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

Л.С. Глоба

ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ

Опорний конспект лекцій
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. Опорний конспект лекцій для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» [Електронне видання]. Кафедра комп’ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 40 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

Лекція 1. Основи баз даних та систем керування базами даних

Поняття даних та інформації є фундаментальним для розуміння принципів побудови сучасних інформаційних систем. Дані є первинним представленням фактів, явищ або процесів, що фіксуються у формалізованому вигляді для подальшої обробки. Вони можуть бути числовими, текстовими, графічними або представленими в іншій формі, але самі по собі не мають завершеного змістового навантаження. Інформація, на відміну від даних, є результатом інтерпретації, обробки та аналізу даних, що набуває змісту та використовується для прийняття рішень. Таким чином, інформація є осмисленими даними, що мають контекст і практичну цінність для користувача або системи.

У сучасних інформаційних технологіях процес перетворення даних на інформацію реалізується за допомогою програмних засобів, алгоритмів та спеціалізованих систем, серед яких центральне місце займають бази даних і системи керування базами даних. Це зумовлено необхідністю ефективного зберігання, структурування, пошуку та обробки великих обсягів даних, що виникають у процесі діяльності організацій, підприємств та інформаційних систем різного призначення.

База даних є організованою сукупністю взаємопов’язаних даних, що зберігаються у структурованому вигляді та призначені для спільного використання різними користувачами або програмними системами. Важливою ознакою бази даних є її логічна узгодженість, мінімізація надлишковості та забезпечення цілісності даних. База даних не є простою сукупністю файлів, а представляє собою структуровану модель предметної області, що відображає реальні об’єкти, їх властивості та взаємозв’язки.

Система керування базами даних (СКБД) є програмним забезпеченням, що забезпечує створення, модифікацію, збереження, пошук і обробку даних у базі даних. Вона виступає як посередник між користувачем або прикладною програмою та фізичним сховищем даних. Основні функції СКБД містять управління структурою бази даних, забезпечення цілісності даних, контроль доступу, підтримку багатокористувацького режиму роботи, резервне копіювання та відновлення даних.

Інформаційна система є більш широким поняттям і містить не лише базу даних і СКБД, але й апаратні засоби, програмне забезпечення, користувачів, а також організаційні процедури, що забезпечують збір, зберігання, обробку та передачу інформації. Таким чином, база даних є ядром інформаційної системи, а СКБД – інструментом для роботи з цим ядром.

Призначення баз даних у сучасних інформаційних системах визначається потребою ефективного управління інформаційними ресурсами. У сучасних умовах обсяги даних постійно зростають, а вимоги до швидкості доступу, надійності та безпеки стають все більш жорсткими. Бази даних дають змогу централізовано зберігати дані, уникати їх дублювання, забезпечувати узгодженість і актуальність інформації, а також реалізовувати складні механізми пошуку та аналізу.

Використання баз даних є характерним для різноманітних сфер діяльності, зокрема банківських систем, електронної комерції, систем управління підприємствами, медичних інформаційних систем, освітніх платформ, соціальних мереж тощо. У кожному з цих випадків база даних виконує роль структурованого сховища інформації, що підтримує роботу прикладних програм і забезпечує доступ користувачів до необхідних даних.

Архітектура систем керування базами даних визначає принципи організації та взаємодії компонентів СКБД. Однією з ключових концепцій є багаторівнева архітектура, яка передбачає розділення логіки представлення даних, логіки їх обробки та фізичного зберігання. Такий підхід дає змогу підвищити гнучкість системи, забезпечити незалежність даних від програм і спростити їх модифікацію.

СКБД зазвичай містить такі основні компоненти: підсистема керування даними, що відповідає за фізичне зберігання даних на носіях; підсистема обробки запитів, що виконує аналіз і оптимізацію запитів користувачів; підсистема керування транзакціями, що забезпечує узгодженість даних у багатокористувацькому середовищі; підсистема безпеки, що контролює доступ до даних; а також інтерфейси взаємодії з користувачами та прикладними програмами.

Важливою характеристикою сучасних СКБД є підтримка транзакцій – логічно завершених операцій над даними, які виконуються як єдине ціле. Транзакції забезпечують властивості надійності та цілісності даних, зокрема атомарність, узгодженість, ізольованість та довговічність. Це дає змогу гарантувати правильність обробки даних навіть у випадку збоїв або паралельного доступу кількох користувачів.

Одним із ключових принципів організації баз даних є розділення рівнів представлення даних. Виділяють три основні рівні: зовнішній, концептуальний та внутрішній. Така трирівнева архітектура є основою для забезпечення незалежності даних, що означає можливість змінювати структуру даних без впливу на прикладні програми або користувачів.

Зовнішній рівень представлення даних відповідає за відображення даних для конкретних користувачів або груп користувачів. Він визначає, які саме дані доступні користувачу та в якому вигляді вони представлені. Наприклад, різні користувачі можуть бачити одну і ту ж базу даних по-різному залежно від їхніх ролей і прав доступу. Зовнішній рівень дає змогу приховати складність внутрішньої організації даних і забезпечити зручний інтерфейс для роботи.

Концептуальний рівень є логічною моделлю всієї бази даних. Він описує структуру даних, їх взаємозв’язки, обмеження цілісності та правила організації. На цьому рівні визначаються сутності, атрибути, зв’язки між сутностями, а також основні обмеження, що забезпечують коректність даних. Концептуальний рівень є незалежним від конкретної реалізації СКБД і не враховує фізичні аспекти зберігання даних.

Внутрішній рівень представлення даних відповідає за фізичну організацію даних на носіях інформації. Він визначає, як саме дані зберігаються у пам’яті комп’ютера або на зовнішніх носіях, які використовуються структури даних, індекси, методи доступу та інші технічні механізми. Внутрішній рівень є найбільш близьким до апаратного забезпечення і орієнтований на забезпечення ефективності зберігання та швидкості доступу до даних.

Розділення цих рівнів дає змогу досягти логічної та фізичної незалежності даних. Логічна незалежність означає можливість змінювати концептуальну схему без необхідності зміни зовнішніх схем, тоді як фізична незалежність дає змогу змінювати внутрішню організацію даних без впливу на концептуальний рівень.

Таким чином, бази даних та системи керування базами даних є ключовими компонентами сучасних інформаційних систем. Вони забезпечують ефективне управління даними, підтримують багатокористувацький доступ, гарантують цілісність і безпеку інформації, а також дають змогу реалізувати складні алгоритми обробки даних. Розуміння основних понять, архітектури та рівнів представлення даних є необхідною умовою для подальшого вивчення методів проєктування, реалізації та оптимізації баз даних.

Лекція 2. Історія розвитку систем керування базами даних та їх класифікація

Розвиток систем керування базами даних є логічним результатом еволюції підходів до зберігання та обробки інформації в обчислювальних системах. На початкових етапах розвитку обчислювальної техніки дані зберігалися у вигляді окремих файлів, організованих за допомогою файлових систем. Такий підхід отримав назву файлової організації даних і був характерним для перших поколінь інформаційних систем.

Файлова система зберігання даних передбачає, що кожна прикладна програма самостійно визначає формат, структуру та способи доступу до даних. Дані зберігаються у файлах, які можуть мати різну організацію – послідовну, індексно-послідовну або пряму. У межах такого підходу кожна програма працює зі своїми файлами, що призводить до виникнення значної кількості незалежних наборів даних.

Основною проблемою файлових систем є надлишковість даних. Оскільки різні програми можуть використовувати однакову інформацію, вона часто дублюється у різних файлах. Це призводить до збільшення обсягу зберігання, а також до труднощів із забезпеченням узгодженості даних. У разі зміни інформації необхідно оновлювати всі копії, що підвищує ймовірність помилок.

Ще одним суттєвим обмеженням є відсутність централізованого управління даними. У файлових системах немає єдиного механізму контролю доступу, забезпечення цілісності та узгодженості даних. Кожна програма реалізує ці функції самостійно, що ускладнює розробку та супровід програмного забезпечення.

Також файловим системам притаманна жорстка залежність між даними та програмами. Формат даних визначається у програмному коді, тому будь-яка зміна структури файлів потребує внесення змін до всіх програм, що працюють із цими даними. Це значно знижує гнучкість системи та ускладнює її розвиток.

Зростання обсягів даних і складності інформаційних систем зумовило необхідність створення більш ефективних засобів управління даними. У 1960-х роках почали з’являтися перші системи керування базами даних, які запропонували новий підхід до організації інформації – централізоване зберігання та незалежність даних від прикладних програм.

Одними з перших моделей даних, що використовувалися у СКБД, були ієрархічна та мережева. Ієрархічна модель організовує дані у вигляді дерева, де кожен елемент має одного батьківського елемента. Такий підхід був реалізований, зокрема, у системі IBM IMS, яка широко застосовувалася у корпоративних системах.

Мережева модель даних є більш гнучкою і дає змогу встановлювати складні зв’язки між записами, де один елемент може мати кілька батьків. Ця модель була стандартизована організацією CODASYL і використовувалася у багатьох ранніх СКБД.

Наступним етапом розвитку стало впровадження реляційної моделі даних, запропонованої Edgar F. Codd у 1970 році. Реляційна модель базується на представленні даних у вигляді таблиць, що значно спрощує їх розуміння та обробку. Вона швидко стала домінуючою завдяки своїй простоті, формальній строгості та підтримці декларативних мов запитів.

З розвитком реляційної моделі з’явилися такі СКБД, як Oracle Database, MySQL, PostgreSQL, які широко використовуються і сьогодні. Вони забезпечують високий рівень абстракції, підтримують складні запити та гарантують цілісність даних.

У подальшому розвиток СКБД привів до появи об’єктно-орієнтованих та об’єктно-реляційних моделей, які поєднують можливості традиційних баз даних із принципами

об’єктно-орієнтованого програмування. Це дало змогу працювати зі складними структурами даних, такими як мультимедійні об’єкти, географічні дані та інші нетипові формати.

Сучасний етап розвитку характеризується появою нереляційних баз даних, відомих як NoSQL-системи. Вони орієнтовані на роботу з великими обсягами неструктурованих або слабко структурованих даних і забезпечують високу масштабованість. До таких систем належать, наприклад, MongoDB та Cassandra.

Переваги використання систем керування базами даних у порівнянні з файловими системами є суттєвими. Насамперед, СКБД забезпечують централізоване управління даними, що дає змогу уникнути їх дублювання та забезпечити узгодженість. Дані зберігаються в єдиній базі, а доступ до них регулюється через СКБД.

Ще однією важливою перевагою є незалежність даних від програм. Завдяки цьому зміни у структурі бази даних не потребують модифікації прикладного програмного забезпечення, що значно спрощує супровід системи.

СКБД також забезпечують механізми контролю цілісності даних, що дають змогу автоматично перевіряти правильність введеної інформації. Крім того, вони підтримують багатокористувацький режим роботи, що є критично важливим для сучасних інформаційних систем.

Важливим аспектом є забезпечення безпеки даних. СКБД дають змогу реалізувати різні рівні доступу, автентифікацію користувачів та захист інформації від несанкціонованого доступу. Також вони підтримують резервне копіювання та відновлення даних, що підвищує надійність системи.

Класифікація систем керування базами даних може здійснюватися за різними ознаками, зокрема за моделями даних та за архітектурою.

За моделями даних СКБД поділяються на ієрархічні, мережеві, реляційні, об’єктно-орієнтовані, об’єктно-реляційні та нереляційні. Ієрархічні СКБД використовують деревоподібну структуру даних, мережева модель підтримує складні зв’язки між елементами, реляційна модель базується на таблицях, об’єктно-орієнтовані системи працюють із об’єктами, а NoSQL-системи орієнтовані на гнучкі структури даних.

За архітектурою СКБД поділяються на централізовані, клієнт-серверні та розподілені. Централізовані системи передбачають зберігання даних на одному сервері, що спрощує управління, але обмежує масштабованість. Клієнт-серверна архітектура розділяє функції між сервером бази даних і клієнтськими програмами, що дає змогу підвищити продуктивність і гнучкість.

Розподілені СКБД передбачають зберігання даних на кількох вузлах мережі. Це дає змогу забезпечити високу відмовостійкість, масштабованість та ефективну обробку великих обсягів даних. У таких системах дані можуть бути розподілені або репліковані між вузлами.

Таким чином, розвиток систем керування базами даних пройшов шлях від простих файлових систем до складних розподілених платформ, що забезпечують ефективне управління великими обсягами інформації. Сучасні СКБД є невід’ємною складовою інформаційних систем і дають змогу реалізувати складні задачі обробки, аналізу та зберігання даних у різних сферах діяльності.

