом для знаходження кількості курсів для студента:

```
mysql> SELECT student.student_name,COUNT(*)
       FROM student,course
       WHERE student.student_id=course.student_id
            GROUP BY student_name;
```

Наступний запит показує приклад групової конкатенації результатів:

```
mysql> SELECT student_name,
         GROUP_CONCAT(DISTINCT test_score
                      ORDER BY test_score DESC SEPARATOR ' ')
       FROM student
            GROUP BY student_name;
```

💻 Практичне завдання

Для запропонованої у попередній лабораторній роботі розробити запити вибору даних з кожної таблиці за певними критеріями (деякі приклади наведено

у додатку Б). Мінімум 5 запитів для кожної таблиці, обов’язково показати використання підсумкових функцій та приклади розв’язку задач, які мають обмеження та результат підсумкових функцій.

? Контрольні запитання

1. Які особливості використання запитів SELECT?
2. Яким чином поєднати дві або три у одному запиті?
3. Яким чином накласти умову на результат підсумкової функції (наприклад, знайти студентів із середнім балом більшим або рівним 60)?

Тема 6. Розробка запитів з підзапитами

💾 Теоретичні відомості

Підзапит – це вираз `SELECT` у середині іншого запиту. Наприклад,

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

Основні переваги підзапитів:

- можливість структурувати та ізолювати частини запиту;
- можливість виконати складні операції без виразів `JOIN` і `UNION`;
- можливість підвищити читабельність запитів.

Найбільш поширена форма використання підзапитів така:

```
non_subquery_operand comparison_operator (subquery)
```

де `comparison_operator` – це один за таких операторів:

```
=  >  <  >=  <=  <>  !=  <=>
```

Наприклад,

```
... WHERE 'a' = (SELECT column1 FROM t1)
```

Також можливо створювати підзапити у формі:

```
non_subquery_operand LIKE (subquery)
```

Ремарка. Підзапит може бути лише у правій частині операції порівняння.

Наступним є приклад поширеного запиту для пошуку «найкращого результату», що не можна реалізувати засобами виразу `JOIN`:

```
SELECT * FROM t1
  WHERE column1 = (SELECT MAX(column2) FROM t2);
```

Для коректності операції порівняння, якщо ліва частина операції – скалярне значення, то і результат підзапиту у правій частині також повинен бути скалярним значенням.

💻 Практичне завдання

Для запропонованої у попередній лабораторній роботі розробити мінімум 5 запитів із підзапитами. Запити повинні показати особливості роботи із підсумковими функціями.

? Контрольні запитання

1. Які обмеження при використанні підзапитів?
2. Наведіть особливості застосування виразу ANY у підзапитах.
3. Наведіть особливості застосування виразу ALL у підзапитах.

Тема 7. Розробка збережуваних процедур і функцій

💾 Теоретичні відомості

MySQL дозволяє використовувати різні типи зебережуваних об’єктів:

- збережувана процедура – об’єкт, що створено виразом `CREATE PROCEDURE` та приведено у дію виразом `CALL`;
- збережувана функція – об’єкт, що створено виразом `CREATE FUNCTION`;
- тригер – об’єкт, що прив’язано до певної таблиці та активується при виконанні певної події;
- представлення – об’єкт, що є віртуальною таблицею, визначеною виразом `VIEW`.

Вираз для створення збережуваних процедур або функцій має такий синтаксис:

```
CREATE
    [DEFINER = user]
    PROCEDURE [IF NOT EXISTS] sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = user]
    FUNCTION [IF NOT EXISTS] sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic: {
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
}

routine_body:
    Valid SQL routine statement
```

Кожна збережувана підпрограма може включати кілька виразів SQL, які розділено крапкою з комою (`;`). Наприклад, така процедура має тіло, що визначено операторами `BEGIN ... END`, які містять вираз `SET` і цикл `REPEAT` для повторення іншого виразу `SET`:

```
CREATE PROCEDURE dorepeat(p1 INT)
BEGIN
  SET @x = 0;
  REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
END;
```

Для заміни роздільника при визначенні збережуваних підпрограм використовують команду `delimiter`. Такий приклад показує як створити та викликати процедуру:

```
mysql> delimiter //

mysql> CREATE PROCEDURE dorepeat(p1 INT)
    -> BEGIN
    ->   SET @x = 0;
    ->   REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> CALL dorepeat(1000);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x;
+------+
| @x   |
+------+
| 1001 |
+------+
     1 row in set (0.00 sec)
```

💻 Практичне завдання

Для кожного запиту на вибір даних з попередньої роботи розробити збережувану процедуру або функцію для серверу MySQL. Показати коректність роботи розроблених підпрограм. Розробити скрипти для вимірювання часу виконання збережуванинх процедур, порівняти результати вимірювання часу із часом, що необхідно на виконання еквівалентних запитів.

? Контрольні запитання

1. Які обмеження характерні для підпрограм у SQL?
2. Які відмінності між процедурами та функціями у SQL?
3. Чи дозволяють збережувані підпрограми збільшити швидкість роботи інформаційної системи?

Тема 8. Розробка тригерів

💾 Теоретичні відомості

Тригер – це іменований об’єкт бази даних, що асоціюється з таблицею та активується, коли певна подія відбувається. Наприклад, тригери використовуються для перевірок даних, що додаються або змінюються.

