Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра комп'ютерних наук

Л.С. Глоба

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ

Методичні вказівки до практичних робіт
для здобувачів вищої освіти,
які навчаються за спеціальностями
галузі «Інформаційні технології»

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

Л.С. Глоба Організація баз даних та знань. Методичні вказівки до практичних робіт для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» [Електронне видання]. Кафедра комп’ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 58 с.

Дисципліна «Організація баз даних та знань» формує у здобувачів теоретичні та практичні знання щодо принципів організації великих обсягів даних, методів їх структуризації, побудови моделей даних, а також технологій зберігання, обробки та аналізу інформації у сучасних інформаційних системах.

ЗМІСТ

ТЕМА 1. ПОНЯТТЯ БАЗ ДАНИХ І СИСТЕМ КЕРУВАННЯ БАЗАМИ ДАНИХ

Практична робота 1. Встановлення та первинне налаштування системи керування базами даних

Мета роботи – ознайомитися з процесом встановлення системи керування базами даних, виконати її первинне налаштування та перевірити працездатність шляхом створення тестової бази даних.

Ключові положення

Система керування базами даних є програмним забезпеченням, що забезпечує створення, зберігання, обробку та захист даних. Вона виступає як проміжний рівень між користувачем або прикладною програмою і фізичним сховищем інформації. Використання СКБД дозволяє централізовано керувати даними, забезпечувати їх цілісність та ефективно організовувати доступ до них.

Процес встановлення СКБД є важливим етапом, від якого залежить подальша стабільність і продуктивність системи. Під час встановлення визначаються основні параметри, такі як місце зберігання даних, налаштування доступу, конфігурація мережевих служб та початкові облікові записи.

Сучасні СКБД, такі як PostgreSQL або MySQL, підтримують автоматизовані процедури встановлення, що спрощує початкову конфігурацію системи. Проте навіть у цьому випадку необхідно розуміти основні параметри налаштування та їх вплив на роботу системи.

Після встановлення СКБД виконується її первинне налаштування. Воно включає створення адміністративного облікового запису, налаштування прав доступу, визначення параметрів з’єднання та конфігурацію служб. Особливу увагу слід приділити безпеці, зокрема встановленню надійного пароля для адміністратора та обмеженню доступу до системи.

СКБД зазвичай працює у клієнт-серверному режимі. Це означає, що серверна частина відповідає за обробку запитів і зберігання даних, а клієнтська частина забезпечує взаємодію користувача з системою. Для підключення до СКБД використовуються спеціальні клієнтські програми або інтерфейси командного рядка.

Після встановлення і налаштування необхідно перевірити працездатність системи. Це виконується шляхом створення тестової бази даних, таблиць і виконання простих SQL-запитів. Така перевірка дозволяє переконатися, що система працює коректно і готова до використання.

Важливим аспектом є розуміння структури зберігання даних у СКБД. Дані організовані у вигляді баз даних, які містять таблиці, індекси та інші об’єкти. Кожна база даних є логічно відокремленою і може використовуватися для окремої інформаційної системи.

Таким чином, встановлення та первинне налаштування СКБД є базовими операціями, що забезпечують підготовку системи до подальшої роботи. Розуміння цього процесу є необхідною умовою для ефективного використання баз даних.

Практичне завдання

1. Обрати систему керування базами даних для встановлення (PostgreSQL або MySQL).
2. Завантажити інсталяційний пакет із офіційного джерела.
3. Виконати встановлення СКБД із використанням стандартних параметрів.
4. Налаштувати адміністративний обліковий запис.
5. Запустити сервер баз даних і перевірити його працездатність.
6. Підключитися до СКБД за допомогою клієнтської програми.
7. Створити тестову базу даних.
8. Створити таблицю з декількома атрибутами.
9. Додати кілька записів до таблиці.
10. Виконати запит на вибірку даних із таблиці.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно визначитися з вибором СКБД. Доцільно використовувати безкоштовні системи, такі як PostgreSQL або MySQL, які широко застосовуються у навчальних і практичних цілях. Необхідно перейти на офіційний сайт відповідної системи і завантажити інсталяційний пакет, що відповідає операційній системі.

На першому етапі виконується встановлення СКБД. Під час встановлення необхідно обрати каталог для розміщення програми і даних, а також задати пароль адміністратора. Рекомендується використовувати стандартні налаштування, якщо немає спеціальних вимог.

Після завершення встановлення необхідно запустити сервер баз даних. У більшості випадків сервер запускається автоматично як системна служба. Необхідно перевірити його стан за допомогою відповідних інструментів операційної системи або засобів СКБД.

На наступному етапі виконується підключення до СКБД. Для цього можна використовувати командний рядок або графічний інтерфейс. Під час підключення необхідно вказати ім’я користувача, пароль, адресу сервера та порт.

Після успішного підключення необхідно створити тестову базу даних. Це дозволить перевірити працездатність системи і виконати подальші операції. Далі у створеній базі даних потрібно створити таблицю з декількома атрибутами різних типів.

Після створення таблиці необхідно додати кілька записів. Це дозволить перевірити коректність операцій вставки даних. Далі потрібно виконати запит на вибірку і переконатися, що дані зберігаються і відображаються правильно.

У процесі виконання роботи необхідно звернути увагу на можливі помилки, зокрема неправильні параметри підключення, відсутність доступу до сервера або помилки у SQL-запитах. У разі виникнення помилок необхідно проаналізувати їх причини і усунути.

Після завершення роботи доцільно зупинити сервер баз даних або перевести його у потрібний режим роботи. Це є важливою складовою адміністрування системи.

Контрольні питання

1. Що таке система керування базами даних?
2. Які функції виконує СКБД?
3. Які етапи встановлення СКБД?

4. Що таке клієнт-серверна архітектура?
5. Які параметри використовуються для підключення до СКБД?
6. Як створюється база даних?
7. Що таке таблиця у базі даних?
8. Які основні операції можна виконувати з таблицями?
9. Як перевірити працездатність СКБД?
10. Які помилки можуть виникати при встановленні та налаштуванні?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис процесу встановлення СКБД;
– опис налаштування системи;
– приклади створення бази даних і таблиці;
– результати виконання SQL-запитів;
– аналіз отриманих результатів;
– висновки.

Практична робота 2. Ознайомлення з інтерфейсом системи керування базами даних

Мета роботи – ознайомитися з інтерфейсом системи керування базами даних, вивчити основні елементи середовища, навчитися виконувати базові операції з об’єктами бази даних та SQL-запитами.

Ключові положення

Інтерфейс системи керування базами даних є основним засобом взаємодії користувача із системою. Він забезпечує доступ до функцій створення, редагування та аналізу баз даних. Інтерфейс може бути реалізований у вигляді командного рядка або графічного середовища, що надає зручні інструменти для роботи з даними.

У сучасних СКБД, таких як PostgreSQL або MySQL, широко використовуються графічні інтерфейси, які дозволяють візуально працювати з об’єктами бази даних. Вони містять панелі навігації, редактори запитів, засоби перегляду структури таблиць та результати виконання запитів.

Основним елементом інтерфейсу є дерево об’єктів бази даних. Воно відображає структуру бази даних, включаючи бази даних, схеми, таблиці, представлення, індекси та інші об’єкти. Використання дерева дозволяє швидко знаходити необхідні елементи і виконувати над ними операції.

Редактор SQL-запитів є важливим компонентом інтерфейсу. Він дозволяє вводити, редагувати та виконувати запити мовою SQL. У редакторі можуть бути доступні засоби підсвічування синтаксису, автодоповнення та перевірки помилок, що спрощує роботу користувача.

Результати виконання запитів відображаються у вигляді таблиць, що дозволяє аналізувати отримані дані. Користувач може виконувати сортування, фільтрацію та інші операції безпосередньо у середовищі СКБД.

Інтерфейс також надає засоби для створення і редагування об’єктів бази даних. Наприклад, створення таблиці може виконуватися як за допомогою SQL-запиту, так і за допомогою візуального конструктора. Це дозволяє обрати найбільш зручний спосіб роботи.

Важливим елементом є керування підключеннями до бази даних. Інтерфейс дозволяє створювати, редагувати та зберігати параметри підключення, що спрощує доступ до різних баз даних.

Крім того, інтерфейс може містити засоби адміністрування, які дозволяють керувати користувачами, правами доступу та іншими параметрами системи. Це є важливим для забезпечення безпеки і стабільності роботи бази даних.

Таким чином, інтерфейс СКБД є комплексним інструментом, що забезпечує ефективну роботу з базами даних. Його використання дозволяє спростити виконання складних операцій і підвищити продуктивність роботи користувача.

Практичне завдання

1. Запустити систему керування базами даних.
2. Підключитися до сервера баз даних.
3. Ознайомитися з головним вікном інтерфейсу.
4. Вивчити структуру дерева об’єктів бази даних.
5. Переглянути список доступних баз даних.
6. Відкрити одну з баз даних і переглянути її об’єкти.
7. Ознайомитися з редактором SQL-запитів.
8. Виконати простий SQL-запит для вибірки даних.
9. Переглянути результати виконання запиту.
10. Ознайомитися з засобами створення таблиць.

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно запустити систему керування базами даних і встановити підключення до сервера. Для цього потрібно використати параметри, задані під час встановлення системи, зокрема ім’я користувача, пароль, адресу сервера та порт.

Після підключення необхідно ознайомитися з головним вікном інтерфейсу. Слід звернути увагу на основні елементи, такі як панель навігації, область редагування запитів і область відображення результатів. Важливо зрозуміти призначення кожного з цих елементів.

Далі необхідно дослідити дерево об’єктів бази даних. Для цього слід розгорнути список баз даних, обрати одну з них і переглянути її структуру. Особливу увагу потрібно приділити таблицям, оскільки вони є основними об’єктами зберігання даних.

На наступному етапі слід відкрити редактор SQL-запитів і виконати простий запит. Наприклад, можна виконати запит для вибірки всіх записів із таблиці. Це дозволить перевірити працездатність системи і ознайомитися з процесом виконання запитів.

Після виконання запиту необхідно проаналізувати отримані результати. Слід звернути увагу на структуру таблиці результатів, назви стовпців і значення даних. Це допоможе зрозуміти, як система відображає інформацію.

Також необхідно ознайомитися із засобами створення таблиць. Для цього можна використати як SQL-запити, так і візуальні інструменти. Це дозволить порівняти різні способи створення об’єктів бази даних.

У процесі виконання роботи доцільно експериментувати з інтерфейсом, відкривати різні об’єкти, змінювати параметри відображення та виконувати різні запити. Це сприятиме кращому розумінню можливостей системи.

Після завершення роботи необхідно коректно завершити сеанс роботи із СКБД, закривши підключення до сервера. Це дозволяє уникнути помилок і забезпечує стабільну роботу системи.

Контрольні питання

1. Що таке інтерфейс СКБД?
2. Які типи інтерфейсів існують?
3. Які основні елементи інтерфейсу СКБД?
4. Що таке дерево об’єктів бази даних?
5. Яке призначення редактора SQL-запитів?
6. Як виконати SQL-запит?
7. Як переглянути результати запиту?
8. Які засоби створення таблиць існують?
9. Як встановити підключення до бази даних?
10. Чому важливо коректно завершувати роботу із СКБД?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис інтерфейсу СКБД;
– опис основних елементів середовища;
– приклади виконання SQL-запитів;
– результати виконання завдань;
– висновки.

Практична робота 3. Аналіз предметної області та визначення інформаційних об’єктів і їх атрибутів

Мета роботи – навчитися виконувати аналіз предметної області, виділяти основні інформаційні об’єкти, визначати їх атрибути та встановлювати взаємозв’язки між ними.

Ключові положення

Аналіз предметної області є початковим етапом проєктування бази даних, який визначає якість майбутньої інформаційної системи. На цьому етапі формується уявлення про структуру даних, їх властивості та взаємозв’язки. Правильний аналіз дозволяє створити логічно узгоджену модель даних і уникнути проблем на подальших етапах розробки.

Предметна область визначає сферу діяльності, для якої створюється база даних. Це може бути навчальний процес, система обліку товарів, банківська система або інша галузь. У межах предметної області необхідно визначити основні об’єкти, які підлягають зберіганню у базі даних.

Інформаційний об’єкт є абстракцією реального об’єкта або явища, що має значення для предметної області. Наприклад, у системі обліку студентів такими об’єктами можуть бути студент, дисципліна, викладач. Кожен об’єкт характеризується набором атрибутів, які описують його властивості.

Атрибут є характеристикою інформаційного об’єкта. Наприклад, для об’єкта студент атрибутами можуть бути прізвище, ім’я, дата народження, номер групи. Важливо визначити повний і водночас мінімально необхідний набір атрибутів, щоб уникнути надлишковості даних.

У процесі аналізу необхідно враховувати типи атрибутів. Атрибути можуть бути простими або складеними, обов’язковими або необов’язковими, а також можуть мати множинні значення. Це впливає на подальше проєктування структури бази даних.

Важливим елементом є визначення ключових атрибутів, які дозволяють однозначно ідентифікувати об’єкти. Такий атрибут або їх комбінація використовується як основа для формування первинного ключа у таблиці.

Крім визначення об’єктів і атрибутів, необхідно встановити зв’язки між об’єктами. Зв’язки відображають взаємодію між об’єктами і визначають структуру даних. Наприклад, студент може бути пов’язаний із дисципліною через факт навчання.

Аналіз предметної області також включає визначення обмежень, що накладаються на дані. Це можуть бути обмеження на значення атрибутів, правила взаємодії між об’єктами або інші умови, що забезпечують коректність даних.

Важливим аспектом є формалізація отриманих результатів. Вона передбачає створення опису об’єктів, їх атрибутів і зв’язків у вигляді структурованої моделі. Це є основою для подальшого побудови концептуальної моделі даних.

Таким чином, аналіз предметної області є фундаментальним етапом, що визначає структуру майбутньої бази даних. Від якості виконання цього етапу залежить ефективність роботи всієї інформаційної системи.

Практичне завдання

1. Обрати предметну область для аналізу (наприклад, навчальний процес, бібліотека, інтернет-магазин).
2. Визначити основні інформаційні об’єкти предметної області.
3. Сформувати перелік атрибутів для кожного об’єкта.
4. Визначити ключові атрибути для кожного об’єкта.
5. Встановити зв’язки між об’єктами.
6. Визначити типи атрибутів (обов’язкові, необов’язкові, складені).
7. Описати обмеження, що накладаються на дані.
8. Сформувати структурований опис предметної області.
9. Перевірити повноту і узгодженість визначених об’єктів і атрибутів.
10. Підготувати результати аналізу для подальшого моделювання.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно чітко визначити предметну область. Рекомендується обрати таку область, яка є зрозумілою і має чітко визначені об’єкти. Це дозволить уникнути помилок і спростить процес аналізу.