Лекція 3. Моделі даних та їх порівняльна характеристика

Моделі даних є ключовим поняттям у теорії та практиці організації баз даних, оскільки саме вони визначають спосіб представлення, структурування та взаємозв'язку даних у системі. Модель даних є формальним описом того, як дані організовані, які обмеження на них накладаються та які операції можуть виконуватися над ними. Вибір моделі даних безпосередньо впливає на ефективність обробки інформації, складність розробки програмного забезпечення та можливості масштабування системи.

У процесі розвитку інформаційних систем сформувалося кілька базових моделей даних, серед яких найбільш значущими є ієрархічна, мережева та реляційна. Кожна з цих моделей має свої особливості, переваги та обмеження, що визначають доцільність її використання у конкретних умовах.

Ієрархічна модель даних є однією з перших моделей, що застосовувалися у системах керування базами даних. Вона базується на деревоподібній структурі, де дані організовані у вигляді вузлів, пов'язаних між собою відношеннями «батько–нащадок». Кожен нащадок має лише одного батька, але батько може мати кілька нащадків. Така структура є природною для представлення даних, що мають чітку ієрархію, наприклад організаційні структури або файлові системи.

Класичним прикладом реалізації ієрархічної моделі є система IBM IMS, яка широко використовувалася у корпоративних інформаційних системах. Основною перевагою цієї моделі є висока швидкість доступу до даних при виконанні операцій, що відповідають ієрархічній структурі. Проте її суттєвим недоліком є низька гнучкість, оскільки зміна структури даних або реалізація складних зв'язків між елементами є складною задачею.

Мережева модель даних є розвитком ієрархічної моделі і дає змогу усунути деякі її обмеження. У цій моделі дані організовані у вигляді графа, де записи можуть мати кілька батьків і кілька нащадків. Це дає змогу реалізувати більш складні зв'язки між даними, що відповідає реальним предметним областям.

Мережева модель була стандартизована організацією CODASYL і широко застосовувалася у 1970-х роках. Основною перевагою цієї моделі є її гнучкість у представленні зв'язків між даними. Водночас вона є складною у реалізації та використанні, оскільки потребує детального опису шляхів доступу до даних і складних механізмів навігації.

Реляційна модель даних є найбільш поширеною у сучасних інформаційних системах. Вона була запропонована Edgar F. Codd і базується на представленні даних у вигляді таблиць, які називаються відношеннями. Кожна таблиця містить рядки (кортежі) та стовпці (атрибути), що описують властивості об'єктів.

Основною перевагою реляційної моделі є її простота та формальна строгість. Дані у таблицях є логічно незалежними, а доступ до них здійснюється за допомогою декларативних мов запитів, таких як SQL. Це дає змогу користувачам формулювати запити без необхідності знання внутрішньої структури зберігання даних.

Реляційна модель також забезпечує високий рівень незалежності даних, підтримку цілісності та можливість виконання складних запитів. Вона стала основою для більшості сучасних СКБД, зокрема PostgreSQL та MySQL.

Незалежно від конкретної моделі, усі моделі даних оперують базовими елементами, серед яких ключовими є сутності, атрибути та зв'язки.

Сутність є об’єктом предметної області, який має самостійне значення і може бути однозначно ідентифікований. Наприклад, у базі даних університету сутностями можуть бути студент, викладач, дисципліна. Кожна сутність характеризується певним набором властивостей.

Атрибут є властивістю сутності, що описує її характеристики. Наприклад, для сутності студент атрибутами можуть бути прізвище, ім’я, дата народження, номер залікової книжки. Атрибути визначають структуру даних і тип інформації, що зберігається у базі даних.

Зв’язок визначає взаємодію між сутностями. Він показує, як об’єкти предметної області пов’язані між собою. Наприклад, студент може бути пов’язаний із дисципліною через зв’язок «вивчає». Зв’язки можуть мати різну кардинальність, наприклад один-до-одного, один-до-багатьох або багато-до-багатьох.

У різних моделях даних ці елементи реалізуються по-різному. В ієрархічній моделі сутності представлені у вигляді вузлів дерева, атрибути є властивостями цих вузлів, а зв’язки реалізуються через ієрархічні відношення. У мережевій моделі зв’язки мають більш складний характер і можуть бути представлені у вигляді довільних графових структур. У реляційній моделі сутності представлені у вигляді таблиць, атрибути – це стовпці таблиць, а зв’язки реалізуються через зовнішні ключі.

Порівняльна характеристика моделей даних дає змогу визначити їх сильні та слабкі сторони. Ієрархічна модель є простою у реалізації та забезпечує високу продуктивність для задач, що відповідають її структурі. Проте вона є негнучкою і не підходить для складних предметних областей із великою кількістю взаємозв’язків.

Мережева модель є більш гнучкою і дає змогу ефективно представляти складні зв’язки між даними. Водночас вона є складною у використанні, оскільки потребує явного визначення шляхів доступу до даних, що ускладнює розробку програм.

Реляційна модель забезпечує баланс між гнучкістю та простотою. Вона дає змогу легко змінювати структуру даних, підтримує декларативні запити та забезпечує високий рівень незалежності даних. Її основним недоліком може бути зниження продуктивності при роботі з дуже великими обсягами даних або складними зв’язками, хоча сучасні СКБД значною мірою вирішують ці проблеми.

Таким чином, вибір моделі даних залежить від вимог конкретної інформаційної системи. Ієрархічна та мережева моделі мають історичне значення і використовуються у спеціалізованих системах, тоді як реляційна модель є стандартом для більшості сучасних застосувань. Розуміння принципів побудови моделей даних та їх характеристик є необхідним для ефективного проєктування та реалізації баз даних.

Лекція 4. Реляційна модель даних та її основні поняття

Реляційна модель даних є базовою теоретичною та практичною основою сучасних систем керування базами даних. Вона була запропонована Edgar F. Codd і базується на використанні математичного апарату теорії множин та реляційної алгебри. Основна ідея моделі полягає у представленні даних у вигляді відношень, що інтерпретуються як таблиці з чітко визначеною структурою.

Реляційна модель забезпечує високий рівень абстракції, що дає змогу користувачам працювати з даними без урахування фізичних аспектів їх зберігання. Це робить її універсальною та придатною для реалізації широкого спектра інформаційних систем – від простих облікових систем до складних корпоративних платформ, що використовують такі СКБД, як PostgreSQL або Oracle Database.

Основним структурним елементом реляційної моделі є відношення. Відношення є множиною кортежів, які мають однакову структуру. У практичній реалізації відношення відповідає таблиці, що містить стовпці та рядки. Кожне відношення має ім'я та описується схемою, яка визначає набір атрибутів і їх типи.

Атрибут є властивістю відношення і відповідає стовпцю таблиці. Кожен атрибут має ім'я та визначений тип значень, які він може приймати. Атрибути описують характеристики об'єктів предметної області. Наприклад, у відношенні Студент атрибутами можуть бути прізвище, ім'я, номер залікової книжки.

Домен визначає множину допустимих значень для атрибута. Він задає тип і обмеження значень, що можуть зберігатися у відповідному стовпці. Наприклад, домен для атрибута дата народження є множиною всіх коректних дат, а для атрибута оцінка – множиною допустимих числових значень у певному діапазоні. Використання доменів дає змогу забезпечити коректність і узгодженість даних.

Кортеж є окремим елементом відношення і відповідає рядку таблиці. Він містить конкретні значення атрибутів, що описують один об'єкт предметної області. Кортежі у відношенні не впорядковані, тобто порядок рядків у таблиці не має значення з точки зору реляційної моделі.

Одним із ключових понять реляційної моделі є ключ. Ключ є атрибутом або набором атрибутів, що дає змогу однозначно ідентифікувати кожен кортеж у відношенні. Вибір правильного ключа є важливим етапом проєктування бази даних, оскільки він впливає на забезпечення цілісності та ефективність обробки даних.

Первинний ключ є основним ідентифікатором кортежів у відношенні. Він повинен бути унікальним для кожного запису та не може містити невизначених значень. Наприклад, у відношенні Студент первинним ключем може бути номер залікової книжки або інший унікальний ідентифікатор. Первинний ключ забезпечує унікальність записів і є основою для встановлення зв'язків між таблицями.

Зовнішній ключ є атрибутом або набором атрибутів, що використовується для встановлення зв'язку між двома відношеннями. Він вказує на первинний ключ іншого відношення і забезпечує зв'язність даних. Наприклад, у відношенні Успішність може бути атрибут, що посилається на первинний ключ відношення Студент. Таким чином реалізується зв'язок між студентами та їх оцінками.

Використання зовнішніх ключів є основою для забезпечення посилальної цілісності, що означає узгодженість зв'язків між таблицями. СКБД контролює, щоб значення зовнішнього ключа відповідали існуючим значенням первинного ключа у пов'язаній таблиці.

Реляційна модель також передбачає наявність ряду обмежень, що забезпечують коректність і узгодженість даних. Одним із базових є обмеження цілісності сутності. Воно вимагає, щоб первинний ключ кожного відношення був унікальним і не містив невизначених значень. Це гарантує можливість однозначної ідентифікації кожного запису.

Іншим важливим обмеженням є посилальна цілісність. Вона забезпечує коректність зв'язків між відношеннями. Якщо у відношенні використовується зовнішній ключ, його

значення повинні відповідати значенням первинного ключа у пов’язаному відношенні або бути відсутніми, якщо це дозволено правилами.

Також важливими є доменні обмеження, що визначають допустимі значення атрибутів. Вони забезпечують правильність даних на рівні окремих полів і можуть включати перевірку типу, діапазону значень, формату тощо.

Крім того, у реляційній моделі можуть застосовуватися додаткові обмеження, що визначають бізнес-правила предметної області. Наприклад, може бути встановлено обмеження, що оцінка студента повинна знаходитися у певному діапазоні або що дата завершення події не може бути ранішою за дату її початку. Такі обмеження реалізуються за допомогою механізмів СКБД.

Важливою особливістю реляційної моделі є те, що вона не допускає дублювання кортежів у відношенні, оскільки відношення визначається як множина, а не як список. Це забезпечує унікальність записів і спрощує обробку даних.

Ще одним принципом є відсутність значущості порядку атрибутів і кортежів. У реляційній моделі порядок стовпців і рядків не має логічного значення, що відрізняє її від файлових систем, де порядок може бути важливим.

Таким чином, реляційна модель даних забезпечує формально визначений, логічно узгоджений та гнучкий підхід до організації даних. Використання таких понять, як відношення, атрибут, домен, кортеж і ключі, дає змогу створювати ефективні структури даних, а система обмежень гарантує їх цілісність і коректність. Це робить реляційну модель основою для побудови більшості сучасних інформаційних систем.

Лекція 5. Основи реляційної алгебри та сучасні моделі даних NoSQL

Реляційна алгебра є формальною математичною основою реляційної моделі даних і визначає набір операцій, за допомогою яких виконується обробка даних у відношеннях. Вона базується на теорії множин і дає змогу формалізувати запити до бази даних. Саме на принципах реляційної алгебри побудовано мову SQL, яка використовується у більшості сучасних систем керування базами даних, зокрема у PostgreSQL та MySQL.

Основною ідеєю реляційної алгебри є те, що всі операції виконуються над відношеннями і результатом кожної операції також є відношення. Це забезпечує замкненість операцій і дає змогу комбінувати їх для формування складних запитів.

Однією з базових операцій є вибірка. Вибірка полягає у відборі кортежів із відношення, що задовольняють задану умову. Вона дає змогу отримати підмножину рядків таблиці, наприклад, вибрати студентів певного курсу або записи з оцінкою вище заданого значення. Вибірка не змінює структуру відношення, а лише обмежує кількість кортежів.

Проєкція є операцією, що дає змогу вибрати певні атрибути з відношення. У результаті проєкції формується нове відношення, яке містить лише вибрані стовпці. Ця операція використовується для зменшення обсягу даних і отримання лише необхідної інформації. Важливою особливістю проєкції є усунення дублікатів, оскільки результатом є множина.

Операція об’єднання використовується для поєднання двох відношень з однаковою структурою. У результаті формується нове відношення, що містить усі кортежі обох відношень без повторень. Ця операція аналогічна об’єднанню множин у математиці.

Різниця відношень є операцією, що дає змогу отримати кортежі, які належать одному відношенню, але відсутні в іншому. Вона також застосовується до відношень із однаковою структурою і використовується для виключення певних даних.

