МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАШИНОБУДІВНИЙ КОЛЕДЖ
ДОНБАСЬКОЇ ДЕРЖАВНОЇ МАШИНОБУДІВНОЇ АКАДЕМІЇ

ЦИКЛОВА КОМІСІЯ КОМП`ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ

КОНСПЕКТ ЛЕКЦІЙ

з дисциплін
«Бази даних»,
«Організація баз даних та знань»

для студентів денної та заочної форм навчання спеціальностей:
5.05010101 – Обслуговування програмних систем і комплексів
(122 – Комп`ютерні науки)
5.05010301 – Розробка програмного забезпечення
(121 – Інженерія програмного забезпечення)

Краматорськ
2018

Конспект лекцій з дисциплін «Бази даних», «Організація баз даних та знань». для студентів спеціальностей 5.05010101 – Обслуговування програмних систем і комплексів (122 – Комп`ютерні науки) та 5.05010301 – Розробка програмного забезпечення (121 – Інженерія програмного забезпечення). Денна та заочна форми навчання. Включає теоретичний матеріал з курсу «Бази даних», «Організація баз даних та знань» / Укл. Ахромов М.О. – Краматорськ: МК ДГМА, 2017. – 148 с.

Укладач: Ахромов М.О., викладач кваліфікаційної категорії «спеціаліст вищої категорії»

РОЗГЛЯНУТО ТА СХВАЛЕНО
на засіданні циклової комісії комп’ютерно-інтегрованих технологій

Протокол від 28 серпня 2018 р.
№ 1
Голова циклової комісії

_____ Н.В. Новікова

ЗМІСТ

Лекція

Тема: ***«Інформація, дані та інформаційні системи»***

План лекції

1 Інформація й дані
2 Інформаційні системи
3 Основні підходи до обробки інформації в автоматизованих інформаційних системах

Зміст лекції

1 Інформація й дані

Перш ніж перейти до обговорення поняття інформаційної системи (ІС), спробуємо з'ясувати, що ж розуміється під словом "інформація". Відповісти на це питання і просто, і складно: слово "інформація" пов'язане із широким колом понять.

Змістовна сторона поняття "інформація" дуже багатогранна й не має чітких семантичних меж. Однак завжди можна сказати, що можна з нею робити. Саме відповідь на це запитання найчастіше й цікавить як системних аналітиків і розроблювачів ІС, так і користувачів інформації (її основних споживачів).

З погляду як користувачів, так і розроблювачів ІС в інформації є одна важлива властивість - вона є одиницею даних, яка підлягає обробці. Звичайно інформація надходить споживачеві саме у вигляді даних: таблиць, графіків, малюнків, фільмів, усних повідомлень, які фіксують у собі інформацію певної структури й типу. Таким чином, дані виступають як засіб подання інформації у певній, фіксованій формі, придатній для обробки, зберігання й передачі. Хоча дуже часто терміни "інформація" й "дані" виступають як синоніми, варто пам'ятати про цю їхню істотну відмінність. Саме в даних інформація знаходить інтерпретацію у конкретній ІС.

При згадуванні про "форму" подання інформації варто сказати ще про одну, "людську" властивість інформації - її сприйняття різними категоріями людей. Дані можуть бути згруповані спільно у документ. Документ може мати або не мати певної внутрішньої структури. Дані можуть бути відображені на екрані дисплея комп'ютера. Документи можуть мати аудіо- або відеоформу. Розробляючи ІС, ніколи не слід забувати, для кого вони (системи) створюються і хто буде їх використовувати. Форма подання інформації в ІС визначає також і категорії користувачів. ІС створюються для конкретних груп користувачів, тобто вони, як правило, проблемно-орієнтовані.

Інформація є даними, яким надається деякий зміст (інтерпретація) у конкретній ситуації у рамках деякої системи понять. Інформація подається за допомогою кодування даних і здобувається шляхом їхнього декодування та інтерпретації.

У цьому визначенні фіксується три основних перетворення інформації й даних у процесі їхньої обробки в ІС: інформація – дані, дані – дані, дані – інформація.

На рисунку 1 наведені дві сторони визначення поняття інформації: функціональна й представницька. Перша загалом визначає коло дій над інформацією, а друга – результат виконання цих дій.

Рисунок 1 – Зміст поняття "інформація"

2 Інформаційні системи

Основною метою створення ІС є задоволення інформаційних потреб користувачів шляхом надання необхідної їм інформації на основі збережених даних. Потреба в інформації як такій не вичерпує поняття інформаційних потреб. Звичайно в поняття інформаційних потреб включають певні вимоги до якості інформаційного обслуговування й поводження системи в цілому (продуктивність, актуальність і надійність даних, орієнтація на користувача та ін.).

Під інформаційною системою розуміється організаційна сукупність технічних засобів, технологічних процесів і кадрів, що реалізують функції збору, обробки, зберігання, пошуку, видачі й передачі інформації.

Необхідність підвищення продуктивності праці у сфері інформаційної діяльності приводить до того, що в якості зовнішніх засобів зберігання й швидкого доступу до інформації найчастіше використовуються засоби обчислювальної техніки (цифровий і аналоговий) на основі комп'ютерів. Сучасні ІС - складні комплекси апаратних і програмних засобів, технології й персоналу, які ще називають автоматизованими інформаційними системами.

Структурно ІС містять у собі апаратне (hardware), програмне (software), комунікаційне (netware), проміжного шару (middleware), лінгвістичне й організаційно-технологічне забезпечення.

Апаратне забезпечення ІС містить у собі широкий набір засобів обчислювальної техніки, передачі даних, а також цілий ряд спеціальних технічних пристроїв (пристрою графічного відображення інформації, аудио- і відеопристрою, засобу мовного уведення й т.д.). Апаратне забезпечення є основою будь-якої ІС.

Комунікаційне (мережне) забезпечення містить у собі комплекс апаратних мережних комунікацій і програмних засобів підтримки комунікацій в ІС. Воно має істотне значення при створенні розподілених ІС й ІС на основі Інтернету.

Програмне забезпечення ІС забезпечує реалізацію функцій введення даних, їх розміщення на носіях, модифікації даних, доступ до даних, підтримку функціонування устаткування. Програмне забезпечення можна розділити на системне (яке завершує процес вибору апаратно-програмного рішення або платформи) і користувацьке (яке застосовується для вирішення завдань задоволення потреб користувача у комп'ютерному середовищі).

Лінгвістичне забезпечення ІС призначене для вирішення завдань формалізації змісту повнотекстової і спеціальної інформації для створення пошукового образу даних (профілю). У класичному змісті звичайно воно включає процедури індексування текстів, їхню класифікацію і тематичну рубрикацію. Найчастіше ІС, що містять складно-структуровану інформацію, містять у собі тезауруси термінів і понять. Сюди можна віднести й створення процесорів спеціалізованих формальних мов кінцевих користувачів, наприклад, мов для маніпулювання бухгалтерською інформацією і т.д. Найчастіше працям з розроблення лінгвістичного забезпечення не надається належного значення. Подібні недогляди найчастіше призводять до несприйняття користувачами самої інформації. Це стосується в першу чергу вузько спеціалізованих ІС.

У міру зростання складності і масштабів ІС важливу роль починає відігравати організаційно-технологічне забезпечення, що з'єднує різнорідні компоненти (апаратури, програми й персонал) у єдину систему й забезпечує процедури її керування й функціонування. Недооцінка цієї складової ІС найчастіше призводить до зриву строків впровадження системи й виведення її на виробничі потужності.

На рисунку 2 наведені функції ІС через її основні структурні компоненти.

Рисунок 2 – Визначення інформаційної системи

3 Основні підходи до обробки інформації в автоматизованих інформаційних системах

Одним з головних питань розроблення програмного забезпечення ІС є питання про співвіднесення програм і даних, тому що вирішення цього питання, в остаточному підсумку, визначає вибір алгоритмів обробки інформації, апаратних засобів і технологічної платформи. Фундаментальним принципом у вирішенні питання про співвіднесення програм і даних є концепція незалежності прикладних програм від даних, і неважливо, яка обробка даних передбачається: централізована або розподілена. Суть цієї концепції полягає не стільки у відділенні програм від даних, скільки у розгляді їх як самостійних взаємодіючих об'єктів.

Однією з останніх модифікацій цього принципу є концепція незалежності прикладних програм від даних разом із процедурами їхньої обробки (об’єктно-орієнтований підхід у програмуванні), що дозволяє вирішити ряд питань обробки даних, пов'язаних з інтерпретацією семантичного змісту даних.

Формування концепції БД і створення на її основі методу баз даних для вирішення завдань обробки інформації відбулося у 1962 році. До середини 60-х років минулого століття основною концепцією побудови програмного забезпечення були концепція файлової системи і так званий позадачний метод. Наприкінці 80-х років минулого століття була запропонована концепція об’єктно-орієнтованих баз даних й об’єктно-орієнтований підхід розроблення програм на основі обробки подій. На рисунок 3 наведені основні ознаки для кожної з зазначених вище концепцій. На рисунок 4 проведене зіставлення основних методів обробки даних.

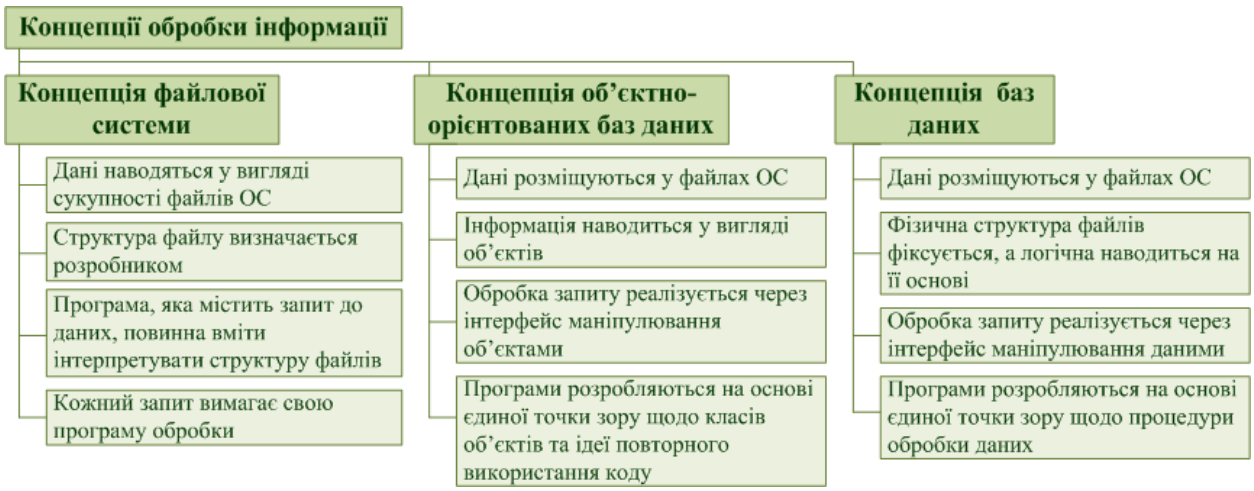


Рисунок 3 – Основні концепції обробки інформації

Рисунок 4 – Основні проблеми методів обробки інформації

Основний зміст позадачного методу зводиться до декомпозиції програми зі своїми окремими блоками даних та алгоритмами; методу баз даних – до наявних окремих описів логічної структури даних та єдиної точки зору щодо процедури обробки даних; об'єктно- орієнтованого методу – полягає в тому, що програми розглядаються як сукупність об'єктів, між якими відбувається обмін інформацією.

Об'єкту притаманні такі властивості:

- інкапсуляція – об'єкти наділяються структурою й мають певне поводження (набором операцій). Операції над об'єктами становлять його методи. Структура об'єкта захована від користувача, що маніпулює об'єктом через його операції. Об'єкт розглядається як абстракція реального світу Для того щоб об'єкт виконав деяку дію, йому потрібно надіслати повідомлення. Об'єкт взаємодіє з іншими об'єктами через події;
- спадкування – являє собою механізм, що дозволяє робити одні об'єкти з інших, при цьому властивості батьківського об'єкта зберігаються у нащадка;

- поліморфізм – різні об'єкти можуть одержувати однакові повідомлення, але реагувати на них по-різному відповідно до реалізації своїх однойменних методів.

Отже, при розробленні автоматизованих ІС слід пам’ятати:

- ІС створюються для конкретних груп користувачів, тобто вони є проблемно-орієнтовані;
- основною метою створення ІС є задоволення інформаційних потреб користувачів шляхом надання необхідної їм інформації на основі збережених даних;
- сучасні ІС - складні комплекси апаратних і програмних засобів, технології й персоналу, які ще називають автоматизованими інформаційними системами;
- дотримання концепції незалежності прикладних програм від даних разом із процедурами їхньої обробки (об’єктно-орієнтований підхід у програмуванні) дозволяє вирішити ряд питань обробки даних, пов'язаних з інтерпретацією семантичного змісту даних;
- ІС працює з упорядкованими взаємозалежними даними, які зберігаються з мінімальною надлишковістю та становлять базу даних. Системою керування базами даних називається сукупність програмних засобів, необхідних для використання БД і подання розробникам і користувачам безлічі різних подань даних.

Лекція

Тема: ***«Загальні поняття баз даних та предметної області»***

План лекції

1 Базові поняття баз даних
2 Поняття предметної області

Зміст лекції

1 Базові поняття баз даних

Використання баз даних і інформаційних систем стає невід'ємній складовій ділової діяльності сучасної людини і функціонування процвітаючих організацій. У зв'язку з цим великої актуальності набуває освоєння принципів побудови і ефективного вживання відповідних технологій і програмних продуктів: систем управління базами даних, CASE-систем автоматизації проектування і інших.

Дані – це інформація, представлена в певному вигляді, яка дозволяє автоматизувати її збір, зберігання і подальшу обробку людиною або інформаційним засобом. Для комп'ютерних технологій дані – це інформація у дискретному, фіксованому вигляді, зручна для зберігання, обробки на ЕОМ, а також для передачі по каналами зв'язку.

Банк даних є різновидом ІС, в якій реалізовані функції централізованого зберігання і накопичення оброблюваної інформації, організованої в одну або декілька баз даних.

Банк даних (БНД) в загальному випадку складається з наступних компонентів: бази (декількох баз) даних, системи управління базами даних, словника даних, адміністратора, обчислювальної системи і обслуговуючого персоналу.

База даних (БД) є сукупністю спеціальним чином організованих даних, що зберігаються в пам'яті обчислювальної системи і об'єктів, що відображують стан, і їх взаємозв'язків в даної наочної області.

Логічну структуру яка зберігається в базі даних називають моделлю представлення даних. До основних моделей представлення даних (моделям даних) відносяться наступні: ієрархічна, мережева, реляційна, реляційна для поста, багатовимірна і об'єктно-орієнтована.

Додаток є програмою або комплексом програм, що забезпечують автоматизацію обробки інформації для прикладного завдання. Нами розглядаються додатки, що використовують БД. Додатки можуть створюватися в середовищі або поза середовищем СКБД — за допомогою

системи програмування, що використовує засоби доступу до БД, наприклад, Delphi або C++ Builder. Додатки, розроблені в середовищі СКБД часто називають додатками СУБД, а додатки, розроблені зовні СУБД, — зовнішніми застосуваннями.

Для роботи з базою даних частенько досить засобів СКБД і не потрібно використовувати додатки, створення яких вимагає програмування. Додатки розробляють головним чином у випадках, коли потрібно забезпечити зручність роботи з БД некваліфікованим користувачам або інтерфейс СКБД не владнує користувачів.

Словник даних (СД) є підсистемою БНД, призначеною для централізованого зберігання інформації про структури даних, взаємозв'язках файлів БД один з одним, типах даних і форматах їх вистави, приладдя даних користувачам, кодах захисту і розмежування доступу і тому подібне

Функціонально СД присутній у всіх БНД, але що не завжди виконує ці функції компонент має саме такий назву. Найчастіше функції СД виконуються СКБД і викликаються з основного меню системи або реалізуються за допомогою її утиліт.

Адміністратор бази даних (АБД) є особа або група осіб, що відповідають за вироблення вимог до БД, її проектування, створення, ефективне використання і супровід.

Функції адміністратора бази даних

− проектування й розробка описів даних на різних рівних та їхнє узгодження.

− розробка структур зберігання і стратегій доступу до даних згідно а вимогами

− ефективного (чи економічного) зберігання, швидкої обробки та виведення даних за бажанням користувача,

− реструктуризація ти реорганізація БД у випадку зміни вимог до характеристик зберігання й обробки даних

− проектування та застосування механізмів захисту даних

− реєстрація користувачів (Імен, паролів), визначення їхніх прав доступу та повноважень

− розробка й використання механізмів резервного копіювання даних та їхнє відновлення під час перебоїв.

− настроювання роботи БД (підвищення продуктивності механізмів обробки даних, підтримання планової надлишковості забезпечення ефективності їхнього зберігання),

− систематичне наглядання за використанням бази даних.

У обчислювальній мережі АБД, як правило, взаємодіє з адміністратором мережі. У обов'язку останнього входять контроль за функціонуванням апаратно-програмних засобів мережі, реконфігурація мережі, відновлення програмного забезпечення після збоїв і відмов устаткування, профілактичні заходи і забезпечення розмежування доступу.

Обчислювальна система (ВС) є сукупністю взаємозв'язаних і погоджено діючих ЕОМ або процесорів і інших пристроїв, що забезпечують автоматизацію процесів прийому, обробки і видачі інформації споживачам.

Оскільки основними функціями БНД є зберігання і обробка даних, то використовувана ВС, разом з прийнятною потужністю центральних процесорів (ЦП) повинна мати достатній об'єм оперативної і зовнішньої пам'яті прямого доступу.

Обслуговуючий персонал виконує функції підтримки технічних і програмних засобів в працездатному стані. Він проводить профілактичні, регламентні, відновні і інші роботи по планах, а також в міру необхідності.

2 Поняття предметної області

Основним призначенням інформаційної системи (ІС) є оперативне забезпечення користувача інформацією про зовнішній світ шляхом реалізації питально-відповідного відношення. Питально-відповідні відношення дозволяють виділити для ІС певний її фрагмент - предметну область, - який буде втілений в автоматизованій ІС. Інформація про зовнішній світ подається в ІС у формі даних, що обмежує можливості змістовної інтерпретації інформації й конкретизує семантику її подання в ІС. Сукупність цих виділених для ІС даних, зв'язків між ними й операцій над ними утворить інформаційну й функціональну моделі предметної області (ПО), що описують її стан з певною точністю. Інформаційна й функціональна моделі ПО є вхідними даними для процесу проектування БД.

Сукупність реалій (об'єктів) зовнішнього світу - об'єктів, про які можна питати, - утворює об'єктне ядро предметної області, яке має онтологічний статус. Не можна одержати в ІС відповідь на запитання про те, що їй невідомо. Термін "об'єкт" є первинним поняттям. Синонімами терміна "об'єкт" є "реалія, сутність, річ". Сутність ПО є результатом абстрагування реального об'єкта шляхом виділення й фіксації набору його властивостей.

Прикладами сутностей (з погляду ІС) або об'єктів (з погляду зовнішнього світу) є окремий студент, група студентів, аудиторія, час занять, слова, числа, символи. Звичайно вважається, що бути об'єктом - означає бути дискретним і помітним.

З об'єктами пов'язано дві проблеми: ідентифікація й адекватний опис. Для ідентифікації використовують ім'я. Використовується тільки вказівна функція імені. Ім'я - це прямий спосіб ідентифікації об'єкта. До непрямих способів ідентифікації об'єкта відносять визначення об'єкта через його властивості (характеристики або ознаки).

Об'єкти взаємодіють між собою через свої властивості, що породжує ситуації. Ситуації - це взаємозв’язки, які виражають взаємини між об'єктами. Ситуації у ПО описуються за допомогою висловлювань про ПО з

використанням виразів і обчисленнями предикатів, тобто формальної, математичної логіки.

Методи математичної логіки дозволяють формалізувати ці твердження і подати їх у вигляді, придатному для аналізу.

Розглянемо висловлювання: студент Іванов А.А, народився у 1982 році. Воно виражає такі властивості об'єкта "Іванов А.А.":

- у явному вигляді - рік народження;
- у неявному вигляді - приналежність до студентів.

Перша властивість встановлює зв'язок між об'єктами "Іванов А.А." і "Рік народження", а друга - між об'єктами "Іванов А.А." і "Безліч студентів". Формалізація цього висловлювання представляється як результат присвоювання значень змінним, які входять у предикати:

НАРОДИВСЯ (Іванов А.А., 1982)

Є СТУДЕНТОМ (Іванов А.А.)

На рисунку 1 наведений один із підходів до класифікації ситуацій у рамках ПО.

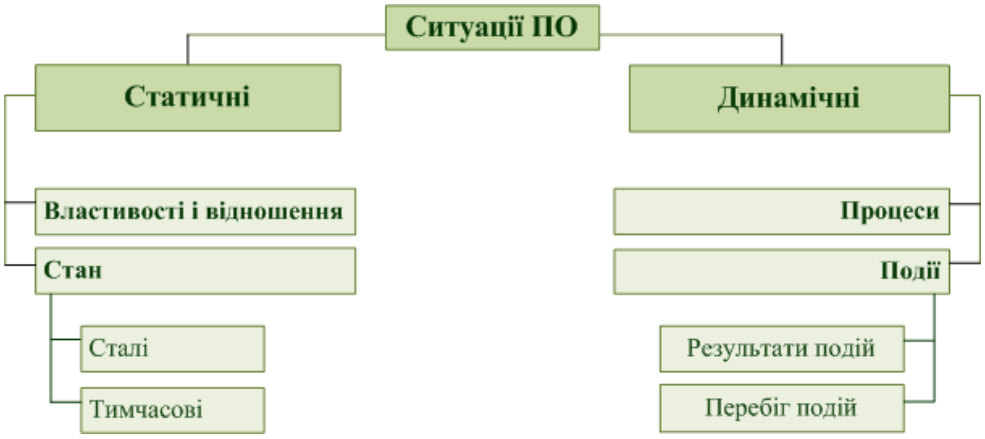


Рисунок 1 – Класифікація ситуацій ПО

Розрізняють статичні й динамічні ситуації. Прикладами статичних ситуацій є такі ситуації, як кольори, вік. Прикладами динамічних ситуацій є такі ситуації, як випекти хліб.

Наведена класифікація вводить у ПО два важливі аспекти - простір і час, до того ж час і як момент, і як інтервал. ПО існує у просторі і часі, тобто їй притаманні часові і просторові відношення і зв’язки. Слід розрізняти реальний час зовнішнього світу та його відображення у БД та у джерелах інформації. У БД взаємозв’язки, залежні від часу, фіксуються тільки після реєстрації в БД. Таким чином, ПО у кожний певний момент часу являє собою відокремлену сукупність визначених об’єктів і ситуацій, яку називають станом ПО.

Предметна область - це цілеспрямована первинна трансформація картини зовнішнього світу у деяку картину, певна частина якої фіксується в ІС як алгоритмічна модель фрагменту дійсності.

Лекція

Тема: ***«Системи управління базами даних»***

План лекції

1 Поняття та функції системи керування базами даних
2 Класифікація СУБД

Зміст лекції

1 Поняття та функції системи керування базами даних

Система керування базами даних (СКБД) – це комплекс мовних і програмних засобів, призначений для створення, ведення і спільного використання БД багатьма користувачами. Зазвичай СКБД розрізняють по використовуваній моделі даних. Так, СКБД, засновані на використанні реляційної моделі даних, називають реляційними СКБД.

Одними з перших СКБД є наступні системи: IMS (IBM, 1968т.), IDMS (Cullinet, 1971 р.), ADABAS (Software AG, 1969 р.) і ІНЕС (ВНІЇСИ АН СРСР, 1976 р.). Кількість сучасних систем керування базами даних обчислюється тисячами.

СКБД відноситься до спеціального забезпечення комп'ютера. Він є інструментом для розробки БД.

Функції СУБД:

– опис даних.

В БД існують досить жорсткі обмеження по маніпулюванню даними які відображають певну область. Як правило задаються описи незмінних властивостей в БД. Такий опис називається описом структури БД або схемою БД.

– маніпулювання даними.

До складу СКБД входять оператори пошуку в БД, керування даними, обміном даними та ін..

– заповнення БД та генератор звітів.

Більшість СКБД мають спеціальні засоби для введення та керування даними, для одержання вихідних форм (звітів).

– діалогові засоби та мова запитів.

Для зручності роботи користувачів і підвищення оперативності доступу до даних більшість функцій СКБД може здійснюватися через дисплей (діалоговий режим). За допомогою дисплею здійснюється перегляд БД, керування нею, введення запитів до БД які формуються на спеціальній мові.

– сервіс.

Тут розуміють ряд функцій, що забезпечують цілісність БД та різного роду довідкових функцій. Одна з важливих сервісних функцій є журнал. Системний журнал засіб поновлення БД на довільний момент в минулому. Робота Системного журналу основана на тому, що в процесі існування СКБД запам’ятовуються всі зміни які проводяться в БД, це дає можливість поновити БД в момент її цілісності.