На першому етапі необхідно виділити основні інформаційні об’єкти. Для цього слід проаналізувати предметну область і визначити, які сутності мають значення для системи. Важливо не перевантажувати модель зайвими об’єктами і зосередитися на основних.

На наступному етапі необхідно визначити атрибути для кожного об’єкта. Атрибути повинні описувати основні властивості об’єкта і бути достатніми для вирішення задач системи. При цьому слід уникати дублювання інформації.

Далі необхідно визначити ключові атрибути. Вони повинні забезпечувати унікальність кожного об’єкта. Якщо один атрибут не може виконувати цю функцію, слід використовувати комбінацію атрибутів або ввести штучний ідентифікатор.

Після цього необхідно встановити зв’язки між об’єктами. Для цього слід проаналізувати, як об’єкти взаємодіють між собою. Важливо визначити тип зв’язків і їх логіку.

Також необхідно визначити обмеження на дані. Це дозволить забезпечити коректність і узгодженість інформації. Обмеження можуть стосуватися значень атрибутів або взаємозв’язків між об’єктами.

На завершальному етапі необхідно оформити результати аналізу у вигляді структурованого опису. Це може бути текстовий опис або таблиці, що містять інформацію про об’єкти і їх атрибути.

У процесі виконання роботи необхідно перевірити логічність і повноту отриманої моделі. Важливо переконатися, що всі необхідні об’єкти визначені і між ними встановлені правильні зв’язки.

Контрольні питання

1. Що таке предметна область?
2. Що таке інформаційний об’єкт?
3. Що таке атрибут?
4. Які типи атрибутів існують?
5. Що таке ключовий атрибут?
6. Як визначаються зв’язки між об’єктами?
7. Які обмеження можуть накладатися на дані?
8. Чому важливий аналіз предметної області?
9. Які помилки можуть виникати при аналізі?
10. Яка роль цього етапу у проєктуванні бази даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис предметної області;
– перелік інформаційних об’єктів;

– перелік атрибутів для кожного об’єкта;
– визначення ключових атрибутів;
– опис зв’язків між об’єктами;
– обмеження на дані;
– висновки.

Практична робота 4. Побудова концептуальної моделі даних для заданої предметної області

Мета роботи – навчитися будувати концептуальну модель даних на основі результатів аналізу предметної області, визначати сутності, атрибути та зв’язки між ними.

Ключові положення

Концептуальна модель даних є узагальненим описом предметної області, що відображає її структуру без прив’язки до конкретної системи керування базами даних. Вона дозволяє формалізувати результати аналізу та підготувати основу для подальшого логічного проєктування.

Основою концептуальної моделі є поняття сутності. Сутність відповідає інформаційному об’єкту предметної області і описує клас однотипних об’єктів. Наприклад, студент, викладач або дисципліна можуть розглядатися як сутності. Кожна сутність характеризується набором атрибутів, які визначають її властивості.

Атрибути сутності визначають інформацію, що зберігається про об’єкт. Вони можуть бути простими або складеними, обов’язковими або необов’язковими. Важливо правильно визначити набір атрибутів, щоб забезпечити повноту і точність опису сутності.

Особливу роль відіграють ідентифікатори сутностей. Ідентифікатор є атрибутом або набором атрибутів, що дозволяє однозначно ідентифікувати кожен екземпляр сутності. Він є основою для подальшого визначення ключів у реляційній моделі.

Зв’язки між сутностями відображають взаємодію між об’єктами предметної області. Вони визначають, як сутності пов’язані між собою і які відношення між ними існують. Наприклад, студент може бути пов’язаний із дисципліною через факт навчання.

Зв’язки характеризуються кардинальністю, яка визначає кількість екземплярів однієї сутності, пов’язаних з екземплярами іншої. Основними типами є один-до-одного, один-до-багатьох і багато-до-багатьох. Правильне визначення кардинальності є важливим для побудови коректної моделі.

Концептуальна модель часто представляється у вигляді ER-моделі. ER-модель використовує графічне подання, у якому сутності, атрибути та зв’язки зображаються у вигляді діаграми. Це дозволяє наочно відобразити структуру даних і спростити її аналіз.

Побудова концептуальної моделі включає кілька етапів. Спочатку визначаються сутності на основі аналізу предметної області. Потім для кожної сутності визначаються атрибути і ідентифікатори. Після цього встановлюються зв’язки між сутностями і визначається їх кардинальність.

Важливим аспектом є перевірка узгодженості моделі. Необхідно переконатися, що всі сутності і зв’язки логічно пов’язані, а атрибути коректно описують відповідні об’єкти. Це дозволяє уникнути помилок на подальших етапах проєктування.

Таким чином, концептуальна модель є важливим етапом розробки бази даних, що забезпечує формалізацію предметної області і створює основу для подальшого проєктування.

Практичне завдання

1. Використати результати аналізу предметної області з попередньої роботи.
2. Визначити перелік сутностей предметної області.
3. Сформувати перелік атрибутів для кожної сутності.
4. Визначити ідентифікатори сутностей.
5. Встановити зв’язки між сутностями.
6. Визначити кардинальність кожного зв’язку.
7. Побудувати концептуальну модель у вигляді ER-діаграми.
8. Перевірити правильність і повноту моделі.
9. Внести необхідні корективи до моделі.
10. Підготувати опис отриманої моделі.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно використати результати аналізу предметної області, отримані у попередній роботі. Якщо аналіз був виконаний некоректно або неповно, необхідно внести уточнення перед побудовою моделі.

На першому етапі необхідно визначити сутності. Для цього слід виділити основні об’єкти предметної області, які мають бути представлені у базі даних. Сутності повинні бути чітко визначені і не дублювати одна одну.

Далі необхідно визначити атрибути кожної сутності. Атрибути повинні повністю описувати сутність, але не містити надлишкової інформації. Особливу увагу слід приділити вибору ідентифікатора.

На наступному етапі встановлюються зв’язки між сутностями. Необхідно визначити, які сутності взаємодіють між собою і яким чином. Для кожного зв’язку потрібно визначити його кардинальність.

Після цього виконується побудова ER-діаграми. Для цього можна використати спеціалізовані інструменти або виконати побудову вручну. Важливо дотримуватися стандартних позначень і забезпечити зрозумілість діаграми.

Після побудови моделі необхідно перевірити її узгодженість. Слід переконатися, що всі сутності мають ідентифікатори, всі зв’язки визначені правильно, а атрибути відповідають сутностям.

У разі виявлення помилок або неточностей необхідно внести зміни до моделі. Це є нормальним етапом процесу проєктування.

На завершальному етапі необхідно підготувати текстовий опис моделі, що містить перелік сутностей, їх атрибутів і зв’язків.

Контрольні питання

1. Що таке концептуальна модель даних?
2. Що таке сутність?
3. Що таке атрибут сутності?

4. Що таке ідентифікатор?
5. Які типи зв’язків існують?
6. Що таке кардинальність зв’язків?
7. Що таке ER-модель?
8. Які етапи побудови концептуальної моделі?
9. Чому важлива перевірка моделі?
10. Яка роль концептуальної моделі у проєктуванні бази даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис предметної області;
– перелік сутностей;
– перелік атрибутів;
– опис зв’язків і їх кардинальності;
– ER-діаграма;
– висновки.

ТЕМА 2. ПРОЄКТУВАННЯ БАЗ ДАНИХ

Практична робота 5. Побудова ER-діаграми предметної області

Мета роботи – навчитися будувати ER-діаграму предметної області, відображати на ній сутності, атрибути, зв'язки та кардинальність, а також формалізувати структуру даних для подальшого проєктування бази даних.

Ключові положення

ER-діаграма є одним із основних засобів концептуального моделювання даних. Вона використовується для графічного подання структури предметної області та дозволяє наочно відобразити сутності, їх атрибути і зв'язки між ними. Використання ER-діаграм є важливим етапом проєктування бази даних, оскільки дає змогу ще до реалізації системи виявити помилки, уточнити логіку предметної області та узгодити структуру даних.

Основу ER-діаграми становлять сутності. Сутність є узагальненим представленням реального об'єкта або явища, що має значення для інформаційної системи. Наприклад, у навчальній системі такими сутностями можуть бути студент, викладач, дисципліна, група. Сутність описує не окремий екземпляр, а клас однотипних об'єктів, для яких характерний спільний набір властивостей.

Для кожної сутності визначаються атрибути. Атрибут є властивістю сутності, що описує її характеристики. Наприклад, сутність студент може містити атрибути прізвище, ім'я, дата народження, номер залікової книжки. Атрибути дають змогу конкретизувати інформацію, яка буде зберігатися у майбутній базі даних. У процесі побудови ER-діаграми важливо визначити лише ті атрибути, які дійсно мають значення для розв'язання задач предметної області.

Серед атрибутів особливе значення мають ідентифікаційні атрибути. Вони дозволяють однозначно відрізнити один екземпляр сутності від іншого. Такі атрибути у подальшому стають основою первинних ключів у реляційній схемі. Якщо природного ідентифікатора немає, допускається використання штучного коду або номера.

ER-діаграма також відображає зв'язки між сутностями. Зв'язок показує, яким чином об'єкти предметної області взаємодіють між собою. Наприклад, між сутностями студент і дисципліна може існувати зв'язок вивчає, а між викладачем і дисципліною – викладає. Саме зв'язки формують логіку майбутньої бази даних і дають змогу зрозуміти, як окремі інформаційні об'єкти пов'язані між собою.

Для кожного зв'язку визначається кардинальність. Вона показує, скільки екземплярів однієї сутності може бути пов'язано з одним екземпляром іншої сутності. Основними варіантами є зв'язки один-до-одного, один-до-багатьох та багато-до-багатьох. Наприклад, один викладач може викладати багато дисциплін, а одна дисципліна може вивчатися багатьма студентами. Правильне визначення кардинальності є важливим, оскільки воно безпосередньо впливає на подальший перехід до реляційної схеми.

У ER-моделюванні важливо розрізняти прості та складені атрибути, а також обов'язкові й необов'язкові атрибути. Простий атрибут не ділиться на складові частини, тоді як складений може бути поділений, наприклад адреса на місто, вулицю, номер будинку. Обов'язкові атрибути повинні мати значення для кожного екземпляра сутності, тоді як

необов’язкові можуть залишатися невизначеними. Такі особливості важливо враховувати ще на етапі побудови ER-діаграми.

ER-діаграма є не просто рисунком, а формалізованим поданням моделі даних. Вона повинна бути логічно узгодженою, зрозумілою та повною. Це означає, що всі основні сутності предметної області повинні бути відображені, атрибути – відповідати сутностям, а зв’язки – коректно описувати взаємодію між об’єктами. Надмірна кількість сутностей або дублювання атрибутів ускладнюють модель і свідчать про недостатньо якісний аналіз предметної області.

Побудова ER-діаграми починається з аналізу результатів попередніх етапів проєктування. Насамперед необхідно узагальнити перелік інформаційних об’єктів, їх атрибутів і зв’язків, визначених під час аналізу предметної області. Далі виконується графічне представлення цих елементів у вигляді діаграми із використанням стандартних позначень.

Практичне значення ER-діаграм полягає у тому, що вони є основою для подальшого логічного проєктування бази даних. На основі ER-діаграми формуються таблиці, визначаються первинні та зовнішні ключі, а також правила реалізації зв’язків у реляційній моделі. Тому якісно побудована ER-діаграма значно спрощує наступні етапи проєктування.

Ще однією важливою перевагою ER-діаграм є можливість використання їх як засобу комунікації між розробниками, замовниками та користувачами системи. На відміну від суто технічних описів, графічна модель є більш наочною і зрозумілою. Це дає змогу обговорювати структуру майбутньої бази даних ще до початку програмної реалізації і вчасно вносити зміни.

Таким чином, ER-діаграма є важливим інструментом концептуального проєктування баз даних, який забезпечує наочне та логічно узгоджене подання предметної області. Її побудова дозволяє систематизувати результати аналізу, виявити помилки у структурі даних і створити основу для переходу до реляційної схеми.

Практичне завдання

1. Використати результати аналізу предметної області та концептуальної моделі, отримані у попередніх роботах.
2. Визначити перелік сутностей, що повинні бути відображені на ER-діаграмі.
3. Визначити атрибути для кожної сутності.
4. Виділити ідентифікаційні атрибути.
5. Встановити зв’язки між сутностями.
6. Визначити кардинальність для кожного зв’язку.
7. Побудувати ER-діаграму предметної області з використанням стандартних позначень.
8. Перевірити правильність розташування сутностей, атрибутів і зв’язків.
9. Проаналізувати повноту та логічну узгодженість діаграми.
10. Підготувати діаграму до використання на етапі логічного проєктування.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно використати результати попереднього аналізу предметної області. Якщо під час попередніх робіт були визначені сутності, атрибути

і зв’язки, їх слід переглянути, уточнити та узгодити перед побудовою ER-діаграми. У разі виявлення суперечностей або надлишкових елементів їх потрібно усунути ще до початку графічного моделювання.

На першому етапі необхідно визначити перелік сутностей, які будуть включені до ER-діаграми. Слід виділити лише ті сутності, які мають самостійне значення для предметної області і реально будуть використовуватися у майбутній базі даних. Не потрібно включати випадкові або другорядні об’єкти, якщо вони не впливають на структуру системи.

Після визначення сутностей потрібно сформувати перелік їх атрибутів. Для кожної сутності слід записати основні характеристики, які дозволяють повно її описати. При цьому слід уникати дублювання атрибутів у різних сутностях, якщо це не обумовлено логікою предметної області. Особливу увагу потрібно приділити ідентифікаційним атрибутам, оскільки вони є основою для однозначного визначення екземплярів сутностей.

На наступному етапі необхідно встановити зв’язки між сутностями. Для цього слід проаналізувати взаємодію між об’єктами предметної області та визначити, які відношення між ними є суттєвими. Назва зв’язку повинна відображати його зміст і бути зрозумілою. Для кожного зв’язку необхідно визначити його кардинальність, тобто кількість екземплярів однієї сутності, що можуть бути пов’язані з екземплярами іншої.

Під час побудови діаграми доцільно використовувати стандартні умовні позначення. Сутності зазвичай подаються у вигляді прямокутників, атрибути – у вигляді овалів або у вигляді текстового переліку всередині блоку сутності, а зв’язки – у вигляді ліній або окремих графічних елементів залежно від обраної нотації. Важливо, щоб усі позначення були послідовними в межах однієї діаграми.

Після побудови ER-діаграми необхідно перевірити її логічну узгодженість. Слід переконатися, що всі сутності мають атрибути, кожна сутність має ідентифікаційний атрибут, усі зв’язки є обґрунтованими, а їх кардинальність визначена правильно. Також потрібно оцінити, чи не містить діаграма зайвих елементів або дублювання інформації.

