Скорочений конспект лекцій з дисципліни бази даних

Розробив: Євген Гофман, доцент каф. програмних засобів

Зміст

1. Загальні відомості про системи баз даних

Файлові системи та системи баз даних

Відомі два підходи до організації збереження інформації: **файлові системи та системи баз даних.**

З точки зору прикладної програми файл – це пойменована область зовнішньої пам'яті, до якої можна записувати або зчитувати дані. Така організація дозволяє досягти високої швидкості обробки інформації, але має цілу низку недоліків.

По-перше, така організація неодмінно викликає дублювання даних.

По-друге, дублювання даних при обробці може призвести до їхнього протиріччя між собою.

Наприклад, інформація про товарну продукцію може зберігатися у декількох відділах: відділі закупівель та відділі збуту. Оскільки зміна цих даних може здійснюватися різними особами, то відсутня можливість виявити порушення несуперечності інформації, що зберігається.

Крім того, формати даних жорстко зафіксовані в обробляючих програмах, що викликає виникнення можливих помилок, а також незручності в разі необхідності модифікації цих форматів.

Тому виникає необхідність відокремити дані від їх опису, визначити таку організацію зберігання даних із урахуванням зв'язків між ними, яка дозволила би використовувати ці дані одночасно для багатьох прикладних програм.

В результаті виникла нова концепція організації інформаційного забезпечення – концепція інтеграції даних, центральне місце в якої займає поняття «бази даних». При такому підході всі дані зберігаються в одному місці окремо від прикладних програм.

Централізований підхід щодо збереження та керування даними має наступні переваги:

1. Значно зменшується надлишок інформації.

2. Зменшення надлишку інформації дозволяє усунути протиріччя та забезпечити логічну цілісність даних.

3. Доступ до даних може здійснюватися одночасно багатьма прикладними програмами чи користувачами.

4. Завдяки повному контролю над базою даних з'являється можливість забезпечення заходів безпеки баз даних.

База даних – це певним чином організована сукупність даних, які відображають стан об'єктів деякої предметної області та відносини між цими об'єктами.

Предметною областю (ПО) називається сукупність об'єктів, предметів реального світу, що розглядаються у контексті задачі, що вирішується. У якості предмету реального світу в даному визначенні можуть бути як матеріальний об'єкт, так і процес, явище тощо.

Ціль створення БД міститься у відокремленні подання БД користувачу від її фізичного подання.

Виділяють зовнішній рівень – це вміст БД, яким бачить його окремий користувач (тобто для користувача зовнішнє подання саме й є база даних). Подання користувачем деякої задачі відображає його конкретні інформаційні потреби. Зовнішнє подання містить тільки ті об'єкти ПО, їхні характеристики та зв'язки між ними, що цікавлять користувача. Зовнішнє подання з точки зору прикладного програміста відображає елементи даних у їх взаємозв'язку.

Та внутрішній (фізичний) рівень пов'язаний з організацією зберігання даних на фізичних носіях інформації. З внутрішнім поданням зв'язується внутрішня схема БД, яка визначає типи структур даних, що використовуються, та їх взаємозв'язки. На цьому рівні здійснюється взаємодія СКБД з методами

доступу операційної системи з метою розміщення даних на запам’ятовуючих пристроях, створення індексів, вибірки даних тощо. На внутрішньому рівні зберігається наступна інформація:

− розподіл дискового простору для зберігання даних і індексів;
− опис подробиць збереження записів (із вказуванням реальних розмірів елементів даних);
− відомості про розміщення записів;
− відомості про стиснення даних і обраних методах їхнього шифрування.

Система баз даних – це система спеціальним чином організованих даних (баз даних), програмних, технічних, мовних, організаційно-методичних засобів, які призначені для забезпечення централізованого накопичення та колективного багатоцільового використання даних.

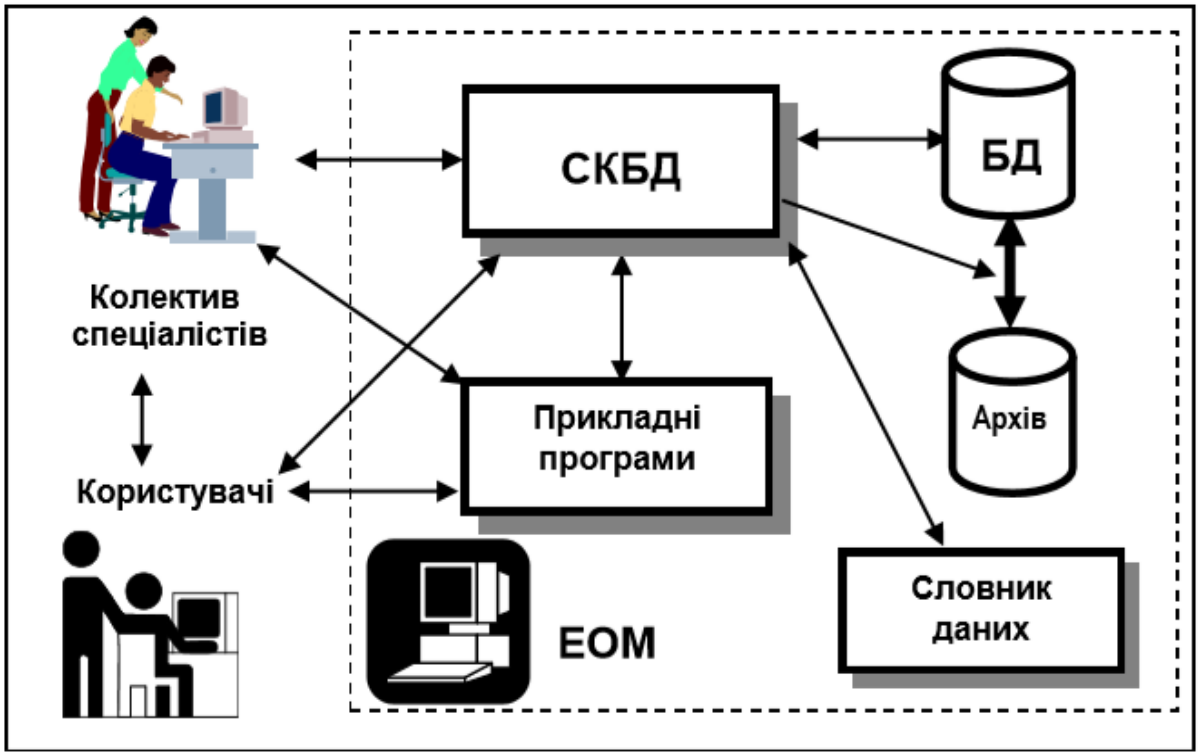

Рисунок 1.1 - Структура системи баз даних

СКБД – це комплекс програмних і лінгвістичних засобів, які забезпечують створення, завантаження, супроводження та експлуатацію бази даних

СКБД грає центральну роль у функціонуванні системи баз даних. Вона володіє засобами безпосереднього доступу до БД і надає кожній прикладній програмі інтерфейс для взаємодії з БД.

Архітектура клієнт сервер

При такому підході система баз даних складається з двох частин: набору клієнтів і сервера.

Клієнт (зазвичай називається локальним вузлом – local node), керує інтерфейсом користувача та логікою застосувань, діючи як складна робоча станція, на якій виконуються застосування користувача. Клієнт одержує від користувача запит, перевіряє синтаксис, генерує запит до бази даних на мові SQL або іншій мові бази даних і передає його серверу. Отримавши відповідь, клієнт форматує отримані дані для надання користувачеві.

Сервер (зазвичай його називають віддаленим вузлом – remote node) – це взагалі СКБД. Він знаходиться на спеціально виділених комп’ютерах і підтримує всі основні функції СКБД: визначення даних, обробка даних, захист і цілісність даних, адміністрування тощо. Сервер одержує та обробляє запити до бази даних, а потім передає отримані результати клієнту.

Таблиця 1.1 – Функції клієнта й сервера

Клієнт	Сервер
Керує інтерфейсом користувача	Приймає та обробляє запити з боку клієнта до бази даних
Приймає й перевіряє синтаксис запиту користувача	Перевіряє повноваження користувачів
Виконує застосування	Гарантує дотримання обмежень цілісності
Генерує запит до бази даних і передає його серверу	Виконує запити/оновлення й повертає результати клієнту
Відображає отримані результати користувачу	Підтримує системний каталог Забезпечує паралельний доступ до бази даних Забезпечує управління відновленням

2. Теорія проектування реляційних баз даних

Побудова ER - діаграми

На етапі концептуального проектування у якості моделі ПО використовується модель типу «сутність – зв'язок» (ER – Entity Relation), що була запропонована паном Ченом [33] у 1976 р.