Тригер можна створити для активації, коли дані додаються, змінюються або видаляються. Наприклад, перед додаванням кожного нового рядка у таблицю.

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
       FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.01 sec)
```

Вище наведено приклад тригера для обчислення суми після кожної операції додавання нового рядка.

💻 Практичне завдання

1. Для кожної таблиці бази даних з попередньої лабораторної роботи розробити журналювання операцій оновлення та видалення даних. Для цього до бази даних необхідно додати таблицю з назвою log, що має поля table – назва таблиці, operation – виконана операція, time – повний час виконання операції.

? Контрольні запитання

1. Що таке тригер?
2. Які обмеження мають тригери?
3. Які переваги застосування тригерів?

Тема 9. Створення бази даних у документ-орієнтованій системі керування базами даних MongoDB

💾 Теоретичні відомості

MongoDB – документо-орієнтована база даних. Кожен запис у MongoDB – документ, структура якого складається із пар поле-значення. Документи аналогічні об'єктів JSON. Значення полів можуть включати інші документи, масиви, масиви документів.

Документи об'єднують у колекції, колекції у бази даних. Команда use використовується для використання або створення (якщо не існує) бази даних. Наприклад,

```
use myDB
```

створить, якщо не існує базу myDB.

Колекції визначаються також динамічно. Наприклад, команди

```
use myNewDB
db.myNewCollection1.insertOne( { x: 1 } )
```

створюють колекцію `myNewCollection1` з одним документом, де поле `x` дорівнює `1`. Кожна наступна команда `insertOne` буде додавати новий документ до цієї колекції.

Колекції можна створити явним чином із використанням команди `db.createCollection()`. Загальний синтаксис цієї команди такий:

```
db.createCollection( <name>,
   {
     capped: <boolean>,
     timeseries: {                  // Added in MongoDB 5.0
        timeField: <string>,        // required for time series collections
        metaField: <string>,
        granularity: <string>
     },
     expireAfterSeconds: <number>,
     clusteredIndex: <document>,    // Added in MongoDB 5.3
     changeStreamPreAndPostImages: <document>,  // Added in MongoDB 6.0
     size: <number>,
     max: <number>,
```

```
      storageEngine: <document>,
      validator: <document>,
      validationLevel: <string>,
      validationAction: <string>,
      indexOptionDefaults: <document>,
      viewOn: <string>,
      pipeline: <pipeline>,
      collation: <document>,
      writeConcern: <document>
   }
)
```

Команда `createCollection`, наприклад, може бути використана для створення колекції з даними про погоду за останні 24 години:

```
db.createCollection(
    "weather24h",
    {
       timeseries: {
          timeField: "timestamp",
          metaField: "data",
          granularity: "hours"
       },
       expireAfterSeconds: 86400
    }
)
```

Документи є об’єктами типу BSON. Наприклад,

```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
             }
```

Документи мають такі обмеження щодо імен полів:

- ім’я `_id` зарезервовано для первинного ключа; його значення повинно бути унікальним у межах колекції;
- сервер дозволяє зберігати поля з іменами, що мають крапку (.) і dзнак долара (`$`).

💻 Практичне завдання

Для моделі предметної області (лабораторна робота № 4) розробити базу документ-орієнтовану модель і базу даних (MongoDB). Запропоновувати селектори для наповнення бази даних інформацією, які будуть відповідні до запитів додавання тестових даних лабораторної роботи № 7.

? Контрольні запитання

1. Що таке документо-орієнтована база даних?
2. Які типи нереляційних баз даних Вам відомі?
3. Назвіть переваги та недоліки нереляційних баз даних.
4. Які типові юз-кейси для нереляційних баз даних?

Тема 10. MongoDB: селектори обробки і пошуку інформації

💾 Теоретичні відомості

Метод `find` використовують для пошуку даних у колекції. Розглянемо використання цього методу на прикладі такої колекції:

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Для того щоб вибрати усі документи колекції використовуємо пустий документ як параметр методу `find`:

```
db.inventory.find( {} )
```

що є еквівалентом до

```
SELECT * FROM inventory;
```

Наступний приклад показує як вибрати усі документи де поле status дорівнює значенню "D".

```
db.inventory.find( { status: "D" } )
```

Цей селектор відповідає запиту

```
SELECT * FROM inventory WHERE status = "D"
```

Еквівалентом запиту `SELECT * FROM inventory WHERE status in ("A", "D")` є такий селектор:

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

Селектор

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

демонструє як вибрати документи за статусом “A” **і** qty < 30 (еквівалентний запит SQL: `SELECT * FROM inventory WHERE status = "A" AND qty < 30`).

Селектор

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
```

демонструє як вибрати документи за статусом “A” **або** qty < 30.

Регулярні вирази можна використовувати для пошуку колекціями, наприклад, такий селектор:

```
db.inventory.find( {
     status: "A",
     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} )
```

є еквівалентом запиту

```
SELECT * FROM inventory WHERE status = "A" AND ( qty < 30 OR item LIKE "p%")
```

💻 Практичне завдання

Для документ-орієнтованої бази даних з попередньої лабораторної роботи розробити селектори, які будуть аналогічними за поведінкою до розроблених у лабораторній роботі № 7. Порівняти час роботи запитів для MySQL з часом роботи селекторів MongoDB.