Декартовий добуток є операцією, що формує всі можливі комбінації кортежів із двох відношень. Якщо одне відношення містить m кортежів, а інше – n, результат міститиме m×n кортежів. Ця операція є основою для реалізації більш складних операцій, зокрема з'єднання.

З'єднання є однією з найважливіших операцій реляційної алгебри. Воно дає змогу об'єднувати дані з двох відношень на основі певної умови. Найчастіше використовується з'єднання за рівністю значень атрибутів, наприклад, при об'єднанні таблиць студентів і їх успішності за ідентифікатором студента. З'єднання є похідною операцією, що базується на декартовому добутку з подальшою вибіркою.

Реляційна алгебра забезпечує формальний апарат для опису операцій над даними, але з розвитком інформаційних технологій виникла потреба у нових підходах до організації даних, що привело до появи сучасних моделей даних, зокрема NoSQL.

NoSQL-бази даних орієнтовані на роботу з великими обсягами даних, що можуть бути неструктурованими або слабко структурованими. Вони відрізняються від реляційних систем гнучкою схемою, високою масштабованістю та орієнтацією на розподілені обчислювальні середовища.

Однією з найбільш поширених моделей NoSQL є документоорієнтована модель. У цій моделі дані зберігаються у вигляді документів, зазвичай у форматі JSON або подібному до нього. Документи можуть мати різну структуру, що забезпечує гнучкість у роботі з даними. Прикладом такої системи є MongoDB, яка широко використовується у вебзастосунках.

Іншою важливою моделлю є колонкова модель даних. У ній дані організовані у вигляді колонкових сімейств, що дає змогу ефективно зберігати та обробляти великі обсяги інформації. Колонкові бази даних оптимізовані для аналітичних запитів і обробки великих масивів даних. Прикладом є Apache Cassandra, яка забезпечує високу масштабованість і відмовостійкість.

Графова модель даних використовується для представлення даних у вигляді вузлів і зв'язків між ними. Вона є особливо ефективною для задач, де важливими є взаємозв'язки між об'єктами, наприклад у соціальних мережах або системах рекомендацій. Прикладом графової бази даних є Neo4j.

Основні принципи організації NoSQL баз даних відрізняються від традиційних реляційних підходів. Одним із ключових принципів є відмова від жорсткої схеми даних. Це означає, що структура даних може змінюватися без необхідності модифікації всієї бази даних, що забезпечує гнучкість і швидкість розробки.

Іншим важливим принципом є горизонтальне масштабування. NoSQL-системи орієнтовані на розподілену архітектуру, де дані зберігаються на багатьох вузлах. Це дає змогу обробляти великі обсяги даних і забезпечувати високу продуктивність.

Також важливим є принцип відмовостійкості. Дані у NoSQL-системах часто реплікуються між кількома вузлами, що дає змогу забезпечити їх доступність навіть у випадку відмови окремих компонентів системи.

У багатьох NoSQL-системах використовується принцип, що пов'язаний із компромісом між узгодженістю, доступністю та стійкістю до розділення мережі. Це означає, що система може гарантувати лише частину цих властивостей одночасно, залежно від вимог конкретного застосування.

Таким чином, реляційна алгебра є теоретичною основою для обробки даних у реляційних системах, тоді як сучасні моделі NoSQL забезпечують нові можливості для роботи з великими та складними наборами даних. Розуміння як класичних, так і сучасних підходів до організації даних є необхідним для ефективного проєктування інформаційних систем.

Тема 2. Проєктування баз даних

Лекція 6. Аналіз предметної області та побудова концептуальної моделі даних

Проєктування бази даних починається з аналізу предметної області, який є одним із найважливіших етапів розробки інформаційної системи. Предметна область є частиною реального світу або діяльності, для якої створюється база даних. Вона містить об’єкти, процеси, правила та взаємозв’язки, що підлягають формалізації та подальшій реалізації у вигляді структури даних.

Аналіз предметної області спрямований на виявлення суттєвих характеристик системи, визначення її меж, а також встановлення основних інформаційних потреб користувачів. На цьому етапі формується розуміння того, які дані повинні зберігатися, як вони пов’язані між собою і які операції над ними будуть виконуватися. Важливою умовою якісного аналізу є тісна взаємодія з експертами предметної області, користувачами системи та замовниками.

Процес аналізу зазвичай починається зі збору вимог до бази даних. Вимоги визначають, які функції повинна виконувати система, які дані необхідно зберігати та які обмеження повинні бути враховані. Збір вимог може здійснюватися різними методами, зокрема шляхом інтерв’ювання користувачів, аналізу існуючих документів, спостереження за роботою системи або використання анкетування.

Вимоги до бази даних поділяються на функціональні та нефункціональні. Функціональні вимоги визначають, які саме операції повинні виконуватися системою, наприклад, збереження інформації про студентів, облік оцінок, формування звітів. Нефункціональні вимоги описують характеристики системи, такі як продуктивність, надійність, безпека, масштабованість.

Після збору вимог здійснюється їх формалізація. Це означає перетворення неструктурованої інформації, отриманої від користувачів, у чітко визначені специфікації. Формулювання вимог повинно бути однозначним, повним і несуперечливим, оскільки від цього залежить якість подальшого проєктування бази даних.

Наступним етапом є визначення інформаційних об’єктів, які відображають сутності предметної області. Інформаційний об’єкт є узагальненим представленням реального об’єкта або явища, що має значення для системи. Наприклад, у системі обліку навчального процесу такими об’єктами можуть бути студент, викладач, дисципліна, група.

Для кожного інформаційного об’єкта визначаються атрибути, які описують його властивості. Атрибути повинні бути вибрані таким чином, щоб повністю характеризувати об’єкт і водночас не містити надлишкової інформації. Наприклад, для об’єкта студент атрибутами можуть бути прізвище, ім’я, дата народження, номер студентського квитка.

Важливим аспектом є визначення зв’язків між інформаційними об’єктами. Зв’язки відображають взаємодію між сутностями і є основою для побудови структури бази даних. Наприклад, між об’єктами студент і дисципліна може існувати зв’язок «вивчає», а між викладачем і дисципліною – зв’язок «викладає». Зв’язки характеризуються кардинальністю, яка визначає кількість екземплярів сутностей, що можуть брати участь у зв’язку.

Після визначення об’єктів, атрибутів і зв’язків здійснюється побудова концептуальної моделі даних. Концептуальна модель є абстрактним представленням структури даних, що не

залежить від конкретної системи керування базами даних. Вона відображає логічну організацію даних і є основою для подальшого логічного та фізичного проєктування.

Найбільш поширеним засобом побудови концептуальної моделі є використання ER-моделі (Entity-Relationship), яка дає змогу наочно представити сутності, їх атрибути та зв'язки між ними. У такій моделі сутності зазвичай зображуються у вигляді прямокутників, атрибути – у вигляді овалів, а зв'язки – у вигляді ромбів.

Побудова концептуальної моделі передбачає кілька етапів. Спочатку визначаються основні сутності предметної області. Далі для кожної сутності визначаються атрибути та ключові поля, що дають змогу однозначно ідентифікувати об'єкти. Після цього встановлюються зв'язки між сутностями та визначається їх кардинальність.

На цьому етапі також важливо врахувати обмеження, що накладаються на дані. Це можуть бути обмеження унікальності, обов'язковості заповнення атрибутів, допустимих значень та інші правила, що визначають коректність даних.

Концептуальна модель повинна бути зрозумілою як для розробників, так і для представників предметної області. Вона використовується як засіб комунікації між учасниками проєкту і дає змогу узгодити структуру даних до початку реалізації системи.

Після побудови концептуальної моделі здійснюється її перевірка та валідація. Це означає перевірку відповідності моделі вимогам користувачів, а також виявлення можливих помилок або суперечностей. У разі необхідності модель коригується і уточнюється.

Таким чином, аналіз предметної області та побудова концептуальної моделі даних є основою для створення ефективної бази даних. Цей етап дає змогу правильно визначити структуру даних, уникнути помилок на ранніх стадіях розробки та забезпечити відповідність системи потребам користувачів. Якісно виконаний аналіз є запорукою успішної реалізації інформаційної системи та її подальшого розвитку.

Лекція 7. ER-моделі та побудова ER-діаграм

ER-модель є одним із основних інструментів концептуального проєктування баз даних. Вона дає змогу формалізувати структуру предметної області у вигляді наочної моделі, що відображає сутності, їх атрибути та зв'язки між ними. ER-модель (Entity-Relationship) використовується на етапі концептуального проєктування і є незалежною від конкретної системи керування базами даних.

Основна мета ER-моделі полягає у створенні логічного опису даних, який є зрозумілим як для розробників, так і для представників предметної області. Це дає змогу узгодити вимоги до бази даних ще до етапу її реалізації, що знижує ймовірність помилок і спрощує подальшу розробку.

Основними елементами ER-моделі є сутності, атрибути та зв'язки. Сутність є об'єктом предметної області, який має самостійне значення і може бути ідентифікований. Наприклад, у системі обліку навчального процесу сутностями можуть бути студент, викладач, дисципліна.

Атрибут є властивістю сутності, що описує її характеристики. Атрибути визначають, які саме дані зберігаються про об'єкт. Наприклад, для сутності студент атрибутами можуть бути прізвище, ім'я, дата народження. Атрибути можуть бути простими або складеними, а також можуть мати обмеження щодо допустимих значень.

Зв’язок є логічним відношенням між сутностями, що відображає їх взаємодію у предметній області. Наприклад, між сутностями студент і дисципліна може існувати зв’язок «вивчає». Зв’язки є ключовим елементом ER-моделі, оскільки саме вони визначають структуру даних і забезпечують їх узгодженість.

У графічному представленні ER-діаграм сутності зазвичай зображуються у вигляді прямокутників, атрибути – у вигляді овалів, а зв’язки – у вигляді ромбів. Така форма подання дає змогу наочно відобразити структуру бази даних і спростити її аналіз.

Типи зв’язків між сутностями визначають характер взаємодії між об’єктами. Виділяють три основні типи зв’язків: один-до-одного, один-до-багатьох та багато-до-багатьох.

Зв’язок один-до-одного означає, що кожному екземпляру однієї сутності відповідає не більше одного екземпляра іншої сутності. Такий тип зв’язку використовується відносно рідко, наприклад, для зв’язку між сутністю користувач і його профіль.

Зв’язок один-до-багатьох є найбільш поширеним. Він означає, що одному екземпляру сутності може відповідати кілька екземплярів іншої сутності. Наприклад, один викладач може викладати кілька дисциплін, але кожна дисципліна має одного викладача.

Зв’язок багато-до-багатьох означає, що кілька екземплярів однієї сутності можуть бути пов’язані з кількома екземплярами іншої сутності. Наприклад, студенти можуть вивчати кілька дисциплін, а кожна дисципліна може вивчатися багатьма студентами. У реляційних базах даних такий зв’язок зазвичай реалізується через додаткову проміжну таблицю.

Кардинальність зв’язків визначає кількість екземплярів сутностей, що можуть брати участь у зв’язку. Вона є важливою характеристикою, оскільки впливає на структуру бази даних і правила її реалізації. Кардинальність задається у вигляді мінімальної та максимальної кількості зв’язаних екземплярів.

Наприклад, у зв’язку між студентом і дисципліною може бути визначено, що студент повинен вивчати щонайменше одну дисципліну, але може вивчати багато дисциплін. У такому випадку кардинальність буде від одного до багатьох. Аналогічно визначаються обмеження для інших типів зв’язків.

Побудова ER-діаграми є процесом, що передбачає послідовне виконання кількох етапів. Спочатку визначаються сутності предметної області на основі зібраних вимог. Далі для кожної сутності визначаються атрибути, що описують її характеристики.

Після цього встановлюються зв’язки між сутностями та визначається їх кардинальність. На цьому етапі також визначаються ключові атрибути, що дають змогу однозначно ідентифікувати екземпляри сутностей.

Наступним кроком є графічне представлення моделі у вигляді ER-діаграми. При цьому важливо забезпечити її зрозумілість, уникати надлишкових елементів і чітко відобразити всі сутності та зв’язки.

Після побудови діаграми виконується її перевірка та уточнення. Необхідно переконатися, що модель відповідає вимогам предметної області, не містить суперечностей і повністю описує необхідні дані.

ER-модель може бути розширена за рахунок введення додаткових елементів, таких як слабкі сутності, багатозначні атрибути, ієрархії узагальнення та спеціалізації. Це дає змогу більш точно відобразити складні структури даних.

Таким чином, ER-модель є ефективним інструментом концептуального проєктування баз даних. Вона дає змогу формалізувати структуру предметної області, забезпечити узгодженість вимог і створити основу для подальшого переходу до логічної моделі даних. Правильна побудова ER-діаграм є ключовим етапом у розробці якісної та ефективної бази даних.