Конструктивними елементами моделі служать сутності, атрибути та зв'язки . Основним конструктивним елементом являється сутність.

Користувач описує ті об'єкти ПО, що його зацікавили, за допомогою сутностей, потім визначає властивості сутностей, використовуючи атрибути, та, нарешті, описує відповідності між сутностями за допомогою зв'язків.

Сутність – це збиральне поняття, деяка абстракція реально існуючого об'єкта навколишнього світу, процесу чи явища.

Згідно з Ченом, об'єкти, що описуються сутностями, можна підрозділити на правильні об'єкти та слабкі об'єкти. Слабким називається об'єкт, який знаходиться в залежності від деякого іншого об'єкта, тобто він не може існувати, якщо не існує цей інший об'єкт.

Наприклад, сутність ПОСТАЧАЛЬНИК являється правильним об'єктом, а ПОСТАЧАЄМИЙ_ТОВАР – слабким об'єктом.

Атрибут – це пойменована характеристика сутності.

Атрибут являється засобом, за допомогою якого визначаються властивості сутності. Атрибути мають сенс тільки по відношенню до сутностей, яких вони визначають. Для ідентифікації конкретних екземплярів сутностей використовуються спеціальні атрибути – ідентифікатори (ключі). Це може бути один або декілька атрибутів, значення яких дозволяють беззаперечно відрізняти один екземпляр сутності від іншого.

Наприклад, ідентифікуючий атрибут (ключ) ТАБЕЛЬНИЙ_НОМЕР.

Зв'язки виступають у моделі в якості засобу, за допомогою якого представляються стосунки між двома й більш сутностями та атрибутами. Розрізняють три види зв'язків, що позначаються 1:1 (« один до одного »), 1:M («один до багатьох») та M:N (« багато до багатьох»).

Під час аналізу предметної області (ПО) можливе виявлення трьох типів зв'язку:

− сутність-сутність;

− сутність-атрибут;

− атрибут-атрибут.

Зв'язок «один до одного» (між двома типами сутностей)

При такому зв'язку кожному екземпляру сутності А відповідає один і тільки один екземпляр сутності В і, навпаки, кожному екземпляру сутності В відповідає один і тільки один екземпляр сутності А. При цьому ідентифікація екземплярів сутності унікальна в обох напрямках.

Зв'язок «один до багатьох» (між двома типами сутностей)

При такому зв'язку кожному екземпляру сутності А може відповідати 0, 1 або декілька екземплярів сутності В , але кожному екземпляру сутності В відповідає тільки один екземпляр сутності А. Це означає, що ідентифікація екземплярів сутності унікальна тільки в напрямку від В до А .

Наприклад, кожний продавець може обслуговувати декілька замовлень, але кожне замовлення обслуговується лише одним продавцем.

Зв'язок «багато до багатьох» (між двома типами сутностей)

Це відображення є найбільш складним видом зв'язку між сутностями. При цьому кожному екземпляру сутності А може відповідати 0, 1 або декілька екземплярів сутності В і навпаки, кожному екземпляру сутності В може відповідати 0, 1 чи декілька екземплярів сутності А, тобто ідентифікація екземплярів сутності неунікальна в обох напрямках.

Наприклад, зв'язок між сутностями ПРОДАВЕЦЬ і ПОКУПЕЦЬ.

Зв'язок «один до одного» (між двома атрибутами)

Якщо при описанні сутності використовуються два унікальних ідентифікатора, наприклад, КОД_КЛІЄНТА та НОМЕР_ПАСПОРТУ, то між ними існує зв'язок 1:1.

Зв'язок «один до багатьох» (між двома атрибутами)

Наприклад, між двома атрибутами НОМЕР_ГРУПИ й ПРІЗВИЩЕ_ СТУДЕНТА існує зв'язок 1:М.

Зв'язок «багато до багатьох» (між двома атрибутами)

Наприклад, атрибути ВИКЛАДАЧ і ДИСЦИПЛІНА сутності ВИКЛАДАННЯ_ДИСЦИПЛІНИ пов'язані зв'язком M:N.

Сутності, атрибути і зв'язки є конструктивними елементами, з використанням яких створюється локальне подання – модель фрагменту предметної області.

Ключовий атрибут – це унікальний ідентифікатор екземпляру сутності. Розрізняють первинні і зовнішні ключі. Первинний ключ має містити множину унікальних значень, використовується для ідентифікації екземплярів сутності в окремій таблиці даних. Зовнішній ключ використовується для посилання на множину значень первинного ключа іншої таблиці, може містити повторювані значення, але тільки з переліку первинного ключа, на який посилається.

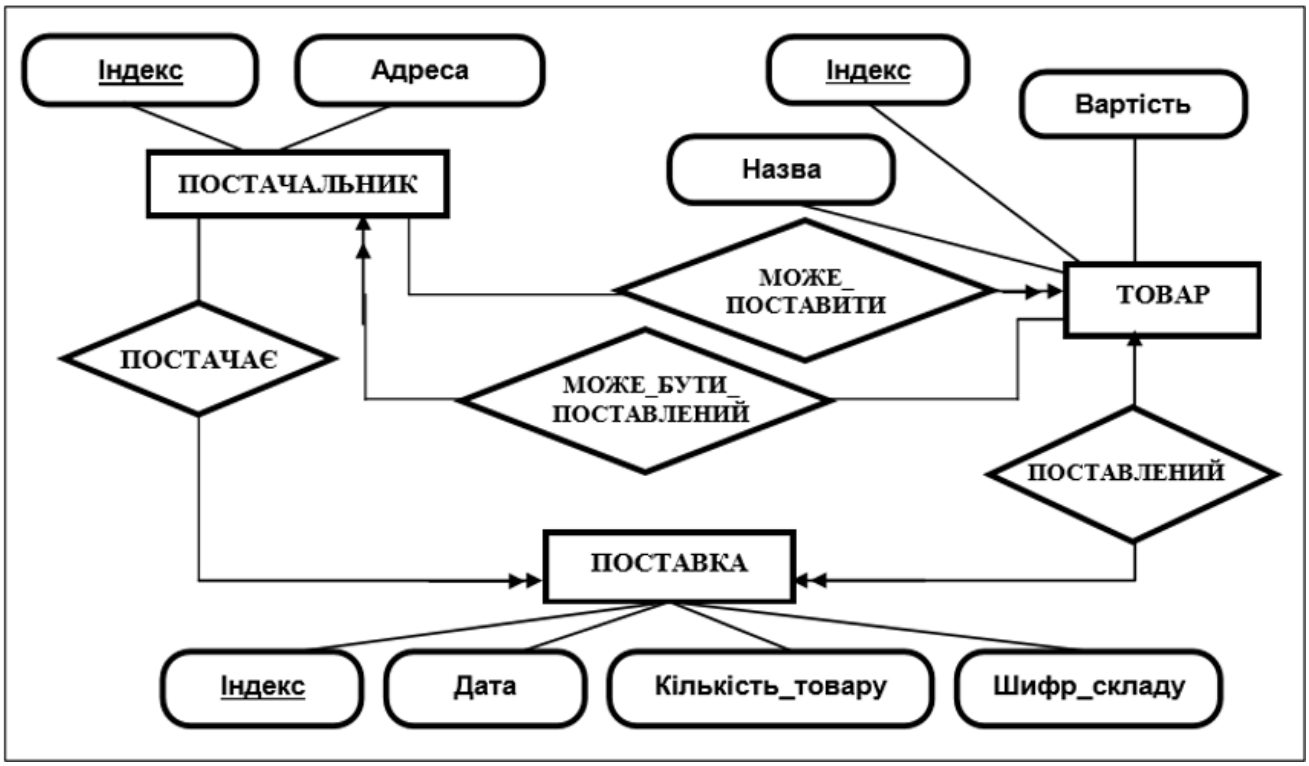

Рисунок 2.1 – приклад локального подання «Постачальник – Поставка – Товар»

Об’єднання локальних подань у загальну ER - діаграму

Після етапу формування локальних подань здійснюється їх об’єднання в єдину глобальну інформаційну структуру. Ця структура не тільки взаємно відповідає кожному поданню, але й уявляє собою інтегровану БД у стиснутому вигляді. При великій кількості подань (більше 5–6) зазвичай використовується бінарне об’єднання. При цьому результат об’єднання N1 об’єктів одного подання з N2 об’єктами другого подання дає (N1 + N2 – X) об’єктів, де X – кількість об’єктів, які співпадають в об’єднаних поданнях.

Існують три основні концепції об’єднання: ідентичність, агрегація та узагальнення.