? Контрольні запитання

1. Які типи селекторів відомі?
2. Наведіть перелік операції порівняння для селекторів.
3. Яким чином видалити документ із колекції?
4. Чи є аналог підзапиту для колецій?

Тема 11. Створення бази даних типу ключ-значення на базі Redis

💾 Теоретичні відомості

Redis — це сховище структури даних із відкритим вихідним кодом (ліцензія BSD), яке використовується як база даних, кеш-пам’ять, брокер повідомлень і механізм потокового передавання. Redis надає такі структури даних, як рядки, хеші, списки, набори, відсортовані набори із запитами діапазонів, растрові зображення, гіперлоглоги, геопросторові індекси та потоки.

Ключі Redis є бінарними, це означає, що ви можете використовувати будь-яку двійкову послідовність як ключ, від рядка на зразок "foo" до вмісту файлу JPEG. Порожній рядок також є дійсним ключем.

Ще кілька правил щодо ключів:

- Дуже довгі ключі не є гарною ідеєю. Наприклад, ключ розміром 1024 байти є поганою ідеєю не тільки з точки зору пам’яті, але й тому, що пошук ключа в наборі даних може вимагати кількох дорогих порівнянь ключів. Навіть якщо поставлене завдання полягає в тому, щоб зіставити існування великого значення, його хешування (наприклад, за допомогою SHA1) є кращою ідеєю, особливо з точки зору пам’яті та пропускної здатності.
- Дуже короткі ключі часто не є гарною ідеєю. Немає сенсу писати "u1000flw" як ключ, якщо замість цього можна написати "user:1000:followers". Останній є більш читабельним, а доданий простір незначний у порівнянні з простором, який використовується самим ключовим об’єктом і об’єктом значення. Хоча короткі ключі, очевидно, споживатимуть трохи менше пам’яті, ваше завдання — знайти правильний баланс.
- Спробуйте дотримуватися схеми. Наприклад, "тип об'єкта:id" є хорошою ідеєю, як у "user:1000". Крапки або тире часто використовуються для багатослівних полів, як-от «comment:4321:reply.to» або «comment:4321:reply-to».

💻 Практичне завдання

Для таблиць-довідників з лабораторної роботи № 6 розробити нереляційні бази даних для Redis. Порівняти час обробки довідникових даних у MySQL і Redis (для цього необхідно розробити коди додавання, оновлення, вибору і видалення даних цих таблиць еквівалентні до аналогічних запитів з лабораторної роботи № 7).

? Контрольні запитання

1. Що таке база даних типу ключ-сховище?
2. Які юз-кейси використання бази даних ключ-сховище?
3. Наведіть приклади можливого використання Redis для вашої предметної області.

Індивідуальне завдання. Розробка інформаційної системи на базі файл-серверної СКБД SQLite

💻 Практичне завдання

Використовуючи будь-яку мову програмування розробити інформаційну систему (програму-клієнта) для обробки бази даних еквівалентної до розробленої протягом виконання лабораторних робіт. База даних повинна зберігатися у файл-серверної СКБД SQLite. Результат для перегляду виконання запитів, розроблених у лабораторній роботі № 7, необхідно передбачити відповідні пункти меню інформаційної системи.

Глосарій

1. *База даних* (англ. database) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування.
2. *База даних «ключ-значення» (англ. key-value database або англ. key–value store)* – парадигма сховищ даних, створена для зберігання, виймання та керування асоціативними масивами – структурою даних, більш знаною як словник або геш.
3. *Відношення* – фундаментальне поняття реляційної моделі даних, воно може буде представлене у вигляді таблиці, стовпці (поля, атрибути) якої відповідають входженням доменів у відношення, а рядки (записи, кортежі) — наборам з n значень, що взяті з початкових доменів.
4. *Дані* – це набір конкретних значень параметрів, що характеризують певний об’єкт або процес.
5. *Документо-орієнтована система керування базами даних (англ. document-oriented database)* – система керування базами даних, спеціально призначена для зберігання ієрархічних структур даних (документів) і зазвичай реалізована за допомогою підходу NoSQL.
6. *Запит* – це формулювання своєї інформаційної потреби користувачем деякої бази даних або інформаційної системи, наприклад, пошукової системи. Для складання запиту використовується мова пошукових запитів.
7. *Інформаційна система* – комунікаційна система, що забезпечує збирання, пошук, оброблення та пересилання інформації
8. *Модель даних* – результат систематизації інформації, відображення її властивостей, структури, зв’язків між її елементами.
9. *Нормалізація схеми бази даних* – покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.
10. *Нормальна форма* — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних.
11. *Предметна область* – частина реального або уявного світу.
12. *Реляційна база даних* – база даних, заснована на реляційній моделі даних.
13. *Реляційна модель даних* — логічна модель даних, що уперше була запропонована британським ученим співробітником компанії IBM Едгаром Франком Коддом, яка ґрунтується на використанні таблиць для збереження властивостей.
14. *Реляційна система керування базами даних (РСКБД; інакше Система керування реляційними базами даних, СКРБД)* – СКБД, що керує реляційними базами даних.