На завершальному етапі необхідно оформити ER-діаграму у придатному для аналізу вигляді. Якщо діаграма побудована вручну, вона повинна бути чіткою і розбірливою. Якщо використовується програмний засіб моделювання, потрібно перевірити коректність позначень і зручність сприйняття діаграми. Отриманий результат слід підготувати для подальшого використання при переході до логічної моделі бази даних.

Контрольні питання

1. Що таке ER-діаграма?
2. Яке призначення ER-діаграми у проєктуванні бази даних?
3. Що таке сутність у ER-моделі?
4. Що таке атрибут сутності?
5. Які атрибути називаються ідентифікаційними?
6. Що таке зв’язок між сутностями?
7. Які типи кардинальності зв’язків існують?
8. Чому важливо правильно визначати кардинальність зв’язків?
9. Які помилки можуть виникати при побудові ER-діаграми?
10. Яка роль ER-діаграми при переході до реляційної схеми?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– короткий опис предметної області;
– перелік сутностей;
– перелік атрибутів сутностей;
– опис зв’язків між сутностями;
– визначення кардинальності зв’язків;
– побудована ER-діаграма;
– аналіз отриманого результату;
– висновки.

Практична робота 6. Визначення сутностей, атрибутів та зв’язків у концептуальній моделі бази даних

Мета роботи – закріпити навички визначення сутностей, їх атрибутів та зв’язків у межах концептуальної моделі, навчитися уточнювати структуру даних і забезпечувати її логічну узгодженість.

Ключові положення

Концептуальна модель бази даних є узагальненим поданням предметної області, що відображає її основні інформаційні об’єкти та взаємозв’язки між ними. Вона формується на основі аналізу предметної області та є основою для подальшого переходу до логічного і фізичного проєктування. Важливим етапом побудови такої моделі є уточнення складу сутностей, їх атрибутів та зв’язків.

Сутність у концептуальній моделі відповідає класу об’єктів предметної області. Вона описує множину однотипних об’єктів, які мають спільні властивості. Важливо, щоб кожна сутність була логічно самостійною і не дублювала інші сутності. Надмірна кількість сутностей або їх неправильне визначення призводить до ускладнення структури бази даних.

Атрибути визначають властивості сутності і повинні повністю описувати її стан. При цьому необхідно дотримуватися принципу достатності: атрибутів має бути достатньо для опису об’єкта, але вони не повинні містити надлишкової інформації. Надлишковість ускладнює підтримку даних і може призводити до помилок.

Особливе значення мають ключові атрибути, які дозволяють однозначно ідентифікувати екземпляри сутності. Вибір ключового атрибута або їх комбінації є важливим завданням, оскільки він впливає на подальшу реалізацію бази даних. У разі відсутності природного ключа може бути введений штучний ідентифікатор.

Атрибути можуть мати різну структуру. Вони можуть бути простими або складеними, однозначними або багатозначними. Складені атрибути можуть бути поділені на підатрибути, що дозволяє більш детально описати об’єкт. Багатозначні атрибути можуть містити кілька значень, що потребує особливого підходу при проєктуванні.

Зв’язки між сутностями визначають взаємодію між об’єктами предметної області. Вони відображають логіку системи і є важливим елементом концептуальної моделі. Кожен

зв’язок повинен мати чітке змістове обґрунтування і відповідати реальним відношенням між об’єктами.

Тип зв’язку визначається його кардинальністю. Основними варіантами є один-до-одного, один-до-багатьох і багато-до-багатьох. Вибір типу зв’язку впливає на подальшу реалізацію у реляційній моделі. Наприклад, зв’язок багато-до-багатьох потребує введення додаткової сутності.

Важливим аспектом є узгодженість моделі. Сутності, атрибути і зв’язки повинні бути логічно пов’язані і не суперечити один одному. У процесі уточнення моделі можуть виявлятися помилки або неточності, які необхідно виправляти до переходу на наступні етапи.

Процес визначення сутностей, атрибутів і зв’язків є ітераційним. Це означає, що модель може уточнюватися і змінюватися у процесі її побудови. Такий підхід дозволяє поступово вдосконалювати структуру даних і досягти її оптимального вигляду.

Таким чином, визначення сутностей, атрибутів і зв’язків є ключовим етапом формування концептуальної моделі бази даних. Від правильності виконання цього етапу залежить якість подальшого проєктування і ефективність роботи інформаційної системи.

Практичне завдання

1. Використати результати попередніх робіт з аналізу предметної області.
2. Уточнити перелік сутностей предметної області.
3. Перевірити доцільність кожної сутності.
4. Визначити атрибути для кожної сутності.
5. Виділити ключові атрибути.
6. Визначити типи атрибутів (простий, складений, багатозначний).
7. Встановити зв’язки між сутностями.
8. Визначити кардинальність зв’язків.
9. Перевірити узгодженість моделі.
10. Підготувати уточнену концептуальну модель.

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно використати результати попередніх етапів проєктування. Особливу увагу слід приділити переліку сутностей, визначених раніше. Якщо деякі сутності дублюють одна одну або не мають самостійного значення, їх необхідно об’єднати або виключити.

На першому етапі слід уточнити склад сутностей. Для цього необхідно проаналізувати їх роль у предметній області і визначити, чи є вони необхідними для опису системи. Сутності повинні відповідати реальним об’єктам і бути логічно обґрунтованими.

Далі необхідно визначити атрибути кожної сутності. При цьому слід перевірити, чи не містять атрибути зайвої або дубльованої інформації. Якщо деякі атрибути можуть бути винесені в окрему сутність, це слід зробити для спрощення моделі.

На наступному етапі необхідно визначити ключові атрибути. Вони повинні забезпечувати унікальність кожного екземпляра сутності. Якщо природного ключа немає, доцільно використовувати штучний ідентифікатор.

Після цього слід встановити зв’язки між сутностями. Необхідно визначити, які сутності взаємодіють між собою і як саме. Для кожного зв’язку потрібно визначити його кардинальність і перевірити її відповідність логіці предметної області.

Особливу увагу слід приділити узгодженості моделі. Слід переконатися, що всі елементи моделі взаємопов’язані і не суперечать один одному. У разі виявлення помилок необхідно внести зміни і повторно перевірити модель.

На завершальному етапі необхідно оформити уточнену модель у вигляді структурованого опису або діаграми. Отриманий результат слід використовувати як основу для подальшого логічного проєктування бази даних.

Контрольні питання

1. Що таке сутність у концептуальній моделі?
2. Що таке атрибут сутності?
3. Які атрибути називаються ключовими?
4. Які типи атрибутів існують?
5. Що таке зв’язок між сутностями?
6. Які типи зв’язків існують?
7. Що таке кардинальність зв’язку?
8. Чому важливо перевіряти узгодженість моделі?
9. Які помилки можуть виникати при визначенні сутностей?
10. Яка роль цього етапу у проєктуванні бази даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– уточнений перелік сутностей;
– перелік атрибутів;
– визначення ключових атрибутів;
– опис зв’язків і їх кардинальності;
– аналіз узгодженості моделі;
– висновки.

Практична робота 7. Перетворення ER-моделі у реляційну схему бази даних

Мета роботи – навчитися виконувати перетворення концептуальної ER-моделі у реляційну схему, визначати таблиці, атрибути, первинні та зовнішні ключі, а також коректно реалізовувати зв’язки між сутностями.

Ключові положення

Перетворення ER-моделі у реляційну схему є ключовим етапом логічного проєктування бази даних. На цьому етапі абстрактна концептуальна модель, що описує

предметну область у вигляді сутностей і зв’язків, трансформується у формалізовану структуру таблиць, яка може бути реалізована у системі керування базами даних.

Основою цього процесу є правила відображення елементів ER-моделі у реляційні структури. Кожна сутність перетворюється у таблицю, а її атрибути стають стовпцями цієї таблиці. Екземпляри сутності відповідають рядкам таблиці. Такий підхід забезпечує прямий зв’язок між концептуальною моделлю і її реалізацією.

Важливим етапом є визначення первинних ключів. Первинний ключ забезпечує унікальність кожного запису у таблиці і дозволяє однозначно ідентифікувати об’єкти. У більшості випадків він формується на основі ідентифікатора сутності, визначеного на попередньому етапі. Якщо природний ключ є складним або нестабільним, доцільно використовувати штучний ідентифікатор.

Зв’язки між сутностями реалізуються за допомогою зовнішніх ключів. Для зв’язку один-до-багатьох зовнішній ключ додається у таблицю, що відповідає стороні «багатьох». Це дозволяє встановити зв’язок між записами різних таблиць і забезпечити цілісність даних.

Зв’язок один-до-одного може бути реалізований різними способами. Найчастіше використовується додавання зовнішнього ключа до однієї з таблиць або об’єднання двох таблиць в одну, якщо це не порушує логіку предметної області.

Зв’язок багато-до-багатьох не може бути безпосередньо представлений у реляційній моделі. Для його реалізації створюється окрема таблиця, яка містить зовнішні ключі обох пов’язаних таблиць. Така таблиця може також містити додаткові атрибути, що описують зв’язок.

Особливу увагу слід приділити слабким сутностям. Вони перетворюються у таблиці, але їх первинний ключ формується з урахуванням зовнішнього ключа, що посилається на сильну сутність. Це дозволяє відобразити залежність слабкої сутності.

Під час перетворення необхідно враховувати обмеження цілісності. Вони забезпечують коректність даних і включають обмеження унікальності, обмеження на допустимі значення атрибутів і правила посилальної цілісності. Правильне визначення обмежень є важливим для забезпечення стабільної роботи бази даних.

Важливим аспектом є перевірка отриманої реляційної схеми. Необхідно переконатися, що всі сутності представлені у вигляді таблиць, усі зв’язки реалізовані, а ключі визначені коректно. Також слід перевірити відсутність надлишковості та логічних суперечностей.

Таким чином, перетворення ER-моделі у реляційну схему є процесом формалізації структури даних, що дозволяє перейти від абстрактного опису предметної області до конкретної реалізації у базі даних.

Практичне завдання

1. Використати ER-діаграму, побудовану у попередній роботі.
2. Визначити перелік сутностей, що підлягають перетворенню у таблиці.
3. Перетворити кожну сутність у таблицю.
4. Визначити атрибути таблиць.
5. Визначити первинні ключі для кожної таблиці.
6. Реалізувати зв’язки один-до-багатьох за допомогою зовнішніх ключів.
7. Реалізувати зв’язки багато-до-багатьох через окремі таблиці.
8. Визначити обмеження цілісності.
9. Перевірити узгодженість отриманої схеми.

10. Підготувати опис реляційної схеми бази даних.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно використати ER-діаграму, побудовану у попередній роботі. Важливо переконатися, що вона є коректною і не містить помилок, оскільки всі подальші перетворення базуються саме на цій моделі.

На першому етапі необхідно перетворити сутності у таблиці. Для кожної сутності створюється окрема таблиця, що містить усі її атрибути. Назви таблиць повинні бути зрозумілими і відповідати сутностям предметної області.

Далі необхідно визначити первинні ключі. Якщо у сутності вже визначено ідентифікатор, його слід використати як первинний ключ. У разі відсутності такого атрибута необхідно ввести штучний ідентифікатор.

На наступному етапі реалізуються зв’язки між таблицями. Для зв’язків один-до-багатьох необхідно додати зовнішній ключ до таблиці, що відповідає стороні «багатьох». Важливо переконатися, що зовнішній ключ посилається на правильний первинний ключ.

Для зв’язків багато-до-багатьох необхідно створити окрему таблицю. Вона повинна містити зовнішні ключі обох пов’язаних таблиць. За необхідності до цієї таблиці можна додати додаткові атрибути.

Після цього необхідно визначити обмеження цілісності. Слід встановити обмеження унікальності, перевірки значень та посилальної цілісності. Це дозволить забезпечити коректність даних у базі.

На завершальному етапі необхідно перевірити отриману схему. Слід переконатися, що всі таблиці пов’язані між собою, ключі визначені правильно, а структура є логічно узгодженою.

У разі виявлення помилок необхідно внести зміни і повторно перевірити схему. Отриманий результат слід оформити у вигляді таблиць або текстового опису.

Контрольні питання

1. Що таке реляційна схема бази даних?
2. Як сутність перетворюється у таблицю?
3. Що таке первинний ключ?
4. Що таке зовнішній ключ?
5. Як реалізується зв’язок один-до-багатьох?
6. Як реалізується зв’язок багато-до-багатьох?
7. Що таке обмеження цілісності?
8. Чому важливо перевіряти реляційну схему?
9. Які помилки можуть виникати при перетворенні?
10. Яка роль цього етапу у проєктуванні бази даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– ER-діаграма;

– перелік таблиць;
– опис атрибутів;
– визначення первинних і зовнішніх ключів;
– обмеження цілісності;
– аналіз отриманої схеми;
– висновки.

Практична робота 8. Визначення первинних та зовнішніх ключів у реляційних таблицях

Мета роботи – навчитися визначати первинні та зовнішні ключі у реляційних таблицях, забезпечувати зв’язки між таблицями та контролювати цілісність даних.

Ключові положення

Ключі є основними елементами реляційної моделі даних, що забезпечують ідентифікацію записів і встановлення зв’язків між таблицями. Вони відіграють ключову роль у забезпеченні цілісності даних і правильній організації структури бази даних.

Первинний ключ є атрибутом або набором атрибутів, що однозначно визначає кожен запис у таблиці. Його основною властивістю є унікальність значень і відсутність невизначених значень. Первинний ключ гарантує, що кожен запис можна однозначно ідентифікувати, що є необхідною умовою для коректної роботи бази даних.

Вибір первинного ключа є важливим етапом проєктування. Ключ повинен бути стабільним, тобто його значення не повинні змінюватися з часом. Крім того, він має бути мінімальним і не містити зайвих атрибутів. У випадках, коли природний ключ є складним або незручним, використовується штучний ідентифікатор.

У деяких таблицях може використовуватися складений первинний ключ, який складається з кількох атрибутів. Такий підхід застосовується у випадках, коли жоден окремий атрибут не забезпечує унікальність записів.

Зовнішній ключ є атрибутом або набором атрибутів, що посилається на первинний ключ іншої таблиці. Він використовується для встановлення зв’язків між таблицями і забезпечення посилальної цілісності. Завдяки зовнішнім ключам забезпечується узгодженість даних у різних таблицях.

Зовнішній ключ визначає залежність між таблицями. Наприклад, таблиця, що містить інформацію про оцінки студентів, може містити зовнішній ключ, який посилається на таблицю студентів. Це дозволяє встановити зв’язок між оцінкою і конкретним студентом.

При визначенні зовнішніх ключів необхідно враховувати обмеження посилальної цілісності. Це означає, що значення зовнішнього ключа повинні відповідати існуючим значенням первинного ключа або бути відсутніми, якщо це допускається.

Важливим аспектом є поведінка системи при зміні або видаленні записів. У таких випадках можуть застосовуватися різні стратегії, зокрема заборона операції, каскадне оновлення або встановлення значення у невизначений стан. Вибір стратегії залежить від логіки предметної області.