2 Класифікація СУБД

Зазвичай сучасна СУБД містить наступні компоненти (рисунок 1.1):

- ядро, яке відповідає за управління даними в зовнішній і оперативній пам'яті і журналізацію
- процесор мови бази даних, що забезпечує оптимізацію запитів на витягання і зміну даних і створення, як правило, машинно-незалежної виконуваної внутрішньої коди
- підсистему підтримки часу виконання, яка інтерпретує програми маніпуляції даними, що створюють призначений для користувача інтерфейс з СУБД
- а також сервісні програми (зовнішні утиліти), що забезпечують ряд додаткових можливостей по обслуговуванню інформаційної системи.

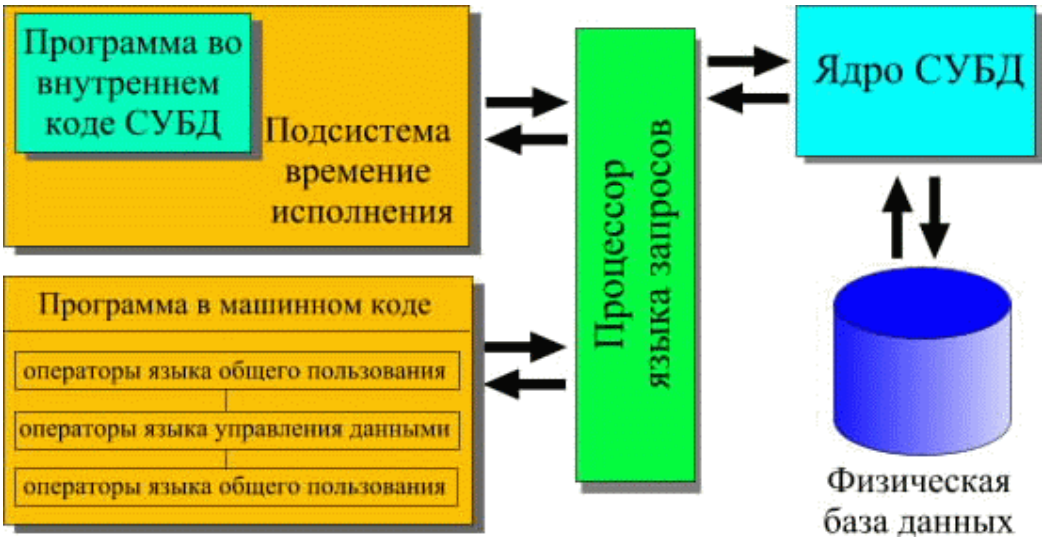


Рисунок 1.1 – Компоненти СУБД

Класифікація СУБД. У загальному випадку під СУБД можна розуміти будь-який програмний продукт, що підтримує процеси створення, ведення і використання БД.

До СУБД відносяться наступні основні види програм:

- повнофункціональні СУБД;
- сервери БД;
- клієнти БД;
- засоби розробки програм роботи з БД.

Повнофункціональні СУБД (ПФСУБД) є традиційними СУБД, які спочатку з'явилися для великих машин, потім для міні-машин і для ПЕВМ. З числа всіх СУБД сучасні ПФСУБД є найбільш багаточисельними і потужними по своїх можливостях. До ПФСУБД відносяться, наприклад, такі пакети як: DataEase, DataFlex, dBase IV, Microsoft Access, Microsoft FoxPro і Paradox.

Зазвичай ПФСУБД мають розвинений інтерфейс, що дозволяє за допомогою команд меню виконувати основні дії з БД: створювати і модифікувати структури таблиць, вводити дані, формувати запити, розробляти звіти, виводити їх на друк і тому подібне Для створення запитів і звітів не обов'язкове програмування, а зручно користуватися мовою QBE (Query By Example — формулювання запитів за зразком). Багато ПФСУБД включають засоби програмування для професійних розробників.

Сервери БД призначені для організації центрів обробки даних в мережах ЕОМ. Ця група БД в даний час менш багаточисельна, але їх кількість поступово зростає. Сервери БД реалізують функції управління базами даних, запрошувані іншими (клієнтськими) програмами зазвичай за допомогою операторів SQL.

Прикладами серверів БД є наступні програми: NetWare SQL (Novell), MS SQL Server (Microsoft), InterBase (Borland), SQLBase Server (Gupta), Intelligent Database (Ingress).

В ролі клієнтських програм для серверів БД в загальному випадку можуть використовуватися різні програми: ПФСУБД, електронні таблиці, текстові процесори, програми електронної пошти і так далі При цьому елементи пари «клієнт — сервер» можуть належати одному або різним виробникам програмного забезпечення.

Засоби розробки програм роботи з БД можуть використовуватися для створення різновидів наступних програм:

- клієнтських програм;
- серверів БД і їх окремих компонентів;
- призначених для користувача застосувань.

Програми першого і другого вигляду досить нечисленні, оскільки призначені, головним чином, для системних програмістів. Пакетів третього вигляду значно більше, але менше, ніж повнофункціональних СУБД.

До засобів розробки призначених для користувача застосувань відносяться системи програмування, наприклад Clipper, всілякі бібліотеки програм для різних мов програмування, а також пакети автоматизації розробок (у тому числі систем типа клієнт-сервер). У числі найбільш поширених можна назвати наступні інструментальні системи: Delphi і Power Builder (Borland), Visual Basic (Microsoft), SILVERRUN (Computer Advisers Inc.), S-Designor (SDP і Powersoft) і ERwin (LogicWorks).

По характеру використання СУБД ділять на персональних і розрахованих на багато користувачів.

Персональні СУБД зазвичай забезпечують можливість створення персональних БД і недорогих застосувань, що працюють з ними. Персональні

СУБД або розроблені з їх допомогою застосування частенько можуть виступати в ролі клієнтської частини розрахованої на багато користувачів СУБД. До персональних СУБД, наприклад, відносяться Visual FoxPro, Paradox, Clipper, dBase, Access і ін.

Розраховані на багато користувачів СУБД включають сервер БД і клієнтську частину і, як правило, можуть працювати в неоднорідному обчислювальному середовищі (з різними типами ЕОМ і операційними системами). До розрахованих на багато користувачів СУБД відносяться, наприклад, СУБД Oracle і Informix.

По використовуваній моделі даних СУБД (як і БД), розділяють на ієрархічних, мережевих, реляційних, об'єктно-орієнтованих і інші типи. Деякі СУБД можуть одночасно підтримувати декілька моделей даних.

З точки зору користувача, СУБД реалізує функції зберігання, зміни (поповнення, редагування і видалення) і обробки інформації, а також розробки і здобуття різних вихідних документів.

Для роботи з тією, що зберігається в базі даних інформацією СУБД надає програмам і користувачам наступних двох типів мов:

– мова опису даних — високорівнева непроцедурна мова декларативного типа, призначена для опису логічної структури даних;

– мова маніпулювання даними — сукупність конструкцій, що забезпечують виконання основних операцій по роботі з даними: введення, модифікацію і вибірку даних по запитах.

Перераховані вище функції СУБД, у свою чергу, використовують наступні основні функції нижчого рівня, які назвемо низькорівневими:

– управління даними в зовнішній пам'яті;
– управління буферами оперативної пам'яті;
– управління транзакціями;
– ведення журналу змін в БД;
– забезпечення цілісності і безпеки БД.

Забезпечення цілісності БД складає необхідну умову успішного функціонування БД, особливо для випадку використання БД в мережах. Цілісність БД є властивість бази даних, що означає, що в ній міститься повна, несуперечлива і така, що адекватно відображає наочну область інформація. Підтримка цілісності БД включає перевірку цілісності і її відновлення в разі виявлення протиріч в базі даних. Цілісний стан БД описується за допомогою обмежень цілісності у вигляді умов, яким повинні задовольняти що зберігаються в базі дані.

Прикладом таких умов може служити обмеження діапазонів можливих значень атрибутів об'єктів, відомості про яких зберігаються до БД, або відсутності записів, що повторюються, в таблицях реляційних БД.

Забезпечення безпеки досягається в СУБД шифруванням прикладних програм, даних, захисту паролем, підтримкою рівнів доступу до бази даних і до окремих її елементів (таблицям, формам, звітам і т. д.).

Лекція

Тема: ***«Життєвий цикл та методологія проектування»***

План лекції

1 Життєвий цикл баз даних
2 Методологія проектування
    2.1 Процес проектування.
    2.2 Критерії оцінювання
    2.3 Інформаційні вимоги

Зміст лекції

1 Життєвий цикл баз даних

Процес створення такої структури бази даних, яка б відповідала вимогам користувачів, називається проектуванням бази даних. Його можна порівняти зі зведенням нової будівлі: визначення вимог, проектування, конструювання і, нарешті, реалізація.

Життєвий цикл системи баз даних є концепцією, в межах якої корисно й зручно розглядати розвиток такої системи. Він, як і життєвий цикл будь-якої програмної системи, складається з двох основних фаз: проектування та реалізації (рисунку 1)

Фаза проектування поділяється на такі етапи:

- визначення стратегії;
- аналіз предметної області;
- концептуальне моделювання;
- логічне й фізичне проектування.

Фаза реалізації складається з таких пунктів:

- власне програмна реалізація;
- документування;
- дослідне впровадження.

Рисунок 1 – Етапи життєвого циклу БД

2 Методологія проектування

Методологія проектування баз даних — це сукупність принципів, методів, інструментів і засобів, що застосовуються для послідовного розроблення структури бази даних. Оскільки система баз даних складається з програм і даних, методологія проектування баз даних розглядається як невід'ємна частина загальної методології проектування програмних систем.

До методології проектування баз даних висуваються певні вимоги. Прийнятною вважається база даних, яка відповідає вимогам користувачів (ефективність, адаптивність, незалежність, захищеність, цілісність тощо) і вимогам до апаратного забезпечення. Методологія має бути достатньо гнучкою, доступною розробникам із різним досвідом проектування, що використовують різні моделі даних і різне програмне забезпечення СКБД.

Методологія проектування баз даних визначає:

- процес проектування;
- методику виконання розрахунків і критеріїв оцінювання альтернативних рішень на кожному етапі проектування;
- інформаційні вимоги як вихідні дані для процесу проектування;

- засоби опису вихідних даних і відображення результатів кожного етапу проектування.

2.1 Процес проектування.

Для баз даних можна застосувати ітераційне низхідне проектування. Процес проектування добре структурований, оскільки кожний його етап завершується певним результатом, а також тому, що допускається ітераційне повторення попередніх етапів, якщо отриманий результат не відповідає вимогам замовника або системним вимогам. Це дає можливість переглядати й змінювати проектні рішення на будь- якому етапі.

З проектуванням тісно пов’язане експертне оцінювання проекту. Мета експертизи - знайти помилки й виправити їх на ранніх етапах проектування. Зазвичай експертиза виконується після завершення кожного з етапів.

Етап проектування БД вважається одним із самих складних етапів створення БД, який не має явно вираженого початку й закінчення. У порівнянні з аналізом вимог до БД або розробкою додатків, проектування БД, на думку багатьох провідних фахівців, є невдало структурованим завданням. Якщо всі етапи створення БД перекриваються один з одним у своїй послідовності, то етап проектування перекривається з усіма іншими етапами. Проектування починається з моменту прийняття стратегічних рішень і триває на етапах реалізації й тестування.

Процес проектування БД охоплює кілька основних сфер:

- проектування об'єктів БД (таблиці, подання, індекси, тригери, збережені процедури, функції, пакети) для подання даних ПО в БД;
- проектування інтерфейсу взаємодії з БД (форми, звіти й т.д.), тобто проектування додатків, які будуть супроводжувати дані в БД і реалізовувати питально-відповідні відношення на цих даних;
- проектування БД під конкретне обчислювальне середовище або інформаційну технологію (архітектура "клієнт-сервер", паралельні архітектури, розподілене обчислювальне середовище);
- проектування БД під призначення системи (інтелектуальний аналіз даних, OLAP, OLTP і т.д.).

2.2 Критерії оцінювання

Оцінювання необхідне для ухвалення рішень за наявності альтернатив. Труднощі у визначенні критеріїв і виборі альтернатив пов’язані з тим, що часто розробляється кілька проектів структури бази даних і потрібно оцінити, який з них є кращим. Зробити це буває досить складно.

Критерії є кількісні (час обробки запитів, вартість операцій маніпулювання даними, витрати пам’яті тощо) та якісні (гнучкість, адаптивність, сприйнятливість та сумісність).

2.3 Інформаційні вимоги

Визначаючи вимоги до інформації, врахуйте, що є інформація, яка стосується структури даних (опис даних та зв’язків безвідносно до конкретних способів їхнього використання й обробки), та інформація про спосіб використання даних (опис вимог до обробки даних).

Засоби опису

Це мовні засоби, призначені для опису результатів виконання кожного етану проектування. А саме, йдеться про такі засоби.

Природна мова, якою строго означуються всі необхідні для опису результатів проектування поняття. Використовується, як правило, на етапі визначення стратегії.

Стандартні форми, анкети та бланки. Використовуються переважно на етапі аналізу.

Спеціальні формалізовані мови концептуального моделювання (семантичні мережі, числення предикатів та ER-мови). Використовуються переважно на етапі концептуального моделювання.

Формалізовані мова означення даних (МОД) і мова маніпулювання даними (ММД). Використовуються на етапі логічного проектування. Зазвичай з цією метою застосовують мову SQL.

Лекція

Тема: ***«Архітектура баз даних»***

План лекції

1 Загальні поняття архітектури баз даних
2 Концептуальний рівень
3 Зовнішній рівень
4 Внутрішній рівень

Зміст лекції

1 Загальні поняття архітектури баз даних

Термін «архітектура» може мати різні тлумачення. Наприклад, коли вказуються функціональні модулі системи й способи їхньої взаємодії, йдеться про функціональну архітектуру. Спосіб реалізації функцій системи, її компоненти та взаємозв'язки між ними фіксує архітектура реалізації системи. У цьому контексті можна також згадати архітектуру технічних засобів систем. Говорячи ж про термін «архітектура БД», ми матимемо на увазі архітектуру інформаційного забезпечення.

Архітектура БД уперше була специфікована дослідницькою групою ANSI/X3/SPARC Study Croup on Data Base Management Systems (ANSI - Американський національний інститут стандартів, X3 - його комітет обчислювальної техніки й обробки інформації, SPARC - підкомітет ANSI/X3 з планування стандартів ANSI), Метою дослідницької групи було визначення галузей і технологій баз даних, в яких було б доречно проводити стандартизацію, а також вироблення рекомендацій для роботи в кожній із таких галузей. Група дійшла висновку, що, можливо, єдиним аспектом систем баз даних, який можна стандартизувати, є інтерфейси. Нею було докладено чимало зусиль для визначення загальної архітектури системи баз даних. Запропонована цією групою архітектура стала класичною та є актуальною й донині.

Основною ідеєю специфікації ANSI SPARC є виділення трьох архітектурних рівнів бази даних, а саме: зовнішнього, концептуального та внутрішнього.

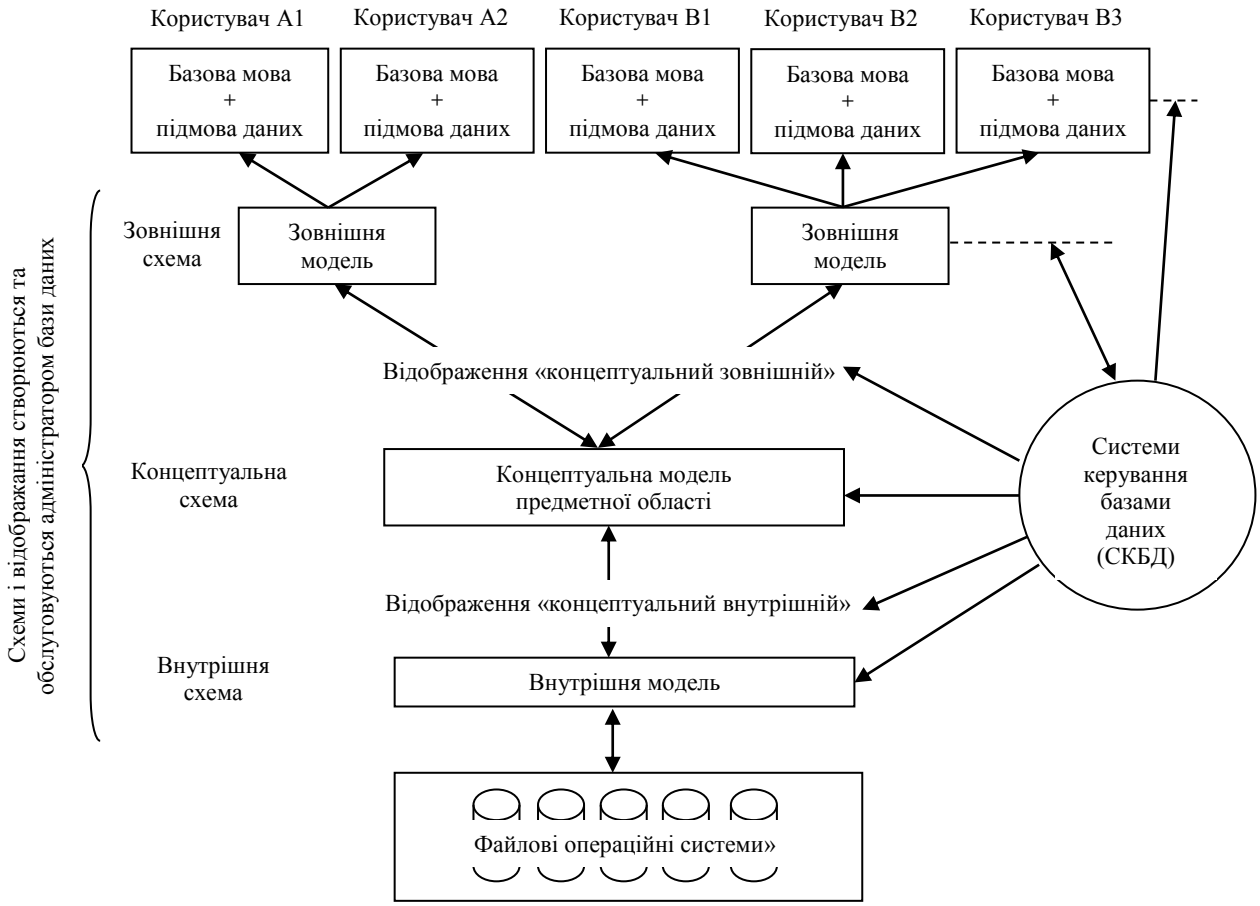


Рисунок 1.1 – Трирівнева архітектура бази даних

2 Концептуальний рівень

На концептуальному рівні здійснюється інтегрований опис предметної області, для якої розробляється БД, незалежно від її сприйняття окремими користувачами та способів реалізації в комп'ютерній системі. Дамо означений основних понять, що використовуються на концептуальному рівні,

Предметний область (ПО) – частина ризького світу для якої здійснюється концептуальне моделювання.

Концептуальна модель ПО – формальне зображення сукупності думок, які можливі стани ПО, а також переходи з одного стану з інший (включно з класифікацією наявних у ПО сутностей, чинних правил, законів, обмежень тощо).

Концептуальне моделювання ПО – процес побудови концептуальної моделі ПО, яка б відображувала ПО з урахуванням вимог, висунутих до цього процесу.

Концептуальна схема – фіксація концептуальної моделі ПО засобами конкретних мов моделей даних. У СКБД концептуальна модель подається у вигляді концептуальної схеми.

Властивості концептуальної моделі (схеми) й характерні особливості концептуального моделювання:

– спільне ти однозначне тлумачення предметної області всіма зацікавленими особами. До розробки складної бази даних залучається великий колектив: експерти, системні аналітики, проектувальники, розробники, ті, хто займається впровадженням і супроводом. Усі вони повинні однозначно розуміти, чим є ПО, в чому зміст використати понять, як вони взаємопов'язані між собою, які обмеження вибуваються до моделі ПО тощо. Спільність понять має забезпечувати концептуальна модель;

– концептуальна схема відображує лише концептуальна важливі аспекти ПО, виключаючи будь-які аспекти зовнішнього або внутрішнього відображення даних. Ця модель не повинна відображувати конкретні потреби окремих користувачів або застосувань. Вона має фіксували, чим є ПО в цілому, а не з точки зору інтересів або потреб користувачів. Для отримання цілісного уявлення про ПО її модель має інтегрувати думки, погляди та інтереси окремих користувачів, але саме інтегрувати, а не виражати їхні конкретні побажання;

– визначення допустимих меж еволюції бази даних. У процесі експлуатації база даних може розвиватися, проте цей розвиток може відбуватися тільки в межах, допустимих для концептуальної схеми;

– відображення зовнішніх схем на внутрішню. Саме через концептуальну схему зовнішні дані відображуються на внутрішні, й навпаки. У такий спосіб створюється єдина основа для опису даних і підтримки цих відображень;

– забезпечення незалежності даних. Наявність відображень концептуальний-зовнішній і концептуальний-внутрішній дає змогу вирішувати проблему логічної та фізичної незалежності даних. Будь-які зміни в тій чи іншій зовнішній моделі не повинні спричиняти зміни в концептуальній або внутрішній моделях. У цьому випадку має змінитися тільки відповідне відображення «концептуальний-зовнішній». Аналогічно, будь-які зміни у внутрішній моделі не зачіпають концептуальну модель і моделі зовнішнього рівня, а тільки приводять до змін відображення «концептуальній-внутрішній».

– централізоване адміністрування. Саме через концептуальну схему здійснюється адміністрування баз даних.

– стійкість. Концептуальна схема не має підладжуватися до вимог тих чи інших користувачів (зовнішній рівень) або до вимог зберігання даних (внутрішній рівень). Будучи моделлю ПО, вона має змінюватися тільки тоді, коли входить у суперечність із нею.

Існує багато мов, які претендують на роль мов концептуального моделювання ПО. Най популярнішими і широковживаними є мови, що належать до класу так званих графічних мов, які оперують поняттями «сутність-атрибут-зв'язок» (Entity-Relationship language).

3 Зовнішній рівень

Через зовнішній рівень користувачі та застосування отримують доступ до бази даних. Мета зовнішнього рівня – надити користувачу/застосуванню лише ті дані, які йому потрібні (а отже, до яких дозволений доступ) і в потрібному вигляді. Це індивідуальний рівень користувача, яким може бути кінцевий користувач, програміст чи застосування. Кожен з них має свою мову спілкування: для кінцевого користувача - це спеціальна мова запитів, для програміста - одна з мов програмування, розширена командами звернення до СКБД, для застосувань це, як правило, певний стандартний інтерфейс звернення по бази даних через СКБД.

Зовнішня модель – це засоби зображеним концептуальної моделі ПО з урахуванням інтересів конкретних користувачів або застосувань. Кожна зовнішня модель подається в СКБД у вигляді зовнішньої схеми.

Зовнішній рівень виконує такі функції.

– забезпечує зображення даних зручним для людини або застосування способом. Ступінь незалежності зовнішнього зображення під концептуального рівня визначається потужністю засобів опису відображення «концептуальний-зовнішній»;

– сприяє вирішенню проблеми безпеки (захисту) даних. Надаючи користувачу лише ті дані, що його цікавлять, ми залишаємо поза межами його доступу решту даних;

– сприяє вирішенню проблеми логічної незалежності даній. Це досягається завдяки відображенню «концептуальній-зовнішній», що встановлює відповідність між концептуальною схемою і конкретною зовнішньою схемою. Потужність його засобів визначає ступінь логічної незалежності застосувань від даних.

4 Внутрішній рівень

Внутрішня модель є відображенням концептуальної моделі ПО з урахуванням способів зберігання даних і методів доступу до них. Внутрішня модель відображується в СКБД у вигляді внутрішньої схеми. Внутрішня модель - це не модель фізичної пам'яті з характеристиками конкретних пристроїв зберігання даних (циліндри, доріжки тощо); вона описується у вигляді нескінченної абстрактної лінійної пам'яті, яка може структуруватися за допомогою інших абстрактних понять на зразок блоків, кластерів, індексів тощо.

Доступ до фізичної пам'яті надасться за допомогою опису відображень внутрішньої моделі на фізичну пам'ять операційної системи.

Загалом внутрішній рівень виконує такі функції .

– забезпечує настроювання бази даних для підвищення продуктивності обробки даних, опису й підтримки планованої надлишковості;

– дає змогу описувати й підтримувати структури зберігання та методи доступу;

– сприяє вирішенню проблеми фізичної незалежності даних, зміни у внутрішній схемі не повинні призводити до змін у зовнішній схемі;

– сприяє вирішенню проблеми безпеки (захисту) даних;

– вирішує проблему відображення даних на структури ОС, у яких дані зберігаються (до таких структур належать зокрема файли).

Лекція

Тема: ***«Поняття про моделювання даних»***

План лекції

1 Поняття та класифікація моделей даних
    1.1 Дані та їхня семантика
    1.2 Моделювання даних
2 Ієрархічна модель
3 Мережева модель

Зміст лекції

1 Поняття та класифікація моделей даних

1.1 Дані та їхня семантика

Слово «дані» походить від латинського «datum» – факт, проте дані не завжди відповідають конкретним чи навіть реальним фактам. Іноді вони неточні або описують те, чого насправді не Існує Данини ми вважатимемо опис будь-якого явища (чи ідеї), що викликає зацікавленість через певні потреби. З даними нерозривно пов’язані їхня інтерпретація (або семантика), тобто той вміст, який їм приписується.