15. *СУБД* – система управління базами даних, англ. Database Management System, DBMS.
16. *Сутності предметної області* є результатом абстрагування шляхом і фіксації важливих властивостей об’єктів або процесів предметної.
17. *Ядро предметної області* – сукупність об’єктів або процесів, що входять до моделей предметної області.
18. NoSQL (зазвичай розшифровується як англ. non SQL або англ. non relational, іноді англ. not only SQL) – база даних, яка забезпечує механізм зберігання та видобування даних відмінний від підходу таблиць-відношень в реляційних базах даних.

Список літератури

1. ДСТУ 2392-94 Інформація та документація. Базові поняття.
2. Програмне та алгоритмічне забезпечення сховищ та просторів даних: монографія / Н. Б. Шаховська ; Нац. ун-т "Львів. політехніка". Л. : Вид-во Львів. політехніки, 2010. 194 с.
3. Сховища та простори даних: монографія / Н. Б. Шаховська, В. В. Пасічник ; Національний ун-т "Львівська політехніка". Л. : НУ "Львівська політехніка", 2009. 244 с.
4. Реляційні бази даних: табличні алгебри та SQL-подібні мови / В. Н. Редько [и др.]. К. : Видавничий дім "Академперіодика", 2001. 198 с.
5. Часові бази даних: моделі та методи реалізації : [монографія] / П. І. Жежнич ; Національний ун-т "Львівська політехніка". Л. : Видавництво Національного ун-ту "Львівська політехніка", 2007. 259 с.
6. Бази даних MySQL : навч. посіб. / Н. Р. Балик, В. І. Мандзюк. Т. : Навчальна книга - Богдан, 2010. 157 с.
7. Meier A., Kaufmann M. SQL & NoSQL Databases: Models, Languages, Consistency Options and Architectures for Big Data Management. Springer Fachmedien Wiesbaden, 2019.
8. Darmont J., Novikov, B., Wrembel, R. New Trends in Databases and Information Systems: ADBIS 2020 Short Papers, Lyon, France, August 25–27, 2020, Proceedings. Communications in Computer and Information Science. Vol. 1259. Springer International Publishing, 2020.
9. Sciore E. Database Design and Implementation: Second Edition. Springer International Publishing, 2020. 458 p.
10. Як відбувається пошук в MySQL – керівництво для початківців. URL: http://yoip.com.ua/yak-vidbuvayetsya-poshuk-v-mysql-kerivnitstvo-dlya-pochatkivtsiv/
11. 11 типів сучасних баз даних. URL: https://proglib.io/p/11-tipov-sovremennyh-baz-dannyh-kratkie-opisaniya-shemy-i-primery-bd-2020-01-07
12. SQL база даних. URL: https://www.ukraine.com.ua/blog/programming/sql-baza-dannih-dlya-chego-prednaznachena-baza-dannih.html
13. Що таке база даних? URL: http://apeps.kpi.ua/shco-take-basa-danykh
14. 11 типів сучасних баз даних: короткий опис, схеми і приклади БД. URL: https://senior.ua/articles/11-tipv-suchasnih-baz-danih-korotkiy-opis-shemi--prikladi-bd

Додаток А. Індивідуальні теми для створення баз даних

У табл. А.1 наведено формулювання варіантів завдань для розробки тематичних баз даних та інформаційних систем при виконанні лабораторних робіт та індивідуального завдання.

Таблиця А.1 – Індивідуальні теми для створення баз даних

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
1	**База даних «Сесія»** База даних повинна містити інформацію про студентів, групи і предмети, з яких здаються іспити або заліки. Кожен студент за сесію складає декілька іспитів і заліків, з яких він отримує оцінку або відмітку про залік. Групі студентів іспит (залік) призначається одночасно. Під час сесії для викладачів формується розклад, для керівництва звіт за результатами складання	Студенти Викладачі Предмети	ПІБ студента Номер залікової книжки Номер групи ПІБ викладача Посада викладача Назва предмету Обсяг годин предмету Дата проведення іспиту Підсумковий бал
2	**База даних «Розклад занять»** При складанні розкладу занять працівники навчальної частини повинні враховувати зайнятість та типи аудиторій, кількість студентів у групі, наявність певних мультимедійних засобів. Лекції можуть проводитися декільком групам одночасно. Розклад складається за днями тижня та номерами пар. Після складання розкладу автоматично формуються версії для студентів (розклад групи), викладачів (персональний розклад), аудиторій (групи і викладачі в аудиторії).	Групи Викладачі Предмети Аудиторії	Номер групи Спеціальність Факультет Назва дисципліни Тип заняття День тижня Час проведення Регулярність (щотижнева, чисельник або знаменник) Номер аудиторії Номер корпусу Тип аудиторії Кількість місць