Ключі також впливають на продуктивність бази даних. Первинні ключі зазвичай індексуються, що забезпечує швидкий доступ до записів. Зовнішні ключі також можуть використовуватися для оптимізації запитів, що виконують з'єднання таблиць.

Правильне визначення ключів дозволяє уникнути дублювання даних і забезпечити логічну структуру бази даних. Помилки у визначенні ключів можуть призвести до втрати цілісності даних і ускладнення обробки інформації.

Таким чином, первинні та зовнішні ключі є фундаментальними елементами реляційної моделі, що забезпечують ідентифікацію даних і зв'язки між таблицями. Їх правильне використання є необхідною умовою для побудови ефективної бази даних.

Практичне завдання

1. Використати реляційну схему, отриману у попередній роботі.
2. Визначити первинні ключі для кожної таблиці.
3. Перевірити унікальність значень первинних ключів.
4. Визначити зовнішні ключі для встановлення зв'язків між таблицями.
5. Встановити відповідність зовнішніх ключів первинним ключам.
6. Визначити типи зв'язків між таблицями.
7. Задати обмеження посилальної цілісності.
8. Перевірити коректність зв'язків між таблицями.
9. Проаналізувати можливі помилки при порушенні цілісності.
10. Підготувати опис структури ключів.

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно використати реляційну схему бази даних, побудовану у попередній роботі. Важливо переконатися, що структура таблиць є коректною і містить усі необхідні атрибути.

На першому етапі необхідно визначити первинні ключі. Для цього слід проаналізувати кожну таблицю і визначити атрибут або набір атрибутів, що однозначно ідентифікують записи. Якщо такі атрибути відсутні, необхідно ввести штучний ідентифікатор.

Після визначення первинних ключів необхідно перевірити їх унікальність. Для цього можна виконати відповідні SQL-запити або проаналізувати дані у таблицях. У разі виявлення дублювання необхідно внести зміни до структури таблиці.

На наступному етапі необхідно визначити зовнішні ключі. Для цього слід встановити, які таблиці пов'язані між собою, і визначити атрибути, що використовуються для зв'язку. Зовнішні ключі повинні посилатися на первинні ключі інших таблиць.

Далі необхідно задати обмеження посилальної цілісності. Це дозволить забезпечити коректність зв'язків між таблицями і запобігти появі некоректних даних.

Після цього необхідно перевірити роботу встановлених ключів. Для цього можна виконати операції вставки, оновлення і видалення даних і проаналізувати реакцію системи. Це дозволить переконатися у правильності налаштування обмежень.

У процесі виконання роботи необхідно звернути увагу на типи зв'язків між таблицями і відповідність зовнішніх ключів цим зв'язкам. У разі виявлення помилок необхідно внести зміни до структури бази даних.

На завершальному етапі необхідно оформити результати роботи у вигляді опису, що містить перелік первинних і зовнішніх ключів та пояснення їх призначення.

Контрольні питання

1. Що таке первинний ключ?
2. Які властивості має первинний ключ?
3. Що таке зовнішній ключ?
4. Яке призначення зовнішнього ключа?
5. Що таке посилальна цілісність?
6. Які типи зв’язків існують між таблицями?
7. Що таке складений ключ?
8. Які помилки можуть виникати при визначенні ключів?
9. Як перевірити унікальність ключа?
10. Чому важливо правильно визначати ключі?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– перелік таблиць;
– визначення первинних ключів;
– визначення зовнішніх ключів;
– опис зв’язків між таблицями;
– перевірка цілісності даних;
– висновки.

Практична робота 9. Нормалізація таблиць бази даних до третьої нормальної форми

Мета роботи – навчитися виконувати нормалізацію реляційних таблиць, визначати функціональні залежності та приводити структуру бази даних до третьої нормальної форми для усунення надлишковості і забезпечення цілісності даних.

Ключові положення

Нормалізація є процесом впорядкування структури бази даних, спрямованим на усунення надлишковості та аномалій при роботі з даними. Вона базується на аналізі функціональних залежностей між атрибутами та передбачає послідовне приведення таблиць до нормальних форм.

Основною метою нормалізації є забезпечення логічної узгодженості даних і спрощення їх обробки. Ненормалізовані таблиці можуть містити дубльовану інформацію, що призводить до проблем при додаванні, зміні або видаленні даних. Такі проблеми називаються аномаліями оновлення.

Функціональна залежність визначає, як значення одного атрибута залежить від значення іншого. Якщо значення атрибута A однозначно визначає значення атрибута B, говорять, що B функціонально залежить від A. Аналіз таких залежностей є основою процесу нормалізації.

Перша нормальна форма передбачає, що всі атрибути таблиці є атомарними, тобто не можуть бути розділені на складові частини. Це означає, що кожне поле містить лише одне значення, а не список або множину значень. Приведення до першої нормальної форми є базовою вимогою для реляційних таблиць.

Друга нормальна форма застосовується до таблиць, що мають складений первинний ключ. Вона вимагає, щоб кожен неключовий атрибут повністю залежав від усього ключа, а не лише від його частини. Якщо існують часткові залежності, таблиця розбивається на кілька таблиць.

Третя нормальна форма передбачає відсутність транзитивних залежностей між атрибутами. Це означає, що неключові атрибути не повинні залежати один від одного. Якщо така залежність існує, вона усувається шляхом винесення залежних атрибутів у окрему таблицю.

Процес нормалізації виконується поетапно. Спочатку таблиця приводиться до першої нормальної форми, потім до другої і, нарешті, до третьої. Кожен етап спрямований на усунення певного типу залежностей і покращення структури даних.

Важливою перевагою нормалізації є зменшення надлишковості даних. Це дозволяє зменшити обсяг зберігання і спростити оновлення інформації. Крім того, нормалізація підвищує цілісність даних, оскільки зменшує ймовірність появи суперечностей.

Водночас надмірна нормалізація може призвести до ускладнення структури бази даних і зниження продуктивності запитів. Тому у практиці часто використовується компроміс між нормалізацією і ефективністю роботи системи.

Таким чином, нормалізація є важливим етапом проєктування бази даних, що дозволяє створити логічно узгоджену і ефективну структуру даних.

Практичне завдання

1. Використати таблиці, отримані у попередніх роботах.
2. Проаналізувати структуру таблиць на наявність надлишкових даних.
3. Визначити функціональні залежності між атрибутами.
4. Привести таблиці до першої нормальної форми.
5. Перевірити наявність часткових залежностей і привести таблиці до другої нормальної форми.
6. Виявити транзитивні залежності і привести таблиці до третьої нормальної форми.
7. Розбити таблиці на кілька взаємопов’язаних таблиць за необхідності.
8. Визначити первинні і зовнішні ключі для нових таблиць.
9. Перевірити цілісність і узгодженість отриманої структури.
10. Порівняти структуру до і після нормалізації.

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно обрати таблицю або набір таблиць, які будуть піддаватися нормалізації. Доцільно використовувати приклади з попередніх робіт, де вже визначені атрибути і зв'язки.

На першому етапі необхідно перевірити відповідність таблиці першій нормальній формі. Слід переконатися, що всі атрибути є атомарними і не містять множинних значень. У разі виявлення порушень необхідно розділити такі атрибути на окремі поля або таблиці.

Далі необхідно визначити первинний ключ і проаналізувати залежності між атрибутами. Якщо таблиця має складений ключ, слід перевірити, чи всі неключові атрибути залежать від усього ключа. У разі часткових залежностей необхідно виконати декомпозицію таблиці.

На наступному етапі необхідно виявити транзитивні залежності. Для цього слід проаналізувати, чи не залежать неключові атрибути один від одного. Якщо такі залежності існують, відповідні атрибути необхідно винести у окремі таблиці.

Після виконання декомпозиції необхідно визначити зв'язки між новими таблицями. Для цього слід встановити зовнішні ключі, що забезпечують цілісність даних.

У процесі виконання роботи необхідно перевірити, чи не втрачено інформацію при розбитті таблиць. Для цього слід переконатися, що всі дані можуть бути відновлені шляхом виконання операцій з'єднання.

На завершальному етапі необхідно порівняти структуру таблиць до і після нормалізації. Слід оцінити, як змінилася структура даних, і визначити переваги отриманої моделі.

Контрольні питання

1. Що таке нормалізація бази даних?
2. Що таке функціональна залежність?
3. Які вимоги першої нормальної форми?
4. Які вимоги другої нормальної форми?
5. Які вимоги третьої нормальної форми?
6. Що таке транзитивна залежність?
7. Які аномалії усуває нормалізація?
8. Які переваги нормалізації?
9. Які недоліки надмірної нормалізації?
10. Яка роль нормалізації у проєктуванні бази даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис вихідної структури таблиць;
– визначення функціональних залежностей;
– етапи нормалізації;
– отримана структура таблиць;
– порівняння результатів;
– висновки.

Практична робота 10. Реалізація обмежень цілісності у структурі бази даних

Мета роботи – навчитися визначати та реалізовувати обмеження цілісності у реляційній базі даних, забезпечувати коректність і узгодженість даних за допомогою вбудованих механізмів СКБД.

Ключові положення

Цілісність даних є однією з основних вимог до баз даних і передбачає коректність, узгодженість та достовірність інформації. Для забезпечення цілісності у реляційних базах даних використовуються спеціальні обмеження, які контролюють допустимість значень і взаємозв'язків між даними.

Обмеження цілісності визначають правила, яким повинні відповідати дані у таблицях. Вони реалізуються на рівні структури бази даних і автоматично перевіряються системою керування базами даних під час виконання операцій вставки, оновлення і видалення даних.

Основними типами обмежень є обмеження домену, обмеження сутності та обмеження посилальної цілісності. Кожен із цих типів виконує свою функцію у забезпеченні коректності даних.

Обмеження домену визначають допустимі значення атрибутів. Вони включають перевірку типів даних, діапазонів значень, формату та інших умов. Наприклад, атрибут, що зберігає дату, повинен містити коректне значення дати, а числовий атрибут може мати обмеження на мінімальне і максимальне значення.

Обмеження сутності забезпечують унікальність записів у таблиці. Вони реалізуються за допомогою первинних ключів, які не допускають дублювання значень і не можуть містити невизначені значення. Це гарантує, що кожен запис може бути однозначно ідентифікований.

Обмеження посилальної цілісності забезпечують узгодженість даних між пов'язаними таблицями. Вони реалізуються за допомогою зовнішніх ключів, які посилаються на первинні ключі інших таблиць. Це означає, що значення зовнішнього ключа повинно відповідати існуючому запису або бути відсутнім, якщо це дозволено.

Важливим аспектом є поведінка системи при зміні або видаленні пов'язаних записів. У таких випадках можуть застосовуватися різні дії, зокрема заборона операції, каскадне оновлення або видалення, а також встановлення значення у невизначений стан. Вибір відповідної дії залежить від логіки предметної області.

Крім основних типів обмежень, використовуються додаткові механізми, такі як обмеження CHECK, унікальні обмеження та значення за замовчуванням. Вони дозволяють задавати додаткові правила і підвищують рівень контролю над даними.

Обмеження цілісності реалізуються на рівні СКБД, наприклад у PostgreSQL або MySQL, що дозволяє автоматично контролювати дані незалежно від прикладних програм. Це підвищує надійність системи і зменшує ймовірність помилок.

Правильне визначення обмежень дозволяє уникнути некоректних даних, зменшити кількість помилок і спростити обробку інформації. Водночас надмірна кількість обмежень може ускладнювати роботу системи і знижувати її продуктивність.

Таким чином, реалізація обмежень цілісності є важливим етапом проєктування бази даних, що забезпечує коректність і узгодженість інформації у системі.

Практичне завдання

1. Використати реляційну схему бази даних, отриману у попередніх роботах.
2. Визначити обмеження домену для атрибутів таблиць.
3. Задати первинні ключі для таблиць.
4. Визначити зовнішні ключі для встановлення зв’язків між таблицями.
5. Реалізувати обмеження посилальної цілісності.
6. Додати обмеження CHECK для контролю значень атрибутів.
7. Задати унікальні обмеження для окремих атрибутів.
8. Встановити значення за замовчуванням для атрибутів.
9. Перевірити роботу обмежень шляхом виконання SQL-запитів.
10. Проаналізувати результати роботи обмежень.

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно використати реляційну схему бази даних, створену у попередніх роботах. Важливо переконатися, що структура таблиць є коректною і містить усі необхідні атрибути.

На першому етапі необхідно визначити обмеження домену. Для кожного атрибута слід встановити відповідний тип даних і, за необхідності, додаткові обмеження на допустимі значення. Це дозволить запобігти введенню некоректних даних.

Далі необхідно визначити первинні ключі. Якщо вони вже визначені, слід перевірити їх коректність. У разі необхідності слід внести зміни до структури таблиць.

На наступному етапі необхідно визначити зовнішні ключі і встановити зв’язки між таблицями. При цьому слід враховувати логіку предметної області і забезпечити відповідність значень.

Після цього необхідно реалізувати обмеження CHECK. Вони дозволяють задавати додаткові умови для значень атрибутів. Наприклад, можна обмежити допустимий діапазон числових значень або формат текстових даних.

Далі необхідно встановити унікальні обмеження і значення за замовчуванням. Це дозволить забезпечити додатковий контроль над даними і спростити їх введення.

На наступному етапі необхідно перевірити роботу обмежень. Для цього слід виконати операції вставки, оновлення і видалення даних і проаналізувати реакцію системи. У разі порушення обмежень система повинна видавати повідомлення про помилку.

У процесі виконання роботи необхідно проаналізувати ефективність використаних обмежень і їх вплив на роботу системи. У разі необхідності слід внести зміни до структури бази даних.

Контрольні питання

1. Що таке цілісність даних?
2. Які типи обмежень цілісності існують?
3. Що таке обмеження домену?
4. Що таке первинний ключ?
5. Що таке зовнішній ключ?
6. Що таке посилальна цілісність?

7. Яке призначення обмеження CHECK?
8. Що таке унікальне обмеження?
9. Як перевіряється робота обмежень?
10. Чому важливо використовувати обмеження цілісності?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис структури бази даних;
– перелік обмежень;
– приклади реалізації обмежень;
– результати перевірки;
– аналіз отриманих результатів;
– висновки.

Практична робота 11. Створення структури бази даних та індексів у СКБД

Мета роботи – навчитися створювати структуру бази даних у системі керування базами даних, визначати таблиці, атрибути, ключі та створювати індекси для підвищення продуктивності роботи запитів.

Ключові положення

Створення структури бази даних є завершальним етапом логічного проєктування і переходом до фізичної реалізації системи. На цьому етапі визначені раніше сутності та зв’язки реалізуються у вигляді таблиць, атрибутів, ключів і обмежень у конкретній системі керування базами даних.