Два та більше елементів у моделі ідентичні, якщо вони мають однакове семантичне (змістове) значення. Але для ідентичних елементів зовсім необов’язково мати однакове синтаксичне представлення.
Агрегація дозволяє розглядати зв’язок між елементами моделі як новий елемент більш високого рівня. *Наприклад, елемент СЛУЖБОВЕЦЬ може бути сприйнятий як агрегація елементів ПРІЗВИЩЕ, НОМЕР_ПАСПОРТУ та АДРЕСА.*
Узагальненням називається абстракція даних, яка дозволяє трактувати клас різних подібних об’єктів як один пойменований узагальнений тип об’єкту. *Наприклад, клас об’єктів {собака, кішка, слон} може бути узагальнений в об’єкт ТВАРИНА.*

Реляційна модель даних. Сім правил побудови ефективної моделі даних

Відношення – це підмножина декартового добутку доменів, або відношення = двомірна таблиця, де в стовпчиках визначаються атрибути, а у рядках екземпляри сутності.

Відношення поділяють на об'єктні та зв'язні.

Об'єктне відношення зберігає дані про об'єкти (екземпляри сутності).

Зв'язне відношення зберігає ключі двох або більше об'єктних відношень, тобто по ключам встановлюються зв'язки між об'єктами відношень.

Клас належності сутності повинний бути або **обов'язковим, або необов'язковим**.

Клас належності визначається по відношенню до зв'язаної сутності, залежить від того, чи можуть зберігатись екземпляри сутності, що не мають зв'язку з іншою сутністю.

Наприклад, клієнти, що нічого ще не купували, або товари, що ніколи ще не продавались.

Правило 1. Якщо ступінь бінарних зв'язків дорівнює 1:1 та клас приналежності обох сутностей є обов'язковим, то потрібне тільки одне відношення. Первинним ключем цього відношення може бути будь-який ключ із двох сутностей.
Правило 2. Якщо ступінь бінарного зв'язку дорівнює 1:1 і клас приналежності однієї сутності є обов'язковим, а іншої – необов'язковим, то необхідна побудова двох відношень – по одному для кожної сутності. При цьому ключ сутності повинний служити первинним ключем для відповідного відношення. Крім того, ключ сутності, для якої клас приналежності є необов'язковим, додається в якості зовнішнього ключа до відношення, яке виділене для сутності з обов'язковим класом приналежності.
Правило 3. Якщо ступінь бінарного зв'язку дорівнює 1:1 і клас належності жодної з сутностей не є обов'язковим, то необхідне використання трьох відношень: по одному для кожної сутності та одного для зв'язку.
Правило 4. Якщо ступінь бінарного зв'язку дорівнює 1:N і клас приналежності n-зв'язкової сутності є обов'язковим, то достатнім є використання двох відношень, по одному на кожну сутність, за умови, що ключ кожної сутності використовується в якості первинного ключа для відповідної сутності. **Додатково ключ 1-зв'язкової сутності повинний бути доданий як атрибут (зовнішній ключ) до відношення, що відводиться n-зв'язковій сутності.**
Правило 5. Якщо ступінь бінарного зв'язку дорівнює 1:n і клас приналежності n-зв'язкової сутності є необов'язковим, то необхідне формування трьох відношень: по одному на кожну сутність і одного відношення для зв'язку. Зв'язкове відношення повинне включати первинні ключі кожної сутності.
Правило 6. Якщо ступінь бінарних зв'язків дорівнює m:n, то для збереження даних потрібні три відношення поза залежністю від класу приналежності обох сутностей. Зв'язкове відношення повинно включати первинні ключі кожної сутності.
Правило 7. У випадку тристороннього зв'язку необхідно використовувати чотири відношення: по одному для кожної сутності й одне відношення для зв'язку. Зв'язкове відношення повинне включати первинні ключі кожної сутності. Аналогічно для n-стороннього зв'язку потрібно n+1 відношення.

Аномалії ненормалізованого відношення. Покроковий метод нормалізації баз даний

При проектуванні схеми реляційної БД атрибути групуються у відношення шляхом виводу реляційної схеми з розробленої ER-діаграми.

При експлуатації БД у відношенні можуть виникнути аномалії, пов'язані з виконанням операції підтримки його в актуальному стані. Виявити фактичний прояв тієї чи іншої аномалії можна, лише врахувавши конкретну семантику даних.

1. Аномалія оновлення (UPDATE). *Заміна керівника відділу призведе до необхідності внесення змін і модифікації по кожному працівнику даного відділу. Отже, для підтримки узгодженості даних необхідно виконати зміну не лише в кортежі бази, а цілий ряд змін, що може розглядатися як аномалія, оскільки характер самої зміни повинен стосуватися лише одного певного запису БД.*

2. Аномалія поповнення (INSERT). *Приймаючи на роботу нового співробітника, необхідно вносити до БД відомості не лише про нього, а й про керівника та тип контракту. Це також можна розглядати як аномалію, тому що не завжди можливо внести до БД відомості про відділ і контракт, оскільки часто співробітників беруть на роботу з випробувальним терміном і лише після його проходження підписують певний вид контракту та визначають відділ, де він працюватиме.*

3. Аномалія вилучання (DELETE). *Припустимо, що звільнились усі працівники певного відділу. Тоді разом з інформацією про них у БД буде втрачено інформацію про цей відділ. Якщо цю інформацію необхідно зберігати, то це також може розглядатись як аномалія.*

4. Надмірність. *Відомості про керівника відділу повторюються в ряді кортежів БД. Основні проблеми із зберіганням надлишку інформації пов'язані не лише з неефективним використанням пам'яті, а й із забезпеченням підтримки узгодженості цих даних.*

Усунути можливість аномалій або, принаймні, зменшити їхню кількість можна, використовуючи формальний метод, запропонований Е. Коддом і який дозволяє проектувальнику знаходити оптимальне угрупування атрибутів для кожного відношення схеми БД.

Термін функціональна залежність (ФЗ) означає: атрибут B відношення R функціонально залежить від атрибута A того ж відношення, якщо в кожний момент часу кожному значенню атрибута A відповідає тільки одне значення атрибута B, зв'язаного з A у відношенні R.

Якщо неключовий атрибут залежить тільки від частини ключа, то він знаходиться в **частковій функціональній залежності**.

Якщо для атрибутів A, B, C виконуються умови A → B і B → C, але обернена залежність відсутня, то C знаходиться в **транзитивній залежності** від A.

У відношенні R атрибут B **багатозначно залежить** від атрибута A (A→>B), якщо кожному значенню A відповідає численність значень B, ніяк не зв'язаних з іншими атрибутами з R.

1НФ. Відношення знаходиться в першій нормальній формі (1НФ), якщо всі його неключові атрибути прості, тобто містять атомарні, неподільні значення.
2НФ. Відношення знаходиться в 2НФ, якщо воно знаходиться в 1НФ і в ньому відсутні часткові функціональні залежності описових атрибутів від атрибутів первинного ключа.
3НФ. Відношення знаходиться в 3НФ, якщо воно знаходиться в 2НФ і не містить транзитивних залежностей.
НФБК. Відношення знаходиться в НФБК тоді й тільки тоді, коли детермінанти є потенційними ключами. Іншими словами, на діаграмі функціональної залежності стрілки будуть починатися тільки з потенційних ключів.
4НФ. Відношення знаходиться в 4НФ, якщо воно знаходиться в НФБК і в ньому відсутні незалежні багатозначні залежності, тобто всі вони виділені (рознесені) в окремі відношення з тим самим ключем.

3. Мова структурованих запитів (SQL)

Загальні відомості

SQL може використовуватися для маніпуляції з даними (вибірка і модифікація), опрацювання структури бази даних (створення та вилучання таблиць та індексів) і адміністрування бази даних (визначення списку користувачів, призначення прав доступу). Використання SQL має багато переваг, найбільше важливими з яких є:

− SQL підтримується багатьма постачальниками СКБД. Програми, написані з використанням мови SQL, будуть працювати майже на будь-якому сервері бази даних;

− SQL простий у вивченні, оскільки в ньому для опису дій використовуються прості англійські слова, такі, як SELECT, INSERT, UPDATE, DELETE, COMMIT, ROLLBACK.