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
3	**База даних «Вступ»** При вступі до ВНЗ абітурієнт подає документи на певні спеціальності. Оригінал документу на підставі якого відбувається вступ може біти покладений у діло тільки однієї спеціальності. Вступ відбувається за балами сертифікатів ЗНО. Автоматично формуються звіти з популярності спеціальностей та формуються рейтинги абітурієнтів.	Абітурієнти Спеціальності Документи	ПІБ абітурієнта Контактні дані абітурієнта Назва документу, його серія, номер, дата видачі, найменування органу, що видав Бали сертифікатів ЗНО Назва спеціальності Факультет
4	**База даних «Готель»** Клієнти готелю резервують кімнати на певні проміжки часу (дні). Кімнати мають кількість спальних місць та тип (економ, стандарт, люкс тощо), за яким визначається вартість проживання. Для адміністрації готелю автоматично формується графік заїздів виїздів, рейтинги популярності кімнат.	Клієнти Кімнати	ПІБ клієнта Кількість осіб, що буде разом з клієнтом Паспортні дані клієнта Номер кімнати Кількість місць у номері Тип номеру Дата та час заїзду/виїзду
5	**База даних «Поліклініка»** Реєстратура поліклініки формує записи пацієнтів до лікарів-спеціалістів. У кожного лікаря є власний кабінет і часи прийому, за якими орієнтуються пацієнти. За результатами прийму лікар ставить діагноз який має стандартний державний шифр і тлумачення. Для кожного лікаря щоденно формується перелік записаних на наступний день пацієнтів, адміністрація поліклініки	Лікарі Пацієнти Діагнози	ПІБ лікаря Спеціальність лікаря Години прийому Номер кабінету ПІБ пацієнта Шифр діагнозу Тлумачення шифру Дата та час запису пацієнта до лікаря Відділення

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
	щотижнево отримує звіт у розрізі діагнозів та активності лікарів.		
6	**База даних «Виклики поліції»** Поліція при виконанні власних обов'язків повинна приїздити на виклики за дзвінками громадян. На виклик відправляється вільний екіпаж, який за його результатами складає звіт, що зберігається в системі. Не всі працівники поліцію вміють водити машину, тому при формуванні екіпажу необхідно забезпечити присутність мінімум одного працівника з посвідченням водія. Щоденно керівництво відділку отримує звіти за викликами.	Адреси Працівники поліції	Вулиця, номер дому і квартира ПІБ персони, що викликала Дата і час виклику Номер екіпажу ПІБ членів екіпажу Звіт екіпажу за викликом
7	**База даних «Штатний розклад»** При складанні шатного розкладу організації штатні одиниці розподіляються за підрозділами. Кожна штатна одиниця характеризується назвою посади, розміром окладу, відсотком надбавки за ненормований робочий день. Кожен підрозділ характеризується назвою, типом, відсотком надбавки за шкідливі умови праці. Кожен працівник займає певну штатну одиницю підрозділу на деяку долю ставки та має персональні надбавки визначені стажем роботи, додатковою освітою. Адміністрація може запросити	Підрозділи Штатні одиниці Працівники	Назва підрозділу Тип підрозділу Відсоток надбавки підрозділу за шкідливі умови Назва посади Розмір окладу за посадою Надбавка посади за ненормований робочий день ПІБ працівника Табельний номер працівника Стаж роботи на момент прийняття на роботу Дата початку роботи на підприємстві Дипломи працівника

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
	звіт для визначення кількості вакантних посад, персональний склад кожного підрозділу.		
8	**База даних «Виклики швидкої»** Карети швидкої допомоги відправляються на виклики за адресами. При виклик характеризується датою і часом. За результатом виклику ставиться діагноз, відбувається або не відбувається госпіталізація. Кожна карета швидкої складається з водія, санітару та лікаря. Для адміністрації формуються звіти, що надають інформацію про активність карет і діагнози.	Адреси Працівники лікарні	Вулиця, номер дому і квартира ПІБ персони, що викликала Дата і час виклику ПІБ пацієнта Вік пацієнта Встановлений діагноз Номер карети швидкої ПІБ водія ПІБ санітару ПІБ лікаря
9	**База даних «Лікарня»** Лікарня складається з відділень, що характеризується назвою, має штат лікарів, один з яких є головним у відділенні. Пацієнти поступають у відділення з певним діагнозом. Кожному пацієнту призначається лікуючий лікар зі складу лікарів відділення. Лікуючий лікар може призначити додатковий огляд лікарями з інших відділень, які складають власні діагнози. Всі діагнози автоматично складаються у єдиний при виписці з лікарні.	Пацієнти Лікарі Відділення	ПІБ пацієнта Дата народження пацієнта Назва відділення ПІБ лікаря Спеціальність лікаря Діагноз Палата Дата початку лікування Дата виписки Відділення
10	**База даних «Страхування»** Договір страхування складається між компанією та клієнтом. Догорів страхування має певний вид, страхову суму,	Клієнти Страхові агенти Договори	ПІБ клієнта Дані клієнта ПІБ страхового агенту Номер страхового договору

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
	страхову премію, вартість (щомісячна або одноразова) та термін дії. Договори складають страхові агенти, які отримають комісійне винагородження за власну роботу. Керівництво бажає автоматично бачити звіти з активності агентів, суму комісійної винагороди кожного агенту та суму надходжень від клієнтів за певний проміжок часу, популярність типів страхових послуг.		Страхова сума Страхова премія Дата укладення Дата припинення дії Що місячна або одноразова вартість Комісія агенту
11	**База даних «Науковий журнал»** У науковий журнал надсилаються авторські статті. Кожна стаття має мінімум одного автора, три анотації (англійська, російська, українська), ключові слова, мову написання та файл з повним текстом, що зберігається на сервері. Для кожної статті редакційна колегія журналу призначає двох рецензентів, які проставляють оцінки (від 1 до 10) за критеріями новизни, актуальності, повноти літературного огляду, повноти викладення матеріалу, якості тексту та пишуть звіт з власним враженням для редакційної колегії, зауваження для авторів. Адміністрація автоматично отримує оцінки та звіти рецензентів, автори автоматично отримують	Автори Статті Рецензенти	ПІБ і контактні дані автора Назва статті Анотації трьома мовами Ключові слова трьома мовами Адреса файлу на сервері ПІБ рецензента Перелік тематичних напрямів рецензенту (трьома мовами) Оцінки статті за критеріями Висновок рецензента для редакції Зауваження рецензента для авторів статті Розділ журналу Підрозділ журналу