Основою структури бази даних є таблиці. Кожна таблиця відповідає окремій сутності і містить атрибути, що описують її властивості. Атрибути визначаються з урахуванням типів даних, що підтримуються СКБД. Правильний вибір типів даних дозволяє оптимізувати використання пам’яті і підвищити продуктивність системи.

При створенні таблиць необхідно визначити первинні ключі, які забезпечують унікальність записів. Також необхідно встановити зовнішні ключі для реалізації зв’язків між таблицями. Це дозволяє забезпечити цілісність даних і правильну організацію структури бази даних.

Після створення таблиць важливим етапом є створення індексів. Індекс є спеціальною структурою даних, що дозволяє прискорити пошук інформації у таблицях. Він функціонує подібно до змісту книги, що дозволяє швидко знайти потрібні дані без перегляду всієї таблиці.

Індекси створюються для одного або кількох атрибутів. Найчастіше вони використовуються для атрибутів, що часто застосовуються у умовах відбору, сортуванні або з’єднаннях таблиць. Використання індексів дозволяє значно зменшити час виконання запитів.

Водночас індекси мають і недоліки. Вони займають додатковий обсяг пам’яті і можуть уповільнювати операції вставки, оновлення і видалення даних, оскільки потребують оновлення індексної структури. Тому необхідно використовувати індекси лише там, де це дійсно необхідно.

Існують різні типи індексів. Найбільш поширеними є індекси на основі дерев, які забезпечують швидкий доступ до даних. Також можуть використовуватися унікальні індекси, що забезпечують відсутність дублювання значень.

У сучасних СКБД, таких як PostgreSQL або MySQL, створення структури бази даних і індексів виконується за допомогою мови SQL. Для цього використовуються оператори визначення даних, які дозволяють створювати таблиці, задавати обмеження і створювати індекси.

Важливим аспектом є перевірка працездатності створеної структури. Після створення таблиць і індексів необхідно виконати тестові запити і оцінити їх продуктивність. Це дозволяє переконатися у правильності реалізації і, за необхідності, внести зміни.

Таким чином, створення структури бази даних і індексів є важливим етапом реалізації інформаційної системи, що забезпечує ефективну роботу з даними і високу продуктивність запитів.

Практичне завдання

1. Використати реляційну схему бази даних, розроблену у попередніх роботах.
2. Створити базу даних у СКБД.
3. Створити таблиці відповідно до розробленої схеми.
4. Визначити типи даних для атрибутів.
5. Задати первинні ключі для таблиць.
6. Встановити зовнішні ключі для зв’язків між таблицями.
7. Створити індекси для атрибутів, що часто використовуються у запитах.
8. Створити унікальні індекси за необхідності.
9. Виконати тестові запити для перевірки структури.
10. Проаналізувати вплив індексів на продуктивність.

Методичні вказівки до виконання практичного завдання

Перед початком роботи необхідно використати реляційну схему бази даних, отриману у попередніх роботах. Слід переконатися, що вона є коректною і не містить помилок.

На першому етапі необхідно створити базу даних у СКБД. Для цього використовується відповідна команда SQL. Після створення бази даних необхідно підключитися до неї для виконання подальших операцій.

Далі необхідно створити таблиці. Для кожної таблиці слід визначити атрибути і їх типи даних. При цьому слід враховувати характер даних і обирати відповідні типи.

Після створення таблиць необхідно задати первинні ключі. Вони повинні забезпечувати унікальність записів. Також необхідно встановити зовнішні ключі для реалізації зв’язків між таблицями.

На наступному етапі необхідно створити індекси. Слід визначити атрибути, які часто використовуються у запитах, і створити для них індекси. Це дозволить підвищити швидкість виконання запитів.

Після створення індексів необхідно виконати тестові запити. Це дозволить оцінити продуктивність системи і перевірити правильність реалізації структури бази даних.

У процесі виконання роботи необхідно звернути увагу на баланс між продуктивністю і використанням ресурсів. Надмірна кількість індексів може призвести до зниження ефективності роботи системи.

На завершальному етапі необхідно проаналізувати результати роботи і зробити висновки щодо ефективності створеної структури бази даних.

Контрольні питання

1. Що таке структура бази даних?
2. Що таке таблиця у СКБД?
3. Що таке індекс?
4. Яке призначення індексів?
5. Які типи індексів існують?
6. Як індекси впливають на продуктивність?
7. Які недоліки використання індексів?
8. Як створюється таблиця у SQL?
9. Як створюється індекс у SQL?
10. Чому важливо тестувати структуру бази даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис структури бази даних;
– перелік таблиць;
– опис атрибутів і типів даних;
– визначення ключів;
– перелік створених індексів;
– результати тестування;
– висновки.

ТЕМА 3. ЕКСПЛУАТАЦІЯ БАЗ ДАНИХ

Практична робота 12. Створення таблиць бази даних засобами мови SQL

Мета роботи – навчитися створювати таблиці бази даних за допомогою мови SQL, визначати атрибути, типи даних, первинні та зовнішні ключі, а також задавати обмеження цілісності.

Ключові положення

Мова SQL є стандартним засобом роботи з реляційними базами даних і використовується для визначення структури, маніпулювання даними та керування доступом. Для створення таблиць застосовуються оператори визначення даних, зокрема команда CREATE TABLE, яка дозволяє описати структуру таблиці та її властивості.

Таблиця є основним елементом реляційної бази даних і використовується для зберігання інформації у вигляді рядків і стовпців. Кожен стовпець відповідає атрибуту, а кожен рядок – окремому запису. При створенні таблиці необхідно визначити назву таблиці, перелік атрибутів і типи даних для кожного з них.

Типи даних визначають формат і допустимі значення атрибутів. Вибір типу даних впливає на ефективність зберігання і обробки інформації. Наприклад, для зберігання текстових значень використовуються символьні типи, для числових – числові, а для дат – спеціальні типи дати і часу.

Важливим елементом є визначення первинного ключа. Він забезпечує унікальність записів у таблиці і не допускає наявності невизначених значень. Первинний ключ може складатися з одного або кількох атрибутів.

Для реалізації зв’язків між таблицями використовуються зовнішні ключі. Вони визначають залежність між таблицями і забезпечують посилальну цілісність даних. Зовнішній ключ повинен посилатися на первинний ключ іншої таблиці.

Окрім ключів, у таблицях можуть використовуватися різні обмеження. Наприклад, обмеження NOT NULL визначає, що атрибут не може мати невизначене значення. Обмеження UNIQUE забезпечує унікальність значень атрибута, а CHECK дозволяє задавати додаткові умови для значень.

При створенні таблиць важливо враховувати логіку предметної області і забезпечити узгодженість структури даних. Неправильне визначення атрибутів або ключів може призвести до проблем у подальшій роботі з базою даних.

У сучасних СКБД, таких як PostgreSQL або MySQL, оператор CREATE TABLE має розширені можливості, що дозволяють задавати складні структури і обмеження.

Після створення таблиць необхідно перевірити їх структуру і переконатися у правильності визначених параметрів. Це можна зробити за допомогою відповідних SQL-запитів або інструментів СКБД.

Таким чином, створення таблиць є базовою операцією у роботі з базами даних, що визначає структуру зберігання інформації і забезпечує основу для подальших операцій.

Практичне завдання

1. Створити нову базу даних або використати існуючу.
2. Визначити перелік таблиць відповідно до розробленої схеми.
3. Створити таблиці за допомогою оператора CREATE TABLE.
4. Визначити атрибути і типи даних.
5. Задати первинні ключі.
6. Встановити зовнішні ключі для зв'язків між таблицями.
7. Додати обмеження NOT NULL, UNIQUE та CHECK.
8. Перевірити створені таблиці.
9. Внести зміни у структуру таблиць за необхідності.
10. Підготувати SQL-запити для створення структури бази даних.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно підготувати реляційну схему бази даних, що містить перелік таблиць, атрибутів і зв'язків. Це дозволить уникнути помилок при створенні таблиць.

На першому етапі необхідно створити базу даних або підключитися до існуючої. Після цього слід перейти до створення таблиць.

Для створення таблиці необхідно використати оператор CREATE TABLE. У ньому слід вказати назву таблиці, перелік атрибутів і їх типи даних. Також необхідно визначити обмеження для атрибутів.

Після визначення атрибутів необхідно задати первинний ключ. Це можна зробити як на рівні окремого атрибута, так і на рівні таблиці.

Далі необхідно визначити зовнішні ключі. Для цього слід вказати атрибути, що посилаються на інші таблиці, і задати відповідні обмеження.

На наступному етапі необхідно додати обмеження цілісності. Зокрема, слід використати NOT NULL для обов'язкових атрибутів, UNIQUE для унікальних значень і CHECK для контролю значень.

Після створення таблиць необхідно перевірити їх структуру. Це можна зробити за допомогою відповідних команд або інструментів СКБД.

У разі виявлення помилок необхідно внести зміни у структуру таблиць. Для цього використовуються відповідні оператори SQL.

На завершальному етапі необхідно оформити створені SQL-запити і підготувати їх для подальшого використання.

Контрольні питання

1. Що таке таблиця у базі даних?
2. Яке призначення оператора CREATE TABLE?
3. Що таке тип даних?
4. Як визначається первинний ключ?
5. Що таке зовнішній ключ?
6. Які обмеження можна задати у таблиці?
7. Що таке NOT NULL?

8. Що таке UNIQUE?
9. Що таке CHECK?
10. Як перевірити структуру таблиці?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис структури таблиць;
– SQL-запити створення таблиць;
– опис використаних обмежень;
– результати перевірки;
– висновки.

Практична робота 13. Додавання, редагування та видалення записів у таблицях бази даних

Мета роботи – навчитися виконувати операції додавання, зміни та видалення записів у таблицях бази даних за допомогою операторів мови SQL, а також контролювати коректність виконання цих операцій.

Ключові положення

Операції додавання, редагування та видалення записів є базовими операціями маніпулювання даними у реляційних базах даних. Вони реалізуються за допомогою операторів мови SQL, що належать до групи DML. Ці оператори дозволяють змінювати вміст таблиць без зміни їх структури.

Оператор INSERT використовується для додавання нових записів у таблицю. При його використанні необхідно вказати значення для атрибутів, що відповідають визначеним типам даних і обмеженням. Додавання записів може виконуватися як для всіх атрибутів, так і для їх частини.

Оператор UPDATE призначений для зміни існуючих записів. Він дозволяє змінювати значення одного або кількох атрибутів для записів, що задовольняють задану умову. Умова задається за допомогою конструкції WHERE. Її відсутність призводить до зміни всіх записів у таблиці.

Оператор DELETE використовується для видалення записів із таблиці. Як і у випадку з UPDATE, він може застосовуватися до всіх записів або лише до тих, що відповідають заданій умові. Неправильне використання цього оператора може призвести до втрати даних.

Під час виконання операцій DML важливо враховувати обмеження цілісності. Система керування базами даних перевіряє коректність значень і зв’язків між таблицями. У разі порушення обмежень операція не виконується.

Особливу увагу слід приділити роботі із зовнішніми ключами. При видаленні записів, що пов’язані з іншими таблицями, можуть застосовуватися різні правила, зокрема заборона видалення або каскадне видалення пов’язаних записів.

Операції зміни даних зазвичай виконуються у межах транзакцій. Це дозволяє забезпечити узгодженість даних і у разі помилки повернути базу даних до попереднього стану.

У сучасних СКБД, таких як PostgreSQL або MySQL, оператори INSERT, UPDATE і DELETE є стандартними і підтримують розширені можливості, що дозволяють виконувати складні операції над даними.

Важливим аспектом є перевірка результатів виконання операцій. Після додавання, зміни або видалення записів необхідно виконати запити вибірки для контролю коректності виконаних дій.

Таким чином, операції маніпулювання даними є основою роботи з базами даних і забезпечують можливість динамічної зміни інформації у системі.

Практичне завдання

1. Використати створені таблиці бази даних.
2. Додати нові записи у таблиці за допомогою оператора INSERT.
3. Виконати додавання кількох записів із різними значеннями.
4. Змінити значення окремих атрибутів за допомогою оператора UPDATE.
5. Використати умови WHERE для вибіркового оновлення.
6. Видалити окремі записи за допомогою оператора DELETE.
7. Виконати видалення записів із використанням умов.
8. Перевірити роботу обмежень цілісності при виконанні операцій.
9. Виконати запити SELECT для перевірки результатів.
10. Проаналізувати зміни у таблицях.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що таблиці бази даних створені і містять початкові дані. Якщо таблиці порожні, слід додати тестові записи.

На першому етапі необхідно виконати операцію додавання даних. Для цього використовується оператор INSERT. Необхідно вказати значення для атрибутів і переконатися, що вони відповідають визначеним типам даних.

Далі необхідно виконати зміну даних за допомогою оператора UPDATE. Особливу увагу слід приділити використанню умови WHERE, щоб уникнути зміни всіх записів у таблиці.

На наступному етапі необхідно виконати видалення записів. Для цього використовується оператор DELETE. Необхідно перевірити, що умова WHERE задана коректно.

Після виконання кожної операції необхідно перевірити результати за допомогою запитів SELECT. Це дозволить переконатися у правильності виконання операцій.

У процесі виконання роботи необхідно звернути увагу на обмеження цілісності. У разі порушення обмежень система повинна видавати повідомлення про помилку.

Також доцільно виконати експерименти з різними варіантами запитів, щоб краще зрозуміти їх поведінку.

На завершальному етапі необхідно проаналізувати отримані результати і зробити висновки щодо особливостей роботи операторів DML.

Контрольні питання

1. Що таке оператор INSERT?
2. Яке призначення оператора UPDATE?
3. Для чого використовується оператор DELETE?
4. Що таке оператор WHERE?
5. Які помилки можуть виникати при виконанні UPDATE?
6. Які наслідки неправильного використання DELETE?
7. Що таке транзакція?
8. Як перевірити результати виконання операцій?
9. Які обмеження впливають на виконання DML?
10. Чому важливо використовувати WHERE?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис виконаних операцій;
– SQL-запити;
– результати виконання;
– аналіз змін у таблицях;
– висновки.

Практична робота 14. Побудова простих SQL-запитів для вибірки даних із таблиць

Мета роботи – навчитися формувати прості SQL-запити для вибірки даних із таблиць, використовувати умови відбору, сортування результатів та отримувати необхідну інформацію з бази даних.

Ключові положення

Оператор SELECT є основним засобом отримання даних із реляційної бази даних. Він дозволяє виконувати вибірку записів із однієї або кількох таблиць і формувати результати у вигляді таблиці. Простий запит SELECT використовується для отримання даних без складних з’єднань і вкладених конструкцій.

Базова структура запиту SELECT містить перелік атрибутів, які потрібно отримати, і джерело даних, тобто таблицю. Якщо необхідно отримати всі атрибути, використовується символ «*». Проте у практиці рекомендується вказувати лише необхідні атрибути для підвищення ефективності роботи.