Таблиця 3.1 – Типи даних SQL

Тип даних	Розмір	Опис
1	2	3
BINARY	1 байт на знак	У полі цього типу можуть зберігатися дані будь-якого типу. Дані не перетворюються (наприклад, у текстові). Дані відображаються в тому ж вигляді, у якому вони вводяться в це поле.
BIT	1 байт	Значення «Так» (Yes) і «Ні» (No), а також поля, що містять одне з двох можливих значень.
TINYINT	1 байт	Ціле значення від 0 до 255.
MONEY	8 байтів	Ціле, що масштабується, від –922 337 203 685 477,5808 до 922 337 203 685 477,5807.
DATETIME (див. DOUBLE)	8 байтів	Дата або час; припустимий будь-який рік від 100 до 9999.
UNIQUEIDENTIFIER	128 бітів	Унікальний ідентифікатор, який використовується при викликах вилучених процедур.
REAL	4 байти	Число з крапкою, що плаває, та одинарною точністю від –3,402823E38 до –1,401298E-45 для негативних значень, від 1,401298E-45 до 3,402823E38 для позитивних значень або значення 0.
FLOAT	8 байтів	Число з крапкою, що плаває, та подвійною точністю від –1,79769313486232E-308 до –4,94065645841247E-324 для негативних значень, від 4,94065645841247E-324 до 1,79769313486232E308 для позитивних значень або значення 0.
SMALLINT	2 байти	Коротке ціле від –32 768 до 32 767 (див. «Примітки»).
INTEGER	4 байти	Довге ціле від –2 147 483 648 до 2 147 483 647.
DECIMAL	17 байтів	Тип даних для збереження точних числових значень від $-10^{28}-1$ до $10^{28}-1$. Точність (1–28) і фактор масштабування (від 0 до заданої точності) визначаються користувачем. За замовчуванням точність і фактор масштабування рівні відповідно 18 і 0.
TEXT	2 байти на знак	Від 0 до 2,14 Гбайт (поле MEMO)
IMAGE	Не обмежено	Від 0 до 2,14 Гбайт. Використовується для об'єктів OLE.
CHARACTER	2 байти на знак	Від 0 до 255 знаків.

Таблиця 3.2 – Команди мови SQL та їх призначення

Команда	Призначення
Команди визначення даних	
alter table	Змінює структуру таблиці
create index	Створює індекс
create table	Створює таблицю
create view	Створює подання
Drop	Вилучає таблицю, індекс, подання
Команди маніпуляції даними	
Delete	Вилучає запису таблиці
Insert	Добавляє запису в таблицю
Update	Змінює дані таблиці
Команда вибірки даних	
Select	Вибирає дані
Команди керування транзакціями	
commit	Робить зміни, проведені з початку транзакції, постійними
rollback	Відкочує всі проведені зміни до крапки збереігання або до початку транзакції
savepoint	Встановлює контрольну крапку, до котрого згодом можна буде виконати відкат
Команди керування даними	
check database	Перевіряє цілісність бази даних
grant	Надає привілеї
revoke	Скасовує надані раніше привілеї

Створення таблиць і зв'язків

За допомогою команд визначення даних SQL можна створювати, вилучати та маніпулювати таблицями (добавляти, вилучати, переставляти стовпчики та змінювати їхні параметри). Щоб створити таблицю треба зробити, щонайменше, наступне: **задати ім'я таблиці, задати імена її стовпчиків, визначити тип даних для кожного стовпчика, визначити (або використовувати по умовчанню) нульовий статус для кожного стовпчика** – припускається або забороняється використання в стовпчику нульових значень.

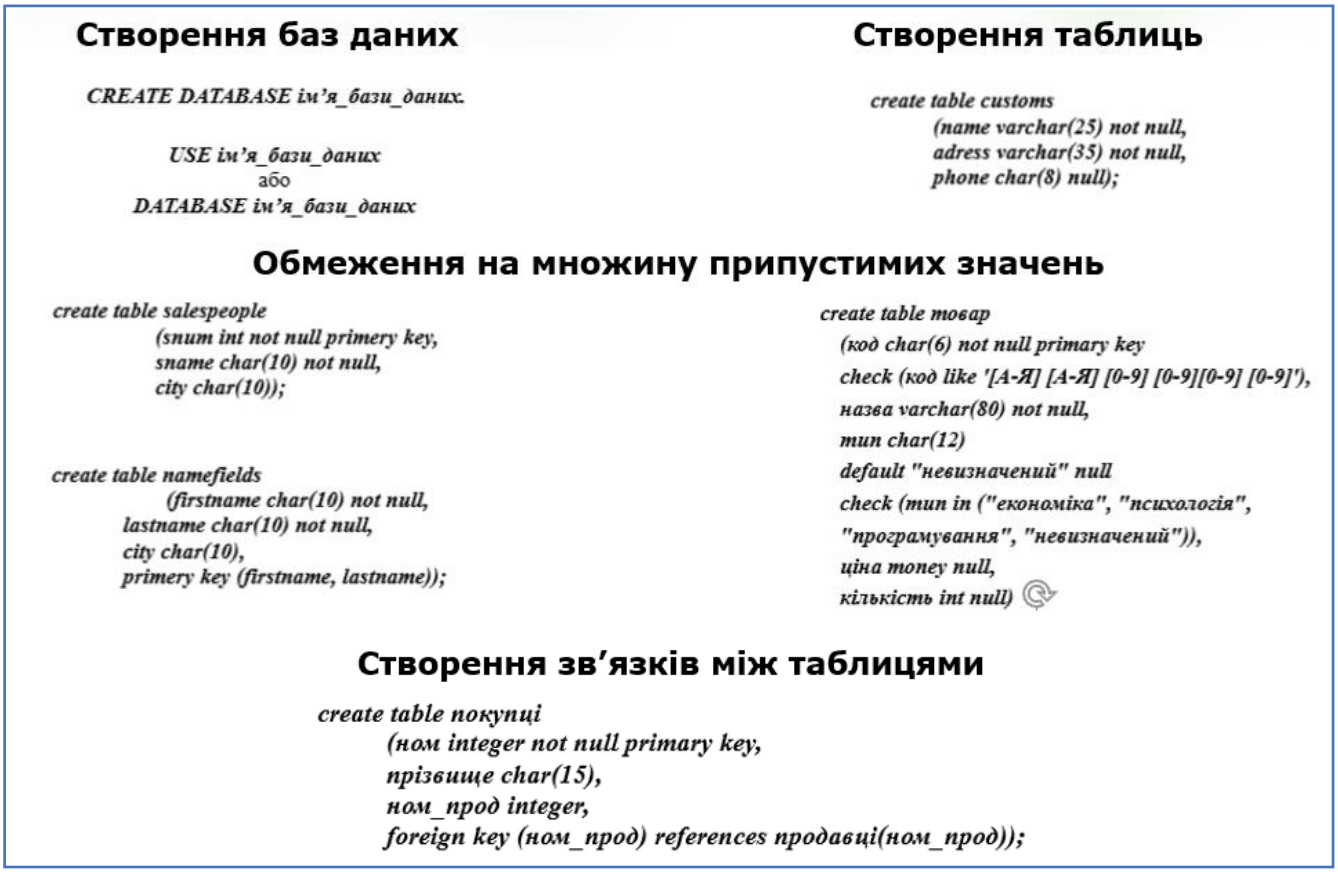

Рисунок 3.1 – приклад синтаксису SQL – команд, що використовуються для визначення даних

Створення індексів

У системах реляційних баз даних механізм індексації є засобом підвищення продуктивності, прискорюючи вибірку інформації з БД.

Індекс (index) – це упорядкований (в алфавітному або чисельному порядку) список вмісту стовпчиків або групи стовпчиків у таблиці.

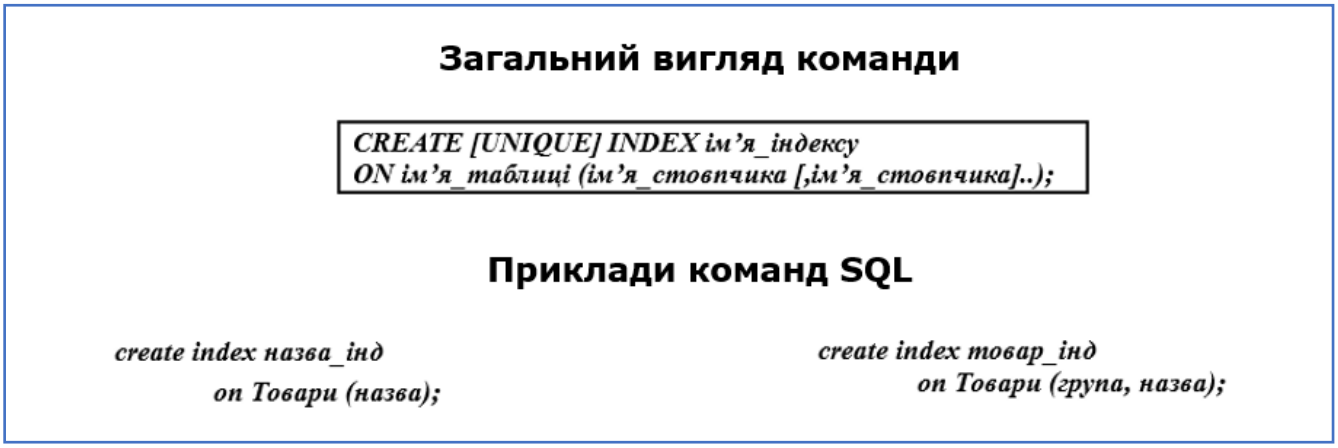

Рисунок 3.2 – приклад використання механізму індексації

Запити на вибірку даних (SELECT)

Для створення запитів на вибірку даних використовується конструкція SELECT. Результатом виконання запиту є таблиця, що зберігається в тимчасовому буфері серверу бази даних. Обрані дані можна використовувати для перегляду, друку звітів, формування графіків або, наприклад, додати їх до постійних (базових) таблиць бази даних.