Дані описуються тією чи іншою мовою і фіксуються на певному носії (скажімо, папері). Зазвичай далі (факти) та їхня семантична інтерпретація фіксуються спільно. Проте в деяких випадках дані й інтерпретація розділяються. Так у зведеній статистичній таблиці зверху, в її шапці, міститься інформація стосовно того, чим є дані (заголовок таблиці, описова інформація, назви стовпців тощо), а нижче розташовуються власне рядки чисел.

Застосування комп'ютерів для введення й обробки даних сприяло ще більшому розділенню даних та їхньої інтерпретації. Оскільки комп’ютери оперують даними як такими, велика частіша інтерпретуючої інформації взагалі явно не фіксується. Програма одержує певні вхідні дані, обробляє їх і видає результат у вигляді сукупності вихідних даних.

З розвитком обчислювальної техніки з'явилася можливість фіксувати за допомогою комп'ютерів семантику даних, тобто закладати інтерпретацію в програми, що оперують даними. Проте за умов спільного використання даних різноманітними прикладними програмами цей підхід ножна застосовувати лише частково. Інакше процес написаний програм, які містять схожі, хоча й не ідентичні механізми інтерпретації, стає неефективним. У подібній ситуації доцільно зв'язувати дані з механізмами інтерпретації, що має забезпечувати уніфікованість подання семантичної інформації. У

результаті змінюється роль даних: їх уже не можна розглядати лише як сукупність бітів, вони отримують певне семантичне навантаження.

Засоби інтерпретації мають бути гнучкими, аби разом з незмінними параметрами даних відображувати й аспекти їхньої еволюції. Цього досягають двома способами. По-перше, можна відтворювати різні погляди на одні й ті самі дані. Наприклад, розглядати певну особу з точки зору кадрової системи як службовця, з погляду виробничих інтересів – як виконавця робіт, а з боку медичного обслуговування – як пацієнта. По-друге, різні дані можна подавати одноманітно. Так, адміністратори, клерки, агенти, секретарі незалежно від роду своєї діяльності можуть розглядатися в кадровій системі як службовці.

1.2 Моделювання даних

Існує багата тинів моделей - фізичні, математичні, економічні тощо, які відображують різні аспекти реального світу. Модель даних відображує уявлення про реальний світ. Проте важливо, аби обсяг знань і семантика даних, відтворені в моделі, були адекватні способу використання даних. Вважатимемо, що модель даних – це сукупність структури даних, операцій над ними (операції маніпулювання даними) та обмежень цілісності. Іншими словами, модель визначає, в який спосіб відбувається об'єднання даних у структури різної складності які існують обмеження на значення дати і як здійснюється оперування цими даними.

Основою для будь-якої структури даних є відображення елементарної одиниці даних у вигляді такої трійки: об'єкт, властивість об'єкта, значення властивості. Сукупність взаємопов’язаних між собою елементарних одиниць даних може відображуватися різноманітними способами, що приводить до формування різних структур, а відтак - різних моделей даних. Моделі даних поділяються на два класи; сильно та слабко типізовані.

У сильно типізованих моделях усі дані мають належати до певної категорії, або типу. Якщо дані не підпадають під жодну з категорій, їх потрібно типізувати штучно. Деякі моделі будуються у такий спосіб, що категорії визначаються наперед і не можуть змінюватися динамічно.

Сильно типізовані моделі мають значні переваги, бо дають змогу побудувати абстракції властивостей даних і дослідити їх у термінах категорій. Більшість моделей, що використовуються в автоматизованих системах, зокрема й базах даних, належать до сильно типізованих.

Для слабко типізованих моделей належність даних до тієї чи іншої категорії немає жодного значення. Категорії використовуються настільки, наскільки це доцільно в кожному конкретному випадку. Окремі дані можуть існувати як незалежно, так і у зв'язку з іншими. Інформація про категорії розглядається як додаткова.

На відміну від сильно типізованих моделей, слабко типізовані забезпечують інтеграцію даних і категорій. Найкращі можливості такої інтеграції надаються численним предикатів, яке у багатьох моделях даних

використовується для зображення знань, що не підтримуються базовими засобами моделювання.

2 Ієрархічна модель

У ієрархічній моделі зв'язку між даними можна описати за допомогою упорядкованого графа (або дерева). Спрощено представлення зв'язків між даними в ієрархічній моделі показане на рисунку 1.1.

Для опису структури (схеми) ієрархічною БД на деякій мові програмування використовується тип даних «дерево».

Тип «дерево» схожий з типами даних «структура» мов програмування ПЛ/1 і Сі і «запис» мови Паскаль. У них допускається вкладеність типів, кожен з яких знаходиться на деякому рівні.

Тип «дерево» є складеним. Він включає підтипи («під дерева»), кожен з яких, у свою чергу, є типом «дерево». Кожен з типів «дерево» складається з одного «кореневого» типа і впорядкованого набору (можливо порожнього) підлеглих типів. Кожен з елементарних типів, включених в типа «дерево», є простим або складеним типом «запис». Простий «запис» складається з одного типа, наприклад, числового, а складений «запис» об'єднує деяку сово-купность типів, наприклад, ціле, рядок символів і покажчик (заслання).

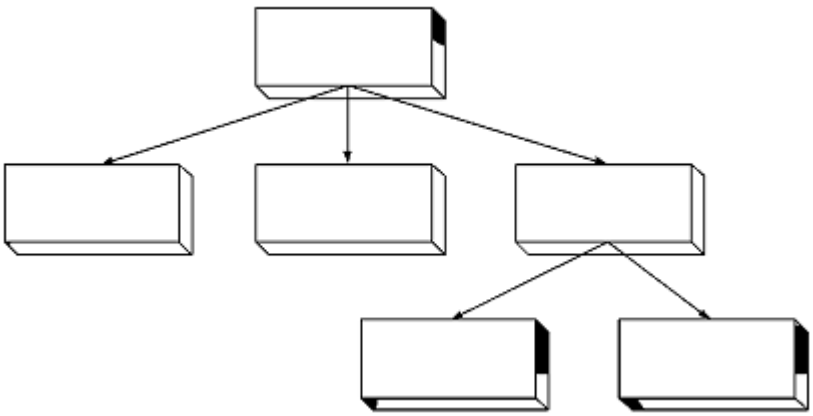


Рисунок 1.1 – Представлення зв'язків в ієрархічній моделі

Кореневим називається тип, який має підлеглих типів і сам не є підтипом. Підлеглий тип (підтип) є нащадком по відношенню до типа, який виступає для нього в ролі предка (батька).

Ієрархічна БД є впорядкованою сукупністю екземплярів даних типа «дерево» (дерев), що містять екземпляри типа «запис» (записи). Часто стосунки спорідненості між типами переносять на стосунки між самими записами. Поля записів зберігають власне числові або символьні значення, які складають основний вміст БД. Обхід всіх елементів ієрархічною БД звичайно виробляється зверху вниз і зліва направо.

Для організації фізичного розміщення ієрархічних даних в пам'яті ЕОМ можуть використовуватися наступні групи методів:

вистава лінійним списком з послідовним розподілом пам'яті(адресна арифметика, лівоспискові структури);

вистава зв'язними лінійними списками (методи, що використовують покажчики і довідники).

До основних операцій маніпулювання ієрархічно організованими даними відносяться наступні:

- пошук вказаного екземпляра БД;
- перехід від одного дерева до іншого;
- перехід від одного запису до іншої усередині дерева;
- вставка нового запису у вказану позицію;
- видалення поточного запису і так далі

Відповідно до визначення типа «дерево», можна укласти, що між предками і нащадками автоматично підтримується контроль цілісності зв'язків. Основне правило контролю цілісності формулюється таким чином: нащадок не може існувати без батька, а у деяких батьків може не бути нащадків. Механізми підтримки цілісності зв'язків між записами різних дерев відсутні.

До достоїнств ієрархічної моделі даних відносяться ефективне використання пам'яті ЕОМ і непогані показники часу виконання основних операцій над даними. Ієрархічна модель даних зручна для роботи з ієрархічно впорядкованою інформацією.

Недоліком ієрархічної моделі є її громіздкість для обробки інформації з досить складними логічними зв'язками, а також складність розуміння для звичайного користувача.

2. Мережева модель

Мережева модель даних дозволяє відображувати всілякі взаємозв'язки елементів даних у вигляді довільного графа, узагальнюючи тим самим ієрархічну модель даних (рисунок 1.2).

Для опису схеми мережевою БД використовується дві групи типів: «запис» і «зв'язок». Тип «зв'язок» визначається для двох типів «запис»: предка і нащадка. Перемінні типа «зв'язок» є екземплярами зв'язків.

Мережева БД складається з набору записів і набору відповідних зв'язків. На формування зв'язку особливих обмежень не накладається. Якщо в ієрархічних структурах запис-нащадок могла мати лише одну запис-предка, то в мережевій моделі даних запис-потомок може мати довільне число записів-предків (звідних батьків).

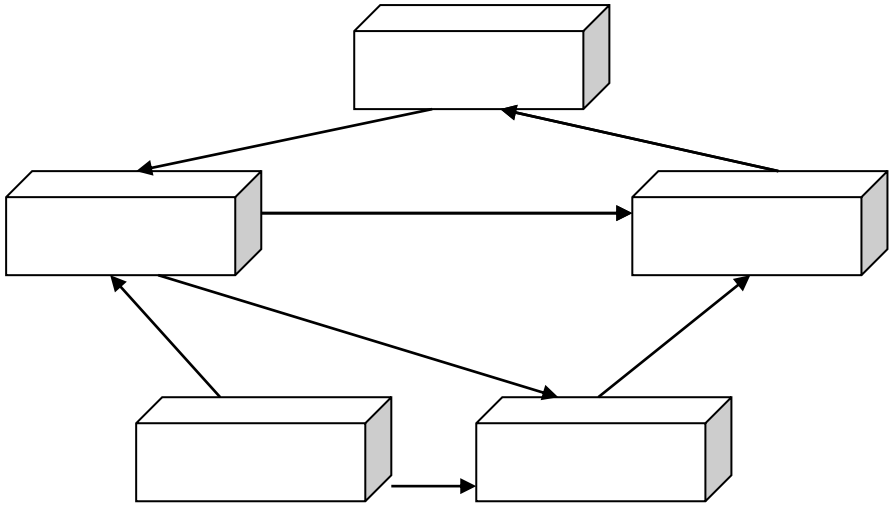


Рисунок 1.2 – Представлення зв'язків в мережевій моделі

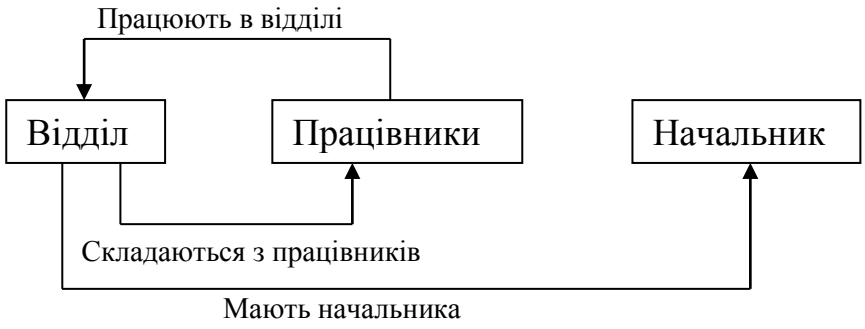


Рисунок 1.3 – Приклад схеми мережевої БД

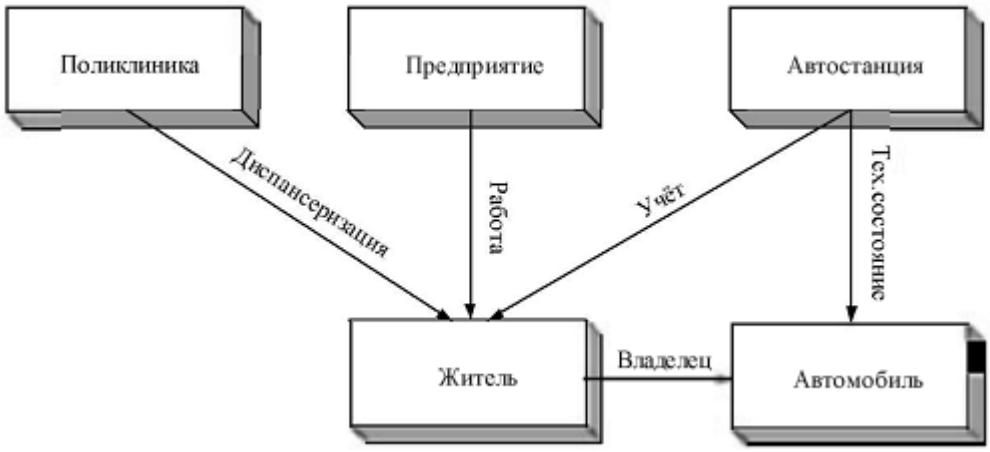


Рисунок 1.4 – Один тип приймає участь у декількох відношеннях

Фізичне розміщення даних в базах мережевого типа може бути організоване практично тими ж методами, що і в ієрархічних базах даних.

До найважливіших операцій маніпулювання даними баз мережевого типа можливо віднести наступні:

- пошук запису в БД;
- перехід від предка до першого нащадка;
- перехід від нащадка до предка;
- створення нового запису;
- видалення поточного запису;
- оновлення поточного запису;
- включення запису в зв'язок;
- виключення запису із зв'язку;
- зміна зв'язків і так далі

Гідністю мережевої моделі даних є можливість ефективної реалізації за показниками витрат пам'яті і оперативності. Порівняно з ієрархічної моделлю мережева модель надає великі можливості в сенсі допустимості утворення довільних зв'язків.

Недоліком мережевої моделі даних є висока складність і жорсткість схеми БД, побудованої на її основі, а також складність для розуміння і виконання обробки інформації в БД звичайним користувачем. Крім того, в мережевій моделі даних ослаблений контроль цілісності зв'язків унаслідок допустимості встановлення довільних зв'язків між записами.

Лекція

Тема: ***«Реляційна модель даних»***

План лекції

1 Основні поняття та складові частини реляційної моделі даних
2 Атрибути і схема відношення
3 Об’єктні та зв’язкові відношення
4 Таблиці. Первинні ключі таблиць

Зміст лекції

1 Основні поняття та складові частини реляційної моделі даних

Основними поняттями реляційних баз даних є: відношення, атрибут, схема відношення, тип даних, домен, кортеж, таблиця, первинний ключ, зв’язок, вторинний ключ, структурна, цілісна та маніпуляційна частини моделі даних.

Значення цих понять розглянемо з використанням прикладу відношення, яке містить інформацію про співробітників (рисунок 1).

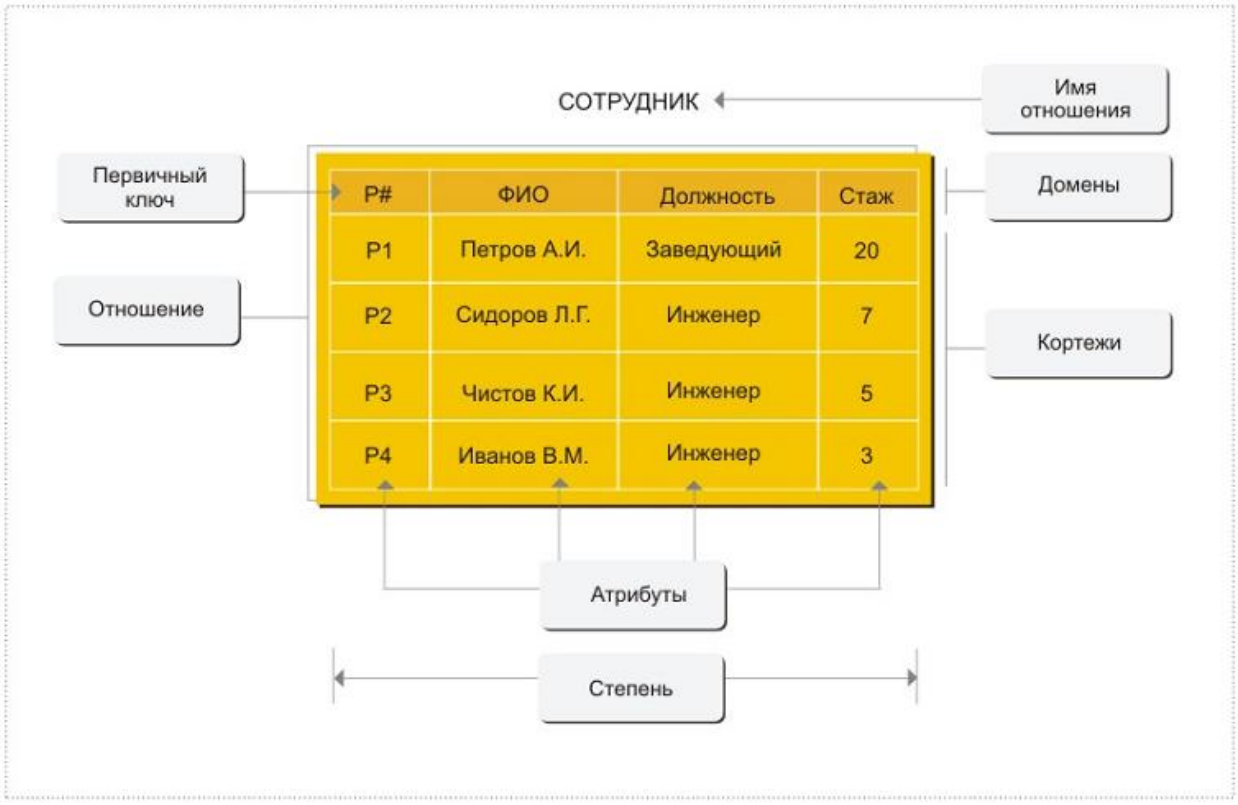


Рисунок 1 – Відношення СПІВРОБІТНИК

Відношення є фундаментальним поняттям і засобом структурування даних у реляційній моделі даних. Відношенням називається будь-який зв'язок між об'єктами або їх властивостями.

Відношення має ім’я, яке задається іменником. Ім’я повинне бути унікальним у базі даних і пояснювати зміст (семантику) відношення. Засобом подання відношення в реляційній базі даних є іменована таблиця.

2 Атрибути і схема відношення

Атрибут - це логічно неподільна одиниця даних, яка визначає певну властивість об'єкта або зв’язок. Кожний атрибут має ім’я, певний тип (рядок символів, число та ін.), значення та інші властивості.

Ім’я атрибута задається іменником в однині. Воно повинне бути унікальним у відношенні, простим і зрозумілим.

Значення атрибута - величина, що характеризує деяку властивість об'єкта або зв’язок. Значення має певний тип, формат, довжину та повинно бути атомарним (неподільним) для даної моделі. Приклади атрибутів та їх значень наведені на рис. 1.6 (первинний ключ виділено напівжирним написанням).

Структура відношення подається у вигляді схеми.

Схемою відношення є кінцева множина впорядкованих імен атрибутів - (A1, A2,..., An). Опис відношення складається з його ім’я та схеми: ІМ’Я (A1, A2,..., An), наприклад:

ВИКЛАДАЧ (Таб_номер, Прізвище, Зарплата, К_кафедри).

Для однозначної ідентифікації кортежів кожне відношення повинно мати первинний ключ. Первинний ключ складається з одного або декількох атрибутів. Наприклад, атрибут «Таб_номер» є первинним ключем відношення «ВИКЛАДАЧ» і визначає кортежі цього відношення.

3 Об’єктні та зв’язкові відношення

Відношення можна поділити на два класи: об'єктні й зв'язкові. Об'єктні відношення зберігають дані про інформаційні об'єкти предметної області. Прикладом об’єктного відношення є відношення «ВИКЛАДАЧ». В об'єктному відношенні не повинно бути рядків з однаковим ключем.

Зв'язкове відношення містить ключі двох або більше об'єктних відношень. За цими ключами зв’язкове відношення встановлює зв'язки між відповідними об’єктними відношеннями. Наприклад, розглянемо два об'єктних відношення:

СТУДЕНТ(Код_студента, Прізвище); ДИСЦИПЛІНА(Код_дисципліни, Назва).

Тоді відношення

ЕКЗАМЕН (Код студента, Код дисципліни, Оцінка) буде зв'язковим між об'єктними відношеннями «СТУДЕНТ» і «ДИСЦИПЛІНА».

У зв'язковому відношенні допускається дублювання значень ключових атрибутів. Крім ключових атрибутів, у зв'язковому відношенні можуть бути й інші атрибути, наприклад, атрибут «Оцінка».

Тип даних, домен і кортеж Значення даних, що зберігаються в реляційній базі даних, мають певний тип, наприклад, текстовий, числовий, грошовий, логічний, дата/час та ін.

Тип даних визначає: множину значень, набір операцій, що можуть бути застосовані до значень заданого типу, та спосіб зовнішнього подання значень. Наприклад, числовий тип визначає множину чисел, над якими допустимі арифметичні операції, а зовнішнє подання - числа.

У реляційній моделі використовуються прості (скалярні) дані, значення яких є атомарними (неподільними). Це означає, що в реляційних операціях не повинна враховуватися внутрішня структура даних. Наприклад, простими даними є дані числового типу та рядки символів, оскільки в операціях вони використовуються як єдине ціле.

Якщо розглядати масив як єдине ціле і не використовувати операції над його елементами, то масив також можна вважати простим типом даних. Більш того, допустимо створити свій, скільки завгодно складний тип даних, описати можливі дії з ним, і якщо в операціях не потрібне знання внутрішньої структури даних, то такий тип даних також буде простим з погляду реляційної теорії. Наприклад, атрибут «Прізвище» (див. рисунок 1), який має текстовий тип і рядки символів, що виражають прізвища співробітників, є атомарними. Якщо рядок розкласти в масив символів, то при переході на такий низький рівень буде втрачено сенс цього рядка як єдиного цілого, тобто це буде вже не прізвище, наприклад, Іванов.

Доменом є підмножина значень простого типу даних, які мають певний сенс і є допустимими для атрибута, що визначений на домені. Домен має унікальне ім'я в межах бази даних. Зазвичай ім’ям домену є ім’я атрибуту

Основне призначення доменів полягає в тому, що вони обмежують порівняння. Некоректно, з логічної точки зору, порівнювати значення з різних доменів, навіть якщо вони мають однаковий тип. Наприклад, значення різних доменів «Номери перепусток» і «Номери відділів» не є порівнянними, хоча домени визначені одним типом цілих чисел. Таким чином, поняття домену допомагає правильно моделювати предметну область.

Кортеж становить набір іменованих значень заданого типу. Іменами значень кортежу є імена атрибутів, а значеннями - припустимі значення домену даного атрибута. Відношення складається з множини кортежів.

4 Таблиці. Первинні ключі таблиць

Таблиці є основною формою подання відношень і основним засобом зберігання даних у реляційній базі даних. Кожному відношенню відповідає таблиця з таким же ім’ям, але, на відміну від відношення, ім’я таблиці є іменником у множині, наприклад, «Викладачі». У реляційній базі даних імена таблиць повинні бути унікальними.

Таблиця складається із заголовку, стовпців і рядків.

Стовпці таблиці називають також полями. На перетині стовпців і рядків містяться значення даних (див. рисунок 1). Стовпець таблиці відповідає атрибуту відношення, має ім’я та містить дані одного з припустимих типів, наприклад, текстового, числового або ін. В одній таблиці стовпці повинні мати унікальні імена, а в різних таблицях імена стовпців можуть бути однакові, наприклад, кожна з таблиць «Викладачі» і «Студенти» може мати поле з ім’ям «Прізвище». Кількість стовпців у таблиці фіксована, список їх імен складає заголовок таблиці. При створенні таблиці її стовпці упорядковуються зліва направо у послідовності введення імен стовпців. Максимально допустиме число стовпців у таблиці, як правило, не вказується. Однак майже у всіх комерційних СКБД ця межа існує та складає приблизно 255 стовпців. У будь-якій таблиці завжди є, як мінімум, один стовпець.

Рядки таблиці відповідають кортежам відношення. На відміну від стовпців, рядки не мають імен, кількість їх не обмежена, а порядок розміщення довільний. Припустиме існування таблиці з нульовою кількістю рядків, яка називається порожньою. Порожня таблиця зберігає структуру, визначену її стовпцями, але в ній не містяться дані. Стандарти реляційних баз даних не накладають обмежень на кількість рядків у таблиці, і в багатьох СКБД розмір таблиць обмежений лише вільним дисковим простором комп'ютера.

Таблиці розміщуються у файлі бази даних. Кожен рядок таблиці відповідає запису файлу, тому рядки таблиці називають також записами.

Первинні ключі ідентифікують рядки оскільки рядки таблиці не мають імен. Первинний ключ становить комбінацію атрибутів таблиці, дані яких однозначно визначають кожен рядок таблиці. Якщо ключ складається з одного атрибуту, він називається простим ключем, а якщо з декількох - складеним ключем. Атрибути, з яких складається ключ, називають ключовими.

Лекція

Тема: ***«Правила Кодда для реляційних баз даних»***

План лекції

1 Правила Кодда для реляційних баз даних
2 Правила реляційної цілісності

Зміст лекції

1 Правила Кодда для реляційних баз даних

0 Основне (правило, що мається на увазі). Система, яка рекламується або проголошується постачальником як реляційна СУБД повинна управляти базами даних виключно способами, відповідними реляційній моделі.

1 Інформаційне правило. Уся інформація, що зберігається в реляційній базі даних, має бути явно, на логічному рівні, представлена єдиним чином: у вигляді значень в реляційних таблицях.