Вибірка даних може бути обмежена за допомогою умови WHERE. Ця конструкція дозволяє відбирати лише ті записи, які відповідають заданим критеріям. Умови можуть включати оператори порівняння, логічні оператори та спеціальні вирази.

Оператори порівняння дозволяють визначати рівність, нерівність, більші або менші значення. Логічні оператори дозволяють комбінувати кілька умов, створюючи складні критерії відбору. Це дозволяє отримувати більш точні результати.

Для роботи з множинами значень використовуються спеціальні оператори. Наприклад, оператор IN дозволяє перевіряти належність значення до заданого набору, а BETWEEN – визначати належність до діапазону значень. Оператор LIKE використовується для пошуку за шаблоном у текстових даних.

Сортування результатів запиту виконується за допомогою конструкції ORDER BY. Вона дозволяє впорядковувати записи за одним або кількома атрибутами. За замовчуванням сортування виконується за зростанням, але може бути заданий порядок за спаданням.

У деяких випадках необхідно обмежити кількість записів, що повертаються запитом. Для цього використовуються відповідні механізми, які залежать від конкретної СКБД. Це дозволяє працювати з великими обсягами даних і отримувати лише необхідну частину результатів.

У сучасних СКБД, таких як PostgreSQL або MySQL, оператор SELECT підтримує широкий набір можливостей, що дозволяє виконувати як прості, так і складні запити.

Важливим аспектом є перевірка правильності запитів. Необхідно переконатися, що запит повертає очікувані результати і не містить синтаксичних або логічних помилок.

Таким чином, побудова простих SQL-запитів є базовою навичкою роботи з базами даних, що дозволяє отримувати необхідну інформацію і аналізувати дані.

Практичне завдання

1. Використати створені таблиці бази даних.
2. Виконати запит SELECT для вибірки всіх записів із таблиці.
3. Виконати вибірку окремих атрибутів.
4. Використати умову WHERE для відбору записів.
5. Застосувати оператори порівняння.
6. Використати логічні оператори для комбінування умов.
7. Застосувати оператори IN, BETWEEN і LIKE.
8. Виконати сортування результатів за допомогою ORDER BY.
9. Обмежити кількість результатів.
10. Проаналізувати отримані результати.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що таблиці бази даних містять дані. Якщо дані відсутні, необхідно додати тестові записи.

На першому етапі необхідно виконати простий запит SELECT для вибірки всіх записів із таблиці. Це дозволить ознайомитися зі структурою даних.

Далі необхідно виконати вибірку окремих атрибутів. Це дозволить отримати лише необхідну інформацію і зменшити обсяг результатів.

На наступному етапі необхідно використати умову WHERE. Слід сформувати прості умови відбору і перевірити їх роботу.

Після цього необхідно застосувати логічні оператори для створення складніших умов. Це дозволить більш точно визначити критерії вибірки.

Далі необхідно використати оператори IN, BETWEEN і LIKE. Це дозволить виконувати більш гнучкий пошук даних.

На наступному етапі необхідно виконати сортування результатів. Слід перевірити різні варіанти сортування і проаналізувати результати.

Після цього необхідно обмежити кількість записів у результаті. Це дозволить працювати з великими обсягами даних.

У процесі виконання роботи необхідно перевіряти правильність запитів і аналізувати отримані результати.

Контрольні питання

1. Що таке оператор SELECT?
2. Яка структура запиту SELECT?
3. Що таке умова WHERE?
4. Які оператори порівняння існують?
5. Що таке логічні оператори?
6. Для чого використовується оператор IN?
7. Що таке оператор BETWEEN?
8. Для чого використовується оператор LIKE?
9. Як виконується сортування результатів?
10. Як обмежити кількість записів?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис виконаних запитів;
– SQL-запити;
– результати виконання;
– аналіз отриманих результатів;
– висновки.

Практична робота 15. Формування запитів із використанням умов відбору та сортування результатів

Мета роботи – навчитися формувати SQL-запити з використанням умов відбору даних, комбінувати критерії пошуку та виконувати сортування результатів для отримання структурованої інформації.

Ключові положення

Формування запитів із умовами відбору є важливим етапом роботи з базами даних, оскільки дозволяє отримувати не всі дані, а лише ті, що відповідають заданим критеріям. Це

підвищує ефективність обробки інформації та дозволяє зосередитися на потрібних результатах.

Основним інструментом для відбору даних є конструкція WHERE, яка використовується у складі оператора SELECT. Вона дозволяє визначати умови, яким повинні відповідати записи, що включаються у результат запиту. Умови можуть базуватися на значеннях атрибутів, їх порівнянні або відповідності певним шаблонам.

Для формування умов відбору використовуються оператори порівняння. Вони дозволяють визначати рівність, нерівність, більші або менші значення. Ці оператори є базовими і використовуються у більшості запитів.

Для побудови складніших умов застосовуються логічні оператори. Вони дозволяють комбінувати кілька умов, що дає змогу створювати гнучкі критерії відбору. Наприклад, можна одночасно враховувати кілька параметрів або визначати альтернативні умови.

Оператор BETWEEN використовується для визначення діапазону значень. Він дозволяє перевіряти, чи входить значення атрибута у заданий інтервал. Це зручно для роботи з числовими і датами.

Оператор IN дозволяє перевіряти належність значення до заданого набору. Він використовується у випадках, коли необхідно порівняти значення з кількома можливими варіантами.

Оператор LIKE використовується для пошуку текстових даних за шаблоном. Він дозволяє виконувати частковий пошук, що є корисним при роботі з текстовою інформацією.

Сортування результатів виконується за допомогою конструкції ORDER BY. Вона дозволяє впорядковувати записи за значеннями одного або кількох атрибутів. Сортування може виконуватися за зростанням або за спаданням.

Використання сортування є важливим для представлення результатів у зручному вигляді. Воно дозволяє структурувати дані і полегшує їх аналіз.

У сучасних СКБД, таких як PostgreSQL або MySQL, механізми відбору і сортування є стандартними і підтримують широкий набір можливостей.

Важливим аспектом є правильність формування умов. Неправильне використання операторів або відсутність логічної узгодженості може призвести до отримання некоректних результатів.

Таким чином, використання умов відбору і сортування дозволяє формувати гнучкі і ефективні запити, що забезпечують отримання необхідної інформації з бази даних.

Практичне завдання

1. Використати таблиці бази даних із попередніх робіт.
2. Виконати запити з використанням простих умов WHERE.
3. Застосувати оператори порівняння для відбору даних.
4. Використати логічні оператори для комбінування умов.
5. Виконати запити з використанням BETWEEN.
6. Використати оператор IN для вибірки даних.
7. Виконати пошук за шаблоном із використанням LIKE.
8. Виконати сортування результатів за допомогою ORDER BY.
9. Застосувати сортування за кількома атрибутами.
10. Проаналізувати результати виконаних запитів.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що таблиці містять достатню кількість даних для аналізу. У разі необхідності слід додати тестові записи.

На першому етапі необхідно сформувати прості запити з умовами WHERE. Слід перевірити, як працюють оператори порівняння і як змінюється результат залежно від умов.

Далі необхідно використати логічні оператори для створення складніших умов. Це дозволить поєднувати кілька критеріїв і отримувати більш точні результати.

На наступному етапі необхідно застосувати оператор BETWEEN для вибірки значень у заданому діапазоні. Слід перевірити його роботу для різних типів даних.

Після цього необхідно використати оператор IN. Це дозволить виконувати вибірку за набором значень.

Далі необхідно виконати пошук за допомогою оператора LIKE. Слід перевірити різні шаблони і проаналізувати результати.

На наступному етапі необхідно виконати сортування результатів. Слід перевірити сортування за зростанням і за спаданням, а також сортування за кількома атрибутами.

У процесі виконання роботи необхідно перевіряти правильність запитів і аналізувати отримані результати.

Контрольні питання

1. Що таке умова WHERE?
2. Які оператори порівняння використовуються у SQL?
3. Що таке логічні оператори?
4. Для чого використовується оператор BETWEEN?
5. Що таке оператор IN?
6. Для чого використовується оператор LIKE?
7. Як виконується сортування результатів?
8. Що таке ORDER BY?
9. Як виконати сортування за кількома атрибутами?
10. Які помилки можуть виникати при формуванні умов?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис виконаних запитів;
– SQL-запити;
– результати виконання;
– аналіз отриманих результатів;
– висновки.

Практична робота 16. Побудова SQL-запитів із використанням з’єднань таблиць

Мета роботи – навчитися формувати SQL-запити із використанням з’єднань таблиць, отримувати пов’язані дані з кількох таблиць та аналізувати результати виконання таких запитів.

Ключові положення

У реляційних базах даних інформація часто розподілена між кількома таблицями відповідно до принципів нормалізації. Для отримання повної інформації про об’єкти предметної області необхідно об’єднувати дані з різних таблиць. Для цього у мові SQL використовуються з’єднання таблиць.

З’єднання таблиць дозволяє формувати результати запитів, що містять дані з кількох джерел. Основою з’єднання є умова, яка визначає, як саме записи з однієї таблиці пов’язуються із записами з іншої. Зазвичай така умова базується на використанні первинних і зовнішніх ключів.

Найбільш поширеним є внутрішнє з’єднання. Воно повертає лише ті записи, для яких виконується умова відповідності між таблицями. Це означає, що у результаті будуть лише ті рядки, які мають відповідні значення у всіх залучених таблицях.

Зовнішні з’єднання дозволяють отримувати записи навіть у випадку відсутності відповідності у пов’язаній таблиці. Ліве з’єднання повертає всі записи з лівої таблиці, навіть якщо для них немає відповідних записів у правій. Праве з’єднання працює аналогічно, але з протилежного боку.

Повне з’єднання дозволяє отримати всі записи з обох таблиць незалежно від наявності відповідностей. Це використовується у випадках, коли необхідно отримати повну картину даних.

З’єднання таблиць виконується за допомогою конструкції JOIN, яка визначає тип з’єднання і умову об’єднання. Умова задається за допомогою конструкції ON і визначає, які атрибути використовуються для встановлення зв’язку.

При використанні з’єднань важливо правильно визначити умову об’єднання. Помилки у цій умові можуть призвести до отримання некоректних результатів або надлишкових даних.

З’єднання можуть виконуватися для двох або більше таблиць. У складних запитах може використовуватися кілька з’єднань, що дозволяє отримувати дані з великої кількості таблиць.

У сучасних СКБД, таких як PostgreSQL або MySQL, з’єднання таблиць є стандартним механізмом роботи з даними і підтримують різні варіанти реалізації.

Важливим аспектом є оптимізація запитів із з’єднаннями. При роботі з великими обсягами даних неправильне використання з’єднань може призвести до значного зниження продуктивності. Тому необхідно використовувати індекси і обирати оптимальні типи з’єднань.

Таким чином, з’єднання таблиць є потужним інструментом, що дозволяє отримувати комплексну інформацію з реляційної бази даних і реалізовувати складні запити.

Практичне завдання

1. Використати таблиці бази даних із попередніх робіт.
2. Виконати запити з використанням внутрішнього з'єднання.
3. Застосувати умову ON для визначення зв'язку між таблицями.
4. Виконати запити з використанням лівого з'єднання.
5. Проаналізувати результати лівого з'єднання.
6. Виконати запити з використанням правого або повного з'єднання.
7. Застосувати з'єднання для трьох і більше таблиць.
8. Використати псевдоніми таблиць.
9. Порівняти результати різних типів з'єднань.
10. Проаналізувати ефективність виконання запитів.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що у базі даних існують пов'язані таблиці, між якими встановлені зв'язки за допомогою ключів.

На першому етапі необхідно виконати запит із використанням внутрішнього з'єднання. Для цього слід обрати дві таблиці, що мають зв'язок, і сформувати запит із використанням конструкції JOIN та умови ON.

Далі необхідно виконати запити із використанням зовнішніх з'єднань. Слід перевірити, як змінюється результат при використанні різних типів з'єднань.

На наступному етапі необхідно виконати з'єднання для кількох таблиць. Це дозволить отримати більш повну інформацію і проаналізувати складніші структури даних.

Після цього доцільно використати псевдоніми для таблиць. Це спрощує запис запитів і підвищує їх читабельність.

У процесі виконання роботи необхідно аналізувати результати запитів і перевіряти їх правильність. У разі виявлення помилок необхідно перевірити умови з'єднання.

На завершальному етапі необхідно проаналізувати ефективність запитів. Слід звернути увагу на швидкість виконання і обсяг оброблюваних даних.

Контрольні питання

1. Що таке з'єднання таблиць?
2. Яке призначення оператора JOIN?
3. Що таке внутрішнє з'єднання?
4. Що таке ліве з'єднання?
5. Що таке праве з'єднання?
6. Що таке повне з'єднання?
7. Що таке умова ON?
8. Для чого використовуються псевдоніми?
9. Які помилки можуть виникати при з'єднанні таблиць?
10. Як оптимізувати запити із з'єднаннями?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис використаних таблиць;
– SQL-запити;
– результати виконання;
– аналіз отриманих результатів;
– висновки.

Практична робота 17. Використання агрегатних функцій та групування даних у SQL-запитах

Мета роботи – навчитися використовувати агрегатні функції у SQL-запитах, виконувати групування даних, формувати узагальнену інформацію та аналізувати результати.

Ключові положення

Агрегатні функції є важливим інструментом аналізу даних у реляційних базах даних. Вони дозволяють виконувати обчислення над наборами записів і отримувати узагальнені результати. На відміну від звичайних запитів, що повертають окремі записи, агрегатні функції формують результати на основі множини даних.

До основних агрегатних функцій належать COUNT, SUM, AVG, MIN та MAX. Функція COUNT визначає кількість записів, SUM – суму значень, AVG – середнє значення, MIN і MAX – мінімальне і максимальне значення відповідно. Ці функції широко використовуються для аналізу статистичних показників.

Агрегатні функції можуть застосовуватися як до всієї таблиці, так і до окремих груп записів. У найпростішому випадку вони використовуються без групування і повертають одне значення для всієї таблиці.

Для більш складного аналізу використовується групування даних. Групування виконується за допомогою конструкції GROUP BY, яка дозволяє об’єднувати записи у групи за значеннями одного або кількох атрибутів. Для кожної групи обчислюються агрегатні значення.

Наприклад, можна згрупувати записи за певним атрибутом і визначити кількість або середнє значення для кожної групи. Це дозволяє отримати структуровану інформацію і виявити закономірності у даних.

Важливим аспектом є те, що всі атрибути, які використовуються у списку вибірки і не входять до агрегатних функцій, повинні бути включені до конструкції GROUP BY. Це забезпечує коректність запиту.

Для відбору груп використовується конструкція HAVING. Вона дозволяє задавати умови для агрегованих даних. На відміну від WHERE, яка застосовується до окремих записів, HAVING працює з результатами групування.