Лекція 8. Логічне проєктування баз даних та перехід до реляційної схеми

Логічне проєктування баз даних є етапом, на якому результати концептуального моделювання трансформуються у формалізовану структуру, що відповідає обраній моделі даних, як правило реляційній. На цьому етапі визначається склад таблиць, їх атрибути, ключі, зв'язки та обмеження, що забезпечують цілісність даних. Результатом логічного проєктування є реляційна схема бази даних, яка може бути безпосередньо реалізована у системах керування базами даних.

Перехід від концептуальної моделі до реляційної схеми базується на формальних правилах відображення елементів ER-моделі у реляційні структури. Основними елементами, що підлягають трансформації, є сутності, атрибути та зв'язки. Кожна сутність, визначена на концептуальному рівні, перетворюється у відношення, тобто таблицю. Атрибути сутності стають атрибутами відношення, а екземпляри сутності відповідають кортежам таблиці.

На початковому етапі виконується перетворення сильних сутностей. Для кожної сильної сутності створюється окреме відношення, яке містить усі її атрибути. Одним із ключових завдань є визначення первинного ключа, що забезпечує унікальність кожного кортежу. Первинний ключ повинен бути мінімальним, стабільним і не містити надлишкових атрибутів. Якщо природний ключ відсутній або є незручним у використанні, вводиться штучний ідентифікатор.

Слабкі сутності також відображаються у вигляді окремих відношень, проте їх первинний ключ формується з урахуванням зовнішнього ключа, що посилається на пов'язану сильну сутність. Це відображає залежність слабкої сутності та забезпечує коректність зв'язків між даними.

Важливим аспектом логічного проєктування є перетворення зв'язків між сутностями. Тип зв'язку визначає спосіб його реалізації у реляційній схемі. Для зв'язку один-до-одного можливі кілька підходів: використання зовнішнього ключа в одній із таблиць або об'єднання двох сутностей в одну таблицю. Вибір залежить від обмежень предметної області та частоти використання даних.

Зв'язок один-до-багатьох реалізується шляхом додавання зовнішнього ключа у відношення, що відповідає стороні «багатьох». Такий підхід є найбільш поширеним і дозволяє ефективно відобразити ієрархічні залежності між об'єктами. Зовнішній ключ посилається на первинний ключ іншого відношення, забезпечуючи логічну зв'язність даних.

Зв'язок багато-до-багатьох не може бути безпосередньо реалізований у реляційній моделі, тому для його відображення створюється окреме відношення, яке містить зовнішні ключі обох пов'язаних сутностей. Таке відношення називається асоціативним і може містити додаткові атрибути, що описують сам зв'язок.

Після визначення структури відношень виконується встановлення ключів. Первинний ключ є основним засобом ідентифікації кортежів. Крім нього, можуть використовуватися альтернативні ключі, які також забезпечують унікальність, але не обрані як основні. У

випадках, коли жоден окремий атрибут не забезпечує унікальність, використовуються складені ключі, що складаються з кількох атрибутів.

Зовнішні ключі відіграють важливу роль у забезпеченні зв'язків між відношеннями. Вони визначають залежності між таблицями і використовуються для реалізації зв'язків, встановлених на концептуальному рівні. Система керування базами даних контролює коректність зовнішніх ключів, що дає змогу забезпечити посилальну цілісність.

На наступному етапі визначаються обмеження цілісності. Обмеження цілісності сутності вимагає, щоб значення первинного ключа були унікальними і не містили невизначених значень. Це забезпечує можливість однозначної ідентифікації кожного запису.

Обмеження посилальної цілісності гарантує, що значення зовнішнього ключа відповідають існуючим значенням первинного ключа у пов'язаному відношенні. У разі порушення цього правила система не дозволяє виконання операції або застосовує визначену стратегію, наприклад заборону видалення або каскадне оновлення.

Доменні обмеження визначають допустимі значення атрибутів. Вони задають типи даних, діапазони значень, формат і інші характеристики. Наприклад, для атрибута, що містить оцінку, може бути визначено допустимий діапазон значень. Такі обмеження забезпечують коректність введених даних і підвищують якість інформації.

Особливу увагу приділяють залежностям між атрибутами. Найбільш важливими є функціональні залежності, які визначають, що значення одного атрибута або групи атрибутів однозначно визначає значення іншого атрибута. Наприклад, ідентифікатор студента однозначно визначає його персональні дані.

Аналіз функціональних залежностей є основою для нормалізації бази даних. Він дає змогу виявити надлишковість і усунути аномалії, що можуть виникати при вставці, оновленні або видаленні даних. Нормалізація спрямована на побудову такої структури даних, яка забезпечує мінімальну надлишковість і максимальну узгодженість.

Водночас під час логічного проєктування необхідно враховувати вимоги до продуктивності. Надмірна нормалізація може призвести до збільшення кількості операцій з'єднання, що негативно впливає на швидкодію системи. Тому важливо знайти баланс між структурною правильністю і ефективністю обробки даних.

У результаті логічного проєктування формується реляційна схема бази даних, яка містить повний опис відношень, їх атрибутів, ключів, обмежень і зв'язків. Ця схема є основою для подальшого фізичного проєктування і реалізації бази даних у конкретній системі.

Таким чином, логічне проєктування баз даних є ключовим етапом, що забезпечує перехід від абстрактної концептуальної моделі до формалізованої структури даних. Правильне визначення таблиць, ключів, обмежень і залежностей дає змогу створити ефективну, узгоджену та масштабовану базу даних, що відповідає вимогам предметної області та забезпечує надійну роботу інформаційної системи.

Лекція 9. Нормалізація баз даних та нормальні форми

Нормалізація баз даних є одним із ключових етапів логічного проєктування, що спрямований на усунення надлишковості даних і забезпечення їх логічної узгодженості. Основною метою нормалізації є побудова такої структури відношень, яка мінімізує

дублювання інформації та запобігає виникненню аномалій при вставці, оновленні та видаленні даних.

Нормалізація базується на аналізі функціональних залежностей між атрибутами. Функціональна залежність визначає відношення між атрибутами, за якого значення одного атрибута або їх сукупності однозначно визначає значення іншого атрибута. Формально це записується як $X \rightarrow Y$, де X є детермінантом, а Y – залежним атрибутом. Наприклад, ідентифікатор студента однозначно визначає його прізвище, ім'я та інші персональні дані.

Функціональні залежності відіграють важливу роль у виявленні надлишковості даних. Якщо деякі атрибути залежать від частини ключа або від інших неключових атрибутів, це свідчить про наявність структурних проблем у базі даних. Саме такі ситуації призводять до аномалій і потребують нормалізації.

Перша нормальна форма (1НФ) є базовою вимогою до реляційних відношень. Вона вимагає, щоб усі атрибути містили атомарні значення, тобто кожне поле таблиці повинно містити лише одне значення, а не множину значень. Крім того, у відношенні не повинно бути повторюваних груп атрибутів. Це означає, що структура таблиці повинна бути простою і не містити вкладених або складених структур.

До приведення у першу нормальну форму таблиця може містити, наприклад, список телефонів у одному полі або повторювані стовпці. У процесі нормалізації такі дані розділяються на окремі записи або виносяться в окремі таблиці. Це забезпечує однорідність структури і спрощує обробку даних.

Друга нормальна форма (2НФ) застосовується до відношень, що мають складений первинний ключ. Вона вимагає, щоб кожен неключовий атрибут повністю залежав від усього первинного ключа, а не від його частини. Якщо деякий атрибут залежить лише від частини ключа, виникає часткова залежність, яка є ознакою ненормалізованої структури.

Для усунення часткових залежностей виконується декомпозиція відношення на кілька таблиць. При цьому кожна таблиця повинна містити атрибути, що повністю залежать від свого ключа. Це дозволяє уникнути дублювання даних і забезпечує більш логічну структуру бази даних.

Третя нормальна форма (3НФ) вимагає відсутності транзитивних залежностей між атрибутами. Транзитивна залежність виникає тоді, коли один неключовий атрибут залежить від іншого неключового атрибута, який, у свою чергу, залежить від первинного ключа. У такому випадку структура таблиці містить надлишкову інформацію.

Для приведення до третьої нормальної форми необхідно виділити транзитивно залежні атрибути в окремі відношення. Це дозволяє усунути дублювання даних і підвищити їх узгодженість. У результаті кожен неключовий атрибут залежить тільки від первинного ключа і не залежить від інших неключових атрибутів.

Нормальна форма Бойса-Кодда (BCNF) є більш строгим варіантом третьої нормальної форми. Вона вимагає, щоб для кожної функціональної залежності $X \rightarrow Y$ детермінант X був суперключем. Це означає, що будь-який атрибут або набір атрибутів, що визначає інші атрибути, повинен однозначно ідентифікувати записи у таблиці.

BCNF усуває деякі аномалії, які можуть залишатися навіть після приведення до третьої нормальної форми. Вона забезпечує більш строгі вимоги до структури бази даних, але може призводити до більш складної декомпозиції і збільшення кількості таблиць.

Процес нормалізації виконується послідовно: спочатку відношення приводиться до першої нормальної форми, потім до другої, третьої і, за необхідності, до BCNF. На кожному етапі виконується аналіз функціональних залежностей і декомпозиція відношень.

Однією з основних переваг нормалізації є усунення надлишковості даних. У ненормалізованих структурах одна і та ж інформація може зберігатися у кількох місцях, що призводить до збільшення обсягу даних і складності їх обробки. Нормалізація дозволяє зберігати кожен факт лише один раз.

Ще однією важливою перевагою є усунення аномалій. Аномалії вставки виникають, коли неможливо додати нові дані без внесення додаткової інформації. Аномалії оновлення виникають, коли зміна одного значення потребує змін у кількох місцях. Аномалії видалення виникають, коли видалення одного запису призводить до втрати важливої інформації. Нормалізація дозволяє уникнути таких проблем.

Нормалізація також забезпечує підвищення узгодженості даних. Оскільки кожен факт зберігається лише в одному місці, зменшується ризик виникнення суперечностей. Це особливо важливо для великих інформаційних систем, де дані активно змінюються.

Водночас нормалізація має і певні обмеження. Надмірна нормалізація може призвести до збільшення кількості таблиць і ускладнення запитів. Це може негативно вплинути на продуктивність системи, оскільки для отримання даних необхідно виконувати більше операцій з’єднання.

Тому у практиці проєктування баз даних часто використовується компромісний підхід, який поєднує нормалізацію і денормалізацію. У деяких випадках допускається свідоме введення надлишковості для підвищення продуктивності, але при цьому необхідно забезпечити контроль узгодженості даних.

Таким чином, нормалізація є важливим інструментом проєктування реляційних баз даних, що дозволяє створити логічно узгоджену структуру, мінімізувати надлишковість і забезпечити ефективну обробку даних. Розуміння нормальних форм і функціональних залежностей є необхідною умовою для розробки якісних інформаційних систем.

Лекція 10. Забезпечення цілісності даних у реляційних базах даних

Цілісність даних є однією з ключових характеристик будь-якої бази даних і визначає ступінь відповідності даних реальному стану предметної області. Забезпечення цілісності означає підтримку правильності, узгодженості та достовірності інформації на всіх етапах її зберігання та обробки. У реляційних базах даних цілісність досягається за рахунок використання системи обмежень і правил, які контролюються системою керування базами даних.

Основою забезпечення цілісності є обмеження цілісності, які визначають допустимі стани даних. Обмеження встановлюються на рівні структури бази даних і автоматично перевіряються під час виконання операцій вставки, оновлення та видалення даних. Це дає змогу запобігти появі некоректної або суперечливої інформації.

Одним із базових типів є обмеження цілісності сутності. Воно вимагає, щоб кожне відношення мало первинний ключ, значення якого є унікальними і не містять невизначених значень. Первинний ключ забезпечує однозначну ідентифікацію кожного запису у таблиці і є фундаментом для побудови зв’язків між таблицями.

Первинний ключ може складатися з одного або кількох атрибутів. У випадку складеного ключа унікальність забезпечується комбінацією значень атрибутів. Важливою вимогою є мінімальність ключа, тобто він не повинен містити зайвих атрибутів.

Іншим важливим типом обмежень є обмеження посилальної цілісності. Воно визначає правила взаємодії між таблицями, що пов'язані через зовнішні ключі. Зовнішній ключ є атрибутом або набором атрибутів, що посилається на первинний ключ іншого відношення. Це дає змогу забезпечити зв'язність даних і підтримувати логічні залежності між ними.