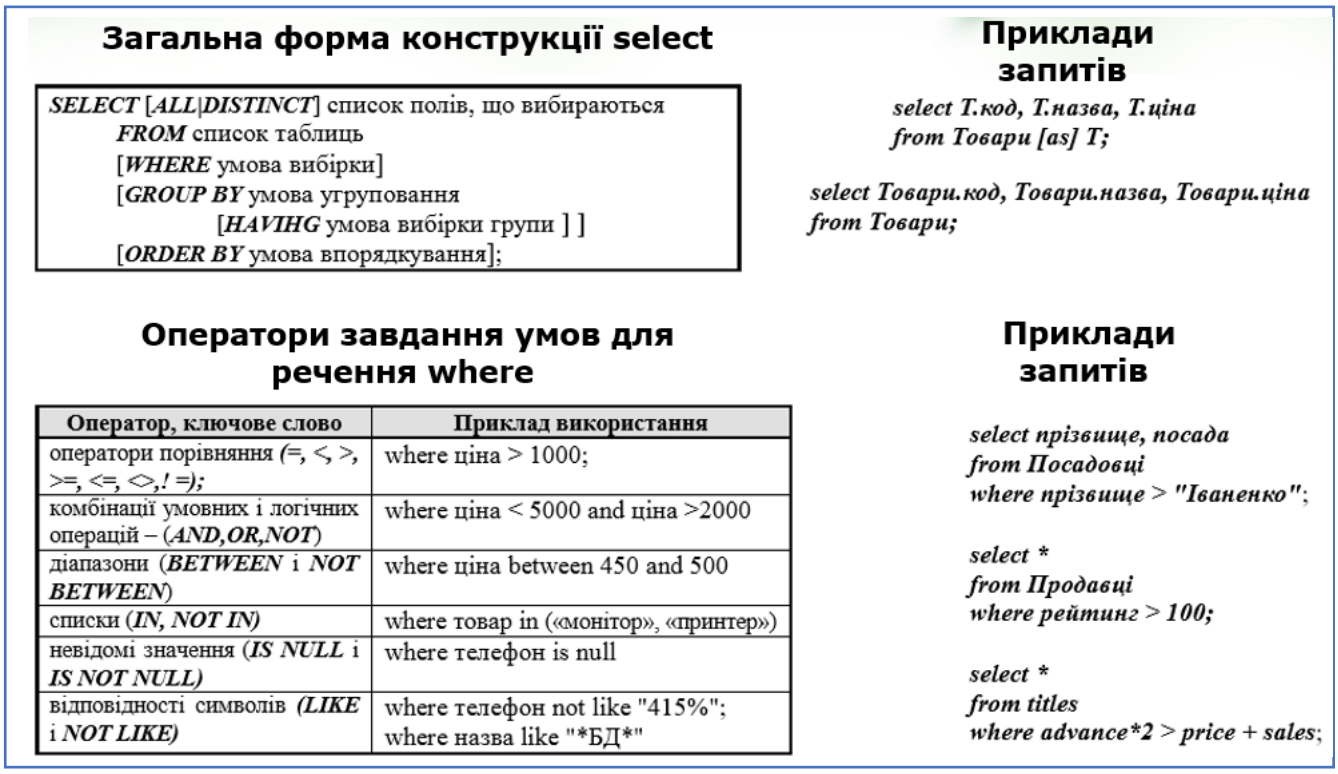

Рисунок 3.3 – приклад використання механізму відбору даних

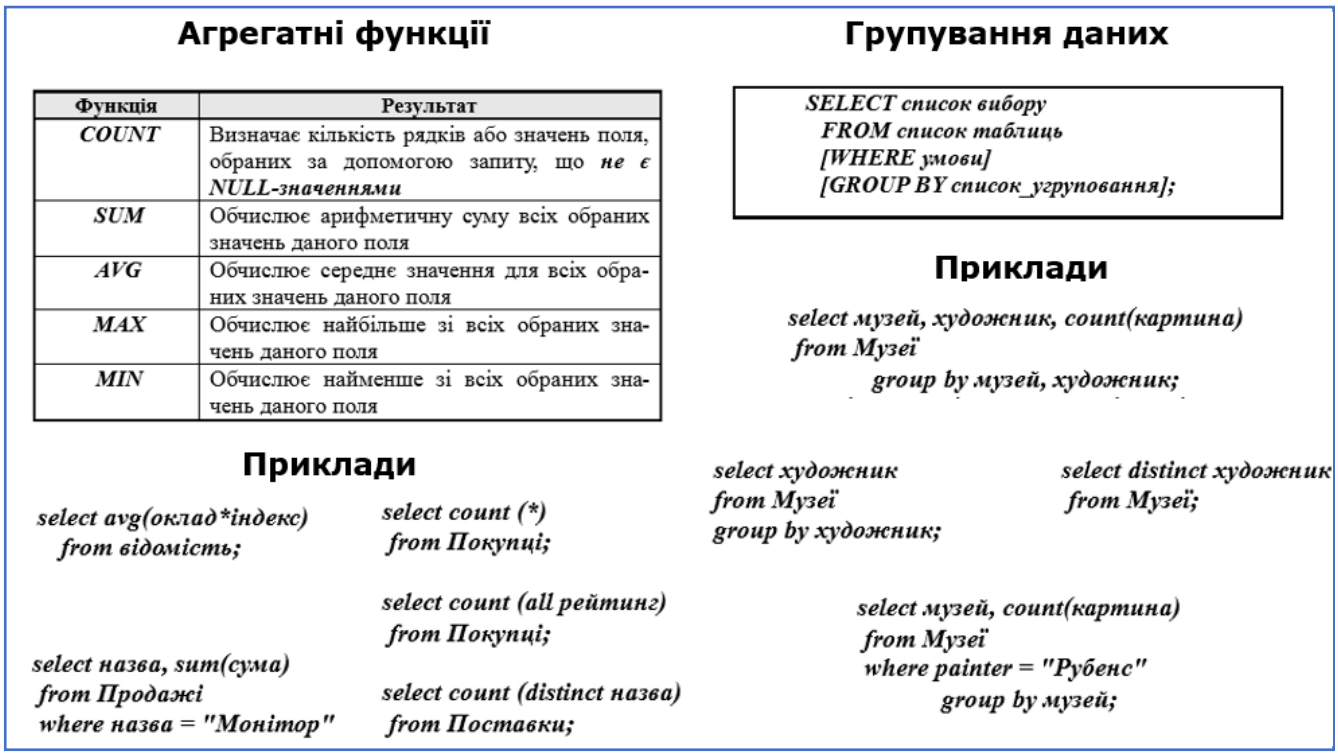

Рисунок 3.4 – приклади використання агрегатних функцій

Запити на модифікацію даних (INSERT, UPDATE, DELETE)

У SQL використовуються три команди модифікації даних:

− INSERT додає нові рядки до таблиці:

− UPDATE змінює існуючі в таблиці рядки;

− DELETE вилучає рядки з таблиці.

Оператор INSERT дозволяє додавати рядки за допомогою ключового слова VALUES або за допомогою оператора SELECT. Для кожного рядку, що додається, використовується свій оператор INSERT.

Порядок слідування стовпчиків в операторі INSERT може бути будь-яким, проте він повинний відповідати порядку слідування значень.

Якщо рядок, який додається, містить значення окремих стовпчиків таблиці, то стовпчики, що залишилися, (що не модифікуються) повинні припускати нульовий статус. У противному випадку команда буде заперечена.

Оператор SELECT у команді INSERT може містити агрегатні функції (Рисунок 3.5).

За допомогою команди DELETE із таблиці вилучається не окреме поле, а рядок цілком.

У предикаті команди DELETE можна використовувати підзапити.

Наприклад, для вилучання всіх покупців, що обслуговуються продавцями з Запоріжжя (Рисунок 3.5).

У предикаті команди UPDATE можна використовувати підзапити.