Агрегатні функції можуть використовуватися разом із іншими конструкціями SQL, такими як з’єднання таблиць або умови відбору. Це дозволяє формувати складні аналітичні запити.

У сучасних СКБД, таких як PostgreSQL або MySQL, агрегатні функції і групування є стандартними засобами аналізу даних.

Важливим аспектом є правильність формування запитів. Неправильне використання агрегатних функцій або групування може призвести до некоректних результатів.

Таким чином, агрегатні функції та групування дозволяють виконувати аналіз даних і отримувати узагальнену інформацію, що є важливою для прийняття рішень.

Практичне завдання

1. Використати таблиці бази даних із попередніх робіт.
2. Виконати запити з використанням функції COUNT.
3. Використати функції SUM і AVG для числових атрибутів.
4. Виконати запити з використанням функцій MIN і MAX.
5. Застосувати агрегатні функції без групування.
6. Виконати групування даних за допомогою GROUP BY.
7. Використати кілька атрибутів у GROUP BY.
8. Застосувати умову HAVING для відбору груп.
9. Поєднати агрегатні функції із умовами WHERE.
10. Проаналізувати результати виконаних запитів.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що таблиці містять достатню кількість даних для виконання аналізу. У разі необхідності слід додати тестові записи.

На першому етапі необхідно виконати прості запити з використанням агрегатних функцій. Слід визначити кількість записів, суму значень і середні значення для вибраних атрибутів.

Далі необхідно виконати групування даних. Для цього слід обрати атрибут, за яким буде виконуватися групування, і застосувати конструкцію GROUP BY.

На наступному етапі необхідно використати кілька атрибутів для групування. Це дозволить отримати більш детальну інформацію.

Після цього необхідно застосувати конструкцію HAVING. Слід сформувати умови для відбору груп і перевірити їх роботу.

Далі необхідно поєднати агрегатні функції з умовами WHERE. Це дозволить обмежити набір даних перед виконанням агрегування.

У процесі виконання роботи необхідно перевіряти правильність запитів і аналізувати отримані результати.

На завершальному етапі необхідно зробити висновки щодо використання агрегатних функцій і групування даних.

Контрольні питання

1. Що таке агрегатна функція?
2. Які основні агрегатні функції існують?
3. Що таке GROUP BY?
4. Яке призначення конструкції HAVING?
5. Чим відрізняється WHERE від HAVING?
6. Як виконується групування за кількома атрибутами?
7. Які помилки можуть виникати при використанні агрегатних функцій?
8. Як перевірити правильність результатів?
9. Яка роль агрегатних функцій у аналізі даних?
10. У яких задачах використовуються агрегатні функції?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис виконаних запитів;
– SQL-запити;
– результати виконання;
– аналіз отриманих результатів;
– висновки.

Практична робота 18. Побудова складних SQL-запитів із використанням підзапитів

Мета роботи – навчитися будувати складні SQL-запити із використанням підзапитів, виконувати вкладену обробку даних, застосовувати підзапити у вибірці, умовах відбору та аналітичних задачах.

Ключові положення

Підзапит є одним із важливих інструментів мови SQL, що дозволяє використовувати результат одного запиту як частину іншого. Такий підхід дає змогу формувати складніші умови обробки даних і отримувати результати, які неможливо або незручно побудувати за допомогою лише простих запитів. Підзапити широко застосовуються у випадках, коли необхідно попередньо визначити проміжний набір даних, а вже потім використати його у зовнішньому запиті.

Підзапит фактично є повноцінним SQL-запитом, який розміщується всередині іншого запиту. Він може повертати одне значення, один стовпець, кілька рядків або навіть тимчасову таблицю. Залежно від цього змінюється спосіб його використання. Підзапит може бути розміщений у частині WHERE, FROM, HAVING або навіть у списку вибірки SELECT.

Одним із найпоширеніших випадків є використання підзапиту в умові WHERE. У такій ситуації зовнішній запит виконує вибірку лише для тих записів, які відповідають результату внутрішнього запиту. Наприклад, можна отримати перелік студентів, середній

бал яких більший за середній бал усіх студентів. У такому випадку підзапит попередньо обчислює середнє значення, а зовнішній запит використовує його в умові порівняння.

Підзапити можуть бути некорельованими і корельованими. Некорельований підзапит виконується незалежно від зовнішнього запиту і передає свій результат як готове значення або набір значень. Такий підхід є простішим для розуміння і часто використовується у типових задачах вибірки та відбору даних.

Корельований підзапит є більш складним, оскільки він залежить від поточного рядка зовнішнього запиту. Це означає, що підзапит виконується окремо для кожного запису зовнішнього запиту. Такий механізм дозволяє реалізувати порівняння всередині груп або виконувати перевірки, які залежать від поточного об'єкта. Водночас корельовані підзапити можуть бути менш ефективними з точки зору продуктивності, особливо при роботі з великими таблицями.

Ще одним важливим випадком є використання підзапитів разом із операторами IN, EXISTS, ANY та ALL. Оператор IN дозволяє перевіряти, чи належить значення атрибута до множини значень, отриманої підзапитом. Це зручно, коли внутрішній запит формує перелік ідентифікаторів або інших значень, за якими потрібно виконати відбір.

Оператор EXISTS використовується для перевірки факту існування хоча б одного запису, що задовольняє певну умову. У цьому випадку важливим є не сам набір даних, а наявність відповідних записів. Такий підхід часто використовується для перевірки наявності пов'язаних записів у іншій таблиці.

Оператори ANY та ALL використовуються для порівняння значення з набором значень, отриманих підзапитом. Вони дозволяють формулювати складні умови типу «більше хоча б одного значення» або «більше всіх значень». Такі конструкції використовуються рідше, але є важливими для реалізації більш складних логічних умов.

Підзапит у частині FROM використовується як тимчасове відношення, що бере участь у подальшій обробці даних. У такому випадку спочатку формується проміжна таблиця, а вже потім до неї застосовуються умови, сортування або групування. Це дає змогу поетапно будувати складні запити і робити їх логіку більш зрозумілою.

Підзапити також можуть використовуватися у списку вибірки SELECT. У такій ситуації для кожного запису зовнішнього запиту обчислюється додаткове значення, наприклад кількість пов'язаних записів або агрегований показник. Такий підхід дозволяє формувати більш інформативні результати без необхідності створення окремих проміжних таблиць.

При побудові складних SQL-запитів із підзапитами важливо враховувати їх вплив на продуктивність. Надмірне використання вкладених запитів, особливо корельованих, може призвести до значного зниження швидкодії системи. У деяких випадках підзапит доцільно замінити з'єднанням таблиць або агрегатним запитом. Проте для навчальних і багатьох практичних задач підзапити залишаються зручним і логічно зрозумілим інструментом.

У сучасних СКБД підзапити є стандартним засобом мови SQL і підтримуються в повному обсязі. Вони широко використовуються в аналітичних запитах, системах звітності, перевірках цілісності та задачах відбору даних за складними критеріями. Правильне використання підзапитів дозволяє будувати гнучкі та потужні засоби обробки даних.

Таким чином, підзапити є важливою складовою SQL, що дозволяє реалізовувати складні механізми вибірки і аналізу інформації. Їх застосування дає змогу працювати з даними на більш високому логічному рівні, поєднуючи декілька етапів обробки у межах одного запиту.

Практичне завдання

1. Використати таблиці бази даних, створені у попередніх роботах.
2. Побудувати запит із некорельованим підзапитом у частині WHERE.
3. Використати підзапит разом з оператором IN.
4. Побудувати запит із використанням оператора EXISTS.
5. Реалізувати підзапит для обчислення агрегованого значення.
6. Побудувати запит із підзапитом у частині FROM.
7. Використати підзапит у списку вибірки SELECT.
8. Реалізувати корельований підзапит.
9. Порівняти результати запитів із підзапитами та аналогічних запитів із JOIN.
10. Проаналізувати правильність і ефективність виконаних запитів.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно переконатися, що у базі даних є кілька взаємопов’язаних таблиць із достатньою кількістю записів. Доцільно використовувати навчальну базу даних, у якій є об’єкти типу студент, дисципліна, оцінка, викладач або інші пов’язані сутності. Це дозволить виконати не лише прості підзапити, а й більш складні варіанти вкладеної обробки.

На першому етапі необхідно побудувати простий підзапит, що повертає одне значення. Наприклад, внутрішній запит може обчислювати середнє, максимальне або мінімальне значення певного атрибута, а зовнішній запит повинен використати цей результат для відбору записів. На цьому етапі важливо зрозуміти сам принцип вкладеного виконання: спочатку формується результат підзапиту, а потім він використовується у зовнішньому операторі SELECT.

Далі необхідно побудувати запит із підзапитом, який повертає множину значень. Для цього доцільно використати оператор IN. Внутрішній запит у такому випадку формує перелік значень, а зовнішній виконує відбір записів, що відповідають цьому переліку. Слід простежити, щоб типи даних у зовнішньому і внутрішньому запиті були сумісними, інакше запит не буде виконано коректно.

Після цього необхідно реалізувати запит із оператором EXISTS. У цьому випадку потрібно перевірити наявність пов’язаних записів у іншій таблиці. Наприклад, можна знайти студентів, для яких існують записи в таблиці оцінок, або дисципліни, для яких існують призначені викладачі. Тут важливо звернути увагу на те, що оператор EXISTS оцінює сам факт існування результату, а не його зміст.

На наступному етапі доцільно побудувати підзапит у частині FROM. Для цього спочатку потрібно сформувати проміжний набір даних, наприклад таблицю з агрегованими показниками, а потім використати цю таблицю як джерело даних у зовнішньому запиті. Такий підхід є зручним, коли аналітичну обробку доцільно розбити на два логічно відокремлені етапи.

Далі слід реалізувати підзапит у списку SELECT. Наприклад, для кожного запису основної таблиці можна обчислити кількість пов’язаних записів, середнє значення або інший показник. У такому випадку підзапит виконується для кожного рядка зовнішнього запиту, тому потрібно уважно стежити за коректністю умов зв’язку.

Окрему увагу необхідно приділити корельованим підзапитам. Для цього слід побудувати запит, у якому внутрішній запит використовує атрибути із зовнішнього. На цьому етапі важливо зрозуміти, що такий підзапит виконується багаторазово, тобто окремо для кожного рядка зовнішнього запиту. Саме тому потрібно не лише досягти правильного результату, а й оцінити доцільність такого способу розв'язання задачі.

Після виконання всіх запитів необхідно порівняти результати підзапитів із результатами аналогічних запитів, побудованих через з'єднання таблиць. Це дозволить побачити, у яких випадках підзапити є більш наочними, а у яких доцільніше використати JOIN. Таке порівняння є важливим не лише для розуміння синтаксису, але й для формування навички вибору найбільш доцільного способу побудови запиту.

У процесі виконання роботи необхідно перевіряти синтаксичну правильність усіх запитів, а також аналізувати отримані результати. У разі виникнення помилок потрібно звертати увагу на узгодження імен атрибутів, типів даних, кількості стовпців у підзапитах і логіку умов відбору. Отримані результати слід зафіксувати і використати для формування висновків щодо особливостей побудови складних SQL-запитів із використанням підзапитів.

Контрольні питання

1. Що таке підзапит у SQL?
2. У яких частинах SQL-запиту може використовуватися підзапит?
3. Чим відрізняється некорельований підзапит від корельованого?
4. Для чого використовується оператор IN разом із підзапитом?
5. Яке призначення оператора EXISTS?
6. У яких випадках підзапит може повертати одне значення?
7. Що таке підзапит у частині FROM?
8. Які особливості має підзапит у списку SELECT?
9. Які проблеми можуть виникати при використанні корельованих підзапитів?
10. У яких випадках доцільно замінити підзапит з'єднанням таблиць?

звіту

– назва практичної роботи;
– мета виконання;
– опис використаних таблиць;
– приклади побудованих підзапитів;
– SQL-запити;
– результати виконання;
– порівняння підзапитів і запитів із JOIN;
– аналіз отриманих результатів;
– висновки.

ТЕМА 4. СУЧАСНІ НАПРЯМИ РОЗВИТКУ БАЗ ДАНИХ І БАЗ ЗНАНЬ

Практична робота 19. Аналіз структури сховища даних та побудова запитів для аналітичної обробки даних

Мета роботи – навчитися аналізувати структуру сховища даних, визначати його основні компоненти, а також будувати SQL-запити для аналітичної обробки інформації.

Ключові положення

Сховище даних є спеціалізованим середовищем зберігання інформації, яке призначене не для оперативного введення та зміни даних, а для їх накопичення, узагальнення та подальшого аналізу. На відміну від традиційних операційних баз даних, що орієнтовані на виконання великої кількості коротких транзакцій, сховища даних створюються для підтримки аналітичних задач, формування звітів, виявлення тенденцій і прийняття управлінських рішень.

Структура сховища даних зазвичай формується на основі інтеграції інформації з кількох джерел. Це можуть бути операційні бази даних, файли, зовнішні інформаційні системи, журнали подій або інші джерела. Перед завантаженням у сховище дані проходять процедури вилучення, очищення, перетворення та узгодження. У результаті формується єдине інформаційне середовище, придатне для аналітичної обробки.

Однією з основних особливостей сховища даних є предметна орієнтованість. Це означає, що інформація у ньому організується не навколо окремих транзакцій, а навколо ключових напрямів діяльності, наприклад продажів, навчального процесу, обліку ресурсів або фінансових показників. Такий підхід дозволяє виконувати аналіз у межах конкретної задачі і формувати більш змістовні аналітичні висновки.

Ще однією важливою особливістю є історичність даних. У сховищі даних часто зберігаються не лише поточні, а й попередні стани інформації. Це дозволяє аналізувати зміни показників у часі, будувати порівняння за періодами, визначати тенденції та виконувати прогнозування. На відміну від операційної бази даних, де значна частина інформації може постійно змінюватися, сховище зберігає історію і дає змогу працювати з накопиченими даними.

У структурі сховища даних важливе місце посідають таблиці фактів і таблиці вимірів. Таблиця фактів містить кількісні показники, що підлягають аналізу. Це можуть бути суми, кількість, тривалість, оцінки, обсяги, витрати та інші числові характеристики. Саме ці дані є основою аналітичних запитів.

Таблиці вимірів містять описові атрибути, за якими виконується аналіз фактів. Наприклад, для аналізу навчального процесу вимірами можуть бути студент, дисципліна, викладач, час, група. Для аналізу продажів вимірами можуть бути товар, покупець, регіон, дата. Виміри дозволяють розглядати факти у різних розрізах і формувати багатовимірну картину даних.

На практиці структура сховища даних часто реалізується у вигляді схеми зірки або схеми сніжинки. У схемі зірки центральне місце займає таблиця фактів, а навколо неї розташовуються денормалізовані таблиці вимірів. Така структура є простою і зручною для

побудови аналітичних запитів. У схемі сніжинки таблиці вимірів можуть бути додатково нормалізовані, що зменшує надлишковість, але ускладнює структуру і запити.