Обмеження посилальної цілісності вимагає, щоб кожне значення зовнішнього ключа відповідало існуючому значенню первинного ключа у пов'язаній таблиці або було відсутнім, якщо це допускається. Якщо ця умова порушується, система не дозволяє виконання операції або застосовує визначену політику обробки.

При видаленні або оновленні записів, на які посилаються зовнішні ключі, можуть застосовуватися різні правила. Одним із варіантів є заборона операції, якщо існують залежні записи. Іншим варіантом є каскадне оновлення або видалення, при якому зміни автоматично поширюються на пов'язані таблиці. Також можливе встановлення значення зовнішнього ключа у невизначений стан.

Крім обмежень сутності та посилальної цілісності, важливу роль відіграють доменні обмеження. Вони визначають допустимі значення атрибутів, включаючи типи даних, діапазони значень, формати та інші правила. Наприклад, для атрибута, що містить дату, може бути встановлено обмеження на допустимий інтервал значень.

Доменні обмеження забезпечують коректність даних на рівні окремих атрибутів і запобігають введенню некоректної інформації. Вони є важливою складовою загальної системи забезпечення цілісності.

Ще одним типом обмежень є обмеження користувача або бізнес-правила. Вони визначають специфічні умови, що випливають із предметної області. Наприклад, може бути встановлено правило, що оцінка студента повинна знаходитися у певному діапазоні або що кількість кредитів дисципліни не може бути від'ємною. Такі обмеження реалізуються за допомогою механізмів СКБД, зокрема перевірок або тригерів.

Важливим аспектом забезпечення цілісності є використання транзакцій. Транзакція є логічно завершеною послідовністю операцій над даними, яка виконується як єдине ціле. Вона забезпечує узгодженість даних навіть у випадку помилок або збоїв.

Транзакції характеризуються властивостями атомарності, узгодженості, ізольованості та довговічності. Атомарність означає, що всі операції транзакції виконуються повністю або не виконуються зовсім. Узгодженість забезпечує перехід бази даних з одного коректного стану в інший. Ізольованість гарантує, що паралельні транзакції не впливають одна на одну. Довговічність означає, що результати виконаних транзакцій зберігаються навіть у разі збоїв.

Первинні та зовнішні ключі є центральними елементами механізму забезпечення цілісності. Первинний ключ гарантує унікальність записів, а зовнішній ключ – узгодженість зв'язків між таблицями. Разом вони формують основу логічної структури бази даних.

Правила підтримки цілісності даних реалізуються автоматично системою керування базами даних. Вони перевіряються при кожній операції, що змінює стан бази даних. Це дозволяє забезпечити постійний контроль за правильністю даних без участі користувача.

У сучасних системах керування базами даних, таких як PostgreSQL або Oracle Database, реалізовано широкий набір засобів для забезпечення цілісності, включаючи обмеження, індекси, тригери та механізми транзакцій.

Водночас важливо розуміти, що забезпечення цілісності є спільним завданням як СКБД, так і розробника бази даних. Неправильне визначення структури або обмежень може призвести до появи помилок, які буде складно виправити на пізніх етапах.

Таким чином, забезпечення цілісності даних є критично важливим аспектом проєктування та експлуатації реляційних баз даних. Використання обмежень, ключів і транзакцій дозволяє підтримувати правильність і узгодженість інформації, що є необхідною умовою ефективної роботи інформаційних систем.

Лекція 11. Фізичне проєктування баз даних та оптимізація структури

Фізичне проєктування баз даних є завершальним етапом розробки, на якому логічна схема перетворюється у конкретну реалізацію в межах обраної системи керування базами даних. На цьому рівні визначаються способи зберігання даних, організація файлів, використання індексів, параметри доступу та інші аспекти, що безпосередньо впливають на продуктивність системи. Якщо логічне проєктування відповідає на питання «що зберігати», то фізичне – «як саме це зберігати».

Фізичне проєктування орієнтоване на ефективне використання ресурсів обчислювальної системи, зокрема оперативної пам’яті, дискового простору та процесорного часу. Воно враховує особливості конкретної СКБД, наприклад PostgreSQL або MySQL, а також характер запитів, що будуть виконуватися над базою даних.

Одним із ключових аспектів фізичного проєктування є вибір структур зберігання даних. Дані у базі даних зберігаються у вигляді файлів, які організовані у сторінки або блоки фіксованого розміру. Сторінка є мінімальною одиницею введення-виведення і містить один або кілька записів. Ефективність доступу до даних значною мірою залежить від того, як ці сторінки організовані на фізичному носії.

Існують різні способи організації файлів даних. Найпростішим є послідовне зберігання, при якому записи розташовуються один за одним у порядку їх додавання. Такий підхід є ефективним для операцій читання великих обсягів даних, але не забезпечує швидкого пошуку окремих записів.

Більш ефективним є використання індексованих структур, які дозволяють швидко знаходити необхідні дані. Індекси є допоміжними структурами, що містять значення атрибутів і посилання на відповідні записи у таблиці. Вони значно прискорюють виконання запитів, особливо тих, що містять умови пошуку.

Найбільш поширеним типом індексу є B-дерево, яке забезпечує логарифмічний час доступу до даних. B-дерева ефективно використовуються для реалізації індексів у більшості реляційних СКБД. Іншим типом є хеш-індекси, які забезпечують дуже швидкий доступ за точним значенням, але менш ефективні для діапазонних запитів.

Використання індексів має як переваги, так і обмеження. З одного боку, вони значно підвищують швидкість виконання запитів. З іншого боку, індекси потребують додаткового місця і збільшують час виконання операцій вставки, оновлення та видалення, оскільки їх необхідно підтримувати в актуальному стані.

Оптимізація структури бази даних полягає у виборі таких фізичних параметрів, які забезпечують максимальну продуктивність системи. До основних методів оптимізації належать вибір відповідних типів даних, створення індексів для часто використовуваних атрибутів, а також організація таблиць з урахуванням характеру запитів.

Важливим аспектом є також денормалізація, яка полягає у свідомому введенні надлишковості даних для зменшення кількості операцій з’єднання. Хоча денормалізація

суперечить принципам нормалізації, вона може бути виправданою у випадках, коли продуктивність є критичним фактором.

Для підвищення ефективності роботи бази даних використовуються також механізми кешування, які дозволяють зберігати часто використовувані дані в оперативній пам’яті. Це зменшує кількість операцій введення-виведення і прискорює виконання запитів.

Ще одним важливим напрямом є оптимізація запитів. Сучасні СКБД містять оптимізатори запитів, які автоматично визначають найефективніший спосіб їх виконання. Проте правильна структура бази даних значною мірою впливає на ефективність роботи оптимізатора.

У сучасних інформаційних системах важливу роль відіграють розподілені бази даних. Вони передбачають зберігання даних на кількох вузлах мережі, що дає змогу забезпечити масштабованість, відмовостійкість і високу продуктивність.

Основними принципами проєктування розподілених баз даних є фрагментація та реплікація даних. Фрагментація полягає у розподілі даних між вузлами за певними правилами. Вона може бути горизонтальною, коли різні записи однієї таблиці зберігаються на різних вузлах, або вертикальною, коли атрибути розподіляються між вузлами.

Реплікація полягає у створенні копій даних на кількох вузлах. Це дозволяє підвищити доступність даних і забезпечити їх збереження у випадку відмови окремих компонентів системи. Проте реплікація ускладнює забезпечення узгодженості даних.

Важливим аспектом розподілених систем є забезпечення узгодженості даних. У таких системах часто виникає компроміс між узгодженістю, доступністю та стійкістю до розділення мережі. Вибір конкретного підходу залежить від вимог до системи.

Фізичне проєктування розподілених баз даних також враховує розташування вузлів, характеристики мережі та типи запитів. Це дозволяє оптимізувати передачу даних і мінімізувати затримки.

Таким чином, фізичне проєктування баз даних є важливим етапом, що забезпечує ефективну реалізацію логічної структури. Вибір структур зберігання, використання індексів, оптимізація та врахування розподіленої архітектури дозволяють створити високопродуктивну та надійну базу даних, що відповідає сучасним вимогам інформаційних систем.

ТЕМА 3. ЕКСПЛУАТАЦІЯ БАЗ ДАНИХ

Лекція 12. Мова SQL та оператори визначення даних

Мова SQL є стандартною мовою для роботи з реляційними базами даних і використовується для створення, модифікації та обробки даних. Вона є декларативною мовою, що означає, що користувач описує, який результат необхідно отримати, а не спосіб його досягнення. SQL використовується у більшості сучасних систем керування базами даних, зокрема PostgreSQL, MySQL та Oracle Database.

Основне призначення SQL полягає у забезпеченні взаємодії користувача або прикладної програми з базою даних. За допомогою SQL можна створювати структуру бази даних, виконувати запити, змінювати дані, а також керувати доступом до них. Це робить SQL універсальним інструментом для роботи з інформаційними системами.

Структура SQL поділяється на кілька основних підмов, кожна з яких відповідає за окремий аспект роботи з базою даних. До них належать мова визначення даних, мова маніпулювання даними, мова керування доступом та мова керування транзакціями. У межах цієї лекції основна увага приділяється мові визначення даних.

Мова визначення даних (DDL – Data Definition Language) використовується для створення і зміни структури бази даних. Вона дозволяє визначати таблиці, їх атрибути, типи даних, ключі та обмеження цілісності. Основними операторами DDL є CREATE, ALTER та DROP.

Оператор CREATE використовується для створення нових об’єктів бази даних, зокрема таблиць, індексів і схем. Найбільш поширеним є створення таблиць, яке передбачає визначення їх структури. Під час створення таблиці задається її ім’я, перелік атрибутів і типи даних для кожного атрибута.

Кожен атрибут таблиці повинен мати визначений тип даних, який визначає множину допустимих значень. Типи даних можуть бути числовими, текстовими, логічними, датами та іншими. Вибір типу даних впливає на обсяг пам’яті, швидкість обробки та можливість виконання операцій над даними.

Крім визначення атрибутів, під час створення таблиці встановлюються обмеження цілісності. Одним із основних є обмеження PRIMARY KEY, яке визначає первинний ключ таблиці. Це обмеження гарантує унікальність значень і відсутність невизначених значень у відповідному атрибуті.

Іншим важливим обмеженням є FOREIGN KEY, яке визначає зовнішній ключ і забезпечує зв’язок між таблицями. Воно гарантує, що значення атрибута відповідають існуючим записам у пов’язаній таблиці.

Також широко використовуються обмеження NOT NULL, яке забороняє відсутність значення у полі, та UNIQUE, яке забезпечує унікальність значень атрибута. Обмеження CHECK дозволяє визначати додаткові умови, яким повинні відповідати значення атрибутів.

Під час створення таблиці також можна задавати значення за замовчуванням для атрибутів за допомогою конструкції DEFAULT. Це дозволяє автоматично заповнювати поля у випадку, якщо значення не було явно задане.

Оператор ALTER використовується для зміни структури вже існуючих таблиць. За його допомогою можна додавати нові атрибути, змінювати їх типи або видаляти непотрібні

елементи. ALTER дозволяє адаптувати структуру бази даних до нових вимог без необхідності створення нових таблиць.

Оператор DROP використовується для видалення об'єктів бази даних. Наприклад, DROP TABLE дозволяє повністю видалити таблицю разом із усіма її даними. Використання цього оператора потребує обережності, оскільки видалені дані не підлягають відновленню без резервної копії.

При створенні таблиць важливо враховувати вимоги до цілісності даних. Правильне визначення обмежень дозволяє запобігти появі помилок і забезпечує узгодженість інформації. Наприклад, використання зовнішніх ключів гарантує коректність зв'язків між таблицями, а обмеження CHECK дозволяють контролювати допустимі значення.

Структура SQL є стандартизованою, але окремі СКБД можуть мати свої розширення. Наприклад, різні системи можуть по-різному реалізовувати типи даних або додаткові функції. Проте основні принципи залишаються однаковими, що забезпечує переносимість знань і навичок.

Важливим аспектом використання SQL є правильна організація структури бази даних. Неправильно визначені атрибути або відсутність обмежень можуть призвести до появи некоректних даних і ускладнити подальшу обробку інформації.

У сучасних інформаційних системах SQL використовується не лише для створення структури бази даних, але й для інтеграції з прикладними програмами. Запити SQL можуть бути вбудовані у програмний код, що дозволяє автоматизувати роботу з даними.