Наприклад, потрібно підвищити комісійні для всіх продавців, які обслуговують не менше двох покупців (Рисунок 3.5).

При відсутності речення WHERE зміни вносяться в усі значення стовпчиків, зазначених у SET.

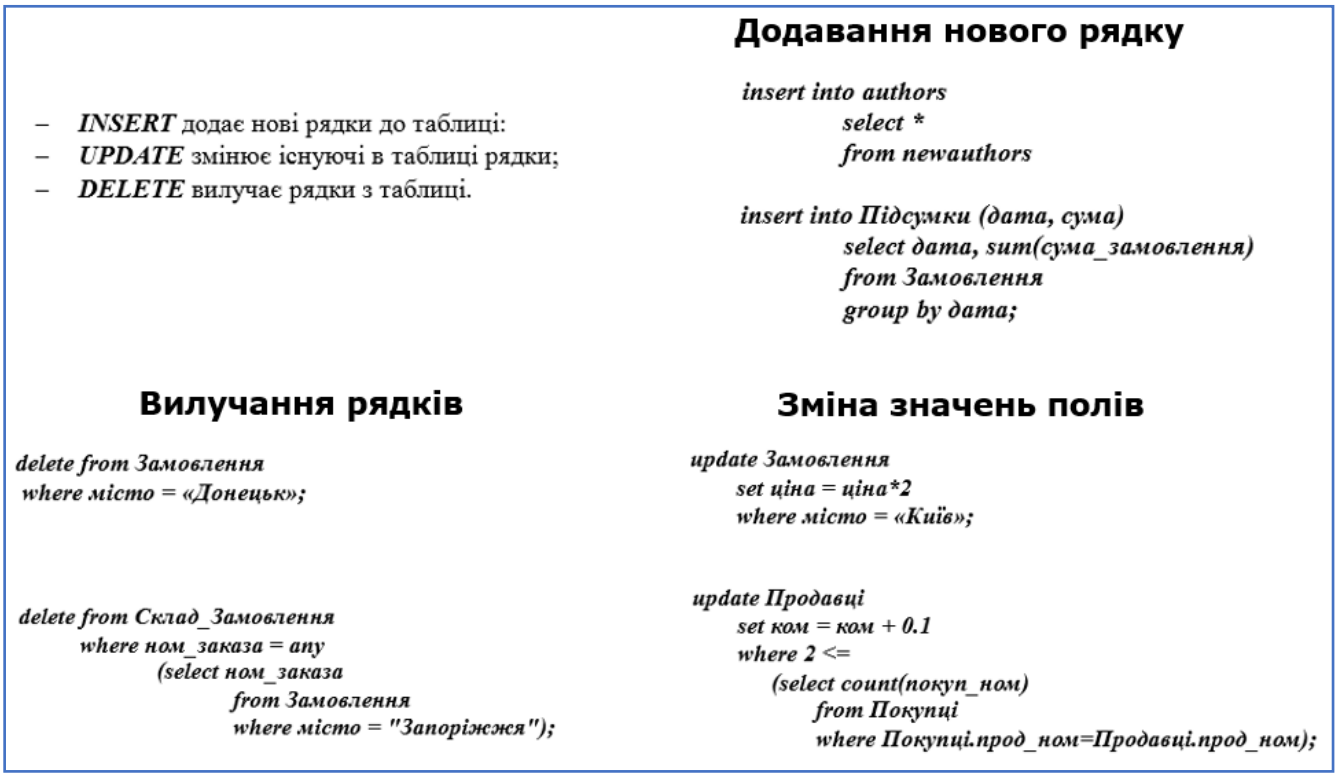

Рисунок 3.5 – приклади використання запитів на оновлення даних

4. Безпека та цілісність даних в реляційних СКБД

При експлуатації баз даних актуальною є проблема захисту інформації, пов'язана з можливістю виникнення наступних ситуацій:

- система може бути зруйнована під час виконання деяких програм, залишивши при цьому базу даних у непередбаченому стані;
- при виконанні конкуруючих програм між ними може виникнути конфлікт, який призведе до одержання неправильних результатів;
- дані можуть бути зіпсовані або змінені анонімним користувачем;
- оновлення можуть змінювати базу даних неприпустимим чином та інші.

Отже, система повинна забезпечувати функції захисту бази даних від подібних проблем, зокрема, відновлення, паралелізм, захист і цілісність.

Відновлення баз даних

Відновлення в системі баз даних означає, в першу чергу, відновлення самої бази даних, тобто повернення її в правильний стан, якщо якийсь збій зробив поточний стан неправильним або підозрілим.

Основний принцип, на якому будується таке відновлення, – це надмірність. Але ця надмірність будується на фізичному рівні та схована від користувача. Це означає, що, якщо будь-яка частина інформації, що міститься в базі даних, може бути реконструйована з іншої збереженої в системі надлишкової інформації, то база даних є такою, що відновлюється.

Транзакція – це логічна одиниця роботи, яка з точки зору відновлення даних розглядається та обробляється системою як єдина неподільна дія.

При цьому під транзакцією у загальному випадку розуміється не одиночна операція, а сукупність узгоджених операцій, що змінюють стан бази даних.

В стандарті ISO SQL/92 вказується, що транзакція автоматично запускається будь-яким оператором, який ініціює транзакцію та який виконується користувачем або програмою (наприклад, SELECT, INSERT або UPDATE). Зміни, внесені до бази даних в ході виконання даної транзакції, не будуть побачені будь-якими іншими транзакціями, що виконуються паралельно, доти, доки ця транзакція не буде явним чином завершена.

Системними називаються збої, що впливають на всі транзакції, що виконуються в даний момент, але не завдають фізичної шкоди самій базі даних. У випадку системного збою механізм відновлення системи баз даних повинний з'ясувати два питання:

– Які транзакції не встигли завершитися до моменту збою та, отже, повинні бути скасовані.

– Які транзакції встигли завершитися до моменту збою, але відповідна інформація ще не переписалася із внутрішніх буферів системи до самої бази даних (фізичної) та, отже, повинні бути виконані повторно.

Паралелізм

У більшості систем керування реляційними базами даних для вирішення проблем паралелізму застосовується механізм, який має назву блокування (locking). Принцип його роботи полягає в тому, що коли користувач вибирає якісь дані з БД, СКБД автоматично зачиняє їх «на замок» із тим, щоб ніякий інший користувач не міг оновлювати ці дані доти, доки блокування не буде зняте.

Існує два основні види блокування: блокування без взаємного доступу (монопольне блокування), що має також назву X-блокування (X lock – eXclusive lock) і блокування зі взаємним доступом (спільне блокування), що має назву S-блокування (S-lock – Shared lock).

X- і S-блокування називають також блокуваннями запису та читання, відповідно. Між цими видами блокування існують такі залежності:

1. Якщо транзакція A блокує кортеж p без можливості взаємного доступу (X-блокування), то запит іншої транзакції B із блокуванням цього кортежу p буде скасований.

2. Якщо транзакція A блокує кортеж p із можливістю взаємного доступу (S-блокування), то
– запит із боку деякої транзакції B на X-блокування кортежу p буде заборонений;
– запит із боку деякої транзакції B на S-блокування кортежу p буде прийнятий (тобто транзакція B також буде блокувати кортеж p за допомогою S-блокування).

Блокування в транзакціях зазвичай задаються неявним чином. Наприклад, запит «на витяг кортежу» є неявним запитом із S-блокуванням, а запит «на оновлення кортежу» – неявним запитом із X-блокуванням відповідного кортежу. Зокрема, у стандарті SQL не передбачене явне завдання блокування.

Привілеї

Привілеї (privileges)– це сукупність дій, які дозволено виконувати користувачеві по відношенню до конкретної таблиці або подання.

В стандарті ISO визначаються наступний набір привілеїв:

- SELECT – право вибирати дані з таблиці;
- INSERT – право додавати нові рядки до таблиці;
- UPDATE – право змінювати дані в таблиці;
- DELETE – право вилучати рядки з таблиці;
- REFERENCES – право посилатися на стовпці вказаної таблиці в описах вимог підтримки цілісності даних;
- USAGE – право використання доменів, перевірки наборів символів і трансляції.

Привілеї (права або повноваження) пов’язані як с користувачами, так і з таблицями. Коли користувач створює таблицю або подання, він автоматично стає її володарем і отримує по відношенню до неї повний набір привілеїв. Для надання доступу до цих об’єктів іншим користувачам необхідно, щоб володар явним чином надав їм ці права.