2 Правило гарантованого логічного доступу. До кожного наявного в реляційній базі атомарного значення має бути гарантований доступ за допомогою вказівки імені реляційної таблиці, значення первинного ключа і імені стовпця.

3 Правило наявності значення. У повністю реляційній СУБД мають бути спеціальні індикатори (відмінні від порожнього символьного рядка або рядка з одних пропусків і відмінні від нуля або якого-небудь іншого числового значення) для вираження (на логічному рівні, систематично і незалежно від типу даних) того факту, що значення відсутнє щонайменше з двох різних причин : його дійсно немає, або воно непридатне до цієї позиції. СУБД повинна не лише відбивати цей факт, але і поширювати на такі індикатори свої функції маніпулювання даними незалежно від їх типу.

4 Правило динамічного діалогового реляційного каталогу. Опис бази даних виглядає логічно як звичайні дані, так що авторизовані користувачі і прикладні програми можуть вживати для роботи з цим описом ту ж реляційну мову, що і при роботі із звичайними даними.

5 Правило повноти мови роботи з даними. Скільки б багато в СУБД не підтримувалося мов і режимів роботи з даними, має бути принаймні одна мова, що виражається у вигляді командних рядків в деякому зручному синтаксисі, який би дозволяв формулювати:

- визначення даних;
- визначення правил цілісності;
- маніпулювання даними (у діалоги і з програми);

– визначення таблиць (у тому числі можливості їх модифікації), що виводяться;

– визначення правил авторизації;

– межі транзакцій.

6 Правило модифікації таблиць. У СУБД повинен існувати коректний алгоритм, що дозволяє автоматично для кожної таблиці визначати під час її створення, чи може вона використовуватися для вставки і видалення рядків і які із стовпців допускають модифікацію, і що заносить отриману таким чином інформацію в системний каталог.

7 Правило множинності операцій. Можливість операції базовими або такими, що виводяться таблицями поширюється повністю не лише на видачу інформації з БД, але і на вставку, модифікацію і видалення даних.

8 Правило фізичної незалежності. Діалогові оператори і прикладні програми на логічному рівні не повинні страждати від яких-небудь змін у внутрішньому зберіганні даних або в методах доступу СУБД.

9 Правило логічної незалежності. Діалогові оператори і прикладні програми на логічному рівні не повинні страждати від таких змін в базових таблицях, які зберігають інформацію і теоретично допускають незмінність цих операторів і програм.

10 Правило збереження цілісності. Діалогові оператори і прикладні програми не повинні змінюватися при зміні правил цілісності в БД (що задаються мовою роботи з даними і що зберігаються в системному каталозі).

11 Правило незалежності від распределенности. Діалогові оператори і прикладні програми на логічному рівні не повинні страждати від здійснюваного фізичного рознесення даних (якщо спочатку СУБД працювала з нерозподіленими даними) або перерозподілу (якщо СУБД дійсно розподілена).

12 Правило непорушення реляційної мови. Якщо в реляційній СУБД є мова низького рівня (для роботи з окремими рядками), він не повинен дозволяти порушувати або "обходити" правила, сформульовані на мові високого рівня і занесені в системний каталог.

2 Правила реляційної цілісності

У реляційній моделі даних є два загальних правила цілісності. Ці два правила відносяться до потенційних ключів і зовнішніх ключів.

У відношенні можуть існувати декілька одиночних або складених атрибутів, які однозначно ідентифікують кортеж відношення. Це — потенційні ключі.

Говорять, що безліч атрибутів До = {Ai, Aj ..., Ak} стосунки r є потенційним ключем r тоді і лише тоді, коли задовольняються два незалежних від часу умови:

унікальність: у довільний заданий момент часу жодні два різні кортежі r не мають одного і того ж значення для Ai, Aj ..., Ак;

мінімальність: жоден з атрибутів Ai, Aj .... Ак не може бути виключений з К без порушення унікальності.

Відношення може мати декілька потенційних ключів. Ключ, що містить два і більш за атрибут, називається складеним ключем. Кожне відношення володіє хоч би одним можливим ключем, оскільки у відношенні не може бути однакових кортежів, а це означає, що, щонайменше, комбінація всіх його атрибутів задовольняє умові унікальності. Потенційні ключі, дозволяючи гарантовано виділити точно один кортеж, забезпечують основний механізм адресації на рівні кортежів реляційної моделі.

Один з можливих ключів (вибраний довільним чином) береться за його первинний ключ. Зазвичай первинним ключем призначається той можливий ключ, яким найпростіше користуватися при повсякденній роботі. Останні можливі ключі, якщо вони є, називаються альтернативними ключами. Для індикації зв'язку між стосунками використовуються зовнішні ключі.

Зовнішній ключ — це набір атрибутів одного відношення, що є потенційним ключем іншого відношення.

Завдяки наявності в'язок між потенційними і зовнішніми ключами забезпечується взаємозв'язок кортежів певних стосунків.

Відношення, що містить зовнішній ключ, називається дочірнім або відношенням, що посилається . А відношення, що містить пов'язаний із зовнішнім ключем потенційний ключ, — батьківським або цільовим відношенням.

Стосунки не можуть розглядатися як статичні об'єкти, оскільки вони призначені для віддзеркалення деякої частини реального світу, а ця частина реального світу може змінюватися в часі. Тому і стосунки змінюються в часі: кортежі можуть додаватися, віддалятися або модифікуватися. Проте, передбачається, що сама схема відношення інваріантна в часі. Відношення повинне сприйматися як безліч можливих станів, які може приймати відношення.

Лекція

Тема: ***«Цілісність даних у базі даних. Структурна частина реляційної моделі»***

План лекції

1 Структурна частина реляційної моделі
- 1.1 Кількісні характеристики відношення
- 1.2 Фундаментальні властивості відношень
- 1.3 Потенційний, альтернативний і первинний ключі відношення

Зміст лекції

Ядром будь-якої бази даних є модель даних. Вона становить множину структур даних, обмежень цілісності й операцій маніпулювання даними. За допомогою моделі даних можуть бути представлені об'єкти предметної області та взаємозв'язки між ними. Найбільш поширене трактування реляційної моделі даних належить К. Дейту. Згідно з трактуванням Дейта у склад реляційної моделі входять три частини: структурна, цілісна та маніпуляційна.

1 Структурна частина реляційної моделі

Структурна частина описує об'єкти, які розглядаються реляційною моделлю, а єдиною структурою даних, яка використовується в реляційних базах даних, є нормалізоване n-арне відношення.

Нехай задані n множин даних D1, D2,..., Dn, які є доменами. Тоді відношенням R називається множина n впорядкованих кортежів <d1, d2,..., dn, де d1 ∈ D1 (символ ∈ означає «елемент з»), d2 ∈ D2,..., dn ∈ Dn. Кортеж становить підмножину пар {<ім’я атрибуту>, <значення>} і є унікальним у відношенні. У математичному сенсі відношення R є підмножиною декартового добутку доменів D1 x D2 x...x Dn. Відношення R включає заголовок і тіло.

Заголовок (схема) відношення містить фіксовану кількість імен атрибутів відношення. Імена атрибутів мають бути унікальними в межах відношення і можуть збігатися з іменами відповідних доменів. Заголовок не змінюється під час роботи з базою даних. Якщо у відношенні змінені, додані або видалені атрибути, то маємо інше відношення (навіть з колишнім ім'ям).

Тілом відношення є набір кортежів. Воно може бути модифікованим під час роботи з базою даних, тобто кортежі можуть змінюватися, додаватися і видалятися.

1.1 Кількісні характеристики відношення

Ступінь (арність) відношення - це число атрибутів відношення. Відношення ступеня один називають унітарним, ступеня два - бінарним, ступеня три - тернарним, а ступеня n - n-арним (відношення має n атрибутів). Ступінь відношення не змінюється в часі.

Кардинальність (потужність) відношення - це кількість кортежів відношення. Кардинальне число відношення змінюється в часі. Наприклад, відношення «Викладачі» є 4-арним, а його кардинальність - 4.

1.2 Фундаментальні властивості відношень

Властивості відношень безпосередньо виходять з наведеного вище теоретико-множинного визначення поняття відношення, яке є множиною кортежів. Фундаментальними властивостями відношень є:

1. Однотипність кортежів. Усі елементи відношення є однотипними кортежами, що дозволяє вважати кортежі аналогом рядків у таблиці.

2. Обмеженість відношень. Відношення включає не всі можливі кортежі з декартового добутку доменів, на яких воно визначене. Для кожного відношення є критерій, який дозволяє визначити ті кортежі, що входять у відношення. Цей критерій визначає сенс або семантику відношення, є логічним виразом і називається предикатом відношення R.

3. Відсутність кортежів-дублікатів і впорядкованості кортежів. У відношенні відсутні однакові кортежі і кортежі не впорядковані зверху вниз, оскільки відношенням є множина кортежів, а згідно з теорією множин кожна множина складається з різних елементів, які не впорядковані. Звідси витікає, що посилання на кортежі у відношенні повинно мати первинний ключ.

4. Відсутність впорядкованості атрибутів. Атрибути відношень не впорядковані, оскільки за визначенням кортеж становить підмножину пар {<ім’я атрибуту>, <значення>} і для посилання на значення використовується ім’я атрибуту. Ця властивість дозволяє модифікувати схеми існуючих відношень шляхом додавання або видалення атрибутів.

5. Атомарність значень атрибутів. Значення всіх атрибутів повинні бути атомарними. Це є наслідком визначення домену як потенційної множини значень простого типу, тобто серед значень домену не може бути відношення.

1.3 Потенційний, альтернативний і первинний ключі відношення

Потенційний ключ відношення - це мінімальний набір атрибутів, який може бути використаний для однозначної ідентифікації будь-якого кортежу відношення. Ключі відношень бувають атомарними і складеними. Атомарний ключ складається з єдиного атрибуту, а складений ключ - із набору атрибутів. Складений ключ не повинен мати зайвих атрибутів. У його

склад включаються тільки ті атрибути, без яких не можлива ідентифікація кортежів.

Відношення має принаймні один потенційний ключ. Дійсно, якщо ніякий атрибут або група атрибутів не є потенційним ключем, то через унікальність кортежів усі атрибути разом утворюють потенційний ключ.

Відношення може мати декілька потенційних ключів. Один із них оголошується первинним ключем (ПК), а інші - альтернативними ключами.

Поняття потенційного ключа є семантичним і відображає трактування понять предметної області. Наприклад, розглянемо відношення «ВИКЛАДАЧ»

На перший погляд може здатися, що в цьому відношенні є три потенційних ключі: «Таб_номер», «Прізвище» і «Зарплата». У кожній колонці таблиці містяться унікальні дані. Проте серед викладачів можуть бути викладачі з однаковою зарплатою і однаковим прізвищем. Табельний же номер по суті, є унікальним для кожного викладача, і саме розуміння семантики даних привело до твердження, що в даному відношенні є тільки один потенційний ключ - «Таб_номер».

Таким чином, потенційні ключі служать єдиним засобом адресації кортежів у відношенні. Тільки знання значень потенційного ключа кортежу дозволяє точно визначити первинний ключ відношення.

У практичній реалізації реляційної СКБД відношення є математичним аналогом таблиць. Таблиця є найбільш звичним і зручним вмістищем для зберігання інформації, має фіксовану кількість пойменованих і впорядкованих стовпців та необмежену кількість рядків. У таблиці 1 наведені терміни, які вживаються як синоніми.

Таблиця 1 – Порівняння термінології

Реляційний термін	Табличний термін у СКБД
Відношення	Таблиця
Заголовок відношення	Заголовок таблиці
Тіло відношення	Тіло таблиці
Атрибут відношення	Найменування стовпця (поля) таблиці
Кортеж відношення	Рядок (запис) таблиці
Ступінь відношення	Кількість стовпців таблиці
Кардинальність відношення	Кількість рядків таблиці
Домен	Тип даних (базовий або користувача)

Лекція

Тема: ***«Теоретико-множинні операції реляційної алгебри»***

План лекції

1 Загальні поняття реляційної алгебри
2 Операція об'єднання
3 Операція перетин
4 Операція різниці
5 Декартовий добуток

Зміст лекції

1 Загальні поняття реляційної алгебри

Запропонувавши реляційну модель даних, Кодд створив і інструмент для зручної роботи із стосунками - реляційну алгебру. Кожна операція цієї алгебри використовує одну або декілька таблиць (стосунків) як її операнди і отримує в результаті нову таблицю, тобто дозволяє "розрізати" або "склеювати" таблиці (рисунок 1.1).

Реляційна алгебра в явному виді представляє набір операцій, які можна використовувати, щоб повідомити систему, як в базі даних з певних стосунків реально побудувати необхідне відношення.

Реляційні оператори мають одну важливу властивість: вони замкнуті відносно поняття відношення. Це означає, що вирази реляційної алгебри визначаються над стосунками реляційних БД і результатом обчислення також являються стосунки. Оскільки результатом будь-якої реляційної операції є деяке відношення, можна утворювати реляційні вирази, в яких замість відношення-операнда деякої реляційної операції знаходиться вкладене реляційне вираження.

Вирази реляційної алгебри будуються на основі операцій (високого рівня) алгебри, і подібно до того, як інтерпретуються арифметичні і логічні вирази, вираження реляційної алгебри також має процедурну інтерпретацію. Іншими словами, запит, представлений на мові реляційної алгебри, може бути вичислений на основі обчислення елементарних операцій алгебри з урахуванням їх старшинства і можливої наявності дужок.

Набір основних операцій алгебри складається з восьми операцій, які діляться на два класи - теоретико-множинні операції і спеціальні реляційні операції, доповнені деякими спеціальними операціями, специфічними для баз даних.

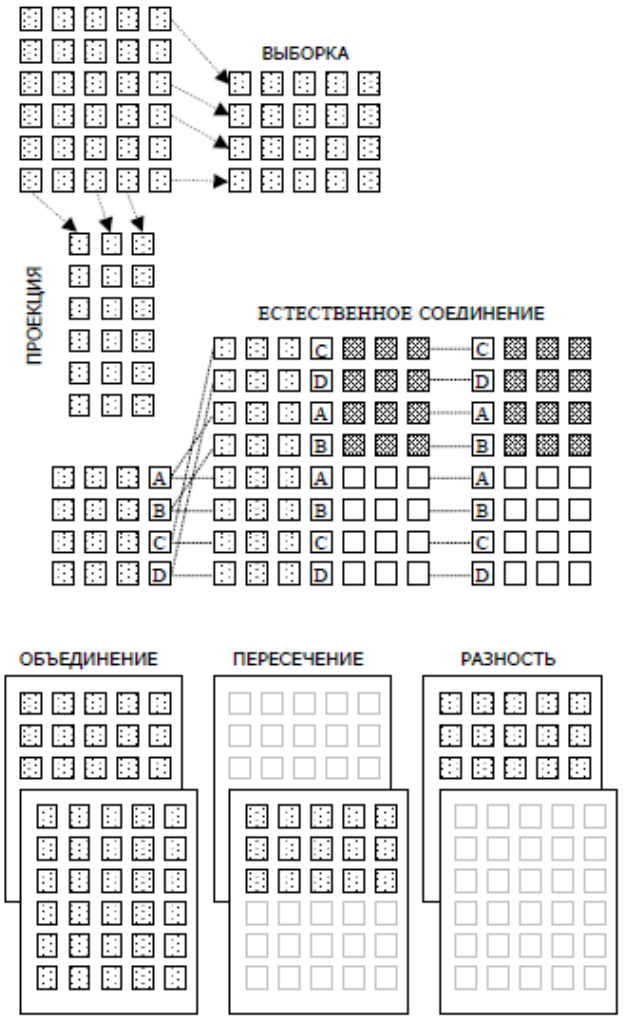


Рисунок 1.1 – Деякі операції реляційної алгебри

До складу теоретико-множинних операцій входять традиційні операції над множинами:

– об'єднання;
– перетин;
– різниця;
– декартовий добуток.

Спеціальні реляційні операції включають:

– вибірку;
– проекцію;
– природне з'єднання;
– ділення.

Операції об'єднання, перетину і різниці вимагають від операндів сумісності за типом. Два відношення сумісні за типом, якщо:

– кожне з них має одну і ту ж безліч імен атрибутів (одна і та ж міра)

– відповідні атрибути (з однаковими іменами) визначені на одному і тому ж домені.

2 Операція об'єднання

Отношение A

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21

Отношение B

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1896	Галкин	Иваново	40

Об'єднанням двох сумісних за типом стосунків A і B (A ∪ B) називається відношення з тим же заголовком, як в стосунках A і B, і з тілом, що складається з безлічі кортежів t, що належать A або B або обом стосункам

A ∪ B

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21
1896	Галкин	Иваново	40

При виконанні операції об'єднання двох стосунків створюється відношення, що включає кортежі, що входять хоч би в один із стосунків-операндів. Зверніть увагу, що кортежі, що повторюються, віддаляються за визначенням відношення.

3 Операція перетин

Перетином двох сумісних за типом стосунків A і B (A ∩ B) називається відношення з тим же заголовком, як в стосунках A і B, і з тілом, що складається з безлічі кортежів t, що належать одночасно обом стосункам A і B. Операція перетину двох стосунків створює відношення, що включає усі кортежі, що входять в обидва відношення-операнди.

A ∩ B

ID_NUM	NAME	CITY	AGE
1809	Иванов	Москва	45

4 Операція різниці

Різницею двох сумісних за типом стосунків A і B (A - B) називається відношення з тим же заголовком, як в стосунках A і B, і з тілом, що складається з безлічі кортежів t, що належать відношенню A і що не належать відношенню B.

Відношення, що є різницею двох стосунків включає усі кортежі, що входять в перше відношення, такі, що жоден з них не входить в друге відношення.

A – B

ID_NUM	NAME	CITY	AGE
1996	Петров	Нижний Новгород	39
1777	Сидоров	Рязань	21

5 Декартовий добуток

Декартовий добуток двох стосунків A і B (A x B), де A і B не мають загальних імен атрибутів, визначається як відношення із заголовком, що є зчепленням двох заголовків початкових стосунків A і B, і тілом, що складається з безлічі кортежів t таких що першим є будь-який кортеж відношення A, а другим, - будь-який кортеж, що належить відношенню B. Кардинальне число результуючого відношення дорівнює добуток кардинальних чисел початкових стосунків, а міра дорівнює сумі мір.

Нехай відношення A - містить імена усіх поточних постачальників, а відношення B - номери усіх поточних деталей. Тоді A ´ B - це усе поточні пари постачальник - деталь і деталь - постачальник

Отношение A

S1
S2
S3

Отношение B

P1
P2
P3
P4

A × B

S1	P1
S1	P2
S1	P3
S1	P4

S2	P1
S2	P2
S2	P3
S2	P4

S3	P1
S3	P2
S3	P3
S3	P4

На практиці явне використання операція декартовий добуток вимагається тільки для дуже складних запитів. Ця операція включена в реляційну алгебру з концептуальних міркувань: (декартовий добуток вимагається як проміжний крок при визначенні операції θ - з'єднання, яка використовується досить часто).

Лекція

Тема: ***«Проектування баз даних»***

План лекції

1 Визначення стратегії як перший етап проектування баз даних
2 ER- моделювання предметної області.

Зміст лекції

1 Визначення стратегії як перший етап проектування баз даних

Метою етапу визначення стратегії є формування спільно із замовником прикладних моделей, вироблення переліку рекомендацій і прийняття погодженого плану, складеного з врахуванням наявних організаційних, фінансових і технічних обмежень, що відображує як поточні, так і майбутні потреби організації.

Опис

Детальний аналіз структури організації може бути початковою базою для розробки перспективного плану створення системи, але витрати на його проведення навряд чи будуть економічно виправданими. Як правило, стратегія розробки інформаційної системи визначається в результаті узагальненого аналізу, на підставі якого потім будується великомасштабна модель прикладної області. Стратегія повинна визначатися в досить стислі терміни з тим, щоб результати проектування не втрачали актуальності.

Результати цього етапу повинні узгоджуватися один з одним і бути досить чіткий сформульовані, щоб замовник міг легко співвіднести запропоновану стратегію зі своїми завданнями і зрозуміти, які саме чинники зумовили ухвалення тих або інших рішень. Крім того, йому має бути викладена перспектива подальшого аналізу, уточнення і передивляється стратегічних рішень.

Результати

Основними результатами цього етапу мають бути:

✓ опис напрямів прикладній діяльності, зокрема формулювання її цілей і завдань, визначення пріоритетів, обмежень, критичних чинників успіху і ключових показників ефективності;

✓ опис цілей і завдань автоматизації, витрат і можливого виграшу;

✓ узагальнена діаграма єств і зв'язків;

✓ узагальнена ієрархічна схема завдань (виробничих і управлінських);

✓ рекомендації відносно майбутньої реалізації і подолання можливих труднощів;

✓ визначення кордонів і обкреслювання сфери вживання системи баз даних;

✓ можлива архітектура системи;
✓ поетапний план проектування бази даних.

Такий підхід до моделювання наочної області передбачає її відображення з трьох різних точок зору:
✓ загальний напрям прикладній діяльності;
✓ прикладні завдання;
✓ інформаційні потреби.

Побудовані на цьому етапі моделі мають бути зрозумілими для замовника, а для того щоб досягти повного узгодження різних точок зору на прикладну область і можливі напрями діяльності, проводяться групові координаційні наради.

Стратегії еволюціонують і розвиваються, обставини і завдання з часом можуть змінюватися, отже неможливо запропонувати директивний метод моделювання стратегій. Тому поважно сполучати неупередженість до нових рішень із здатністю швидко оцінювати альтернативні напрями діяльності з врахуванням заданих обмежень і пріоритетів.

Ключові чинники успіху

На першому етапі слід виділити в першу чергу такі чинники успіху:
✓ використання всіх можливих засобів, що дають можливість підвищити рівень знань наочної області;
✓ активна участь в розробці стратегії осіб, які добре розуміють справжні потреби організації;
✓ проведення плідних нарад з ретельним розглядом всіх питань.

3 ER- моделювання предметної області.

Основні поняття

Метод суттєвість-зв'язок називають також методом «ER-діаграм». Метод заснований на використанні діаграм, званих відповідно діаграмами ER-екземплярів і діаграмами ER-типів.

Основними поняттями методу суттєвість-зв'язок є наступні:
• суттєвість
• атрибут суттєвості
• ключ суттєвості
• зв'язок між суттєвостіми
• міра зв'язку
• клас приналежності екземплярів суттєвості
• діаграми ER-екземплярів
• діаграми ER-типів.

Суттєвість є об'єктом, інформація про яке зберігається в БД. Екземпляри суттєвості відрізняються один від одного і однозначно ідентифікуються. Назвами єств є, як правило, іменники, наприклад: ВИКЛАДАЧ, ДИСЦИПЛІНА, КАФЕДРА, ГРУПА.

Атрибут є властивістю суттєвості. Це поняття аналогічно поняттю атрибуту у відношенні. Так, атрибутами суттєвості ВИКЛАДАЧ може бути його Прізвище, Посада, Стаж (викладацький) і так далі

Ключ суттєвості - атрибут або набір атрибутів, використовуваний для ідентифікації екземпляра суттєвості.

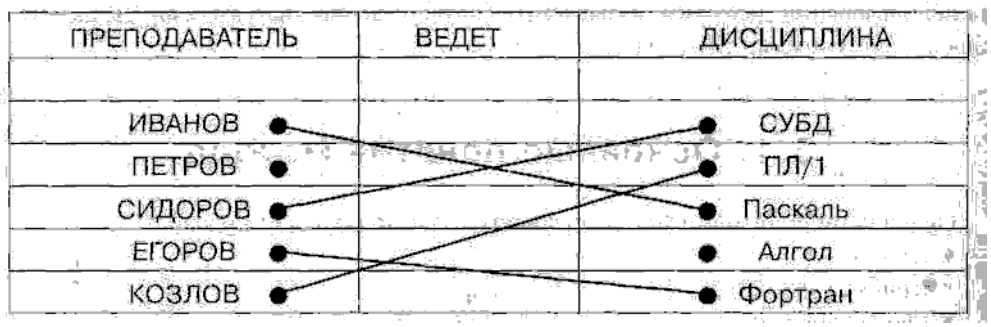
Зв'язок два або більш за суттєвості - передбачає залежність між атрибутами цих єств. Назва зв'язку зазвичай представляється дієсловом. Прикладами зв'язків між суттєвостіми є наступні: ВИКЛАДАЧ ВЕДЕ ДИСЦИПЛІНУ (Іванов ВЕДЕ «Бази даних»), ВИКЛАДАЧ ВИКЛАДАЄ В ГРУППЕ (Іванов ВИКЛАДАЄ В 256 групі), ВИКЛАДАЧ ПРАЦЮЄ НА КАФЕДРІ (Іванов ПРАЦЮЄ НА 25 кафедрі).

Приведені визначення суттєвості і зв'язку не повністю формалізовані, але прийнятні для практики. Слід мати на увазі, що в результаті проектування можуть бути отримані декілька варіантів однієї БД. Так, два різні проектувальники, розглядаючи одну і ту ж проблему з різних точок зору, можуть отримати різні набори єств і зв'язків. При цьому обоє варіанту можуть бути робітниками, а вибір кращого з них буде результатом особистих переваг.

З метою підвищення наочності і зручності проектування для вистави єств, екземплярів єств і зв'язків між ними використовуються наступні графічні засоби:

- діаграми ER-екземплярів
- діаграми ER-типів, або ER-діаграми.