Таким чином, мова SQL є основним інструментом роботи з реляційними базами даних. Оператори визначення даних дозволяють створювати і змінювати структуру бази даних, визначати атрибути і обмеження цілісності. Правильне використання SQL забезпечує надійність, узгодженість і ефективність роботи інформаційних систем.

Лекція 13. Оператори маніпулювання даними та побудова запитів у SQL

Оператори маніпулювання даними (DML – Data Manipulation Language) призначені для роботи з даними, що вже збережені у базі даних. На відміну від операторів визначення даних, які формують структуру, DML дозволяє виконувати операції додавання, зміни, видалення та вибірки даних. Саме ці оператори використовуються у повсякденній роботі з базами даних і забезпечують взаємодію прикладних програм із даними.

Основними операторами DML є INSERT, UPDATE, DELETE та SELECT. Кожен із них виконує конкретну функцію і використовується залежно від поставленої задачі. Усі ці оператори підтримуються сучасними системами керування базами даних, зокрема PostgreSQL, MySQL та Oracle Database.

Оператор INSERT використовується для додавання нових записів до таблиці. При цьому необхідно вказати ім'я таблиці та значення атрибутів, які потрібно додати. Значення можуть задаватися для всіх атрибутів або лише для деяких із них, якщо для інших визначені значення за замовчуванням. Важливо, щоб типи даних відповідали визначеним у структурі таблиці, а також щоб не порушувалися обмеження цілісності.

Оператор INSERT може використовуватися як для додавання одного запису, так і для вставки результатів запиту з іншої таблиці. Це дозволяє ефективно переносити дані між таблицями та формувати нові набори даних.

Оператор UPDATE використовується для зміни існуючих записів у таблиці. Він дозволяє змінювати значення одного або кількох атрибутів для записів, що задовольняють задану умову. Умова визначається за допомогою конструкції WHERE. Якщо умова не задана, зміни застосовуються до всіх записів таблиці, що може призвести до небажаних наслідків.

Оператор UPDATE є важливим інструментом підтримки актуальності даних. Він використовується, наприклад, для зміни персональних даних, оновлення статусів або коригування інформації у системі.

Оператор DELETE призначений для видалення записів із таблиці. Як і у випадку з UPDATE, видалення може виконуватися для всіх записів або лише для тих, що відповідають заданій умові. Використання умови WHERE є критично важливим, оскільки її відсутність призводить до видалення всіх даних із таблиці.

При виконанні операції DELETE необхідно враховувати обмеження посилальної цілісності. Якщо запис пов’язаний із іншими таблицями через зовнішні ключі, система може заборонити його видалення або виконати каскадне видалення залежних записів.

Оператор SELECT є основним засобом отримання даних із бази даних. Він дозволяє формувати запити різної складності, отримуючи дані з однієї або кількох таблиць. У найпростішому випадку SELECT використовується для вибору всіх атрибутів і записів таблиці.

Структура простого запиту SELECT містить перелік атрибутів, які потрібно отримати, ім’я таблиці та, за необхідності, умови відбору. Це дає змогу формувати гнучкі запити і отримувати лише необхідну інформацію.

Важливим елементом запитів є умови відбору, які задаються за допомогою конструкції WHERE. Умови дозволяють обмежити кількість записів, що повертаються запитом. Для формування умов використовуються оператори порівняння, логічні оператори та спеціальні конструкції.

Оператори порівняння дозволяють перевіряти рівність, нерівність, більші або менші значення. Логічні оператори дають змогу комбінувати кілька умов, створюючи складні критерії відбору. Це дозволяє точно визначити, які записи повинні бути включені у результат.

Також використовуються спеціальні оператори, такі як BETWEEN, IN та LIKE. Вони дозволяють виконувати перевірку належності до діапазону, множини значень або шаблону. Це значно розширює можливості формування запитів.

Сортування результатів запиту виконується за допомогою конструкції ORDER BY. Вона дозволяє впорядковувати записи за значеннями одного або кількох атрибутів. За замовчуванням сортування виконується за зростанням, але може бути заданий порядок за спаданням.

Сортування є важливим інструментом для представлення результатів у зручному вигляді. Воно використовується при формуванні звітів, списків або інших результатів, де порядок має значення.

У процесі роботи з DML-операторами важливо враховувати обмеження цілісності. Усі операції повинні відповідати визначеним правилам, інакше система не дозволить їх виконання. Це забезпечує захист даних від некоректних змін.

Крім того, DML-операції виконуються у межах транзакцій, що дозволяє забезпечити узгодженість даних. Якщо під час виконання операції виникає помилка, транзакція може бути скасована, і база даних повертається до попереднього стану.

Таким чином, оператори маніпулювання даними є основним інструментом роботи з інформацією у реляційних базах даних. Вони дозволяють додавати, змінювати, видаляти та отримувати дані, використовуючи гнучкі механізми запитів. Правильне використання DML забезпечує ефективну роботу з базою даних і дозволяє реалізувати широкий спектр задач у інформаційних системах.

Лекція 14. Складні SQL-запити та аналітична обробка даних

Складні SQL-запити є важливим інструментом для отримання, обробки та аналізу даних у реляційних базах даних. Якщо прості запити дозволяють вибрати дані з однієї таблиці, то складні запити дають змогу працювати з кількома таблицями, виконувати обчислення, формувати узагальнену інформацію та отримувати аналітичні результати. Саме такі запити широко використовуються у звітності, бізнес-аналітиці та інформаційних системах.

Основу складних SQL-запитів становить оператор SELECT, який може доповнюватися різними конструкціями, що розширюють його можливості. До таких конструкцій належать з'єднання таблиць, підзапити, агрегатні функції та механізми групування. Усі ці інструменти дозволяють працювати з даними на більш високому рівні абстракції.

Однією з ключових можливостей SQL є використання з'єднань таблиць. З'єднання дозволяє об'єднувати дані з кількох таблиць на основі зв'язків між ними. Це є необхідним у випадках, коли інформація про об'єкти зберігається у різних таблицях, що є типовим для нормалізованих баз даних.

Найбільш поширеним є внутрішнє з'єднання, яке повертає лише ті записи, для яких виконується умова відповідності між таблицями. Наприклад, при об'єднанні таблиць студентів і їх оцінок буде отримано лише ті записи, де існує відповідність за ідентифікатором студента.

Окрім внутрішнього з'єднання, використовуються зовнішні з'єднання, які дозволяють отримати всі записи з однієї таблиці навіть у випадку відсутності відповідних записів в іншій. Це особливо важливо для аналізу неповних даних, наприклад для визначення студентів, які ще не отримали оцінки.

Також застосовуються ліві, праві та повні з'єднання, що відрізняються способом включення записів з відповідних таблиць. Використання правильного типу з'єднання дозволяє точно сформувати результат запиту відповідно до поставленої задачі.

Іншим важливим інструментом є підзапити. Підзапит – це запит, який використовується всередині іншого запиту. Він може застосовуватися для формування умов відбору, обчислення значень або отримання проміжних результатів.

Підзапити можуть бути вкладеними або корельованими. Вкладені підзапити виконуються незалежно від основного запиту і передають свій результат у зовнішній запит. Корельовані підзапити виконуються для кожного рядка основного запиту, що робить їх більш гнучкими, але потенційно менш ефективними з точки зору продуктивності.

Підзапити часто використовуються у конструкціях WHERE та FROM, а також у списку вибраних атрибутів. Вони дозволяють реалізувати складні логічні умови та отримувати дані, які не можуть бути отримані простими запитами.

Особливе місце у складних запитах займають агрегатні функції. Вони дозволяють виконувати обчислення над наборами даних, отримуючи узагальнені значення. До основних агрегатних функцій належать COUNT, SUM, AVG, MIN та MAX.

Агрегатні функції широко використовуються для формування статистичної інформації. Наприклад, можна визначити кількість студентів, середній бал або максимальне значення показника. Це є основою для побудови аналітичних звітів.

Застосування агрегатних функцій часто поєднується з групуванням даних. Групування виконується за допомогою конструкції GROUP BY і дозволяє об'єднувати записи у групи за значеннями певних атрибутів. Для кожної групи обчислюються агрегатні значення.

Наприклад, можна згрупувати студентів за групами і визначити середній бал для кожної групи. Це дозволяє отримати більш структуровану інформацію і виявити закономірності у даних.

Важливим доповненням до групування є конструкція HAVING, яка дозволяє задавати умови для відбору груп. На відміну від WHERE, яка застосовується до окремих записів, HAVING працює з результатами групування. Це дозволяє, наприклад, вибрати лише ті групи, у яких середній бал перевищує певне значення.

Складні SQL-запити також використовуються для побудови аналітичних запитів. Аналітичні запити дозволяють отримувати узагальнену інформацію, виявляти тенденції та виконувати порівняльний аналіз даних.

До аналітичних задач належать обчислення підсумків, визначення рейтингу, аналіз динаміки показників та інші задачі, що потребують обробки великих обсягів даних. Для їх реалізації використовуються поєднання агрегатних функцій, групування, підзапитів і з'єднань таблиць.

У сучасних СКБД аналітичні можливості розширюються за рахунок використання віконних функцій, які дозволяють виконувати обчислення над наборами рядків, пов'язаних із поточним рядком. Це дає змогу реалізувати більш складні аналітичні сценарії без необхідності створення додаткових підзапитів.

Важливим аспектом побудови складних запитів є їх оптимізація. Неправильно сформований запит може призвести до значного зниження продуктивності. Тому необхідно враховувати структуру бази даних, наявність індексів та особливості виконання запитів.

Наприклад, використання з'єднань замість вкладених підзапитів у деяких випадках дозволяє значно підвищити ефективність виконання. Також важливо уникати надлишкових операцій і використовувати лише необхідні атрибути.

Складні SQL-запити є потужним інструментом, що дозволяє працювати з даними на високому рівні. Вони дають змогу не лише отримувати інформацію, але й виконувати її глибокий аналіз.

Таким чином, використання з'єднань, підзапитів, агрегатних функцій і механізмів групування дозволяє будувати ефективні аналітичні запити. Розуміння цих інструментів є необхідною умовою для роботи з сучасними інформаційними системами та забезпечує можливість прийняття обґрунтованих рішень на основі даних.

Лекція 15. Транзакції та керування доступом до даних у реляційних базах даних

У сучасних інформаційних системах бази даних рідко використовуються одним користувачем. Як правило, доступ до даних здійснюється одночасно багатьма користувачами

або прикладними програмами. У таких умовах виникає необхідність забезпечення узгодженості даних, запобігання конфліктам і контролю доступу. Основними механізмами, що вирішують ці задачі, є транзакції та засоби керування доступом до даних.

Транзакція є логічно завершеною послідовністю операцій над базою даних, яка виконується як єдине ціле. Вона може містити одну або кілька команд, наприклад вставку, оновлення або видалення даних. Основною особливістю транзакції є те, що вона або виконується повністю, або не виконується зовсім. Це дозволяє забезпечити коректність стану бази даних навіть у випадку помилок.

Класичним прикладом транзакції є банківська операція переказу коштів, яка складається з двох дій: списання коштів з одного рахунку та зарахування на інший. Якщо одна з цих операцій не буде виконана, система повинна повернутися до початкового стану. Саме це і забезпечують транзакції.

Транзакції характеризуються чотирма основними властивостями, відомими як ACID. Атомарність означає, що всі операції транзакції виконуються як єдине ціле. У випадку помилки жодна з операцій не повинна впливати на базу даних.

Узгодженість гарантує, що після завершення транзакції база даних переходить з одного коректного стану в інший. Це означає, що всі обмеження цілісності повинні залишатися виконаними.

Ізольованість означає, що паралельні транзакції не повинні впливати одна на одну. Кожна транзакція повинна виконуватися так, ніби вона є єдиною у системі. Це запобігає виникненню конфліктів і забезпечує передбачуваність результатів.

Довговічність гарантує, що результати успішно завершеної транзакції зберігаються навіть у випадку збоїв системи. Це досягається за рахунок використання журналів змін і механізмів відновлення.

У багатокористувацьких системах особливо важливим є забезпечення цілісності даних при одночасному доступі. Якщо кілька користувачів одночасно змінюють одні й ті самі дані, можуть виникати конфлікти, що призводять до втрати або спотворення інформації.

Однією з типових проблем є втрата оновлення, коли зміни одного користувача перезаписують зміни іншого. Іншою проблемою є «брудне читання», коли одна транзакція читає дані, які ще не були остаточно зафіксовані іншою транзакцією. Також можливі неповторювані читання та фантомні записи.

Для запобігання таким проблемам використовуються механізми керування конкурентністю. Вони визначають правила взаємодії транзакцій і забезпечують узгодженість даних при паралельному доступі.