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
	оцінки та зауваження рецензентів (анонімно).		
12	**База даних «Агентство нерухомості»** Агентство нерухомості займається оформленням контрактів з нерухомості. Для цього виконують облік продавців нерухомості і клієнтів, що бажають її придбати. Кожен об'єкт нерухомості характеризується адресую, станом (потрібність ремонту), вартістю, згодою чи незгодою власника на торг. Кожен клієнт характеризується персональними даними, максимальною сумою, що може бути витрачена, кількісними вимогами (мінімальний та максимальний поверх, кількість кімнат, бажаний район). Агентство при згоді обох сторін укладаю договір, що містить суму, до якої дійшли сторони, і дату укладання. Договори укладають агенти агентства, які за результатами роботи отримують платню.	Нерухомість Клієнти Агенти Договори	Адреса об'єкта нерухомості Тип (квартира, дім, ділянка землі тощо) ПІБ власника Площа Кількість кімнат Поверх Кількість поверхів Загальна площа Жила площа Площа кухні Тип санвузлу Вартість ПІБ клієнта Орієнтована суму для придбання нерухомості Тип і характеристики об'єкта побажання Контактні дані Номер договору Сума договору Дата підписання Агент, що супроводжує договір.
13	**База даних «Фото-Сервіс»** Фото сервіс на кшталт Instagram дозволяє публікувати власні фото з коментарями, які будуть доступні широкому загалу або тільки друзям. Фото опубліковані для всіх можуть коментуватися будь-ким, а ті, що з обмеженим доступом, тільки друзями.	Клієнти Фотографії	Персональна інформація клієнту Друзі клієнту Назва фото Файл з фото Теги фото Коментарі до фото

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
14	**База даних «Інтернет-Розсилка»** Інтернет видання публікує замітки за різною тематикою. Клієнти сервісу бажають отримувати повідомлення про публікацію нових матеріалів з певними ключовими словами або за обраними темами. Клієнти налаштовують структуру таких повідомлень (необхідність отримання тільки заголовків або заголовки з анотаціями, або поний текст).	Клієнти Матеріали	Персональна інформація клієнту (ПІБ, email тощо) Ключові слова клієнту Теми клієнту Назва матеріалу Анотація матеріалу Ключові слова матеріалу Автори матеріалу Вартість підписки матеріалу Розділ каталогу Підрозділ каталогу
15	**База даних «Туристичне агентство»** Туристичне агентство надає послуги з організації відпочинку на курортах. Кожен курорт має тип (горно-лижний, морський тощо), рівень якості (у зірочках), країна та місце знаходження, пропозиції щодо вартості відпочинку. Клієнт обирає курорт, а працівник агентства оформляє договір. З кожного оформленого договору агент отримує власні комісійні.	Курорти Клієнти Агенти	Назва курорту Тип курорту Якість курорту Місце знаходження Перелік пропозицій щодо вартості відпочинку на курорті Персональні дані замовника Персональні дані агенту Комісійний відсоток агенту Дата укладання договору Термін відпочинку
16	**База даних «Відділ збиту»** Відділ збиту підприємства займається оформлень продаж. Необхідно фіксувати остачу кожного товару на складі після його продажу, дату продажу, кількість одиниць. Кожна одиниця товару має вартість, метрику (штуки, кілограми, метри тощо). Товари відвантажуються за накладною	Товари Замовники Накладні	Назва товари Метрика товару Кількість товару Назва замовника Ідентифікаційний номер замовники Номер накладної Товар накладної Отримувач товару Працівник, що оформив продаж

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
17	**База даних «Порушення правил дорожнього руху»** Система автоматично фіксує порушення правил дорожнього руху шляхом збереження номеру автомобіля. За номером автомобіля визначається його власник, який потім за збереженою адресою отримує повідомлення про штраф. Для кожного порушення призначається працівник, який перевіряє коректність роботи автоматики системи.	Автомобілі Порушення	Номер авто Власник авто Адреса власника авто Тип авто Шифр порушення Штраф за порушення Дата і час порушення Працівник для перевірки коректності призначення Розділ каталогу Підрозділ каталогу
18	**База даних «Прокат автомобілів»** Фірма прокату автомобілів надає інформацію про наявні для прокату авто (виробник, марка, кількість місць для сидіння, колір, вартість) та фіксує угоди з клієнтами (термін, авто).	Авто Клієнти Угоди	Номер авто Виробник авто Марка авто Тип авто Кількість місць для сидіння Колір авто Вартість одного дня прокату Персональні дані клієнта Термін прокату
19	**База даних «Інтернет-магазин»** Інтернет-магазин продає власні товари зареєстрованим клієнтам. Кожен товар має категорію, характеристики, вартість, одиниці вимірювання (штуки, кілограми тощо). Клієнт формує корзину покупця обираючи товари в певній кількості та зазначає адресу доставки.	Товари Клієнти Корзина покупця	Назва товару Категорія товару Характеристики товару Одиниця вимірювання товару Вартість одиниці товару Персональні дані клієнту Дата покупки Кількість обраних одиниць Адреса доставки Розділ каталогу