Приклад діаграми ER-екземплярів



Діаграма ER-екземплярів показує, яку конкретно дисципліну веде кожен з викладачів.

Розглянемо діаграму ER-типів, відповідну розглянутій діаграмі ER-екземплярів.

На початковому етапі проектування БД виділяються атрибути, що становлять ключі єств.

На основі аналізу діаграм ER-типів формуються стосунки проектованої БД. При цьому враховується міра зв'язку єств і клас їх приладдя, яке, у свою

чергу, визначається на основі аналізу діаграм ER-екземплярів відповідних єств.

Міра зв'язку є характеристикою зв'язку між суттєвостіми, який може бути типа: 1:1, 1:M, M:1, M:M.

Клас приналежності (КП) суттєвості може бути: обов'язковим і необов'язковим.

Клас приналежності суттєвості є обов'язковим, якщо всі екземпляри цього суттєвості обов'язково беруть участь в даному зв'язку, інакше клас приналежності суттєвості є необов'язковим.

Варіюючи класом приналежності єств для кожного з названих типів зв'язку, можна отримати декілька варіантів діаграм ER-типів.

Розглянемо приклади деяких з них.

Приклад 1. Зв'язки типа 1:1 і необов'язковий клас приналежності.

У приведеній діаграмі міра зв'язку між суттєвостіми 1:1, а клас приналежності обох єств необов'язковий.

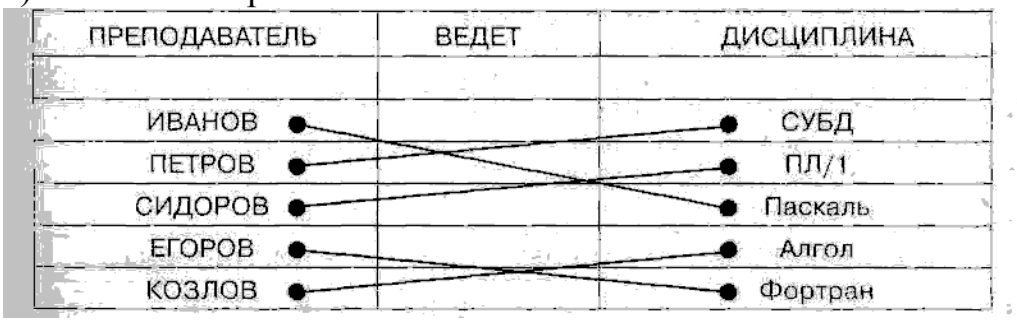
Дійсно, з малюнка видно наступне:

• кожен викладач веде не більш за одну дисципліну, а кожна дисципліна ведеться не більше ніж одним викладачем (міра зв'язку 1:1);

• деякі викладачі не ведуть жодної дисципліни і є дисципліни, які не веде жоден з викладачів (клас приналежності обох єств необов'язковий).

Приклад 2. Зв'язки типа 1:1 і обов'язковий клас приналежності. На мал. приведені діаграми, в яких міра зв'язку між суттєвостіми 1:1, а клас приналежності обох єств обов'язковий.

а) ER-екземплярів



б) ER-типов

Діаграми для зв'язку 1:1 і обов'язковим КП обох єств

В цьому випадку кожен викладач веде одну дисципліну і кожна дисципліна ведеться одним викладачем.

Можливі два проміжні варіанти з необов'язковим класом приналежності однієї з єств.

Зауваження.

- На діаграмах ER-типів обов'язкова участь в зв'язку екземплярів суттєвості наголошується блоком з крапкою усередині, суміжним з блоком цього суттєвості

- При необов'язковій участі екземплярів суттєвості в зв'язку додатковий блок до блоку суттєвості не пристроюється, а крапка розміщується на лінії зв'язку

- Символи на лінії зв'язку вказують на міру зв'язку.

Під кожним блоком, відповідним деякому єству, вказується її ключ, що виділяється підкресленням. Багатокрапка за ключовими атрибутами означає, що можливі інші атрибути суттєвості, але жоден з них не може бути частиною її ключа. Ці атрибути виявляються після формування стосунків.

Розглянемо приклади варіантів з мірою зв'язку 1:М або М:1.

Приклад 3. Зв'язки типа 1:М.

Кожен викладач може вести декілька дисциплін, але кожна дисципліна ведеться одним викладачем.

Приклад 4. Зв'язки типа М:1.

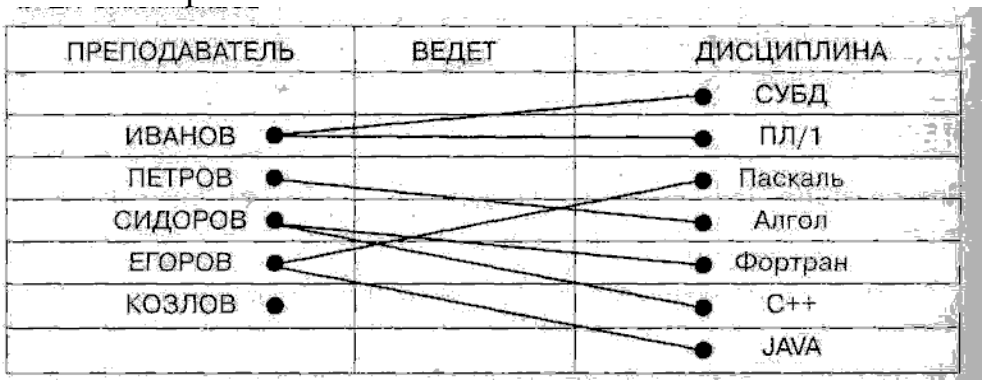
Кожен викладач може вести одну дисципліну, але кожну дисципліну можуть вести декілька викладачів.

Приклади з типом зв'язку 1:М або М:1 можуть мати ряд варіантів, що відрізняються класом приналежності однієї або обох єств. Позначимо обов'язковий клас приналежності символом «О», а необов'язковий - символом «Н», тоді варіанти для зв'язку типа 1:М умовно можна представити як: О-про, О-Н, Н-о, Н-Н. Для зв'язку типа М:1 також є 4 аналогічних варіанту.

Приклад 5. Зв'язки типа 1:М варіант Н-о.

Кожен викладач може вести декілька дисциплін або жодній, але кожна дисципліна ведеться одним викладачем

а) ER-екземплярів



б) ER-типов

Приклад 6. Зв'язки типа M:M.

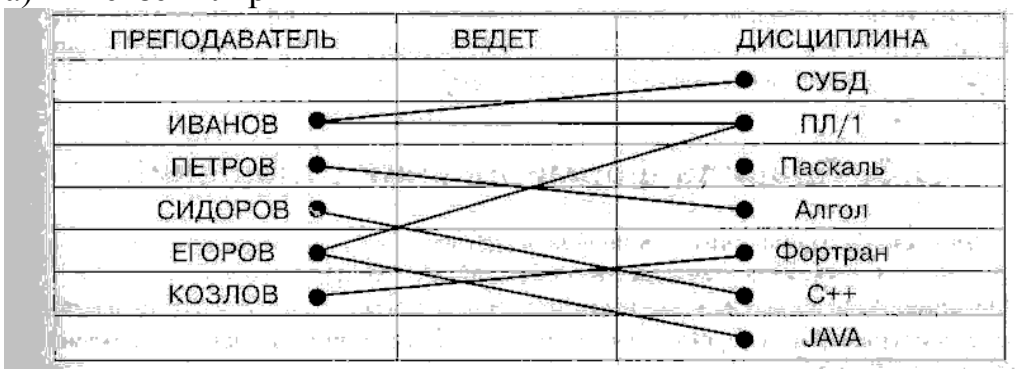
Кожен викладач може вести декілька дисциплін, а кожна дисципліна може вестися декількома викладачами.

Як і в разі інших типів зв'язків, для зв'язку типа M:M можливі 4 варіанти, приладдя єств, що відрізняється класом.

Приклад 7. Зв'язки типа M:M і варіант класу приналежності O-H.

Допустимо, що кожен викладач веде не менше однієї дисципліни, а дисципліна може вестися більш ніж одним викладачем, є і такі дисципліни, які ніхто не веде.

а) ER-екземплярів



б) ER-типов

Виявлення єств і зв'язків між ними, а також формування на їх основі діаграм ER-типів виконується на початкових етапах методу суттєвість-зв'язок.

Лекція

Тема: ***«Залежності між атрибутами»***

План лекції

1 Залежності між атрибутами
2 Аксіоматика функціональних залежностей

Зміст лекції

1 Залежності між атрибутами

Розглянемо основні види залежностей між атрибутами стосунків: функціональні, транзитивні і багатозначні.

Поняття функціональної залежності є базовим, оскільки на його основі формулюються визначення всіх останніх видів залежностей.

Атрибут B функціонально залежить від атрибуту A, якщо кожному значенню A відповідає в точності одне значення B. Математіческьки функціональна залежність B від A позначається записом A→B. Це означає що у всіх кортежах з однаковим значенням атрибуту A атрибут B матиме також одне і те ж значення. Відзначимо, що A і B можуть бути складеними - складатися з двох і більш за атрибути.

Функціональна взаємозалежність. Якщо існує функціональна залежність вигляду A→B і B→A, то між A і B є взаємно однозначна відповідність, або функціональна взаємозалежність. Наявність функціональної взаємозалежності між атрибутами A і B позначимо як

A↔B або B↔A.

Приклад. Хай є деяке відношення, що включає два атрибути, функціонально залежні один від одного. Це серія і номер паспорта (N) і прізвище, ім'я і по батькові власника (ПІБ). Наявність функціональної залежності поля ПІБ від N означає не лише той факт, що значення поля N однозначно визначає значення поля ПІБ, але і те, що одному і тому ж значенню поля N відповідає лише єдине значення поля ПІБ.

Зрозуміло, що в даному випадку діє і зворотна ФЗ: кожному значенню поля ПІБ відповідає лише одне значення поля N. У даному прикладі передбачається, що ситуація наявності повного збігу прізвищ, імен і по батькові двох людей виключена.

Якщо відношення знаходиться в 1НФ, то всі неключові атрибути функціонально залежать від ключа з різною мірою залежності.

Частковою залежністю (частковою функціональною залежністю) називається залежність неключового атрибуту від частини складеного ключа.

Альтернативним варіантом є повна функціональна залежність неключового атрибуту від всього складеного ключа.

Атрибут C залежить від атрибуту A транзитивно (існує транзитивна залежність), якщо для атрибутів A, B, C виконуються умови A→B і B→C, але зворотна залежність відсутня.

Між атрибутами може мати місце багатозначна залежність.

У відношенні R атрибут B багатозначно залежить від атрибуту A, якщо кожному значенню A відповідає безліч значень B, не пов'язаних з іншими атрибутами з R

Багатозначні залежності можуть бути «один до багатьох» (1:M), «багато до одного» (M:1) або «багато до багатьох» (M:M), що позначаються відповідно:

A=>B, A<=B і A<=>B

Взаємно незалежні атрибути. Два або більш за атрибут називаються взаємно незалежними, якщо жоден з цих атрибутів не є функціонально залежним від інших атрибутів.

В разі двох атрибутів відсутність залежності атрибуту A від атрибуту B можна позначити так: A¬→B. Випадок, коли A¬→B і B¬→A, можна позначити A¬=B.

2 Аксіоматика функціональних залежностей

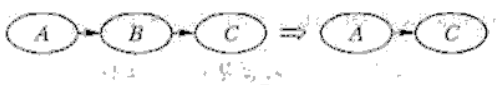
Стосовно заданого реляційного відношення R ми можемо розглядати безліч функціональних залежностей F, які визначені на нім.

Як довів Ст Армстронг, безліч функціональних залежностей F має певні властивості, що перераховані в таблиці 1. У лівому стовпці таблиці згадані властивості, записані в аналітичному вигляді, в правом — в графічному. Більшість тверджень тут приведена у формі «якщо ..., то ... », яку слід розуміти так: якщо деякі функціональні залежності належать безлічі F, то цій безлічі належать і ті залежності, що вказані після слова «то».

Не всі з приведених властивостей є незалежними, а саме властивості (4) -(8) виводяться з (1), (2), (3), який утворюють повну систему аксіом функціональних залежностей.

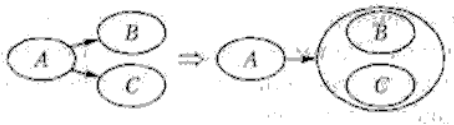
Таблиця 1. Властивості функціональних залежностей

1) Транзитивність: якщо A>B і B>C, то A>C;



2) Проектна: якщо B A, те A>B

3) Адітівность (об'єднання): якщо A>B і A>C, то A>(У, З)



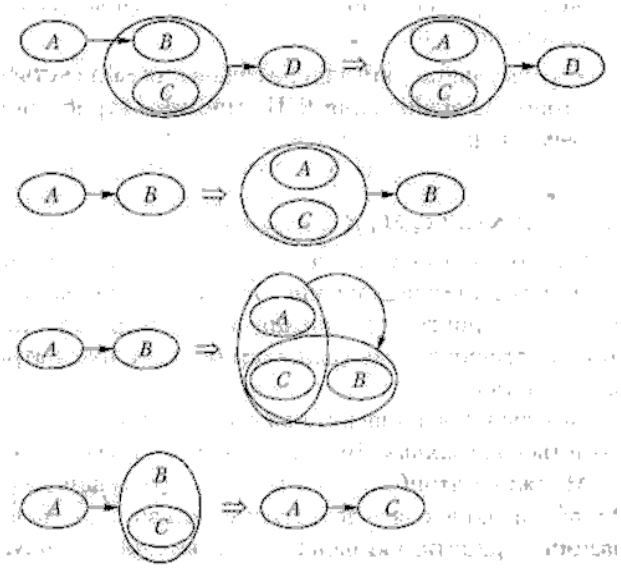
4) Рефлексивність: A>A

5) Псевдотранзитивність: якщо A>B і (У, З) >D, то (A, З) >D

6) Продовження: якщо A>B, то (A, З) >B для будь-якого атрибуту З

7) Поповнення: якщо A>B, то (A, З)>(У, З) для будь-якого атрибуту З

8) Декомпозиція: якщо A>B і З B, то A>C



Лекція

Тема: ***«Перші нормальні форми»***

План лекції

1 Перша нормальна форма
2 Друга нормальна форма
3 Третя нормальна форма
4 Нормальна форма Бойса-Кодда

Зміст лекції

1 Перша нормальна форма

Процес проектування БД з використанням методу нормальних форм є ітераційним і полягає в послідовному перекладі стосунків з першої нормальної форми в нормальні форми вищого порядку по певних правилах. Кожна наступна нормальна форма обмежує певного типа функціональних залежностей, усуває відповідні аномалії при виконанні операцій над стосунками БД і зберігає властивості попередніх нормальних форм.

Виділяють наступну послідовність нормальних форм:

- перша нормальна форма (1НФ);
- друга нормальна форма (2НФ);
- третя нормальна форма (3НФ);
- посилена третя нормальна форма, або нормальна форма Бойса - Кодда (НФБК);
- четверта нормальна форма (4НФ);
- п'ята нормальна форма (5НФ).

Перша нормальна форма. Відношення знаходиться в 1НФ, якщо всі його атрибути є простими (мають єдине значення). Початкове від носіння будується так, щоб воно було в 1НФ.

Перевід відношення в наступну нормальну форму здійснюється методом «декомпозиції без втрат». Така декомпозиція повинна забезпечити те, що запити (вибірка даних по умові) до вихідного відношення і до стосунків, що отримуються в результаті декомпозиції, дадуть однаковий результат.

Основною операцією методу є операція проекції. Пояснимо її на прикладі. Передбачимо, що відносно R(A,B,C,D,E...) усунення функціональної залежності C>d дозволить перевести його в наступну нормальну форму. Для вирішення цього завдання виконаємо декомпозицію відношення R на два нові відношення R1(A, B, 3, E...) і R2(C,D). Відношення R2 є проекцією відношення R на атрибути 3 і D.

Часткова залежність від ключа наводить до наступного:

1. У відношенні присутнє явне і неявне надлишкове дублювання даних

2. Наслідком надлишкового дублювання даних є проблема їх редагування.

Приклад. Нехай дано наступне відношення Students :

StudentID	Name
1	Казаков Петр Владимирович
2	Васильев Иван Аркадьевич
4	Шишкина Дарья Сергеевна

Поле Name містить одночасно прізвище, ім'я і по батькові. Це поле можна розділити на три, тоді отримаємо відношення в 1НФ:

StudentID	LastName	FirstName	MiddleName
1	Казаков	Петр	Владимирович
2	Васильев	Иван	Аркадьевич
4	Шишкина	Дарья	Сергеевна

Слід мати на увазі, що значення полів мають бути логічно неділимі, оскільки прізвище складається з окремих букв, але таке ділення не матиме сенсу для відношення Students.

2 Друга нормальна форма

Відношення знаходиться в 2НФ, якщо воно знаходиться в 1НФ і кожному неключовому атрибуті функціонально повно залежить від первинного ключа (складеного).

Для усунення часткової залежності і перекладу відношення в 2НФ необхідно, використовуючи операцію проекції, розкласти його на декілька стосунків таким чином:

– побудувати проекцію без атрибутів, що знаходяться в частковій функціональній залежності від первинного ключа;

– побудувати проекції на частини складеного первинного ключа і атрибути, залежні від цих частин

Приклад. Розглянемо наступне відношення:

StudentID	LastName	FirstName	MiddleName	Course	Score
1	Казаков	Петр	Владимирович	информатика	5
2	Васильев	Иван	Аркадьевич	математика	4

4	Шишкина	Дарья	Сергеевна	математика	5
2	Васильев	Иван	Аркадьевич	информатика	3

Первинним ключем тут буде {StudentID, Course}, тоді проаналізуємо ФЗ цього відношення, де детермінантом є первинний ключ, а в правій частині неключовий атрибут. Розглянемо наступну ФЗ:

$$\{StudentID, Course\} \rightarrow \{LastName\}$$

Очевидно, що вона є такою, що приводиться, оскільки прізвище залежить тільки від StudentID, а значить існує ФЗ: {StudentID } → {LastName}.

Отже це відношення не знаходиться в 2НФ. Проведемо декомпозицію початкового відношення на наступних два.

Students:

StudentID	**LastName**	**FirstName**	**MiddleName**
1	Казаков	Петр	Владимирович
2	Васильев	Иван	Аркадьевич
4	Шишкина	Дарья	Сергеевна

Scores:

StudentID	**Course**	**Score**
1	информатика	5
2	математика	4
4	математика	5
2	информатика	3

3 Третя нормальна форма

Визначення 1. Відношення знаходиться в 3НФ, якщо воно знаходиться в 2НФ і кожному неключовому атрибуті нетранзитивно залежить від первинного ключа.

Існує і альтернативне визначення.

Визначення 2. Відношення знаходиться в 3НФ в тому і лише в тому випадку, якщо всі неключові атрибути відношення взаємно незалежні і повністю залежать від первинного ключа.

На практиці побудова 3НФ схем стосунків в більшості випадків є достатньою і приведенням до них процес проектування реляційної БД закінчується.

Приклад. Розглянемо наступне відношення Students :

StudentID	LastName	FirstName	MiddleName	GroupID	Supervisor
1	Казаков	Петр	Владимирович	1	Царев С.М.
2	Васильев	Иван	Аркадьевич	2	Пестов Д.Н.
4	Шишкина	Дарья	Сергеевна	1	Царев С.М.

Оскільки існують ФЗ: StudentID → GroupID і GroupID → Supervisor, то атрибут Supervisor транзитивно залежить від первинного ключа StudentID. Отже відношення не знаходиться в 3НФ. Приведемо відношення до 3НФ.

Students:

StudentID	LastName	FirstName	MiddleName	GroupID
1	Казаков	Петр	Владимирович	1
2	Васильев	Иван	Аркадьевич	2
4	Шишкина	Дарья	Сергеевна	1

Groups:

GroupID	Supervisor
1	Царев С.М.
2	Пестов Д.Н.

4 Нормальна форма Бойса-Кодда

При визначенні 3НФ було зроблене допущення про те, що відношення має тільки один потенційний ключ, який і є первинним. Якщо розглянути загальніший випадок, то первинне визначення, це Э. Коддом для 3НФ, опиняється не в усіх випадках задовільним. Зокрема, воно неадекватне при виконанні наступних умов :

1) відношення має два (чи більше) потенційні ключі;
2) ці потенційні ключі є складеними;
3) вони перекриваються (тобто мають принаймні один загальний атрибут).

Тому згодом початкове визначення 3НФ було замінене строгішим визначенням Бойса-Кодда.

На практиці комбінація усіх трьох умов зустрічається рідко, і для стосунків, в яких не виконуються усі ці три умови, 3НФ і нормальна форма Бойса-Кодда повністю еквівалентні.

Нормальна форма Бойса-Кодда (НФБК). Відношення знаходиться в нормальній формі Бойса-Кодда тоді і тільки тоді, коли кожна її нетривіальна

функціональна залежність, що не приводиться ліворуч, має як свій детермінант деякий потенційний ключ.

Приклад. Розглянемо наступне відношення:

StudentID	**CourseID**	**TeacherID**
1	20	23
2	20	12
4	15	9

Кожен кортеж означає, що деякий студент вивчає певну дисципліну у вказаного викладача. При цьому існують наступні обмеження:

1. Кожен викладач веде тільки одну дисципліну.
2. Одну дисципліну може вести декілька викладачів.
3. Для деякого студента одну дисципліну веде тільки один викладач.

У цьому відношенні є два потенційні ключі {StudentID, CourseID} і {StudentID, TeacherID}. Обидва ключі є складеними і вони мають загальний атрибут StudentID, тобто перекриваються. Таким чином виконано усі три умови, які можуть привести до ситуації, коли відношення може знаходитися в 3НФ, але не знаходиться в НФБК.

Очевидно, що відношення знаходиться в 3НФ:

− значення усіх атрибутів неділимі (1НФ);

− кожен неключовий атрибут не приводиться залежить від первинного ключа (2НФ);

− усі неключові атрибути нетранзитивно залежать від потенційного ключа (3НФ).

Проте, в цьому відношенні існує ФЗ TeacherID → CourseID і TeacherID при цьому не є потенційним ключем, отже відношення не знаходиться в НФБК.

Якщо провести декомпозицію цього відношення два: {StudentID, CourseID} і {CourseID, TeacherID}, те втратимо інформацію про те, який саме викладач веде дисципліну для конкретного студента. Це пояснюється тим, що початкове відношення є атомарним, тобто його не можна розбити на дві незалежні проекції.

Таким чином, в процесі нормалізації не завжди є сенс прагнути до атомарних стосунків, але, проте, для основних об'єктів бази даних рекомендується домагатися атомарності.

Якщо відношення знаходиться тільки в 1НФ, але не знаходиться в 2НФ і 3НФ, то можна говорити про надмірність інформації. Надмірність не лише збільшує об'єм і трудомісткість заповнення бази даних, але і може привести до аномалій оновлення. Аномалії можуть призводити до труднощів при вставці, оновленні і видаленні, а головне до спотворення або втрати інформації.

Лекція

Тема: ***«Введення в структуровану мову запитів SQL»***

План лекції

1 Стандарт і реалізація мови SQL
2 Введення в технологію клієнт-сервер
3 Типи команд SQL

Зміст лекції

1 Стандарт і реалізація мови SQL

Зростання кількості даних, необхідність їх зберігання і обробки привели до того, що виникла потреба в створенні стандартної мови баз даних, який міг би функціонувати в багаточисельних комп'ютерних системах різних видів. Дійсно, з його допомогою користувачі можуть маніпулювати даними незалежно від того, чи працюють вони на персональному комп'ютері, мережевій робочій станції або універсальній ЕОМ.

Однією з мов, що з'явилися в результаті розробки реляційної моделі даних, є мова SQL (Structured Query Language), яка в даний час набула дуже широкого поширення і фактично перетворилася на стандартну мову реляційних баз даних. Стандарт на мову SQL був випущений Американським національним інститутом стандартів (ANSI) в 1986 р., а в 1987 р. Міжнародна організація стандартів (ISO) прийняла його як міжнародного. Нинішній стандарт SQL відомий під назвою SQL/92.

З використанням будь-яких стандартів зв'язані не лише багаточисельні і сповна очевидні переваги, але і певні недоліки. Перш за все, стандарти направляють в певне русло розвиток відповідної індустрії; в разі мови SQL наявність твердих засадничих принципів наводить, кінець кінцем, до сумісності його різних реалізацій і сприяє як підвищенню переносимості програмного забезпечення і баз даних в цілому, так і універсальності роботи адміністраторів баз даних. З іншого боку, стандарти обмежують гнучкість і функціональні можливості конкретної реалізації. Під реалізацією мови SQL розуміється програмний продукт SQL відповідного виробника. Для розширення функціональних можливостей багато розробників, що дотримуються прийнятих стандартів, додають до стандартної мови SQL різні розширення. Слід зазначити, що стандарти вимагають від будь-якої закінченої реалізації мови SQL наявності певних характеристик і у загальних рисах відображають основні тенденції, які не лише наводять до сумісності між всіма конкуруючими реалізаціями, але і сприяють підвищенню значущості програмістів SQL і користувачів реляційних баз даних на сучасному ринку програмного забезпечення.