Одним із основних механізмів є блокування. Блокування передбачає тимчасове обмеження доступу до певних даних під час виконання транзакції. Це дозволяє уникнути конфліктів між транзакціями.

Існують різні типи блокувань. Найбільш поширеними є блокування для читання та блокування для запису. Блокування для читання дозволяє іншим транзакціям також читати дані, але забороняє їх зміну. Блокування для запису забороняє як читання, так і зміну даних іншими транзакціями.

Механізм блокування реалізується за допомогою протоколів, одним із яких є двофазне блокування. На першій фазі транзакція отримує всі необхідні блокування, а на другій – звільняє їх. Такий підхід забезпечує узгодженість виконання транзакцій.

Водночас використання блокувань може призводити до взаємних блокувань, коли дві або більше транзакцій очікують одна на одну. Для вирішення цієї проблеми використовуються механізми виявлення та усунення взаємних блокувань.

Альтернативним підходом є використання багатоверсійного керування конкурентністю. У цьому випадку для кожного запису може існувати кілька версій, що дозволяє транзакціям працювати з різними станами даних без взаємного блокування. Це підвищує продуктивність системи і зменшує кількість конфліктів.

Керування доступом до даних також передбачає обмеження прав користувачів. У базах даних визначаються ролі та привілеї, які визначають, які операції може виконувати користувач. Це дозволяє захистити дані від несанкціонованого доступу.

Наприклад, одному користувачу може бути дозволено лише читання даних, тоді як інший має право їх змінювати. Такий підхід забезпечує безпеку і контроль за використанням інформації.

У сучасних системах керування базами даних, таких як PostgreSQL або MySQL, реалізовано розвинені механізми підтримки транзакцій і керування доступом. Вони дозволяють ефективно працювати з великими обсягами даних у багатокористувацькому середовищі.

Таким чином, транзакції та механізми керування доступом є невід'ємною складовою сучасних баз даних. Вони забезпечують цілісність, узгодженість і безпеку даних, дозволяючи системі коректно працювати навіть у складних умовах паралельного доступу. Розуміння цих механізмів є необхідним для ефективного проєктування та експлуатації інформаційних систем.

Лекція 16. Захист та адміністрування баз даних

Захист і адміністрування баз даних є невід'ємною складовою експлуатації інформаційних систем і спрямовані на забезпечення безпеки, надійності та стабільності їх функціонування. Якщо проєктування визначає структуру та логіку роботи з даними, то адміністрування забезпечує їх безперервну, безпечну та ефективну роботу у реальних умовах. У сучасних системах, таких як PostgreSQL, MySQL або Oracle Database, ці функції реалізуються за допомогою спеціалізованих механізмів і інструментів.

Одним із ключових напрямів є забезпечення захисту даних. Захист бази даних включає заходи, спрямовані на запобігання несанкціонованому доступу, збереження конфіденційності інформації та забезпечення її цілісності. У сучасних умовах загрози можуть бути як зовнішніми, так і внутрішніми, тому система захисту повинна бути комплексною.

Основним механізмом захисту є керування доступом користувачів. У базах даних створюються облікові записи користувачів, кожному з яких призначаються певні права. Права визначають, які операції користувач може виконувати: читання, додавання, зміна або видалення даних, а також створення чи модифікація об'єктів бази даних.

Для спрощення керування доступом використовуються ролі. Роль є набором прав, який може бути призначений одному або кільком користувачам. Це дозволяє централізовано керувати правами доступу і спрощує адміністрування системи. Наприклад, можуть існувати ролі адміністратора, розробника та користувача з обмеженими правами.

Важливим аспектом є принцип мінімальних привілеїв. Він передбачає, що користувач отримує лише ті права, які необхідні для виконання його завдань. Це зменшує ризик несанкціонованих дій і підвищує рівень безпеки системи.

Крім контролю доступу, застосовуються механізми автентифікації та авторизації. Автентифікація визначає, ким є користувач, тоді як авторизація визначає, які дії він має право виконувати. У сучасних системах можуть використовуватися різні методи автентифікації, включаючи паролі, сертифікати та інтеграцію з зовнішніми службами.

Ще одним важливим напрямом є резервне копіювання баз даних. Резервне копіювання дозволяє створювати копії даних, які можуть бути використані для відновлення у випадку збоїв, помилок або втрати даних. Це є критично важливим для забезпечення надійності системи.

Існують різні типи резервного копіювання. Повне копіювання передбачає створення повної копії всієї бази даних. Інкрементне копіювання зберігає лише ті дані, що були змінені після останнього копіювання. Диференційне копіювання зберігає зміни, що відбулися після останнього повного копіювання.

Вибір стратегії резервного копіювання залежить від вимог до системи, зокрема від допустимого часу відновлення і обсягу даних. Часто використовується комбінований підхід, який поєднує різні типи копіювання для досягнення оптимального результату.

Відновлення бази даних є процесом повернення системи до працездатного стану після збою. Воно може виконуватися на основі резервних копій та журналів транзакцій. Журнали дозволяють відновити дані до конкретного моменту часу, що є важливим у випадках втрати частини інформації.

Важливим аспектом адміністрування є моніторинг роботи системи. Моніторинг дозволяє відстежувати стан бази даних, її продуктивність, використання ресурсів та наявність помилок. Це дає змогу своєчасно виявляти проблеми і запобігати їх розвитку.

Моніторинг включає аналіз навантаження на систему, швидкості виконання запитів, використання процесора, пам’яті та дискових ресурсів. На основі цих даних адміністратор може приймати рішення щодо оптимізації системи.

Підтримка працездатності бази даних передбачає виконання регулярних адміністративних процедур. До них належать оптимізація структури даних, оновлення статистики, перебудова індексів, а також контроль за цілісністю даних.

Також важливим є оновлення програмного забезпечення СКБД. Нові версії можуть містити виправлення помилок, покращення продуктивності та нові функції. Проте оновлення повинно виконуватися обережно, щоб не порушити роботу системи.

Адміністрування баз даних також включає планування ресурсів. Це означає прогнозування зростання обсягів даних, навантаження на систему та необхідності у додаткових ресурсах. Такий підхід дозволяє забезпечити стабільну роботу системи у довгостроковій перспективі.

Таким чином, захист і адміністрування баз даних є комплексною задачею, що включає керування доступом, резервне копіювання, відновлення, моніторинг і підтримку системи. Від правильності виконання цих функцій залежить безпека, надійність і ефективність роботи інформаційних систем. Розуміння принципів адміністрування є необхідним для забезпечення стабільної роботи сучасних баз даних.

ТЕМА 4. СУЧАСНІ НАПРЯМИ РОЗВИТКУ БАЗ ДАНИХ І БАЗ ЗНАНЬ

Лекція 17. Розподілені системи керування базами даних

Розподілені системи керування базами даних є сучасним етапом розвитку інформаційних технологій, що дозволяє ефективно працювати з великими обсягами даних у мережевому середовищі. У таких системах дані зберігаються не на одному сервері, а розподіляються між кількома вузлами, які можуть бути географічно віддаленими. Це дозволяє підвищити масштабованість, продуктивність і відмовостійкість інформаційних систем.

Розподілена база даних є логічно єдиною системою, хоча фізично її компоненти можуть знаходитися на різних серверах. Користувач або прикладна програма взаємодіє з нею як із єдиною базою даних, не враховуючи її фізичну структуру. Це досягається за рахунок спеціалізованих механізмів, що забезпечують прозорість доступу до даних.

Основними принципами побудови розподілених баз даних є прозорість розташування, прозорість фрагментації та прозорість реплікації. Прозорість розташування означає, що користувач не повинен знати, де фізично знаходяться дані. Прозорість фрагментації означає, що дані можуть бути розділені на частини, але система автоматично об’єднує їх під час виконання запитів. Прозорість реплікації забезпечує роботу з копіями даних як із єдиним джерелом.

Важливим аспектом є організація взаємодії між вузлами системи. Кожен вузол може виконувати функції зберігання та обробки даних, а також брати участь у виконанні запитів. Для цього використовуються протоколи обміну даними та механізми координації.

Одним із ключових методів організації розподілених баз даних є фрагментація даних. Фрагментація полягає у розподілі таблиць або їх частин між різними вузлами системи. Це дозволяє зменшити навантаження на окремі сервери і підвищити ефективність обробки запитів.

Існує кілька видів фрагментації. Горизонтальна фрагментація передбачає розподіл рядків таблиці між вузлами. Наприклад, записи можуть розподілятися за географічною ознакою або за діапазоном значень. Вертикальна фрагментація полягає у розподілі атрибутів таблиці між вузлами. У цьому випадку різні частини одного запису зберігаються у різних місцях.

Також можлива комбінована фрагментація, яка поєднує горизонтальний і вертикальний підходи. Вибір методу фрагментації залежить від характеру запитів і структури даних. Правильно виконана фрагментація дозволяє значно підвищити продуктивність системи.

Іншим важливим механізмом є реплікація даних. Реплікація передбачає створення копій даних на кількох вузлах системи. Це дозволяє забезпечити доступність даних навіть у випадку відмови окремих серверів.

Реплікація може бути повною або частковою. У випадку повної реплікації всі дані дублюються на кожному вузлі. Це забезпечує максимальну доступність, але потребує значних ресурсів. Часткова реплікація передбачає копіювання лише частини даних, що дозволяє зменшити витрати ресурсів.

Важливим аспектом реплікації є забезпечення узгодженості даних. При зміні даних на одному вузлі необхідно забезпечити оновлення всіх копій. Це може виконуватися синхронно

або асинхронно. Синхронна реплікація забезпечує високу узгодженість, але може знижувати продуктивність. Асинхронна реплікація є більш продуктивною, але допускає тимчасову неузгодженість.

Розподілені системи керування базами даних широко використовуються у сучасних інформаційних системах, зокрема у хмарних сервісах і великих вебзастосунках. Прикладом таких систем є Apache Cassandra та MongoDB, які орієнтовані на горизонтальне масштабування і роботу з великими обсягами даних.

Перевагами розподіленого зберігання даних є висока масштабованість, можливість обробки великих обсягів інформації та підвищена відмовостійкість. У разі виходу з ладу одного вузла система може продовжувати роботу за рахунок інших вузлів.

Ще однією перевагою є можливість розподілу навантаження між кількома серверами. Це дозволяє підвищити продуктивність і забезпечити швидкий доступ до даних навіть при великій кількості користувачів.

Водночас розподілені системи мають і певні проблеми. Однією з основних є складність забезпечення узгодженості даних. У розподіленому середовищі важко гарантувати, що всі копії даних є однаковими у кожний момент часу.

Також важливою проблемою є затримка передачі даних між вузлами. Це може впливати на швидкість виконання запитів і загальну продуктивність системи.

Ще одним викликом є складність адміністрування. Розподілені системи потребують налаштування взаємодії між вузлами, контролю реплікації та моніторингу стану всієї системи.

Важливим теоретичним аспектом є компроміс між узгодженістю, доступністю та стійкістю до розділення мережі. У розподілених системах неможливо одночасно забезпечити всі ці властивості на максимальному рівні, тому необхідно обирати пріоритети залежно від вимог системи.

Таким чином, розподілені системи керування базами даних є потужним інструментом для роботи з великими обсягами інформації. Вони забезпечують масштабованість і відмовостійкість, але водночас вимагають складних механізмів управління і контролю. Розуміння принципів їх побудови є необхідним для створення сучасних інформаційних систем.

Лекція 18. Об’єктні, об’єктно-орієнтовані та об’єктно-реляційні системи керування базами даних

У процесі розвитку інформаційних систем виникла необхідність працювати з більш складними структурами даних, ніж ті, що пропонує класична реляційна модель. Це пов’язано з поширенням об’єктно-орієнтованого програмування, яке оперує такими поняттями, як класи, об’єкти, інкапсуляція, наслідування та поліморфізм. Для узгодження підходів до програмування і зберігання даних були розроблені об’єктні, об’єктно-орієнтовані та об’єктно-реляційні системи керування базами даних.

Об’єктні бази даних є підходом, при якому дані зберігаються у вигляді об’єктів, аналогічних тим, що використовуються в об’єктно-орієнтованому програмуванні. Об’єкт містить як дані (атрибути), так і методи для їх обробки. Це дозволяє поєднати структуру даних і логіку їх обробки в одному елементі.

У таких системах підтримуються основні принципи об’єктно-орієнтованого підходу. Інкапсуляція забезпечує приховування внутрішньої структури об’єкта, наслідування дозволяє створювати нові об’єкти на основі існуючих, а поліморфізм забезпечує можливість обробки різних типів об’єктів єдиним способом.