№ з/п	Назва бази даних та опис предметної області	Мінімальний набір базових таблиць	Мінімальний набір полів таблиць
			Підрозділ каталогу
20	**База даних «Музичний каталог»** Онлайн музичний каталог дозволяє зареєстрованим користувачам публікувати власні музичні. Музичні композицію мають назву, авторів, жанр, формат файлу збереження, теги. Автором композиції може бути один або декілька зареєстрованих користувачів. Кожну музичну композицію можна зробити доступною всім відвідувачам каталогу або тільки зареєстрованим користувачам, або обраному колу зареєстрованих користувачів. До кожної композиції автор може додати допоміжні матеріали: текст, ноти, графічний логотип тощо. Коментарі до композицій можуть залишати тільки зареєстровані користувачі. Автор може відповідати на коментарі, формуючи дерево бесіди.	Користувачі Композиції Коментарі	ПІБ користувача Контактні дані користувача Назва композиції Жанр Текс коментарю Розділ каталогу Підрозділ каталогу

Додаток Б. Мінімальний перелік запитів і селекторів для баз даних

У табл. Б.1 наведено формулювання варіантів завдань для розробки запитів і селекторів для тематичних баз даних та інформаційних систем при виконанні лабораторних робіт та індивідуального завдання.

Таблиця Б.1 – Перелік завдань для розробки запитів

№ з/п	Назва бази даних	Завдання для розробки запитів
1	**База даних «Сесія»**	1. Вивести повну інформацію про викладачів та предмети, які вони викладають. 2. Вивести повну інформацію про студентів та предмети, які вони вивчають. 3. Вивести інформацію про розклад складання іспитів та заліків групами студентів по кожному з предметів із зазначенням: - дата проведення контролю; - тип контролю – іспит чи залік; - група; - предмет; - викладач. 4. Вивести інформацію про результати складання іспитів та заліків студентами по кожному предмету по групах. 5. Вивести рейтинг студентів (сума балів за іспити та заліки) по групах та відсортувати за показником рейтингу по кожній групі.
2	**База даних «Розклад занять»**	1. Вивести інформацію про розклад групи. 2. Вивести інформацію про розклад викладача. 3. Вивести інформацію про розклад аудиторії. 4. Вивести інформацію про загальній розклад навчального процесу (групи, викладачі, аудиторії).
3	**База даних «Вступ»**	1. Вивести повну інформацію про абітурієнтів . 2. Вивести повну інформацію про документи абітурієнта.

№ з/п	Назва бази даних	Завдання для розробки запитів
		3. Вивести інформацію про абітурієнтів та спеціальності, до яких вони подали документи на вступ, із зазначенням до якої спеціальності був поданий оригінал документів. 4. Вивести рейтинг спеціальностей. 5. Вивести рейтинг абітурієнтів по кожній спеціальності.
4	**База даних «Готель»**	1. Вивести повну інформацію про всі номери готелю. 2. Вивести повну інформацію про всіх клієнтів готелю. 3. Вивести інформацію про всі відвідування готелю клієнтами із зазначенням дат заїзду та виїзду, обраного номеру та сплаченої вартості проживання. 4. Вивести інформацію про вільні номери готелю для заданого проміжку часу. 5. Вивести повну інформацію про клієнтів, що проживають у готелі на даний час.
5	**База даних «Поліклініка»**	1. Вивести повну інформацію про лікарів поліклініки. 2. Вивести повну інформацію про пацієнтів поліклініки. 3. Вивести інформацію про пацієнтів, записаних на прийом до певного лікаря. 4. Вивести повну інформацію про всі діагнози, що поставили лікарі пацієнту. 5. Вивести повну інформацію про всі діагнози, що поставив лікар пацієнтам.
6	**База даних «Виклики поліції»**	1. Вивести повну інформацію про працівників поліції. 2. Вивести повну інформацію про состав екіпажів поліції. 3. Вивести повну інформацію про виклики поліції. 4. Вивести повну інформацію про звіти екіпажів за викликами.
7	**База даних «Штатний розклад»**	1. Вивести повну інформацію про підрозділи.