Всі конкретні реалізації мови декілька відрізняються один від одного. На користь самих же виробників гарантувати, аби їх реалізація відповідала сучасним стандартам ANSI в частині переносимості і зручності роботи користувачів. Проте, кожна реалізація SQL містить удосконалення, що відповідають вимогам того або іншого сервера баз даних. Ці удосконалення або розширення мови SQL є додатковими командами і опціями, що є додаваннями до стандартного пакету і доступні в даній конкретній реалізації.

В даний час мова SQL підтримується багатьма десятками СУБД різних типів, розроблених для найрізноманітніших обчислювальних платформ, починаючи від персональних комп'ютерів і закінчуючи мейнфреймами.

Всі мови маніпулювання даними, створені для багатьох СУБД до появи реляційних баз даних, були орієнтовані на операції з даними, представленими у вигляді логічних записів файлів. Зрозуміло, це вимагало від користувача детального знання організації зберігання даних і серйозних зусиль для вказівки того, які дані необхідні, де вони розміщуються і як їх отримати.

Дана мова SQL орієнтована на операції з даними, представленими у вигляді логічно взаємозв'язаних совокупностей таблиць-стосунків. Найважливіша особливість його структур — орієнтація на кінцевий результат обробки даних, а не на процедуру цієї обробки. Мова SQL сам визначає, де знаходяться дані, індекси і навіть які найбільш ефективні послідовності операцій слід використовувати для здобуття результату, а тому вказувати ці деталі в запиті до бази даних не потрібно.

2 Введення в технологію клієнт-сервер

У зв'язку з розширенням ринку інформаційних послуг виробники програмного забезпечення стали випускати усе більш інтелектуальні, а значить, і об'ємні програмні комплекси. Багато організацій і окремі користувачі часто не могли розмістити придбані продукти на власних ЕОМ. Для обміну інформацією і її поширення були створені мережі ЕОМ, а узагальнювальні програми і дані стали встановлювати на спеціальних файлових серверах.

Завдяки СУБД, що працюють з файловими серверами, безліч користувачів отримують доступ до одних і тих же баз даних. Спрощується розробка різних автоматизованих систем управління організаціями. Проте, при такому підході вся обробка запитів з програм або з терміналів призначених для користувача ЕОМ на них і виконується, тому для реалізації навіть простого запиту необхідно прочитувати з файлового сервера або записувати на нього цілі файли, а це веде до конфліктних ситуацій і перевантаження мережі. Для виключення вказаних недоліків була запропонована технологія клієнт-сервер, але при цьому знадобилася єдина мова спілкування з сервером – вибір ліг на SQL.

Технологія клієнт-сервер означає такий спосіб взаємодії програмних компонентів, при якій вони утворюють єдину систему. Як видно з самої

назви, існує деякий клієнтський процес, що вимагає певних ресурсів, а також серверний процес, ці ресурси що надає. Зовсім необов'язково, аби вони знаходилися на одному комп'ютері. Зазвичай прийнято розміщувати сервер на одному вузлі локальної мережі, а клієнтів – на інших вузлах.

У контексті бази даних клієнт управляє призначеним для користувача інтерфейсом і логікою додатка, діючи як робоча станція, на якій виконуються додатки баз даних. Клієнт приймає від користувача запит, перевіряє синтаксис і генерує запит до бази даних на мові SQL або іншій мові бази даних, відповідному логіку додатка. Потім передає повідомлення сервера, чекає вступу відповіді і форматує отримані дані для представлення їх користувачеві. Сервер приймає і обробляє запити до бази даних, після чого відправляє отримані результати назад клієнтові. Така обробка включає перевірку повноважень клієнта, забезпечення вимог цілісності, а також виконання запиту і оновлення даних. Окрім цього підтримується управління паралельністю і відновленням. Архітектура клієнт-сервер володіє рядом переваг:

- забезпечується ширший доступ до існуючих баз даних;
- підвищується загальна продуктивність системи: оскільки клієнти і сервер знаходяться на різних комп'ютерах, їх процесори здатні виконувати додатки паралельно. Налаштування продуктивності комп'ютера з сервером спрощується, якщо на нім виконується лише робота з базою даних;
- знижується вартість апаратного забезпечення; досить потужний комп'ютер з великим пристроєм зберігання потрібний лише серверу – для зберігання і управління базою даних;
- скорочуються комунікаційні витрати. Додатки виконують частину операцій на клієнтських комп'ютерах і посилають через мережу лише запити до баз даних, що дозволяє значно скоротити об'єм що пересилаються по мережі даних;
- підвищується рівень несуперечності даних. Сервер може самостійно управляти перевіркою цілісності даних, оскільки лише на нім визначаються і перевіряються всі обмеження. При цьому кожному застосуванню не доведеться виконувати власну перевірку;
- архітектура клієнт-сервер природно відображується на архітектуру
- відкритих систем.

Подальше розширення дворівневої архітектури клієнт-сервер передбачає розділення функціональної частини колишнього, «товстого» (інтелектуального) клієнта на дві частини. У трирівневій архітектурі клієнт-сервер «тонкий» (неінтелектуальний) клієнт на робочій станції управляє лише призначеним для користувача інтерфейсом, тоді як середній рівень обробки даних управляє всією останньою логікою додатка. Третій рівень — сервер бази даних. Ета трирівнева архітектура виявилася більш відповідною для деяких середовищ – наприклад, для мереж Internet і intranet, де як клієнт може виступати звичайний браузер Web-.

3 Типи команд SQL

Реалізація в SQL концепції операцій, орієнтованих на табличне представлення даних, дозволила створити компактну мову з невеликим набором пропозицій. Мова SQL може використовуватися як для виконання запитів до даних, так і для побудови прикладних програм.

Основні категорії команд мови SQL призначені для виконання різних функцій, включаючи побудову об'єктів бази даних і маніпулювання ними, початкове завантаження даних в таблиці, оновлення і видалення існуючої інформації, виконання запитів до бази даних, управління доступом до неї і її загальне адміністрування.

Основні категорії команд мови SQL:

- DDL – мова визначення даних;
- DML – мова маніпулювання даними;
- DQL – мова запитів;
- DCL – мова управління даними;
- команди адміністрування даних;
- команди управління транзакціями

Визначення структур бази даних (DDL)

Мова визначення даних (Data Definition Language, DDL) дозволяє створювати і змінювати структуру об'єктів бази даних, наприклад, створювати і видаляти таблиці. Основними командами мови DDL є наступні: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX.

Маніпулювання даними (DML)

Мова маніпулювання даними (Data Manipulation Language DML) використовується для маніпулювання інформацією усередині об'єктів реляційної бази даних за допомогою трьох основних команд: INSERT, UPDATE, DELETE.

Вибірка даних (DQL)

Мова запитів DQL найбільш відома користувачам реляційної бази даних, не дивлячись на те, що він включає одну команду: SELECT. Ця команда разом зі своїми багаточисельними опціями і пропозиціями використовується для формування запитів до реляційної бази даних.

Мова управління даними (DCL)

Команди управління даними дозволяють управляти доступом до інформації, бази даних, що знаходиться усередині. Як правило, вони використовуються для створення об'єктів, пов'язаних з доступом до даних, а також служать для контролю над розподілом привілеїв між пользователями. Команди управління даними наступні: GRANT, REVOKE.

Команди адміністрування даних

За допомогою команд адміністрування даних користувач здійснює контроль за виконуваними діями і аналізує операції бази даних; вони також можуть виявитися корисними при аналізі производительности системи. Не

слід плутати адміністрування даних з адмініструванням бази даних, яке є загальним управлінням базою даних і має на увазі використання команд всіх рівнів.

Команди управління транзакціями

Існують наступні команди, що дозволяють управляти транзакціями бази даних: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION.

Лекція

Тема: ***«Проектування таблиць бази даних. Типи даних»***

План лекції

Зміст лекції

1 Типи даних

1.1 Класифікація типів даних

У усіх мовах програмування, а також в SQL важливе місце займають підтримувані типи даних. Дані, які зберігаються в пам'яті комп'ютера і піддаються обробці, можна віднести до різних типів. Поняття типу даних виникає природним чином, коли необхідно застосувати до них операції обробки.

У специфікації SQL :2003 визнані п'ять зумовлених загальних типів, усередині яких можуть бути підтипи, :

– строковий (символьний) :
    а) CHARACTER (чи CHAR);
    б) CHARACTER VARYING (чи VARCHAR);
    в) CHARACTER LARGE OBJECT (чи CLOB);
– числовий:
– точні числові типи:
    а) integer;
    б) smallint;
    в) bigint;
    г) numeric;
    д) decimal;
– приблизні числові типи:
    а) real;
    б) double precision;

в) float;

– логічний (булевий) - boolean;

– дати-часу:

а) date;

б) time without time zone*

в) time with time zone;

г) timestamp without time zone;

д) timestamp with time zone;

– інтервальний.

Крім того, існують особливі типи: row (запис), array (масив) і multiset (мультимножина).

1.2 Рядки

Строкові дані (послідовності символів) мають три головні строкові типи. Для стовпця таблиці можна вказати тип character (n) або char(рядок фіксованої довжини), де n - максимальна кількість символів, що містяться в рядку. Якщо (n) не вказано, то передбачається, що рядок складається з одного символу. Якщо в стовпець типу character (n) вводиться m < n символів, то позиції, що залишилися, заповнюються пропусками.

Тип даних CHARACTER VARYING (n) або VARCHAR(рядок змінної довжини) застосовується тоді, коли дані, що вводяться, мають різну довжину і небажано доповнювати їх пропусками. При цьому зберігається тільки та кількість символів, яку ввів користувач. В даному випадку вказівка максимальної кількості символів обов'язкова (на відміну від character).

Дані типів CHARACTER і CHARACTER VARYING можуть брати участь в одних і тих же строкових операціях.

Тип даних CHARACTER LARGE OBJECT (CLOB - великий символьний об'єкт) використовується для представлення дуже великих символьних рядків (наприклад, статей, книг і тому подібне). У деяких СУБД цей тип називається memo, а в інших - text. З дан¬ными цього типу можна виконувати не усі операції, предусмот¬ренные для типів CHARACTER і CHARACTER VARYING. Так, їх не можна використовувати в операціях порівняння, за винятком рівності і нерівності. Крім того, стовпці цього типу не можуть бути первинними і зовнішніми ключами, а також бути оголошені як унікальні значення, що мають. Інакше кажучи, при створенні таблиць за допомогою оператора CREATE і оголошенні стовпців типу CLOB не можна використовувати ключові слова PRIMARY KEY, FOREIGN KEY і UNIQUE.

У наступному прикладі створюється таблиця із звичайним символь¬ным стовпцем і стовпцем типу CLOB, значення якого можуть містити 100 000 символів :

```
CREATE TABLE myTable
( ( FIELD1 CHARACTER (60)
FIELD2 CLOB (100000));
```

1.3 Числа

Числовий тип даних може бути двох видів точний і приблизний. Точні числові типи дозволяють точно виразити значення числа. Деякі величини мають дуже великий діапазон значень, і в таких випадках досить обмежитися деяким наближеним їх представленням з урахуванням технічних можливостей комп'ютера (розмірів регістра).

До точних числових відносяться наступні п'ять типів:

- integer - ціле (без дробової частини) число. Кількість розрядів (точність) залежить від реалізації SQL. У деяких реалізаціях числа цього типу лежать в діапазоні від - 2 147 483 648 до 2 147 483 647 (четырехбайтное ціле число);

- smallint - мале ціле число. Кількість розрядів залежить від реалізації SQL, але не більше кількості розрядів integer в цій же реалізації. У деяких реалізаціях числа цього типу лежать в діапазоні від - 32 768 до 32 767 (двобайтове ціле число);

- bigint - велике ціле число. Кількість розрядів залежить від реалізації SQL і перевищує кількість розрядів числа типу integer;

- numeric (x, y) - число, в якому усього x розрядів (точність), з яких y розрядів (масштаб) відводиться для дробової частини. Якщо y не вказано (numeric (x)), то для дробової частини відводиться кількість розрядів, встановлена в системі за умовчанням. Якщо не вказані ні x, ні y (numeric), то приймаються обоє ці величини, встановлені за умовчанням. Наприклад, якщо вказаний тип numerc (6, 2), то максимальне значення числа дорівнює 9999.99;

- decimal (x, y) - десяткове число, в якому усього x розрядів, з яких y розрядів відводяться для дробової частини. Якщо x або/і y не вказані, то набувають значень за умовчанням. Цей тип дуже схожий на numeric. Відмінність полягає в тому, що якщо в decimal (x, y) вказані x і y менше, ніж допустимі реалізацією SQL, то використовуватимуться останні. Якщо x і y не вказані, то застосовується система умовчань. Наприклад, ви задали для стовпця тип decimal (6, 2). Якщо реалізація SQL дозволяє, то в цей стовпець можна ввести числа, що перевищують 9999.99. На відміну від decimal (x, y), тип numeric (x, y) жорстко задає діапазон можливих значень числової величини.

До приблизних числових типів відносяться наступні три типи:

- real - дійсне число одинарної точності з плаваючою розділовою точкою (ця точка "плаває", з'являючись в різних місцях числа). Наприклад, 5.25, 5.257, 5.2573. Точність представлення числа залежить від реалізації SQL і устаткування. Наприклад, 32-бітовий комп'ютер дає велику точність, чим 16-бітовий;

- double precision - дійсне число подвійної точності з плаваючою розділовою точкою. Точність представлення числа залежить від реалізації SQL і устаткування. Застосовується для представлення наукових даних (наприклад, результатів вимірів) в широкому діапазоні значень, тобто як дуже малих (близьких до 0), так і дуже великих;

- float (x) - дійсне число з плаваючою розділовою точкою і мінімальною точністю x, що займає не більше 8 байтів. Якщо комп'ютер може підтримати вказану точність, використовуючи апаратну одинарну точність, то система використовуватиме арифметику одинарної точності. Якщо вказана точність вимагає арифметики з подвійною точністю, то система використовуватиме її. Цей тип слід застосовувати, якщо передбачається можливість перенесення бази даних на іншу апаратну платформу, що відрізняється розмірами регістрів. Приклад значення типу float : 5.318E-24 (тобто 5.318, помножене на 10 в мірі - 24). Таку ж форму представлення мають і числа типу real і double precision.

При створенні таблиць цілочисельні типи застосовуються для стовпців, що містять різного роду ідентифікатори, наприклад, номера (коди) клієнтів, товарів, замовлень і тому подібне. Зрозуміло, чи ес¬вміст стовпця має бути цілим числом (наприклад, кількість ящиків, пляшок, штук і тому подібне), то тип цього стовпця природно визначити як integer, smallint або bigint.

Допустимо, в таблиці клієнти є стовпець ID_клієнта, з¬що тримає унікальні ідентифікатори клієнтів. Якщо количе¬ство клієнтів не перевищує 32 000, то тип стовпця можна опре¬ділити як smallint. Якщо у вашій таблиці зберігатимуться зведення про сотні тисяч клієнтів, то тип стовпця ID_клієнта слід визначити як integer.

Якщо стовпець в проектованій таблиці повинен містити числа з дробовою частиною, то для нього можна задати який-небудь неце¬лочисленный тип. Якщо ви не упевнені, що застосувати: точні числові типи або приблизні, вибирайте точні (numeric, decimal). Вони вимагають менше ресурсів і дають точ¬ные результати. Якщо в стовпці передбачається зберігати дані з дуже широкого діапазону (і дуже малі, і дуже великі числа), то використовуйте приблизні типи даних (float, real).

1.4 Логічні дані

У SQL тип даних boolean (булевий) має три значення - true, false і unknown. Значення unknown (невідоме) було введене для позначення результату, що виходить при срав¬нении зі значенням null (невизначене). Якщо користувач ще не ввів в елемент таблиці ніякого значення, то цей "порожній" осередок містить значення null, що інтерпретується як невідоме або невизначене значення.

Результатом будь-якої операції порівняння true або false з null або з unknown завжди являється unknown.

У SQL -выражениях логічні значення полягають в кавыч¬ки, наприклад, 'TRUE' або 'true'

1.5 Дата і час

Тип data (дата) призначений для зберігання значень дати, елі¬менти яких розташовані в наступному порядку: рік (4 цифри), дефіс (-), місяць (2 цифри), дефіс, день (2 цифри).

Таким чином, значення дати займають 10 позицій, наприклад,

2005-10-02.

Дані цього типу можуть містити будь-яку дату з 0001 року по 9999 рік.

Для представлення часу передбачено два типи:

- time without time zone (час без часового поясу) предна¬значен для зберігання значень часу, елементи яких розташовані в наступному порядку : годинник, двокрапка, мину¬ти, двокрапка, секунди. Годинник і хвилини представляються дву¬мя цифрами, а секунди можуть бути представлені двома і бо¬леї цифрами (якщо вимагається дробова частина), наприклад 18:35:19.547. Довжина дробової частини секунд залежить від реали¬зации, але внутрішнє представлення часу повинне мати не менше 6 цифр. За умовчанням час цього типу перед¬ставляют без дробової частини секунд. Щоб вказати, що час має бути представлений з л цифрами після роздільник¬ний точки, досить використовувати такий синтаксис: time without time zone (л). Наприклад, щоб окрім секунд ука¬зывались ще і мілісекунди, слід визначити тип як time without time zone (3). Довжина даних рассматривае-мого типу без дробової частини дорівнює 8 символам, а з дробовою частиною - 9 плюс кількість цифр після розділової точки. Для завдання часу без вказівки часового поясу з використанням установок за умовчанням можна использо¬вать короткий синтаксис - time;

- time with time zone (час з часовим поясом) - такий же тип даних, як і time without time zone. Відмінність заклю¬чается лише в тому, що до значення часу додається ще і інформація про різницю між місцевим і всесвітнім вре¬менем. Всесвітній час (Universal Time Coordinated, UTC) - це час по Грінвічу, тобто час нульового меридіана, про¬того, що ходить через м. Грінвіч у Великобританії (Greenwich Mean Time, GMT). Значення різниці між локальним і всесвітнім часом знаходиться в діапазоні від -12:59 до 13:00. Довжина даних даного типу дорівнює довжині даних типу time without time zone плюс 6, оскільки до-полнительная інформація про різницю часів займає 6 позицій (дефіс, знак (+) або (-), 2 цифри для годинника, двоє¬точие, 2 цифри для хвилин).

Для одночасного представлення дати і часу служать сле¬дуючі два типи:

- timestamp without time zone (дата і час без часового поясу). Елементи даних цього типу мають такі ж характеристики, як і для даних типу date і time without time zone, за винятком одного: дані типу timestamp without time zone по умовчанню мають 6 цифр в дробовій частині секунд, а не 0, як в типі time without time zone. Для вказівки кількості цифр в дробовій частині використовується синтаксис timestamp without time zone (л). Якщо дробової частини немає, то дані займають 19 позицій: 10 позицій для дати, один

пропуск і 8 позицій для часу. Якщо визначена дробова частина, то довжина даних дорівнює 20 плюс количествоцифр в дробовій частині секунд;

- timestamp with time zone (дата і час з часовим поясом) - такий же тип даних, як і timestamp with time zone. Відмінність полягає в тому, що до значення часу додається ще і інформація про різницю між місцевим і всесвітнім часом. Додаткова інформація займає 6 позицій. Дані типу timestamp with time zone без дробової частини займають 25 позицій, з дробовою частиною - 26 плюс кількість цифр в дробовій частині секунд.

Щоб представити в SQL -выражении дату, час або дату-час, необхідно використовувати функцію cast об приведення до заданого типу. Допустимо, в таблиці продажу є стовпець дата типу data.

Щоб отримати відомості з цієї таблиці за період після 2005-09-30, слід виконати такий запит:

SELECT * FROM Продажі WHERE Дата > CAST ('2005-09-30' AS DATE);

Тут рядок, що містить дату, приводиться до типу data, і підлозі¬ченный результат бере участь в операції порівняння з даними стовпця Дата.

У мові SQL є три функції, які повертають теку¬щие дату і час :

- current_date - повертає поточну дату (тип date).

Наприклад, 2005-06-18;

- current_time (число) - повертає поточний час (тип time). Цілочисельний параметр число вказує точність представлення секунд.

Наприклад, при число = 2 секунди бу¬дутий представлені з точністю до сотих (дві цифри в дроб¬ний частині) : 12:39:45.27;

- current_timestamp (число) - повертає дату і час (тип TIMESTAMP), Наприклад 2005-06-18 12:39:45.27. Цілочисельний параметр число вказує точність представлення секунд.

1.6 Інтервалів

Інтервал є різницею між двома значеннями типу дата-час. SQL підтримує два типи інтервалів : рік-місяць і день-час. Інтервал типу рік-місяць - це кількість років і місяців між двома датами, а інтервал день-час - кількість днів, годинника, хвилин і секунд між двома моментами в межах одного місяця. Не можна змішувати обчислення, ис¬пользующие інтервал рік-день, з обчисленнями, в яких ис¬користується інтервал день-час.

Інтервал часу можна задати двома способами: у виді на¬чального і кінцевого моментів або у вигляді початкового моменту і тривалості, наприклад:

- (TIME '12: 25:30', TIME '14: 30:00') - Інтервал, заданий початковим і кінцевим моментами;

- (TIME '12: 45:00', INTERVAL '5' HOUR) - Інтервал, заданий початковим моментом і тривалістю в годиннику.

Щоб задати значення типу інтервал, використовується такий син¬таксис:

INTERVAL 'довжина' YEAR | MONTH | DAY | HOUR | MINUTE | SECOND

Тут довжина - довжина інтервалу, після якої вказується еди¬ница виміри (можливі значення вказані через вертикаль¬ную бісові), :

- YEAR - РІК;
- MONTH - місяць;
- DAY - день;
- HOUR - година;
- MINUTE - хвилина;
- SECOND - секунда.

Наприклад, для завдання інтервалу завдовжки 15 днів слідує ис-пользовать вираження interval '15' day.

2 Створення таблиці

Після створення загальної структури бази даних можна приступити до створення таблиць, якими є стосунки, що входять до складу проекту бази даних.

Базовий синтаксис оператора створення таблиці має наступний вигляд:

```
CREATE TABLE имя_таблицы
(имя_столбца тип_данных [NULL | NOT NULL ] [,...n])
```

Головне в команді створення таблиці – визначення імені таблиці і опис набору імен полів, які вказуються у відповідному порядку. Крім того, цією командою обмовляються типи даних і розміри полий таблиці.

Ключове слово NULL використовується для вказівки того, що в даному стовпці можуть міститися значення NULL. Значення NULL відрізняється від пропуску або нуля – до нього удаються, коли необхідно вказати, що дані недоступні, опущені або недопустимі. Якщо вказано ключове слово NOT NULL, то будуть відхилені будь-які спроби помістити значення NULL в даний стовпець. Якщо вказаний параметр NULL, приміщення значень NULL в стовпець дозволене. За умовчанням стандарт SQL передбачає наявність ключового слова NULL.

Ми використовували спрощену версію оператора CREATE TABLE стандарту SQL. Його повна версія наводиться при обговоренні питань забезпечення цілісності даних.

Приклад 3.1. Створити таблицю для зберігання даних про товари, що надходять у продаж в деякій торгівельній фірмі. Необхідно врахувати такі відомості, як назва і тип товару, його ціна, сорт і місто, де товар виробляється.

```
CREATE TABLE Товар
```

```
(Назва        VARCHAR(50) NOT NULL
Ціна  MONEY NOT NULL
Тип   VARCHAR(50) NOT NULL
Сорт  VARCHAR(50)
ГородТовара VARCHAR(50))
```

Приклад 3.2. Створити таблицю для збереження відомостей про постійних клієнтів з вказівкою назв міста і фірми, прізвища, імені і по батькові клієнта, номера його телефону.

```
CREATE TABLE Клієнт
(Фірма VARCHAR(50) NOT NULL
Прізвище VARCHAR(50) NOT NULL
Ім'я VARCHAR(50) NOT NULL
По батькові VARCHAR(50)
ГородКлієнта VARCHAR(50)
Телефон CHAR(10) NOT NULL)
```

3 Зміна таблиці

Структура існуючої таблиці може бути модифікована за допомогою команди ALTER TABLE, спрощений синтаксис якої представлен нижчий:

```
ALTER TABLE имя_таблицы
{[ADD [COLUMN] имя_столбца тип_данных [
NULL | NOT NULL ]]
| [DROP [COLUMN] имя_столбца]}
```

Команда дозволяє додавати і видаляти стовпці, змінювати їх визначення.

Одне з основних правил при додаванні стовпців в існуючу таблицю свідчить: коли в таблиці вже містяться дані, стовпець, що додається, не може бути визначений з атрибутом NOT NULL. Цей атрибут означає, що для кожного рядка даних відповідний стовпець повинен містити деяке значення, тому додавання стовпця з атрибутом NOT NULL наводить до появи протиріччя — вже існуючі рядки даних таблиці не матимуть в новому стовпці ненульових значень.

Проте, існує спосіб додавання обов'язкових полів в існуючу таблицю. Для цього необхідно:

− додати в таблицю новий стовпець, визначивши його з атрибутом NULL (тобто стовпець не зобов'язаний містити яких-небудь значень);

− ввести в новий стовпець які-небудь значення для кожного рядка даних таблиці;

− переконавшись, що новий стовпець містить ненульові значення для кожного рядка даних, змінити структуру таблиці, замінивши атрибут цього стовпця на NOT NULL.