Об’єктні системи керування базами даних дають змогу працювати зі складними структурами, такими як графічні об’єкти, мультимедійні дані, геоінформаційні моделі. Це робить їх придатними для спеціалізованих задач, наприклад у системах автоматизованого проєктування або мультимедійних додатках.

Водночас об’єктні бази даних мають обмежене поширення. Це пов’язано зі складністю стандартизації, відсутністю універсальної мови запитів, аналогічної SQL, а також труднощами інтеграції з існуючими системами.

Об’єктно-орієнтовані системи керування базами даних є розвитком об’єктного підходу і орієнтовані на повну інтеграцію з мовами об’єктно-орієнтованого програмування. У таких системах база даних фактично є розширенням середовища програмування.

У об’єктно-орієнтованих СКБД відсутній розрив між моделлю даних і моделлю програмування. Об’єкти можуть безпосередньо зберігатися у базі даних без необхідності перетворення у табличну форму. Це дозволяє уникнути так званого «імпедансного розриву» між об’єктною та реляційною моделями.

Об’єктно-орієнтовані СКБД використовуються у системах, де важливо зберігати складні об’єкти з великою кількістю взаємозв’язків. Вони забезпечують природне представлення даних, що відповідає структурі програмного коду.

Проте такі системи також мають обмеження. Вони менш універсальні, ніж реляційні бази даних, і мають обмежену підтримку стандартів. Це ускладнює їх використання у великих корпоративних системах.

Об’єктно-реляційні системи керування базами даних є компромісним підходом, який поєднує можливості реляційної моделі з елементами об’єктно-орієнтованого підходу. У таких системах зберігається таблична структура даних, але додається підтримка складних типів, користувацьких типів даних та механізмів наслідування.

Об’єктно-реляційні СКБД дозволяють працювати з вкладеними структурами, масивами, записами та іншими складними типами. Це розширює можливості класичних реляційних систем і дозволяє використовувати їх у більш складних задачах.

Прикладом об’єктно-реляційної системи є PostgreSQL, яка підтримує користувацькі типи даних, розширення та функції, що дозволяють реалізувати складні структури даних.

Основною перевагою об’єктно-реляційного підходу є збереження сумісності з реляційною моделлю і мовою SQL при одночасному розширенні функціональності. Це дозволяє використовувати знайомі інструменти і водночас працювати зі складнішими структурами даних.

Особливості використання об’єктних і об’єктно-реляційних систем проявляються у складних програмних системах. У таких системах дані часто мають складну структуру, що важко відобразити у вигляді простих таблиць. Наприклад, у системах автоматизованого проєктування використовуються об’єкти, що містять геометричні параметри, зв’язки і методи обробки.

У мультимедійних системах необхідно працювати з даними різних типів, такими як зображення, відео та аудіо. Об’єктні підходи дозволяють ефективно зберігати і обробляти такі дані.

У розподілених системах і вебзастосунках об’єктно-реляційні бази даних дозволяють поєднувати гнучкість об’єктного підходу з надійністю реляційної моделі. Це робить їх універсальним рішенням для сучасних інформаційних систем.

Водночас необхідно враховувати складність таких систем. Використання об’єктних можливостей може ускладнювати проєктування і знижувати продуктивність у деяких випадках. Тому важливо обирати підхід залежно від конкретних вимог системи.

Таким чином, об’єктні, об’єктно-орієнтовані та об’єктно-реляційні системи керування базами даних розширюють можливості роботи з даними і дозволяють ефективно вирішувати задачі, пов’язані зі складними структурами. Розуміння їх особливостей дає змогу обрати оптимальний підхід до організації даних у сучасних програмних системах.

Лекція 19. Сховища даних та аналітичні системи

У сучасних інформаційних системах поряд із традиційними базами даних, орієнтованими на обробку транзакцій, широко використовуються сховища даних та аналітичні системи. Їх основне призначення полягає не лише у зберіганні інформації, а й у забезпеченні її аналізу, підтримці прийняття рішень та виявленні закономірностей у великих обсягах даних.

Сховище даних є спеціалізованою базою даних, що призначена для накопичення, інтеграції та аналізу інформації з різних джерел. На відміну від операційних систем, де дані постійно змінюються, у сховищах даних інформація є відносно стабільною і використовується переважно для аналітичних задач. Дані у сховищі організовані таким чином, щоб забезпечити швидкий доступ до великих обсягів інформації.

Основними характеристиками сховищ даних є інтегрованість, предметна орієнтованість, часовий аспект та незмінність. Інтегрованість означає, що дані надходять з різних джерел і приводяться до єдиного формату. Предметна орієнтованість означає, що дані організовані навколо певних бізнес-процесів або об’єктів. Часовий аспект передбачає збереження історичних даних, а незмінність означає, що дані не змінюються, а лише доповнюються.

У контексті аналітичних систем важливим є розрізнення OLTP та OLAP систем. OLTP (Online Transaction Processing) системи орієнтовані на обробку великої кількості коротких транзакцій. Вони забезпечують швидке виконання операцій вставки, оновлення та видалення даних і використовуються у щоденній роботі інформаційних систем.

OLAP (Online Analytical Processing) системи, навпаки, орієнтовані на аналіз даних. Вони дозволяють виконувати складні запити, що обробляють великі обсяги інформації, і використовуються для формування звітів, аналізу тенденцій та підтримки прийняття рішень. OLAP-системи працюють із даними, що вже накопичені у сховищах даних.

Основною відмінністю між OLTP і OLAP є характер навантаження. OLTP-системи виконують велику кількість простих операцій, тоді як OLAP-системи виконують відносно невелику кількість складних аналітичних запитів. Це визначає різні підходи до проєктування і оптимізації цих систем.

У аналітичних системах широко використовуються багатовимірні моделі даних. Вони дозволяють представляти дані у вигляді багатовимірних структур, де кожен вимір відповідає певному аспекту аналізу. Наприклад, у системі продажів вимірами можуть бути час, товар, регіон і клієнт.

Основою багатовимірної моделі є поняття куба даних. Куб даних дозволяє виконувати аналіз за різними вимірами, що дає змогу швидко отримувати узагальнену інформацію. Наприклад, можна визначити обсяг продажів за певний період часу у конкретному регіоні.

Багатовимірні моделі підтримують операції, такі як агрегація, деталізація, поворот вимірів і фільтрація. Це дозволяє виконувати гнучкий аналіз даних і отримувати інформацію у різних розрізах.

Важливим елементом аналітичних систем є вітрини даних. Вітрина даних є підмножиною сховища даних, що орієнтована на конкретну предметну область або групу користувачів. Вона містить лише ті дані, які необхідні для виконання певних аналітичних задач.

Вітрини даних дозволяють підвищити продуктивність системи, оскільки зменшують обсяг даних, що обробляються. Крім того, вони спрощують доступ до інформації і роблять її більш зрозумілою для користувачів.

Процес наповнення сховища даних зазвичай включає етапи вилучення, перетворення та завантаження даних. На першому етапі дані збираються з різних джерел. На другому етапі вони очищуються, узгоджуються і приводяться до єдиного формату. На третьому етапі дані завантажуються у сховище.

Аналітична обробка інформації включає виконання складних запитів, побудову звітів, аналіз тенденцій та прогнозування. Для цього використовуються спеціалізовані інструменти, що дозволяють працювати з великими обсягами даних і отримувати результати у зручному вигляді.

У сучасних системах аналітична обробка може виконуватися у режимі реального часу або з використанням попередньо оброблених даних. Вибір підходу залежить від вимог до системи і доступних ресурсів.

Важливим аспектом є оптимізація аналітичних запитів. Для цього використовуються спеціальні структури даних, індекси та механізми кешування. Це дозволяє значно підвищити швидкість обробки великих обсягів інформації.

Таким чином, сховища даних і аналітичні системи є важливими компонентами сучасних інформаційних систем. Вони забезпечують можливість ефективного аналізу даних, підтримки прийняття рішень і виявлення закономірностей. Розуміння принципів їх побудови і функціонування є необхідним для роботи з великими обсягами інформації у сучасних умовах.

Лекція 20. Бази знань та системи керування знаннями

У сучасних інформаційних системах поряд із традиційними базами даних все більшого значення набувають бази знань, які орієнтовані не лише на зберігання фактів, а й на представлення знань, закономірностей і правил предметної області. Якщо база даних оперує структурованими даними, то база знань містить інформацію, яка може бути використана для логічного висновку, аналізу та прийняття рішень.

Знання у контексті інформаційних систем визначаються як узагальнена, структурована і осмислена інформація, що відображає закономірності предметної області і може бути використана для вирішення задач. На відміну від даних, знання містять не лише факти, але й зв’язки між ними, правила та досвід.

Системи керування знаннями призначені для створення, зберігання, обробки та використання знань. Вони є складнішими за традиційні системи керування базами даних, оскільки повинні забезпечувати не лише доступ до інформації, але й її інтерпретацію. Такі системи широко використовуються у штучному інтелекті, експертних системах, системах підтримки прийняття рішень.

Основними компонентами системи керування знаннями є база знань, механізм логічного висновку та інтерфейс користувача. База знань містить факти і правила, механізм висновку дозволяє отримувати нові знання на основі існуючих, а інтерфейс забезпечує взаємодію користувача з системою.

База знань є центральним елементом системи і містить інформацію про предметну область у формалізованому вигляді. Вона може містити як декларативні знання, що описують факти, так і процедурні знання, що визначають способи їх використання.

Декларативні знання представлені у вигляді тверджень, що описують стан об’єктів або їх властивості. Процедурні знання визначають правила або алгоритми, які дозволяють отримувати нові знання на основі існуючих.

Важливим аспектом є вибір методу представлення знань. Існує кілька основних підходів, кожен із яких має свої особливості та області застосування.

Одним із найпоширеніших є логічний підхід, при якому знання представляються у вигляді логічних формул. Такий підхід дозволяє використовувати формальні методи доведення і забезпечує високий рівень точності, але може бути складним у реалізації.

Іншим підходом є продукційні системи, у яких знання представлені у вигляді правил типу «якщо – то». Наприклад, правило може визначати, що якщо виконуються певні умови, то необхідно зробити певний висновок. Продукційні системи широко використовуються в експертних системах.

Фреймова модель є ще одним методом представлення знань. У цій моделі знання організовані у вигляді структур, що описують об’єкти та їх властивості. Фрейми дозволяють ефективно представляти складні об’єкти і взаємозв’язки між ними.

Семантичні мережі використовуються для представлення знань у вигляді графів, де вузли відповідають об’єктам або поняттям, а зв’язки – відношенням між ними. Такий підхід дозволяє наочно відображати структуру знань і використовується у системах обробки природної мови.

Важливим елементом систем керування знаннями є механізм логічного висновку. Він дозволяє отримувати нові знання на основі існуючих фактів і правил. Існують різні методи висновку, зокрема прямий і зворотний висновок.

Прямий висновок передбачає застосування правил до наявних фактів для отримання нових фактів. Зворотний висновок починається з гіпотези і перевіряє, чи можна її підтвердити на основі наявних знань.

Системи керування знаннями широко використовуються у сучасних інформаційних системах. Однією з найбільш відомих областей є експертні системи, які імітують роботу експерта у певній галузі. Вони використовуються у медицині, технічній діагностиці, фінансовому аналізі.

У системах підтримки прийняття рішень бази знань дозволяють аналізувати ситуації, оцінювати альтернативи і вибирати оптимальні рішення. Це особливо важливо у складних і невизначених умовах.

У сучасних програмних системах бази знань також використовуються для реалізації інтелектуальних функцій, таких як рекомендаційні системи, аналіз текстів, обробка

природної мови. Вони дозволяють підвищити рівень автоматизації і ефективність роботи систем.

Водночас використання баз знань пов’язане з певними труднощами. Однією з основних проблем є складність формалізації знань. Не всі знання можуть бути легко представлені у формальному вигляді, що ускладнює їх використання.

Іншою проблемою є підтримка актуальності знань. У динамічних предметних областях знання швидко застарівають, що вимагає їх постійного оновлення.

Таким чином, бази знань та системи керування знаннями є важливим напрямом розвитку інформаційних технологій. Вони дозволяють працювати не лише з даними, але й із знаннями, забезпечуючи можливість логічного висновку, аналізу та прийняття рішень. Розуміння принципів їх побудови є необхідним для створення сучасних інтелектуальних систем.

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

Опорний конспект лекцій для здобувачів вищої освіти,
які навчаються за спеціальностями галузі «Інформаційні технології»

Українською мовою