Аналітична обробка даних у сховищі передбачає побудову SQL-запитів, що дозволяють отримувати узагальнену інформацію, виконувати групування, обчислення показників та порівняння. Такі запити зазвичай використовують агрегатні функції, групування, сортування, з'єднання таблиць і, за потреби, підзапити. Основною метою є не просто отримання окремих записів, а формування аналітичного представлення інформації.

Під час побудови аналітичних запитів важливо враховувати логіку предметної області. Наприклад, якщо аналізується успішність студентів, доцільно визначати середній бал за дисциплінами, групами або семестрами. Якщо аналізується діяльність підприємства, можна обчислювати сумарний обсяг продажів за місяцями, регіонами або категоріями товарів. Запити повинні бути спрямовані на отримання змістовної інформації, а не лише на технічне відображення даних.

Важливою особливістю аналітичних запитів є використання групування. Воно дозволяє об'єднувати записи за певними ознаками та обчислювати агреговані показники для кожної групи. Наприклад, можна згрупувати дані за роками, факультетами, видами продукції або іншими ознаками. Це дозволяє перейти від великого масиву окремих записів до узагальненого представлення інформації.

У більш складних задачах аналітичної обробки можуть використовуватися кілька рівнів групування, вкладені запити та порівняння агрегованих результатів. Наприклад, можна знайти дисципліни, середній бал за якими є нижчим за загальний середній показник по факультету, або визначити групи, у яких кількість студентів перевищує середнє значення по всьому закладу. Такі запити дають змогу виконувати не лише описовий, а й порівняльний аналіз.

Під час аналізу структури сховища даних слід звертати увагу на те, які таблиці виконують роль фактів, які є вимірами, які атрибути є ключовими, а які використовуються для аналітичної обробки. Також необхідно оцінити, наскільки структура сховища відповідає задачам аналізу. Якщо структура не дозволяє легко формувати потрібні запити, це свідчить про недоліки її проєктування.

Не менш важливим є питання продуктивності аналітичних запитів. Оскільки сховища даних часто містять великі обсяги інформації, запити повинні бути побудовані так, щоб мінімізувати надмірні обчислення і використовувати наявні зв'язки та індекси. Хоча у навчальних прикладах обсяг даних може бути невеликим, важливо вже на цьому етапі розуміти принципи ефективної побудови аналітичних запитів.

Таким чином, аналіз структури сховища даних і побудова запитів для аналітичної обробки є важливими складовими сучасної роботи з інформацією. Це дозволяє перейти від простого зберігання даних до їх змістовного використання, узагальнення і підтримки прийняття рішень.

Практичне завдання

1. Ознайомитися зі структурою навчального сховища даних.
2. Визначити таблицю фактів і таблиці вимірів.
3. Проаналізувати атрибути таблиці фактів.
4. Проаналізувати атрибути таблиць вимірів.
5. Визначити ключові поля та зв'язки між таблицями.

6. Побудувати SQL-запити для вибірки даних із таблиці фактів разом із даними з вимірів.

7. Побудувати запити з використанням агрегатних функцій.

8. Виконати групування даних за одним або кількома вимірами.

9. Побудувати аналітичні запити для отримання узагальнених показників.

10. Проаналізувати результати виконаних запитів і зробити висновки щодо структури сховища даних.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися зі схемою сховища даних, яке буде використовуватися у роботі. Якщо використовується навчальний приклад, слід визначити, які таблиці містять фактичні показники, а які виконують роль вимірів. Доцільно почати з аналізу назв таблиць, їх атрибутів та зв'язків між ними. Уже на цьому етапі потрібно встановити, яка інформація є описовою, а яка – кількісною і придатною для подальшого аналізу.

На першому етапі необхідно визначити таблицю фактів. Для цього потрібно знайти таблицю, що містить числові показники, які можуть бути використані для обчислення сум, середніх значень, мінімумів, максимумів або кількості записів. Після цього слід визначити таблиці вимірів, які пов'язані з таблицею фактів і містять пояснювальні характеристики. Необхідно коротко описати призначення кожної таблиці та вказати, яку роль вона виконує у структурі сховища.

Далі необхідно проаналізувати атрибути таблиці фактів. Потрібно визначити, які з них є ключовими, які містять числові показники, а які служать посиланнями на виміри. Після цього слід проаналізувати таблиці вимірів і встановити, які атрибути можуть використовуватися для групування та аналітичної обробки. Наприклад, це можуть бути дата, категорія, назва дисципліни, назва групи, викладач, регіон або інші ознаки.

На наступному етапі необхідно побудувати базові SQL-запити, які дозволяють об'єднати таблицю фактів із таблицями вимірів. Для цього слід використати з'єднання таблиць за ключовими полями. Результатом мають бути запити, що дозволяють бачити числові показники разом із пояснювальними характеристиками. Це необхідно для переходу від технічного перегляду структури до змістовного аналізу інформації.

Після цього необхідно побудувати аналітичні запити з використанням агрегатних функцій. Доцільно обчислити кількість записів, суму значень, середні показники або інші характеристики, що відповідають змісту предметної області. Наприклад, можна визначити кількість фактів за певними категоріями, середнє значення показника за окремими групами або сумарний обсяг за вибраними періодами.

На наступному етапі необхідно застосувати групування даних. Слід побудувати кілька запитів, у яких результати узагальнюються за одним виміром, а потім за кількома вимірами одночасно. Це дозволить оцінити, наскільки структура сховища придатна для побудови багатовимірних аналітичних запитів. Також доцільно застосувати сортування результатів, щоб полегшити їх подальший аналіз.

Далі необхідно побудувати один або кілька більш змістовних аналітичних запитів. Наприклад, можна визначити групи з найбільшими або найменшими показниками, знайти періоди з найвищою активністю, порівняти значення між різними категоріями або виявити записи, що відхиляються від середнього рівня. Такі запити повинні показувати, що сховище

даних використовується не лише для зберігання інформації, а й для підтримки аналітичних висновків.

У процесі виконання роботи необхідно фіксувати всі побудовані SQL-запити і результати їх виконання. Особливу увагу слід приділити відповідності отриманих результатів структурі сховища. Якщо певні запити важко побудувати або результати виявляються неінформативними, це потрібно зазначити в аналізі як можливий недолік структури сховища.

На завершальному етапі необхідно сформулювати висновки. У висновках слід оцінити, наскільки структура сховища даних відповідає задачам аналітичної обробки, які запити виявилися найбільш інформативними, а також які особливості організації таблиць фактів і вимірів вплинули на зручність аналізу. Отримані результати слід узагальнити у звіті.

Контрольні питання

1. Що таке сховище даних?
2. Чим сховище даних відрізняється від операційної бази даних?
3. Що таке таблиця фактів?
4. Що таке таблиця вимірів?
5. Які атрибути таблиці фактів використовуються для аналітичної обробки?
6. Яку роль виконує групування даних у аналітичних запитах?
7. Які агрегатні функції найчастіше використовуються в аналітичних запитах?
8. Що таке схема зірки?
9. Які переваги має використання сховища даних для аналітичної обробки?
10. Які труднощі можуть виникати під час побудови аналітичних SQL-запитів?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– короткий опис структури сховища даних;
– визначення таблиці фактів і таблиць вимірів;
– опис ключових полів і зв’язків;
– приклади побудованих SQL-запитів;
– результати аналітичної обробки даних;
– аналіз отриманих результатів;
– висновки.

Практична робота 20. Створення простої бази знань та представлення знань у інформаційній системі

Мета роботи – навчитися створювати просту базу знань, формалізувати знання у вигляді правил та фактів, а також реалізовувати їх використання у інформаційній системі.

Ключові положення

База знань є складовою інтелектуальних інформаційних систем і призначена для зберігання структурованих знань про певну предметну область. На відміну від традиційних баз даних, що зберігають факти у вигляді таблиць, база знань містить не лише дані, а й правила, що дозволяють робити висновки та отримувати нову інформацію на основі наявної.

Основою бази знань є поняття знання, яке відображає узагальнену інформацію про об’єкти, їх властивості та взаємозв’язки. Знання можуть бути представлені у різних формах, зокрема у вигляді фактів, правил, логічних висловлювань або семантичних структур. Вибір способу представлення залежить від задачі та особливостей предметної області.

Факти є базовими елементами бази знань і описують конкретні твердження про об’єкти. Наприклад, факт може вказувати, що певний студент навчається на конкретному курсі або що температура перевищує заданий рівень. Факти можуть змінюватися у процесі роботи системи.

Правила визначають логіку обробки знань і дозволяють отримувати нові висновки. Найбільш поширеною формою є продукційні правила виду «якщо – то». Вони описують залежності між фактами і дозволяють системі виконувати логічні висновки. Наприклад, якщо студент має високий середній бал, то він може отримати підвищену стипендію.

Представлення знань у інформаційній системі передбачає їх формалізацію у вигляді структур, які можуть бути оброблені програмними засобами. Для цього використовуються різні моделі, зокрема логічні моделі, фреймові структури, семантичні мережі та продукційні системи. Кожна з них має свої переваги і використовується залежно від задачі.

Важливим елементом є механізм виведення, який забезпечує обробку правил і фактів. Він виконує логічні операції і формує нові знання на основі наявної інформації. Існують різні підходи до виведення, зокрема прямий і зворотний. Прямий підхід полягає у застосуванні правил до наявних фактів, а зворотний – у перевірці, чи можна підтвердити задане твердження.

У сучасних інформаційних системах бази знань можуть реалізовуватися з використанням різних технологій, зокрема мов програмування або спеціалізованих систем. Наприклад, для реалізації логічних моделей можуть використовуватися декларативні мови, такі як Prolog, або традиційні мови програмування із реалізацією правил.

Важливим аспектом є узгодженість і повнота бази знань. Неправильне формулювання правил або відсутність необхідних фактів може призвести до отримання некоректних результатів. Тому процес створення бази знань потребує ретельного аналізу предметної області.

Практичне застосування баз знань включає експертні системи, системи підтримки прийняття рішень, інтелектуальні довідкові системи та інші програмні засоби, що використовують логічний аналіз інформації.

Таким чином, база знань є інструментом формалізації і використання знань, що дозволяє інформаційним системам виконувати інтелектуальні функції і підтримувати процес прийняття рішень.

Практичне завдання

1. Обрати предметну область для створення бази знань.
2. Визначити основні факти предметної області.
3. Сформувати перелік правил типу «якщо – то».
4. Описати структуру бази знань.
5. Реалізувати факти у вигляді структур даних.
6. Реалізувати правила у вигляді логічних конструкцій.
7. Реалізувати простий механізм виведення.
8. Перевірити роботу бази знань на прикладах.
9. Проаналізувати отримані результати.
10. Оформити опис реалізації бази знань.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно обрати предметну область, для якої буде створюватися база знань. Вона повинна бути достатньо простою, щоб можна було чітко визначити факти і правила, але водночас містити логічні залежності між об’єктами.

На першому етапі необхідно визначити факти. Для цього слід описати конкретні твердження, що характеризують предметну область. Факти повинні бути чіткими і не містити суперечностей.

Далі необхідно сформулювати правила. Вони повинні відображати логічні залежності між фактами. Кожне правило має бути сформульоване у вигляді умови і результату.

На наступному етапі необхідно визначити структуру бази знань. Це може бути набір таблиць, списків або інших структур, що дозволяють зберігати факти і правила.

Після цього необхідно реалізувати базу знань у програмному середовищі. Для цього можна використати будь-яку мову програмування або спеціалізований інструмент.

Далі необхідно реалізувати механізм виведення. Він повинен аналізувати факти і застосовувати правила для отримання нових знань.

Після реалізації необхідно перевірити роботу системи на прикладах. Слід задати вхідні дані і перевірити, чи правильно формуються висновки.

У процесі виконання роботи необхідно звернути увагу на коректність логіки і узгодженість правил. У разі виявлення помилок необхідно внести зміни.

На завершальному етапі необхідно оформити результати роботи і зробити висновки щодо ефективності створеної бази знань.

Контрольні питання

1. Що таке база знань?
2. Чим база знань відрізняється від бази даних?
3. Що таке факт?
4. Що таке правило?

5. Що таке продукційне правило?
6. Які існують способи представлення знань?
7. Що таке механізм виведення?
8. Які типи виведення існують?
9. Які помилки можуть виникати при створенні бази знань?
10. У яких системах використовуються бази знань?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис предметної області;
– перелік фактів;
– перелік правил;
– опис реалізації;
– результати роботи системи;
– аналіз отриманих результатів;
– висновки.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Глоба Л. С., Суліма С. В., Скулиш М. А. Робота з базами даних: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Інформаційно-комунікаційні технології» спеціальності 172 «Електронні комунікації та радіотехніка». 2-ге вид. Київ: КПІ ім. Ігоря Сікорського, 2024. 569 с.
2. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Harlow: Pearson, 2014. 1440 p. (додано)
3. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Boston: Pearson, 2016. 1272 p.
4. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill Education, 2019. 1376 p.
5. Sergeiev-Horchynskyi O. O., Hloba L. S., Machalin I. O. Системи баз даних. Комп'ютерний практикум: навч. посіб. Київ: Національний центр «Мала академія наук України», 2021. 128 с.

Допоміжна

6. Perkins L., Redmond E., Wilson J. R. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 2nd ed. Raleigh: Pragmatic Bookshelf, 2018. 360 p.
7. Ploetz A., Kandhare D., Kadambi S., Wu X. Seven NoSQL Databases in a Week: Get Up and Running with the Fundamentals and Functionalities of Seven of the Most Popular NoSQL Databases. Birmingham: Packt Publishing, 2018. 308 p.
8. Leavitt N. Will NoSQL Databases Live Up to Their Promise? IEEE Computer. 2010. Vol. 43(2). P. 12–14.
9. Strauch C. NoSQL Databases. Stuttgart Media University, 2012. 20 p.
10. Mansouri Y., Prokhorenko V., Babar M. A. An Automated Implementation of Hybrid Cloud for Performance Evaluation of Distributed Databases. Journal of Network and Computer Applications. 2020. Vol. 167.
11. Janev V., Graux D., Jabeen H., Sallinger E. Knowledge Graphs and Big Data Processing. Cham: Springer, 2020. 312 p.

Інформаційні ресурси

12. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs
13. Oracle Corporation. MySQL Documentation. URL: https://dev.mysql.com/doc
14. MongoDB Inc. MongoDB Documentation. URL: https://www.mongodb.com/docs
15. SQLite Consortium. SQLite Documentation. URL: https://www.sqlite.org/docs.html
16. Oracle Corporation. Oracle Database Documentation. URL: https://docs.oracle.com/en/database/

Навчальне видання

Глоба Лариса Сергіївна

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ

Методичні вказівки до практичних робіт для здобувачів вищої освіти,
які навчаються за спеціальностями галузі «Інформаційні технології»

Українською мовою