При зміні визначень стовпців слід брати до уваги деякі загальноприйняті правила:

− розмір стовпця може бути збільшений до максимального значення, що допускається відповідним типом даних;

− розмір стовпця може бути зменшений лише в тому випадку, якщо найбільше значення, що міститься в нім, не перевершуватиме його нового розміру;

− кількість розрядів числового типа даних завжди може бути збільшена;

− кількість розрядів числового типа даних може бути зменшена лише в тому випадку, якщо кількість розрядів найбільшого значения у відповідному стовпці не перевершуватиме нового числа розрядів, визначеного для цього стовпця;

− кількість десяткових знаків числового типа даних може бути зменшена або збільшена;

− тип даних стовпця, як правило, може бути змінений. Деякі реалізації фактично можуть обмежити розробника

− у використанні деяких опцій команди ALTER TABLE. Наприклад, може виявитися недопустимим видалення стовпців з існуючої таблиці. Аби добитися цього, спочатку потрібно буде видалити саму таблицю і тільки потім заново її побудувати з потрібними стовпцями. Причому вже внесені до таблиці дані будуть втрачені.

Можливі труднощі, пов'язані з видаленням з таблиці стовпця, який залежить від деякого стовпця іншої таблиці. В такому разі спочатку доведеться видалити обмеження стовпця, а потім сам стовпець.

Приклад 3.3. Додати в таблицю Клієнт поле для номера розрахункового рахунку.

ALTER TABLE Клієнт ADD Рас_счет CHAR(2 0)

4 Видалення таблиці

З часом структура бази даних міняється: створюються нові таблиці, а колишні стають непотрібними і віддаляються з бази даних за допомогою оператора:

DROP TABLE имя_таблицы [RESTRICT | CASCADE]

Слід зазначити, що ця команда видалить не лише вказану таблицю, але і всі вхідні в неї рядки даних. Якщо потрібно видалити з таблиці лише дані, зберігши структуру таблиці, слід скористатися командою DELETE.

Оператор DROP TABLE додатково дозволяє вказувати, чи слід операцію видалення виконувати каскадний. Якщо в операторові вказано

ключове слово RESTRICT, то за наявності в базі даних хоч би одного об'єкту, існування якого залежить від таблиці, що видаляється, виконання оператора DROP TABLE буде скасовано. Якщо вказано ключове слово CASCADE, автоматично віддаляються і всі інші об'єкти бази даних, чиє існування залежить від таблиці, що видаляється, а також інші об'єкти, залежні від об'єктів, що видаляються. Загальний ефект від виконання оператора DROP TABLE з ключовим словом CASCADE може виявитися вельми відчутним, тому подібних операторів слід використовувати з максимальною обережністю.

Найчастіше оператор DROP TABLE використовується для виправлення помилок, допущених при створенні таблиці. Якщо таблиця була створена з некоректною структурою, можна скористатися оператором DROP TABLE для її видалення, після чого створити таблицю заново.

Лекція

Тема: ***«Модифікація таблиць баз даних»***

План лекції

1 Команда ALTER TABLE
2 Додавання стовпця
3 Модифікація стовпця.
4 Видалення стовпця
5 Додавання і видалення обмежень на рівні таблиці

Зміст лекції

1 Команда ALTER TABLE

Команда ALTER TABLE призначена для модифікації структури таблиці. З її допомогою модно змінювати властивості існуючих стовпців, видаляти або додавати в таблицю стовпці, а також управляти обмеженням цілісності як на рівні стовпця, так і на рівні таблиці, тобто виконувати наступні функції:

- додати в таблицю визначення нового стовпця;
- видалити стовпець з таблиці;
- змінити значення за умовчанням для якого-небудь стовпця;
- додати або видалити первинний ключ таблиці;
- додати або видалити зовнішній ключ таблиці;
- додати або видалити умову унікальності;
- додати або видалити умову на значення.

Розглянемо загальний синтаксис команди ALTER TABLE :

ALTER TABLE<ім'я_таблиці>

[ALTER COLUMN <ім'я_стовпця> [SET DEFAULT <вираження>] | [DROP DEFAULT]]

|[ADD <визначення_стовпця>]

|[DROP COLUMN <ім'я_стовпця> [CASCADE] | [RESRTICT]]

|[ADD [<визначення_первинного_ключа>] | [<визначення_зовнішнього_ключа>] | [<умова_унікальності>] | [<умова_на_значення>]]

|[DROP CONSTRAINT <ім'я_обмеження> [CASCATE] | [RESTRICT]]

Команда ALTER TABLE бере на себе усі дії з копіювання даних в тимчасову таблицю, видаленню старої таблиці і створенню замість неї нової таблиці з потрібною структурою і наступним переписуванням в неї даних.

Призначення багатьох параметрів і ключових слів команди ALTER TABLE аналогічно призначенню відповідних параметрів і ключових лов

команди CREATE TABLE (наприклад, синтаксис конструкції <визначення_стовпця> співпадає з синтаксисом аналогічної конструкції CREATE TABLE).

Основні режими використання команди ALTER TABLE наступні:

- додавання стовпця;
- видалення стовпця;
- модифікація стовпця;
- зміна, додавання і видалення обмежень (первинних і зовнішніх ключів, значень за умовчанням).

2 Додавання стовпця.

ALTER TABLE Студенти
ADD Рік_вступу INTEGER NOT NUL DEFAULT YEAR(GETDATE())

У структуру таблиці "Студент" буде доданий ще один стовпець зі значенням за умовчанням, рівним поточному року.

3 Модифікація стовпця.

Для модифікації існуючого стовпця служить ключове слово ALTER COLUMN. Зміна стовпця неможлива, якщо:

- стовпець бере участь в обмеженнях PRIMARY KEY або FOREGIG KEY;
- на стовпець накладені обмеження цілісності CHECK або UNIQUE (виключення - стовпці, що мають тип даних змінної довжини, тобто типи даних, що починаються на var);
- із стовпцем пов'язано значення за умовчанням.

Визначаючи для стовпця новий тип даних, слід пам'ятати про те, що старий тип даних повинен конвертуватися в новий.

ALTER TABLE Студенти
ALTER COLUMN Номер_групи CHAR(6) NOT NULL

4 Видалення стовпця

Не можна видаляти стовпці з обмеженням цілісності CHECK, FOREIGN KEY, UNIQUE або PRIMARY KEY, а також стовпці, для яких визначені значення за умовчанням.

ALTER TABLE Студенти
DROP COLUMN Рік_вступу

5 Додавання і видалення обмежень на рівні таблиці

Додавання зовнішніх ключів в таблицю "Учбовий план" (створення зв'язку з ім'ям FK_Дисципліна і зв'язки з ім'ям FK_Кадровий_склад_) :

ALTER TABLE Вчений план
ADD CONSTRAINT FK_Дисципліна
FOREIGN KEY (ID_Дисципліна)
REFERRNCES Дисципліни

ALTER TABLE Вчений план
ADD CONSTRAINT FK_Кадровий_склад
FOREIGN KEY (ID_Викладач)
REFERRNCES Кадровий_склад

За допомогою конструкції ADD CONSTRAINT створюється пойменоване обмеження. Необхідно відмітити, що видалення будь-якого обмеження на рівні таблиці відбувається тільки по його імені, тому обмеження має бути пойменоване (щоб його можна було видалити).

Додавання значення за умовчанням для стовпця Номер_групи :

ALTER TABLE Студент
ADD CONSTRAINT DEF_Номер_групи DEFAULT 1 FOR Номер_групи

На рівні таблиці буде створено обмеження цілісності з ім'ям DEF_Номер_групи.

Видалення обмежень :

ALTER TABLE Учбовий_план
DROP CONSTRAINT FK_Дисципліна

ALTER TABLE Студент
DROP CONSTRAINT DEF_Номер_групи.

Лекція

Тема: ***«Цілісність даних»***

План лекції

1 Поняття про обмеження цілісності. Класифікація обмежень.
2 Декларативні обмеження цілісності
- 2.1 Цілісність відношень
- 2.2 Цілісність атрибутів
- 2.3 Цілісність зв'язків між відношеннями
- 2.4 Цілісність зв'язків між атрибутами

Зміст лекції

Терміном цілісність даних позначають достовірність і точність інформації, що зберігається в базі. Цілісність досягається забезпеченням відповідності даних певним додатковим обмеженням, крім тих, які накладаються схемою бази на структуру даних та їхні типи.

Обмеження цілісності – це правила, які обмежують усі можливі стани бази даних, а також переходи з одного стану в інший. Таким чином, обмеження цілісності визначають множину «допустимих» станів і переходів між ними. База даних перебуває в цілісному стані, якщо вона відповідає всім визначеним для неї вимогам цілісності.

1 Поняття про обмеження цілісності. Класифікація обмежень.

Обмеження цілісності класифікують за:
- походженням;
- способом підтримки;
- терміном перевірки;
- областю дії;
- можливістю обмежувати переходи бази даних з одного стану в інший.

За походженням обмеження цілісності поділяють на:
- структурні;
- семантичні.

Структурні обмеження цілісності – це обмеження, які випливають із властивостей структури даних, що зберігаються в базі.

Семантичні обмеження цілісності – це обмеження, що накладаються предметною областю, яка моделюється.

За способом підтримки виділяють такі класи обмежень цілісності:

- декларативні;
- процедурні.

Декларативні обмеження цілісності — це обмеження, що фіксують умови, яким має відповідати база даних. Завдання СКБД - не допускати порушення цих умов. Зазвичай декларативні обмеження цілісності визначаються мовою опису структури даних. Прикладом такого обмеження є: «Фонд зарплати факультету має бути рівним сумі фондів зарплати всіх його кафедр». Декларативні обмеження цілісності специфікуються фразою CONSTRAINT в описі таблиці або її полів.

Процедурні обмеження цілісності — це описи дій, спрямованих на забезпечення цілісності. Прикладом такого обмеження є: «Під час зміни фонду зарплати будь-якої з кафедр автоматично змінити фонд зарплати факультету так, щоб він дорівнював сумі фондів зарплати усіх кафедр». Процедурні обмеження цілісності специфікуються тригерами.

За часом перевірки обмеження цілісності поділяються на такі, що:

- перевіряються негайно;
- мають відкладену перевірку.

Обмеження, що перевіряються негайно, перевіряються безпосередньо у момент виконання операції, яка може порушити цілісність. Наприклад, неповторюваність назви факультету перевіряється під час додавання нового рядка до таблиці, яка містить інформацію про факультети, або під час заміни імені факультету в існуючому рядку таблиці. Якщо обмеження порушується, то операція блокується.

Обмеження цілісності з відкладеною перевіркою використовуються у тому випадку, коли для підтримання бази даних у несуперечному стані потрібно виконати дві або більше операції. Прикладом обмеження цілісності, яке не може бути перевірено негайно, є декларативне обмеження щодо фондів заробітної плати факультету та його кафедр. Після додавання нової кафедри з заданим фондом фінансування база даних переходить у суперечний стан, оскільки фонд фінансування факультету стає не рівним сумі фондів фінансування його кафедр. Для того щоб повернути базу даних у несуперечний стан, потрібно виконати ще одну операцію — оновити фонд фінансування факультету. У такому випадку ці дві операції об’єднуються в одну транзакцію і обмеження цілісності перевіряється після її завершення.

За областю дії розрізняють обмеження, що стосуються:

- відношення;
- атрибута;
- зв’язків між відношеннями;
- зв’язків між атрибутами.

За можливістю обмежувати переходи бази даних з одного стану в інший обмеження цілісності поділяються на:

- статичні;
- динамічні.

Статичні обмеження цілісності накладають обмеження на можливі стани бази даних. Наприклад, декларативні обмеження цілісності, що описуються далі, є статичними.

Динамічні обмеження цілісності задають обмеження на можливі переходи бази даних з одного стану в інший.

2 Декларативні обмеження цілісності

Під час розгляду декларативних обмежень цілісності постають такі питання:

- на які об’єкти моделі даних поширюються обмеження цілісності;
- якими є ці обмеження;
- як ті чи інші обмеження цілісності специфікуються;
- які існують механізми підтримання цілісності.

У реляційних базах даних до об’єктів, на які поширюються обмеження цілісності, належать такі:

- відношення;
- атрибути;
- зв’язки між відношеннями;
- зв’язки між атрибутами.

У сучасних СКБД є й багато інших об’єктів бази даних, щодо яких специфікуються обмеження цілісності.

Розглянемо, як задаються обмеження цілісності для об’єктів реляційних СКБД Специфікація буде подана мовою PL/SQL, що застосовується в СКБД Oracle і в якій для специфікації структурних обмежень цілісності використовується фраза CONSTRAINT (складова частина речень CREATE TABLE і ALTER TABLE).

2.1 Цілісність відношень

У реляційній СКБД цілісність відношень визначається за допомогою первинного ключа, для якого мають виконуватися такі обмеження цілісності:

- атрибути первинного ключа не можуть містити NULL-значень;
- значення первинного ключа (як окремого атрибута або сукупності атрибутів) не можуть дублюватися в межах відношення.

Цілісність за первинним ключем специфікується так:

CONSTRAINT <ім’я обмеження> PRIMARY KEY (<перелік полів>)

де <перелік полів> — поля, що складають первинний ключ.

Ця специфікація вказує, які саме поля є ключовими.Для специфікації

можливих ключів використовується фраза UNIQUE.

Механізм підтримки цілісності відношень реалізує СКБД

CREATE TABLE КАФЕДРА
(#D NUMBER(2),
Назва VARCHAR2(9),
Завідувач VARCHAR2(10),
CONSTRAINT DepPK PRIMARY KEY (#D));

2.2 Цілісність атрибутів

У реляційній СКБД цілісність атрибутів може забезпечуватися:

- зазначенням типів даних та їх розмірів (обов'язкова властивість);
- визначенням, чи є обов'язковим значення атрибута (NULL/NOT NULL);
- визначенням, чи може значення атрибута дублюватися (UNIQUE):
- зміною значення атрибута після його введення;
- встановленням умов, яким мають відповідати значення атрибута.

Для похідних атрибутів цілісність має гарантуватися процедурою їхнього обчислення. Цілісність атрибутів специфікується фразами NULL/NOT NULL і UNIQUE у визначенні атрибута (поля).

Якщо унікальними мають бути значення не окремих полів, а сукупності полів, то це вказується так:

CONSTRAINT <ім'я обмеження> UNIQUE (<перелік полів>)

Цілісність атрибутів специфікується також зазначенням обмеження

CONSTRAINT <і м'я обмеження> CHECK (<умова>)

Умова визначається на атрибутах відношення. Фраза CHECK може також вказуватися в означенні атрибута.

CREATE TABLE ГРУПА (#GNUMBER(3),
#D NUMBER(3).
Курс NUMBER(l) CHECK (Course IN(1.2.3.4.5)).
Номер NUMBER(3) CHECK (Номер > 0).
Кількість NUMBER(2J CHECK (Кількість > 0).
#Куратор NUMBER(3) UNIQUE.
CONSTRAINT GrpPK PRIMARY KEY (#G).
CONSTRAINT GrpUNQ UNIQUE (Курс. Номер))

Зазначимо, що коли UNIQUE вказується для групи атрибутів, то йдеться про унікальність саме групи атрибутів, кожний з них окремо не обов'язково має бути унікальним. За допомогою фрази UNIQUE специфікуються також можливі ключі таблиці, тоді вона має використовуватися разом з обмеженням NOT NULL.

Механізм підтримки цілісності атрибутів реалізує СКБД.

2.3 Цілісність зв'язків між відношеннями

Цілісність зв'язків між відношеннями визначається зовнішніми ключами. Під час специфікації зовнішнього ключа вказується відповідний йому первинний ключ іншого відношення. Такий первинний ключ називається референційним.

Відношення, на яке здійснюється посилання за допомогою зовнішнього ключа, називається батьківським, а те, яке посилається, — дочірнім.

Якщо зовнішній ключ певного відношення може містити NULL-значення, це означає, що зв'язок даного відношення з іншим є факультативним. NOT NULL- специфікація стовпців зовнішнього ключа свідчить, що зв'язок є обов'язковим Означення зовнішнього ключа свідчить про наявність цілісності за посиланням: значення зовнішнього ключа має посилатися на значення первинного ключа іншого відношення. Тобто ситуація, коли зовнішній ключ має значення, відсутнє серед значень відповідного первинного ключа, розглядається як ситуація, що порушує цілісність за посиланням.

Цілісність за посиланням в Oracle специфікується так:

CONSTRAINT <назва обмеження> FOREIGN KEY («перелік полів 1>)

REFERENCES <ім'я таблиці>(<перелік полів 2>)

[ON DELETE {CASCADE | SET NULL}]

де «перелік полів 1> — поля, що становлять зовнішній ключ, <ім'я таблиці> — ім'я таблиці референційного ключа, <перелік полі в 2> — поля, з яких складається референційний ключ. В Oracle допускається, щоб «перелік полів 2> був не первинним ключем, а списком довільних полів, які мають специфікацію UNIQUE.

Механізм підтримки цілісності за посиланням реалізує СКБД. Розглянемо, як діє механізм під час видалення рядка з батьківської таблиці. Можливі три варіанти:

- рядок батьківської таблиці може бути видалений лише в тому випадку, якщо немає зовнішніх ключів, що посилаються на значення референційного ключа цього рядка:
- під час видалення рядка батьківської таблиці видаляються також усі рядки в інших таблицях, значення зовнішніх ключів яких посилаються на значення референційного ключа цього рядка (каскадне видалення);
- під час видалення рядка батьківської таблиці всі посилання на значення видаленого референційного ключа замінюються на NULL.

Відсутність фрази [ON DELETE {CASCADE | SET NULL}] вказує на перший варіант. Фраза ON DELETE CASCADE в специфікації референційної цілісності вказує на другий варіант, а ON DELETE SET NULL - на третій.

2.4 Цілісність зв'язків між атрибутами

Цілісність зв'язків між атрибутами визначається умовою, якій має відповідати сукупність атрибутів одного або кількох відношень. Такі умови специфікуються тригерами.

Наприклад, аби вказати, що кількість студентів на лекції не може перевищувати кількість місць в аудиторії, слід означити такий тригер:

```
CREATE TRIGGER Лекція Вставка Оновлення
BEFORE INSERT, UPDATE ON ЛЕКЦІЯ WHEN
(SELECT Місця
FROM АУДИТОРІЯ
WHERE АУДИТОРІЯ.#R =ЛЕКЦІЯ.#R) <
(SELECT Кількість
FROM ГРУПА
WHERE ГРУПА.#Є = ЛЕКЦІЯ.#Є)
BEGIN
ROLLBACK TRANSACTION
END
```

Лекція

Тема: ***«Засоби маніпулювання даними»***

План лекції

1 Додавання даних до таблиці
2 Видалення даних з таблиці
3 Оновлення даних

Зміст лекції

Мова SQL орієнтована на виконання операцій над групами записів, хоча в деяких випадках їх можна проводити і над окремим записом.

Запити дії є досить потужним засобом, оскільки дозволяють оперувати не лише окремими рядками, але і набором рядків. За допомогою запитів дії користувач може додати, видалити або відновити блоки даних. Існує три види запитів дії :

INSERT INTO - запит додавання;
DELETE - запит видалення;
UPDATE - запит оновлення.

1 Додавання даних до таблиці

Оператор INSERT застосовується для додавання записів в таблицю. Формат оператора :

<оператор__вставки>::=INSERT INTO <ім'я_таблиці> [(ім'я_стовпця [,..п])]

{ALUES (значення[,..п])|<SELECT оператор>}

Тут параметр ім'я таблиці є або ім'ям таблиці бази даних, або ім'ям оновлюваного представлення.

Перша форма оператора INSERT з параметром VALUES призначена для вставки єдиного рядка у вказану таблицю. Список стовпців вказує стовпці, яким будуть присвоєні значення в записах, що додаються. Список може бути опущений, тоді маються на увазі усі стовпці таблиці (окрім оголошених як лічильник), причому в певному порядку, встановленому при створенні таблиці. Якщо в операторові INSERT вказується конкретний список імен полів, то будь-які пропущені в нім стовпці мають бути оголошені при створенні таблиці як що допускають значення NULL, за винятком тих випадків, коли при описі стовпця використовувався параметр DEFAULT. Список значень повинен таким чином відповідати списку стовпців :

- кількість елементів в обох списках має бути однаковою;
- повинна існувати пряма відповідність між позицією одного і того ж елементу в обох списках, тому перший елемент списку значень повинен відноситися до першого стовпця в списку стовпців, другий - до другого стовпця і так далі
- типи цих елементів в списку значень мають бути сумісні з типами цих відповідних стовпців таблиці.

Приклад 1. Додати в таблицю TOVAR новий запис.

```
INSERT INFO Tovar (Nazvanie, Tip, Cena)
VALUES("Слов'янський", "шоколад", 12)
```

Якщо стовпці таблиці TOVAR вказані у повному складі і в тому порядку, в якому вони перераховані при створенні таблиці TOVAR, оператора можна спростити.

```
INSERT INTO Tovar VALUES ("Слов'янський", "шоколад", 12)
```

Друга форма оператора INSERT з параметром SELECT дозволяє скопіювати безліч рядків з однієї таблиці в іншу. Пропозиція SELECT може бути будь-яким допустимим оператором SELECT. Рядки, що вставляються у вказану таблицю, в точності повинні відповідати рядкам результуючої таблиці, створеної при виконанні вкладеного запиту. Усі обмеження, вказані вище для першої форми оператора SELECT, застосовні і в цьому випадку.

Оскільки оператора SELECT в загальному випадку повертає безліч записів, то оператор INSERT в цій формі призводить до додавання в таблицю аналогічного числа нових записів.

Приклад 2. Додати в підсумкову таблицю зведення про загальну суму щомісячних продажів кожного найменування товару

```
INSERT INTO
((Nazvanie, Mesyac, Stoimosti)
SELECT Tovar.Nazvanie, Month(Sdelka.Data)
AS Mesyac, Sum(Tovar.Cena*Sdelka.Kolichestvo)
AS Stoimosti
FROM Tovar
INNER JOIN Sdelka
ON Tovar.KodTovara= Sdelka. Kod Tovara
GROUP BY Tovar.Nazvanie, Month(Sdelka.Data)
```

2 Видалення даних з таблиці

Оператор DELETE призначений для видалення групи записів з таблиці.
Формат оператора :

<оператор_видалення> ::=DELETE
FROM <ім'я таблиці>[WHERE <условие_відбору>]

Тут параметр ім'я таблиці є або ім'ям таблиці бази даних, або ім'ям оновлюваного представлення.

Якщо пропозиція WHERE присутня, віддаляються записи з таблиці, що задовольняють умові відбору. Якщо опустити пропозицію WHERE, з таблиці будуть видалені усі записи, проте сама таблиця збережеться.

Приклад 3. Видалити усі торішні угоди.

```
DELETE
FROM Sdelka
WHERE Year(Sdelka.Data)  -  Year(GETDATE())  -  1
```

У наведеному прикладі умова відбору формується з урахуванням року (функція Year) від поточної дати (функція GETDATEQ).

3 Оновлення даних

Оператор UPDATE застосовується для зміни значень в групі записів або в одному записі вказаної таблиці.
Формат оператора :

<оператор зміни> ::=
UPDATE ім'я_таблиці SET ім'я_стовпця=
<вираження>[,..п]
[WHERE <умова відбору>]

Параметр ім'я таблиці - це або ім'я таблиці бази даних, або ім'я оновлюваного представлення. У пропозиції SET вказуються імена одного і більш за стовпці, дані в яких необхідно змінити. Пропозиція WHERE є необов'язковою. Якщо воно опущене, значення вказаних стовпців будуть змінені в усіх рядках таблиці. Якщо пропозиція WHERE присутня, то оновлені будуть тільки ті рядки, які задовольняють умові відбору. Вираження є новим значенням відповідного стовпця і має бути сумісне з ним за типом даних.

Приклад 4. Для товарів першого сорту встановити ціну в значення 140 і залишок - в значення 20 одиниць.

```
UPDATE Tovar SET Tovar.Cena - 140, Tovar.Octatoc=20
WHERE Tovar.Sort - " Perviy "
```

Приклад 5. Збільшити ціну товарів першого сорту на 25%.

```
UPDATE Tovar SET Tovar.Cena=Tovar.Cena* 1.25
WHERE Tovar.Sort - " Перший "
```

Лекція

Тема: ***«Виконання SQL-запитів»***

План лекції

1 Оператор SELECT
2 Оператор FROM
3 Оператор WHERE
    3.1 Порівняння
    3.2 Діапазон
    3.3 Приналежність безлічі
    3.4 Відповідність шаблону
    3.5 Значення NULL
4 Пропозиція ORDER BY

Зміст лекції

1 Оператор SELECT

Оператор SELECT – один з найбільш важливих і найпоширеніших операторів SQL. Він дозволяє виробляти вибірки даних з таблиць і перетворювати до потрібного вигляду отримані результати. Будучи дуже потужним, він здатний виконувати дії, еквівалентні операторам реляційної алгебри, причому в межах єдиної виконуваної команди. При його допомозі можна реалізувати складні і громіздкі условия відбору даних з різних таблиць.