Контрольний слід виконання операцій

Якщо, наприклад, суперечливість даних призводить до підозри, що зроблено несанкціоноване втручання до бази даних, то контрольний слід повинний бути використаний для прояснення ситуації й підтвердження того, що всі процеси знаходяться під контролем. Якщо це не так, то контрольний слід допоможе, принаймні, виявити порушника.

Для зберігання контрольного сліду зазвичай використовується особливий файл, у якому система автоматично записує всі операції, що виконані користувачем при роботі з БД. У деяких системах контрольний слід може фізично зберігатися у файлі відновлення даних, а в інших – в окремому файлі. У будь-якому випадку користувачі повинні мати можливість звертатися до контрольного сліду за допомогою традиційної мови реляційних запитів (звичайно, за умови, що такі запити санкціоновані!).

Типовий запис у файлі контрольного сліду може містити таку інформацію:

- запит (вихідний текст запиту);
- термінал, з якого була викликана операція;
- користувач, який задав операцію;
- дата й час запуску операції;
- базові відношення, кортежі й атрибути, до яких зверталися в процесі виконання операції;
- старі значення;
- нові значення.

Безпека даних. Виборче керування доступом

У загальному випадку правила безпеки задаються п’ятьма компонентами (рисунок 4.1):

1. Ім’я правила, під яким воно буде зареєстровано в системному каталозі.

2. Одна або декілька привілеїв, які задані за допомогою директиви GRANT. У якості привілеїв можна використовувати одну з наведених на слайді:

- SELECT [(список_атрибутів_через_кому)] – витяг;
- INSERT – вставка;
- UPDATE [(список_атрибутів_через_кому)] – оновлення;
- DELETE – вилучення;
- ALL.

Специфікація ALL є скороченим записом використання всіх привілеїв – SELECT (із всіма атрибутами), INSERT, UPDATE (із всіма атрибутами) і DELETE.

3. Діапазон, до якого це правило застосовується та який є деякою підмножиною кортежів єдиного відношення. Він задається за допомогою директиви ON.

4. Один або декілька ідентифікаторів користувачів, які мають задані привілеї в заданому діапазоні, що вказується за допомогою директиви TO. Якщо задана директива ALL, то вона означає всіх відомих системі користувачів.

5. Реакція на порушення правил, яка диктує системі визначені дії, якщо користувач порушив правила безпеки. Вона задається директивою ON ATTEMPTED. По умовчанню приймається відмова у виконанні дії, що запитувалася. Проте в більш складних ситуаціях може виявитися корисним застосувати іншу дію; наприклад, у деяких випадках може знадобитися завершити виконання програми або заблокувати термінал.

Формат команди

CREATE SECURITY RULE правило
GRANT список_привілеїв_через_кому
ON вираз
TO список_користувачів_через_кому
[***ON ATTEMPTED VIOLATION*** дія];

Список привілеїв

- ***SELECT*** [(список_атрибутів_через_кому)] – витяг;
- ***INSERT*** – вставка;
- ***UPDATE*** [(список_атрибутів_через_кому)] – оновлення;
- ***DELETE*** – вилучення;
- ***ALL***.

Приклад 1

```
CREATE SECURITY RULE EX2
      GRANT SELECT (S#, SNAME, CITY)
      ON S
      TO Іваненко, Петренко, Сидоренко;
```

Приклад 2

```
CREATE SECURITY RULE EX3
      GRANT SELECT, INSERT, UPDATE (код_товару, назва),
            DELETE
      ON Товар WHERE Товар. ціна < 100$
      TO ALL;
```

Рисунок 4.1 – визначення правил безпеки

Цілісність даних. Мінімізація впливу людського фактору

Термін цілісність використовується для опису точності та коректності даних у БД. Як при забезпеченні безпеки, так і при підтримці цілісності система повинна містити відомості про ті правила, які користувачу не можна порушувати. При цьому СКБД повинна стежити за дотриманням заданих правил при виконанні користувачем деяких операцій. Відмінністю обмежень цілісності від правил безпеки є те, що перші не залежать від користувача.

Обмеження цілісності зручно класифікувати за чотирма типами (рисунок 4.2):

1. Обмеження цілісності домену визначають множину значень, що утворюють цей домен, тобто просто перераховуються всі припустимі значення домену.
2. Обмеження цілісності атрибута задаються у вигляді окремої частини визначення цього атрибута при створенні відношення (таблиці). Ці обмеження завжди перевіряються негайно, тобто будь-яка спроба (за допомогою операцій вставки INSERT або оновлення UPDATE) ввести до БД некоректне значення атрибута буде заборонена.
3. Обмеження цілісності відношення є правилом, що задається тільки для одного якогось відношення. При цьому вони також перевіряються негайно. *Наприклад, можна задати, що постачальники зі Львову мають статус 20.*
4. Обмеження цілісності бази даних є правилом, що задається для двох і більш пов'язаних між собою відношень. *Наприклад, обмеження, що означає, що постачальники зі статусом менше 20 не можуть поставляти товари в кількості більш 500.*

Приклади, тип 1

```
CREATE DOMAIN Колір CHAR (15)
FORALL Колір
        (Колір = «блакитний» OR
        Колір = «рожевий» OR'
        Колір = «чорний» OR Колір = «зелений»)

CREATE DOMAIN Колір CHAR (15)
  VALUES («блакитний», «рожевий», «чорний», «зелений»)
```

Приклади, тип 3

```
CREATE INTEGRITY RULE правило_5
FORALL Постачальник (IF Постачальник.місто= «Львів»
THEN Постачальник. статус =20)
ON ATTEMPTED VIOLATION REJECT;
```

Приклади, тип 4

```
CREATE INTEGRITY RULE правило_6
FORALL Постачальник (FORALL Постачання
(IF Постачальник.статус <20 AND Постачальник. Код_
        постачальника = Постачання. Код_постачальника
    THEN Постачання.Кількість ≤ 500 ))
    [ON ATTEMPTED VIOLATION REJECT];
```

```
CREATE INTEGRITY RULE правило_6
IF Постачальник.статус <20 AND Постачальник.Код_
        постачальника = Постачання. Код_ постачальника
THEN Постачання.Кількість ≤ 500
[ON ATTEMPTED VIOLATION REJECT];
```

Рисунок 4.2 – приклади правил підтримки цілісності даних

4. Розподілені бази даних

У зв'язку з широким поширенням комп'ютерних мереж і хмарових сервісів намітилася тенденція до децентралізації збереження та обробки даних.

Децентралізований підхід відображає організаційну структуру компанії, що логічно складається з окремих підрозділів, які фізично розподілені по різним офісам, відділенням або філіям, причому кожна окрема структурна одиниця має справу з особистим набором даних, які обробляються.

Розробка розподілених баз даних, які відображають організаційні структури підприємств, дозволяють зробити загальнодоступними дані, що підтримуються кожним підрозділом, забезпечивши при цьому їх зберігання саме в тих місцях, де вони частіше використовуються. Такий підхід розширює можливості спільного використання даних, одночасно підвищуючи ефективність доступу до них.

Розподілена база даних (РБД) – це сукупність логічно взаємозв'язаних баз даних, фізично розподілених у комп'ютерній мережі.

У розподіленій БД (Distributed Database – DDB) не всі дані зберігаються централізовано; вони розподілені по мережі вузлів, які відділені географічно, але зв'язані комунікаційними лініями. Кожний вузол має свою власну базу даних; крім того, він може звертатися до даних, які зберігаються на інших вузлах.

Розподілена СКБД (РСКБД) – це програмний комплекс, який призначений для керування розподіленими базами даних і який дозволяє зробити розподіленість системи прозорою для користувача.

Прозорість означає незалежність даних у розподілених системах, яка передбачає, що користувач у цій системі працює з розподіленою базою даних як з логічно цілісною сукупністю даних, тобто на його роботу не повинно впливати те, як дані розподілені між вузлами мережі.

Стратегії розподілу даних

Розподілена стратегія без дублювання. За такої стратегії визначають дані, які потрібно зберігати в кожному вузлі мережі. При цьому розподілену базу даних проектують як неперетинні між собою підмножини даних, розподілені по вузлах мережі. Проектування даних за такої стратегії є складною задачею.

Ключовим фактором, який впливає на надійність і доступність бази даних, є так звана локалізація посилань. Якщо БД розподілена так, що дані, розміщені в цьому вузлі, викликаються винятково його користувачем, то це свідчить про високий ступінь локалізації посилань.

Якщо подібне розчленування даних здійснити неможливо та для виконання запитів користувача потрібно звертатись за інформацією до інших вузлів, то це свідчить про невисокий ступінь локалізації посилань.

Розглянута стратегія підходить для тих предметних областей, в яких практично немає дублювання даних у різних вузлах мережі; користувач кожного вузла працює із своїми файлами та досить рідко використовує дані інших вузлів мережі.