№ з/п	Назва бази даних	Завдання для розробки запитів
		2. Вивести повну інформацію про штатні одиниці за підрозділами. 3. Вивести повну інформацію про працівників. 4. Вивести повну інформацію про розподіл працівників за штатними одиницями підрозділів. 5. Вивести повну інформацію про вакантні посади з долею ставок за штатними одиницями підрозділів.
8	**База даних «Виклики швидкої»**	1. Вивести повну інформацію про працівників швидкої. 2. Вивести повну інформацію про карети швидкої. 3. Вивести повну інформацію про виклики швидкої. 4. Вивести повну інформацію про пацієнта та виклики швидкої до пацієнта. 5. Знайти екіпажи з найбільшою і найменшою кількістю викликів.
9	**База даних «Лікарня»**	1. Вивести повну інформацію про працівників лікарні. 2. Вивести повну інформацію про пацієнтів лікарні. 3. Вивести інформацію по кожному відділенню. 4. Знайти палату з найменшою кількістю пацієнтів. 5. Знайти середню кількість пацієнтів у палаті по кожному відділенню
10	**База даних «Страхування»**	1. Вивести повну інформацію про страхових агентів 2. Вивести повну інформацію про клієнтів. 3. Знайти клієнта з найбільшою кількістю страхових випадків 4. Знайти агента, що оформив найбільшу кількість договорів у поточному місяці. 5. Знайти середню кількість коштів, що сплачено як страховки, для кожного місяця поточного року.

№ з/п	Назва бази даних	Завдання для розробки запитів
11	**База даних «Науковий журнал»**	1. Вивести повну інформацію про редакційну колегію. 2. Вивести повну інформацію про авторів. 3. Знайти розділ з найбільшою кількістю публікацій. 4. Знайти автора, що опублікував найбільшу кількість статей за останні п’ять років. 5. Знайти автора, що подав роботи у найбільшу кількість рубрик журналу.
12	**База даних «Агентство нерухомості»**	1. Вивести повну інформацію про агентів. 2. Вивести повну інформацію про клієнтів. 3. Знайти клієнта з найбільшою кількістю об’єктів нерухомості 4. Знайти агента, що оформив найбільшу кількість договорів у поточному місяці. 5. Знайти середню кількість коштів, що отримано від оренди, для кожного місяця поточного року.
13	**База даних «Фото-Сервіс»**	1. Вивести повну інформацію про працівників. 2. Вивести повну інформацію про клієнтів. 3. Знайти клієнта з найбільшою кількістю замовлень. 4. Знайти працівника, що оформив найбільшу кількість замовлень. 5. Знайти середню кількість коштів, що отримано від замовників, для кожного місяця поточного року.
14	**База даних «Інтернет-Розсилка»**	1. Вивести повну інформацію про клієнтів. 2. Знайти клієнта з найбільшою кількістю підключень до розсилки. 3. Знайти найпопулярнішу рубрику. 4. Вивести рубрики, що мають кількість підписників, меншу за середню.
15	**База даних «Туристичне агентство»**	1. Вивести повну інформацію про працівників. 2. Вивести повну інформацію про клієнтів. 3. Знайти клієнта з найбільшою кількістю замовлень.

№ з/п	Назва бази даних	Завдання для розробки запитів
		4. Знайти працівника, що оформив найбільшу кількість замовлень. 5. Знайти середню кількість коштів, що отримано від замовників, для кожного місяця поточного року.
16	**База даних «Відділ збиту»**	1. Вивести повну інформацію про працівників. 2. Вивести повну інформацію про клієнтів. 3. Знайти клієнта з найбільшою кількістю замовлень. 4. Знайти працівника, що оформив найбільшу кількість замовлень. 5. Знайти середню кількість коштів, що отримано від замовників, для кожного місяця поточного року.
17	**База даних «Порушення правил дорожнього руху»**	1. Вивести повну інформацію про порушників. 2. Знайти порушника з найбільшою кількістю порушень. 3. Знайти усіх порушників, що мають кількість порушень більше середньої. 4. Знайти порушника з найбільшим штрафом. 5. Знайти середню кількість порушень для кожного місяця року.
18	**База даних «Прокат автомобілів»**	1. Вивести повну інформацію про клієнтів. 2. Знайти клієнта з найбільшою кількістю замовлень. 3. Знайти працівника, що оформив найбільшу кількість замовлень. 4. Знайти середню кількість коштів, що отримано від замовників, для кожного місяця поточного року. 5. Знайти автомобіль, що найчастіше був у прокату.
19	**База даних «Інтернет-магазин»**	1. Вивести повну інформацію про клієнтів. 2. Знайти клієнта з найбільшою кількістю замовлень. 3. Знайти розділ магазину, що має найбільшу кількість замовлень.

№ з/п	Назва бази даних	Завдання для розробки запитів
		4. Знайти середню кількість коштів, що отримано від замовників, для кожного місяця поточного року. 5. Знайти найбільш популярний товар у магазині.
20	**База даних «Музичний каталог»**	1. Вивести розділи каталогу. 2. Знайти розділ каталогу, що має найбільшу кількість музичних композицій. 3. Знайти композицію, що належить до найбільшої кількості жанрів. 4. Знайти виконавців, що мають кількість композицій, більшу за середню. 5. Знайти середню кількість виконавців у кожному розділі каталогу.

Методичне видання
(українською мовою)

Чопоров Сергій Вікторович
Чопорова Оксана Володимирівна
Мильцев Олександр Михайлович
Столярова Анастасія Валеріївна

БАЗИ ДАНИХ

Навчальний посібник
для здобувачів ступеня вищої освіти бакалавра
спеціальності «Інженерія програмного забезпечення»
освітньо-професійної програми «Програмна інженерія»

Рецензент *С. М. Гребенюк*
Відповідальний за випуск *С. В. Чопоров*
Коректор *О. В. Чопорова*