Економічні задачі за своїми інформаційними властивостями характеризуються дуже тісними інформаційними взаємозв'язками, тому для даного класу задач реалізація цієї стратегії досить складна, неефективна й недоцільна.

Розподілена стратегія з дублюванням. Ця стратегія полягає в тому, що база даних проектується як при централізованому підході, але фізично дублюється в кожному вузлі мережі. Кожний вузол має свою копію, продубльовану стільки разів, скільки вузлів у мережі.

Стратегія розподілу з дублюванням найбільш ефективно розв'язує проблеми доступу та вибірки даних із мінімальними витратами часу. Система досить проста при проектуванні.

Переваги цієї стратегії полягають у тому, що зменшуються витрати на передавання інформації та вірогідність виникнення черг, коли кілька користувачів одночасно звертаються до одного файлу БД. Але водночас цю стратегію важко контролювати з точки зору дублювання даних, чим ускладнюється реалізація проблеми узгодженості та цілісності даних. Значно складнішими є проблеми адміністрування та підтримування БД даних в актуальному стані.

Однак разом з перевагами цей підхід характеризується складністю адміністрування та розв'язання проблеми узгодженості файлів БД у різних вузлах мережі. Ця проблема узгодженості досить гостро може постати тоді, коли зв'язок у мережі порушується та в копіях в різних вузлах виникають розбіжності. У цьому разі потрібно розробити спеціальний механізм для узгодження деяких копій БД.

Змішана стратегія розподілу даних поєднує два підходи, пов'язані з розподілом без дублювання та з дублюванням даних, з метою використання їх переваг. Ця стратегія поділяє БД на багато логічних фрагментів, як це зроблено в стратегії розподілу без дублювання.

Крім того, вона повинна дозволяти мати довільну кількість фізичних копій кожного фрагмента. Такий підхід до створення РБД дає змогу дублювати дані довільну кількість разів і в кожному вузлі; водночас у кожному вузлі може міститися потрібна частина бази даних. Система, побудована за цією стратегією, допускає досить просту реалізацію паралельної обробки даних, що скорочує час відгуку системи. Ця стратегія також забезпечує дуже велику надійність даних, за рахунок дублювання даних їх легко можна відновити при помилках чи збоях обладнання. Однак, як і при стратегії дублювання, виникає проблема узгодженості копій бази даних у всіх вузлах мережі.

Локальні дані – це дані, що підтримуються у власній базі даних вузла мережі.
Глобальні дані – це дані, що підтримуються в базі даних, розташування якої відрізняється від розташування хоча б одного з її користувачів.
Агент – це процес, який відповідає за виконання транзакції на конкретному вузлі.
Локальна транзакція – це транзакція, що складається з одного агента.
Глобальна транзакція – це транзакція, що складається з декількох агентів.

Підтримка цілісності даних в розподілених базах даних

В РБД внаслідок розподілення даних по окремим вузлам проблема підтримки цілісності дещо ускладнюється. У розподілених системах виділяють чотири типи збоїв:

- збій транзакції,
- збій вузла,
- збій носія (диска),
- збій комунікаційної лінії.

У всіх перелічених ситуаціях збою транзакції необхідно перервати та виконати відкат бази даних. Для попередження «зависання» системи необхідний контроль за часом виконання транзакції. Якщо ліміт часу, відведеного для виконання транзакції, перевищено, то її потрібно відмітити як таку, що підлягає аварійному завершенню незалежно від результату початкових кроків, і всі файли, задіяні при її виконанні, перевести в початковий стан.

Для підтримання цілісності бази даних при різного роду збоях використовують протоколи журналізації, що вміщують інформацію про всі зміни, які вносяться до бази даних під час виконання транзакції. В одну транзакцію доцільно об'єднувати операції, що виконують логічно зв'язані між собою зміни файлів бази даних. Підтримка транзакцій – це основа забезпечення цілісності БД. У сучасних розподілених СКБД підтримку транзакцій можна виконувати за допомогою двох методів:

1. двофазною фіксацією транзакцій;

2. реплікацією (дублюванням) даних.

При використанні методу з двофазною фіксацією, транзакцію виконують у два етапи, спираючись на такі правила:

1. Якщо хоча б один вузол не може з будь-яких причин зафіксувати транзакцію, то розподілена транзакція припиняється на всіх інших вузлах, які беруть участь у її виконанні.

2. Якщо всі вузли погоджуються з фіксацією транзакції, то вона фіксується на всіх вузлах, які беруть участь в її реалізації.

При **двофазній фіксації** транзакції працюють одночасно з однією розподіленою базою даних. У випадку з **дублюванням даних** кожна транзакція має змогу копіювати необхідні для її виконання дані з розподіленої бази даних і працювати з нею як із своєю особистою базою даних, модифікувати ці дані, а потім після виконання транзакції повертати їх до розподіленої бази даних.

6. Концепція сховищ даних

Сховище даних (DW) – це предметно-орієнтований, інтегрований, прив'язаний до часу та незмінний набір даних, призначений для підтримки процесу прийняття рішень.

Ідея, покладена в основу технології DW, полягає в тому, що проводити оперативний аналіз безпосередньо на базі оперативних інформаційних систем неефективно.

Замість цього, всі необхідні для аналізу дані витягаються із декількох традиційних баз даних, а також зовнішніх джерел (наприклад, статистичних звітів), перетворюються та потім розміщуються в одне джерело даних – DW.

Є ще одна причина, що виправдовує появу окремого сховища – складні аналітичні запити до оперативної інформації гальмують поточну роботу компанії, надовго блокуючи таблиці та захоплюючи ресурси сервера.

Кінцевою метою створення сховища даних є інтеграція корпоративних даних у єдиному репозиторії, звертаючись до якого користувачі зможуть формувати запити, генерувати звіти й виконувати аналіз даних.

Рисунок 6.1 – приклад побудови DW з використанням OLAP кубу

Дані, що надходять у сховище, можуть бути неузгодженими, фрагментованими, схильними до змін, містити дублікати або пропуски. Тому в процесі завантаження даних до сховища вони за допомогою менеджера завантаження:

- очищуються (усунення непотрібної інформації);
- агрегуються (обчислення сум, середніх значень, тощо);

 трансформуються (перетворення типів даних, реорганізація структур зберігання);

 поєднуються із зовнішніх і внутрішніх джерел (приведення до єдиних форматів);

 синхронізуються (відповідність одному моменту часу).

Доставка даних кінцевому користувачеві може бути або прямою, або через інформаційні вітрини – «магазини даних» (data mart – DM).

7. Бази даних NoSQL

Бази даних NoSQL використовують різноманітні моделі даних для доступу до інформації та управління нею. Бази даних таких типів оптимізовані спеціально для додатків, які потребують гнучких моделей даних, а також для роботи з частково структурованими або слабоструктурованими даними. Усе це досягається шляхом пом'якшення жорстких вимог до узгодженості даних, притаманних реляційних типів баз даних.

Залежно від моделі даних, реалізація може відрізнятися. Однак у багатьох базах даних NoSQL використовується текстовий формат обміну даними, що базується на мові Javascript (JSON), – відкритий формат обміну даними, що представляє дані у вигляді набору пар «ім'я-значення».

Існує кілька різних систем баз даних NoSQL у зв'язку з відмінностями у способах роботи та зберігання даних без схем.

Бази даних «ключ-значення»

База даних "ключ-значення" зберігає дані як сукупність пар "ключ-значення", в яких ключ служить унікальним ідентифікатором. Ключі і значення можуть бути будь-що: від простих до складних складових об'єктів.

Бази даних документів

У баз даних документів той самий формат моделі документа, який розробники використовують у коді своїх додатків. Такі бази даних зберігають інформацію у вигляді гнучких, напівструктурованих та ієрархічних за своєю природою об'єктів JSON.

Графові бази даних

Графові бази даних призначені для спрощення розробки та запуску додатків, що працюють із наборами тісно пов'язаних даних. У таких базах даних використовуються вузли для зберігання сутностей інформації та ребра для зберігання взаємозв'язків між цими сутностями. Ребро завжди має початковий і кінцевий вузол, тип та напрямок. Ребра можуть описувати взаємозв'язки типу «предок-нащадок», дії, права володіння тощо.

Переваги використання NoSQL баз даних

NoSQL бази даних можна використовувати в додатках, в яких:

- потрібні гнучкі схеми, що забезпечують більш швидку та ітеративну розробку;
- надається перевага продуктивності, а не висока узгодженість даних і збереження зв'язків між таблицями даних (посилальна цілісність);
- потрібне горизонтальне масштабування шляхом сегментування між серверами;
- підтримується частково структуровані чи неструктуровані дані.