Оператор SELECT – засіб, який повністю абстрагований від питань представлення даних, що допомагає сконцентрувати увагу на проблемах доступу до даних. Приклади його використання наочно демонструють один із засадничих принципів великих (промышленных) СУБД: засоби зберігання даних і доступу до них відокремлені від засобів представлення даних. Операції над даними виробляються в масштабі наборів даних, а не окремих записів. Оператор SELECT має наступний формат:

```
SELECT [ALL | DISTINCT ] {*|[имя_столбца
[AS новое_имя]] [,...n]
FROM имя_таблицы [[AS] псевдонім] [,...n]
[WHERE <условие_поиска>]
[GROUP BY имя_столбца [,...n]]
[HAVING <критерії вибору груп>]
[ORDER BY имя_столбца [,...n]]
```

Оператор SELECT визначає поля (стовпці), які входитимуть в результат виконання запиту. У списку вони розділяються комами і наводяться в такій черговості, в якій мають бути представлені в результате запиту. Якщо використовується ім'я поля, що містить пропуски або роздільники, його слід укласти в квадратні дужки. Символом * можна вибрати всі поля, а замість імені поля застосувати вираження з декількох імен.

Якщо обробляється ряд таблиць, то (за наявності однойменних полей в різних таблицях) в списку полів використовується повна специфікація поля, тобто Імя_табліци.Имя_поля.

2 Оператор FROM

Оператор FROM задає імена таблиць і переглядів, які містять поля, перераховані в операторові SELECT. Необов'язковий параметр псевдонім – це скорочення, що встановлюється для імені таблиці.

Обробка елементів оператора SELECT виконується в наступній послідовності:

FROM – визначаються імена використовуваних таблиць;

WHERE – виконується фільтрація рядків об'єкту відповідно до заданих умов;

GROUP BY – утворюються групи рядків, що мають одне і те ж значение у вказаному стовпці;

HAVING – фільтруються групи рядків об'єкту відповідно до указанным умови;

SELECT – встановлюється, які стовпці мають бути присутніми у вихідних даних;

ORDER BY – визначається впорядкованість результатів виконання операторів.

Порядок пропозицій і фраз в операторові SELECT не може бути змінений. Лише дві пропозиції SELECT і FROM є обов'язковими, всі інші можуть бути опущені. SELECT – закрита операція: результат запиту до таблиці є іншою таблицею. Існує безліч варіантів запису даного оператора, що ілюструється наведеними нижче прикладами.

Приклад 1. Скласти список відомостей про всіх клієнтів.

```
SELECT *
FROM Клієнт
```

Параметр WHERE визначає критерій відбору записів з вхідного набору. Але в таблиці можуть бути присутніми записи, що повторюються (дубликаты). Предикат ALL задає включення у вихідний набір всіх дублікатів,

відібраних по критерію WHERE. Немає необхідності вказувати ALL явно, оскільки це значення діє за умовчанням.

Приклад 2. Скласти список всіх фірм.

```
SELECT ALL Клієнт.Фірма
FROM Клієнт
```

Або (що еквівалентно)

```
SELECT Клієнт.Фірма
FROM Клієнт
```

Результат виконання запиту може містити значення, що дублюються, оскільки на відміну від операцій реляційної алгебри оператор SELECT не виключає значень, що повторюються, при виконанні вибірки даних.

Предикат DISTINCT слід застосовувати в тих випадках, коли потрібно відкинути блоки даних, що містять дублюючі записи у вибраних полях. Значення для кожного з приведених в інструкції SELECT полів мають бути унікальними, аби запис, що містить їх, зміг увійти до вихідного набору.

Причиною обмеження у вживанні DISTINCT є та обставина, що її використання може різко уповільнити виконання запитів.

Відкоректований приклад 2 виглядає таким чином:

```
SELECT DISTINCT Клієнт.Фірма
FROM Клієнт
```

3 Оператор WHERE

За допомогою WHERE-оператора користувач визначає, які блоки даних з приведених в списку FROM таблиць з'являться в результаті запиту. За ключовим словом WHERE слідує перелік умов пошуку, що визначають ті рядки, які мають бути вибрані при выполнении запиту. Існує п'ять основних типів умов пошуку (або предикатів):

Порівняння: порівнюються результати обчислення одного вираження з результатами обчислення іншого.

Діапазон: перевіряється, чи потрапляє результат обчислення вираження в заданий діапазон значень.

Приналежність безлічі: перевіряється, чи належить результат обчислень вираження заданій безлічі значень.

Відповідність шаблону: перевіряється, чи відповідає деяке строкове значення заданому шаблону.

Значення NULL: перевіряється, чи містить даний стовпець визначник NULL (невідоме значення).

3.1 Порівняння

У мові SQL можна використовувати наступних операторів порівняння:
= – рівність;
< – менше;
> – більше;
<= – менше або рівно;
>= – більше або рівно;
<> – не рівно.

Приклад 3. Показати всі операції відпустки товарів об'ємом більше 20.

```
SELECT *
FROM Операція
WHERE Колічество>20
```

Складніші предикати можуть бути побудовані за допомогою логічних операторів AND, OR або NOT, а також дужок, використовуваних для визначення порядку обчислення вираження.

Обчислення вираження в умовах виконується по наступних правилах:

- Вираження обчислюється зліва направо.
- Першими обчислюються підвирази в дужках.
- Оператори NOT виконуються до виконання операторів AND і OR.
- Оператори AND виконуються до виконання операторів OR.

Для усунення будь-якої можливої неоднозначності рекомендується використовувати дужки.

Приклад 4. Вивести список товарів, ціна яких більше або рівна 100 і менше або рівна 150.

```
SELECT Назва, Ціна
FROM Товар
WHERE Цена>=100 And Цена<=150
```

Приклад 5. Вивести список клієнтів з Москви або з Самари.

```
SELECT Прізвище, ГородКлієнта
FROM Клієнт
WHERE ГородКлієнта="Моськва" Or ГородКлієнта="Самара"
```

3.2 Діапазон

Оператор BETWEEN використовується для пошуку значення усередині деякого інтервалу, визначуваного своїми мінімальним і максимальним значеннями. При цьому вказані значення включаються в умову пошуку.

Приклад 6. Вивести список товарів, ціна яких лежить в діапазоні від 100 до 150 (запит еквівалентний прикладу 4.4).

```
SELECT Назва, Ціна
FROM Товар
WHERE Ціна Between 100 And 150
```

При використанні заперечення NOT BETWEEN потрібний, аби значення, що перевіряється, лежало поза кордонами заданого діапазону.

Приклад 7. Вивести список товарів, ціна яких не лежить в діапазоні від 100 до 150.

```
SELECT Товар.Назва, Товар.Ціна
FROM Товар
WHERE Товар.Ціна Not Between 100 And 150
```

Або (що еквівалентно)

```
SELECT Товар.Назва, Товар.Ціна
FROM Товар
WHERE (Товар.Цена<100) OR (Товар.Цена>150)
```

3.3 Приналежність безлічі

Оператор IN використовується для порівняння деякого значення із списком заданих значень, при цьому перевіряється, чи відповідає результат обчислення вираження одному із значень в наданому списку. За допомогою оператора IN може бути досягнутий той же результат, що і в разі вживання оператора OR, проте оператор IN виконується швидшим.

Приклад 8. Вивести список клієнтів з Москви або з Самари
(запит еквівалентний прикладу 4.5).

```
SELECT Прізвище, ГородКлієнта FROM Клієнт
WHERE ГородКлієнта in ("Москва", "Самара")
```

NOT IN використовується для відбору будь-яких значень, окрім тих, які вказані в представленому списку.

Приклад 9. Вивести список клієнтів, що проживають не в Москві і не в Самаре.

```
SELECT Прізвище, ГородКлієнта FROM Клієнт
WHERE ГородКлієнта Not in ("Моськва","самара")
```

3.4 Відповідність шаблону

За допомогою оператора LIKE можна виконувати порівняння вираження із заданим шаблоном, в якому допускається використання символов-заменителей:

– символ % – замість цього символу може бути підставлена будь-яка кількість довільних символів.

– символ _ замінює один символ рядка.

– [ ] – замість символу рядка буде підставлений один з можливих символів, вказаний в цих обмежувачах.

– [^] – замість відповідного символу рядка будуть підставлені всі символи, окрім вказаних в обмежувачах.

Приклад 10. Знайти клієнтів, в яких в номері телефону друга цифра, – 4.

```
SELECT Клієнт.Прізвище, Клієнт.Телефон
FROM Клієнт
WHERE Клієнт.Телефон Like "_4%"
```

Приклад 11. Знайти клієнтів, в яких в номері телефону друга цифра, – 2 або 4.

```
SELECT Клієнт.Прізвище, Клієнт.Телефон FROM Клієнт
WHERE Клієнт.Телефон Like "_[24]%"
```

Приклад 12. Знайти клієнтів, в яких в номері телефону друга цифра, – 2, 3 або 4.

```
SELECT Клієнт.Прізвище, Клієнт.Телефон
FROM Клієнт
WHERE Клієнт.Телефон Like "_[2-4]%"
```

Приклад 13. Знайти клієнтів, в яких в прізвищі зустрічається склад «-ро-».

```
SELECT Клієнт.Прізвище
FROM Клієнт
```

```
WHERE Клієнт.Прізвище Like "%ро%"
```

3.5 Значення NULL

Оператор IS NULL використовується для порівняння поточного значення із значенням NULL – спеціальним значенням, вказуючим на відсутність будь-якого значення. NULL – це не те ж саме, що знак пропуску (пропуск – допустимий символ) або нуль (0 – допустиме число). NULL відрізняється і від рядка нульової довжини (порожнього рядка).

Приклад 14. Знайти співробітників, в яких немає телефону (поле Телефон не містить жодного значення).

```
SELECT Прізвище, Телефон
FROM Клієнт
WHERE Телефон Is Null
```

IS NOT NULL використовується для перевірки присутності значення в полі.

Приклад 15. Знайти співробітників, в яких є телефон (поле Телефон містить яке-небудь значення).

```
SELECT Клієнт.Прізвище, Клієнт.Телефон
FROM Клієнт
WHERE Клієнт.Телефон Is Not Null
```

4 Пропозиція ORDER BY

У загальному випадку рядка в результуючій таблиці SQL-запроса ніяк не впорядковані. Проте їх можна необхідним чином відсортувати, для чого в оператора SELECT поміщається фраза ORDER BY. ORDER BY, яка сортує дані вихідного набору в заданій послідовності. Сортування може выполняьсяпо декільком полям, в цьому випадку вони перераховуються за ключовим словом ORDER BY через кому. Спосіб сортування задається ключовим словом, що вказується в рамках параметра ORDER BY слідом за назвою поля, по якому виконується сортування. За умовчанням реалізується сортування за збільшенням. Явно вона задається ключовим словом ASC. Для виконання сортування в зворотній послідовності необхідно після імені поля, по якому вона виконується, вказати ключове слово DESC. Фраза ORDER BY дозволяє упорядкувати вибрані записи в порядку зростання або убування значень будь-якого стовпця або комбінації стовпців, незалежно від того,

присутні ці стовпці в таблиці результату чи ні. Фраза ORDER BY завжди має бути останнім елементом в операторові SELECT.

Приклад 16. Вивести список клієнтів в алфавітному порядку.

```
SELECT Клієнт.Прізвище, Клієнт.Фірма FROM Клієнт
ORDER BY Клієнт.Прізвище
```

У фразі ORDER BY може бути вказане і більше одного елементу. Головний (перший) ключ сортування визначає загальну впорядкованість рядків результуючої таблиці. Якщо у всіх рядках результуючої таблиці значення головного ключа сортування є унікальними, немає необхідності використовувати додаткові ключі сортування. Проте, якщо значення головного ключа не унікальні, в результуючій таблиці буде присутньо декілька рядків з одним і тим же значенням старшого ключа сортування. В цьому випадку, можливо, доведеться упорядкувати рядки з одним і тим же значенням головного ключа по якому-небудь додатковому ключу сортування.

Приклад 17. Вивести список фірм і клієнтів. Назви фірм упорядкувати в алфавітному порядку, імена клієнтів в кожній фірмі відсортувати в зворотному порядку.

```
SELECT Клієнт.Фірма, Клієнт.Прізвище FROM Клієнт
ORDER BY Клієнт.Фірма, Клієнт.Прізвище DESC
```

Лекція

Тема: ***«Обчислення і підведення підсумків в SQL»***

План лекції

1 Побудова обчислюваних полів
2 Використання підсумкових функцій
3 Пропозиція GROUP BY
4 Пропозиція HAVING

Зміст лекції

1 Побудова обчислюваних полів

У загальному випадку для створення обчислюваного (похідного) поля в списку SELECT слід вказати деяке вираження мови SQL. У цих виразах застосовуються арифметичні операції складання, віднімання, множення і ділення, а також вбудовані функції мови SQL. Можна вказати ім'я будь-якого стовпця (поля) таблиці або запиту, але ис-пользовать ім'я стовпця тільки тієї таблиці або запиту, які вказані в списку пропозиції FROM відповідної інструкції. При побудові складних виразів можуть знадобитися дужки.

З'єднання - це процес, коли дві або більш за таблицю об'єднуються в одну. Здатність об'єднувати інформацію з декількох таблиць або запитів у вигляді одного логічного набору даних обумовлює широкі можливості SQL.

Операція внутрішнього з'єднання в мові SQL називається INNER JOIN і використовується, коли треба включити усі рядки з обох таблиць, що задовольняють умові об'єднання. Внутрішнє з'єднання має місце і тоді, коли в пропозиції WHERE порівнюються значення полів з різних таблиць.

В умовах об'єднання можуть брати участь поля, що відносяться до одного і тому ж типу даних і таких, що містять один і той же вид даних, але вони не обов'язково повинні мати однакові імена. Блоки даних з двох таблиць об'єднуються, як тільки у вказаних полях будуть знайдені співпадаючі значення.

Якщо в пропозиції FROM перераховано декілька таблиць і при цьому не вживається специфікація JOIN, а для вказівки відповідності полів з таблиць використовується умова в пропозиції WHERE, то деякі реляційні СУБД (наприклад, Access) оптимізують виконання запиту, інтерпретуючи його як з'єднання.

Стандарти SQL дозволяють явним чином задавати імена стовпців результуючої таблиці, для чого застосовується фраза AS.

Приклад 1. Розрахувати загальну вартість для кожної угоди. Цей запит використовує розрахунок результуючих стовпців на основі арифметичних виразів.

```
SELECT Товар.Назва, Товар.Ціна, Угода.Кількість, Товар.Ціна*Угода.Кількість AS Вартість
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
```

Приклад 2. Отримати список фірм з вказівкою прізвища і ініціалів клієнтів.

```
SELECT Фирма, Прізвище+"
"+"+Left(Ім'я, 1)+"" .+Left(По батькові, 1)+"".
AS ПІБ
FROM Клієнт
```

У запиті використана вбудована функція Left, що дозволяє вирізувати в текстовій змінній один символ ліворуч.

Приклад 3. Отримати список товарів з вказівкою року і місяця продажу.

```
SELECT Товар.Назва, Year(Угода.Дата)
AS Рік, Month(Угода.Дата) AS Місяць
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
```

У запиті використані вбудовані функції Year і Month для виділення року і місяця з дати.

2 Використання підсумкових функцій

За допомогою підсумкових (агрегатних) функцій у рамках SQL - запроса можна отримати ряд узагальнювальних статистичних відомостей про безліч відібраних значень вихідного набору

Користувачеві доступні наступні основні підсумкові функції:

Count (Вираження) - визначає кількість записів у вихідному наборі SQL -запроса;

Min/Max (Вираження) - визначають найменше і найбільше з безлічі значень в деякому полі запиту;

Avg (Вираження) - ця функція дозволяє розрахувати середнє значення безлічі значень, що зберігаються в певному полі відібраних запитом записів. Воно є арифметичним середнім значенням, тобто сумою значень, що ділиться на їх кількість.

Sum (Вираження) - обчислює суму безлічі значень, що містяться в певному полі відібраних запитом записів.

Найчастіше вираженням виступають імена стовпців. Вираження може обчислюватися і по значеннях декількох таблиць.

Усі ці функції оперують зі значеннями в єдиному стовпці таблиці або з арифметичним вираженням і повертають єдине значення. Функції COUNT, MIN і MAX застосовані як до числових, так і до нечислових полів, тоді як функції SUM і AVG можуть використовуватися тільки у разі числових полів, за винятком COUNT(*). При обчисленні результатів будь-яких функцій спочатку виключаються усі порожні значення, після чого необхідна операція застосовується тільки до конкретних значень стовпця, що залишилися. Варіант COUNT(*) - особливий випадок використання функції COUNT, його призначення полягає в підрахунку усіх рядків в результуючій таблиці, незалежно від того, містяться там порожні, такі, що дублюються або будь-які інші значення.

Якщо до застосування узагальнювальної функції необхідно виключити значення, що дублюються, слід перед ім'ям стовпця у визначенні функції помістити ключове слово DISTINCT. Воно не має сенсу для функцій MIN і MAX, проте його використання може вплинути на результати виконання функцій SUM і AVG, тому необхідно заздалегідь обдумати, чи повинне воно бути присутнім у кожному конкретному випадку. Крім того, ключове слово DISTINCT може бути вказане в будь-якому запиті не більше одного разу.

Підсумкові функції можуть використовуватися тільки в списку пропозиції SELECT і у складі пропозиції HAVING. У усіх інших випадках це неприпустимо. Якщо список в пропозиції SELECT містить підсумкові функції, а в тексті запиту відсутня фраза GROUP BY, що забезпечує об'єднання даних в групи, то жоден з елементів списку пропозиції SELECT не може включати яких-небудь посилань на поля, за винятком ситуації, коли поля виступають аргументами підсумкових функцій.

Приклад 4. Визначити першу за абеткою назву товару.

```
SELECT Min(Товар.Назва) AS Min_Назва
FROM Товар
```

Приклад 5. Визначити кількість угод.

```
SELECT Count(*) AS Кількість_угод
FROM Угода
```

Приклад 6. Визначити сумарну кількість проданого товару.

```
SELECT Sum(Угода.Кількість) AS Кількість_товару
FROM Угода
```

Приклад 7. Визначити середню ціну проданого товару.

```
SELECT Avg(Товар.Ціна) AS Avg_Ціна
FROM Товар INNER JOIN Угода
ON Товар.КодТовара =Угода.КодТовара;
```

Приклад 8. Підрахувати загальну вартість проданих товарів.

```
SELECT Sum(Товар.Ціна*Угода.Кількість) AS Вартість
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
```

3 Пропозиція GROUP BY

Часто в запитах вимагається формувати проміжні підсумки, що зазвичай відображується появою в запиті фрази "для кожного.". Для цієї мети в операторові SELECT використовується пропозиція GROUP BY. Запит, в якому присутній GROUP BY, називається групуючим запитом, оскільки в нім групуються ці, отримані в результаті виконання операції SELECT, після чого для кожної окремої групи створюється єдиний сумарний рядок. Стандарт SQL вимагає, щоб пропозиція SELECT і фраза GROUP BY були тісно пов'язані між собою. За наявності в операторові SELECT фрази GROUP BY кожен елемент списку в пропозиції SELECT повинен мати єдине значення для усієї групи. Більше того, пропозиція SELECT може включати тільки наступні типи елементів : імена полів, підсумкові функції, константи і вирази, що включають комбінації перерахованих вище елементів.

Усі імена полів, приведені в списку пропозиції SELECT, мають бути присутніми і у фразі GROUP BY - за винятком випадків, коли ім'я стовпця використовується в підсумковій функції. Зворотне правило не є справедливим - у фразі GROUP BY можуть бути імена стовпців, відсутні в списку пропозиції SELECT.

Якщо спільно з GROUP BY використовується пропозиція WHERE, то воно обробляється першим, а групуванню піддаються тільки ті рядки, які задовольняють умові пошуку.

Стандартом SQL визначено, що при проведенні групування усі відсутні значення рассматриваютсякак рівні. Якщо два рядки таблиці в одному і тому ж груповаому стовпці містять значення NULL і ідентичні значення в усіх інших непорожніх групованих стовпцях, вони поміщаються в одну і ту ж групу.

Приклад 9. Вичислити середній об'єм купівель, здійснених кожним покупцем.

```
SELECT Клієнт.Прізвище, Avg(Угода.Кількість)
AS Середня_кількість
```

```
FROM Клієнт INNER JOIN Угода
ON Клієнт.КодКлиента = Угода.КодКлиента
GROUP BY Клієнт.Прізвище
```

Фраза "кожним покупцем" знайшла своє віддзеркалення в SQL -запросе у вигляді пропозиції GROUP BY Клиент.Фамилия.

Приклад 10. Визначити, на яку суму був проданий товар кожного найменування.

```
SELECT Товар.Назва
Sum(Товар.Ціна*Угода.Кількість)
AS Вартість
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара GROUP BY Товар.Назва
```

Приклад 11. Підрахувати кількість угод, здійснених кожною фірмою.

```
SELECT Клієнт.Фирма, Count(Угода.КодСделки)
AS Кількість_угод FROM Клієнт INNER JOIN Угода
ON Клієнт.КодКлиента = Угода.КодКлиента GROUP BY Клієнт.Фирма
```

Приклад 12. Підрахувати загальну кількість купленого для кожної фірми товару і його вартість.

```
SELECT Клієнт.Фирма,
Sum(Угода.Кількість) AS Загальна_Кількість
Sum(Товар.Ціна*Угода.Кількість) AS Вартість
FROM Товар INNER JOIN
((Клієнт INNER JOIN Угода
ON Клієнт.КодКлиента = Угода.КодКлиента)
ON Товар.КодТовара = Угода.КодТовара
GROUP BY Клієнт.Фирма
```

Приклад 13. Визначити сумарну вартість кожного товару за кожен місяць.

```
SELECT Товар.Назва,
Month(Угода.Дата) AS Місяць
Sum(Товар.Ціна*Угода.Кількість) AS Вартість
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
GROUP BY Товар.Назва, Month(Угода.Дата)
```

Приклад 14. Визначити сумарну вартість кожного товару першого сорту за кожен місяць.

```
SELECT Товар.Назва,
Month(Угода.Дата) AS Місяць
Sum(Товар.Ціна*Угода.Кількість) AS Вартість
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
WHERE Товар.Сорт="Перший"
GROUP BY Товар.Назва, Month(Угода.Дата)
```

4 Пропозиція HAVING

За допомогою HAVING відбиваються усі попередньо згруповані за допомогою GROUP BY блоки даних, задовольняючі заданим в HAVING умовам. Це додаткова можливість "профільтрувати" вихідний набір.

Умови в HAVING відрізняються від умов в WHERE:

- HAVING виключає з результуючого набору даних групи з результатами агрегованих значень;
- WHERE виключає з розрахунку агрегатних значень по угрупуванню записи, що не задовольняють умові;
- в умові пошуку WHERE не можна задавати агрегатні функції.

Приклад 15. Визначити фірми, у яких загальна кількість угод перевищила 3.

```
SELECT Клієнт.Фирма,
Count(Угода.Кількість) AS Кількість_угод
FROM Клієнт INNER JOIN Угода
ON Клієнт.КодКлиента = Угода.КодКлиента
GROUP BY Клієнт.Фирма
HAVING Count(Угода.Кількість)>3
```

Приклад 16. Вивести список товарів, проданих на суму більше 10000 крб.

```
SELECT Товар.Назва
Sum(Товар.Ціна*Угода.Кількість) AS Вартість
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
GROUP BY Товар.Назва
HAVING Sum(Товар.Ціна*Угода.Кількість)>10000
```

Приклад 17. Вивести список товарів, проданих на суму більше 10000 без вказівки суми.

```
SELECT Товар.Назва
FROM Товар INNER JOIN Угода
ON Товар.КодТовара = Угода.КодТовара
GROUP BY Товар.Назва
HAVING Sum(Товар.Ціна*Угода.Кількість)>10000
```

Лекція

Тема: ***«Транзакції»***

План лекції

1 Як влаштована транзакція
2 Визначення параметрів транзакції

Зміст лекції

Транза́кция (англ. transaction) - група послідовних операцій з базою даних, яка є логічною одиницею роботи з даними. Транзакція може бути виконана або цілком і успішно, дотримуючи цілісність даних і незалежно від що паралельно йдуть інших транзакцій, або не виконана взагалі, і тоді вона не повинна справити ніякого враження. Транзакції обробляються транзакційними системами, в процесі роботи яких створюється історія транзакцій.

Розрізняють послідовні (звичайні), паралельні і розподілені транзакції. Розподілені транзакції мають на увазі використання більш ніж однієї транзакційної системи і вимагають набагато складнішої логіки (наприклад, two - phase commit - двофазний протокол фіксації транзакції). Також в деяких системах реалізовані автономні транзакції, або под-транзакции, які є автономною частиною батьківської транзакції.

1 Як влаштована транзакція

Транзакція складається з будь-яких операторів SQL, які можуть змінити базу даних. У SQL:2003 не передбачений спеціальний оператор початку транзакції. Проте деякі реалізації SQL вимагають наявності такого оператора. Він може виглядати поразному, наприклад, BEGIN WORK, BEGIN TRAN АБО BEG IN. У системах, сумісних з SQL:2003, якщо ніяка транзакція ще не почата, а на виконання вирушає оператор SQL, то автоматично створюється нова транзакція. Наприклад, оператори

CREATE TABLE (створити Таблицю), UPDATE (відновити) I SELECT (вибрати) виконуються тільки в транзакції.

Отже, оператора початку транзакції може і не бути. Проте для її завершення передбачено двох операторів:

COMMIT - завершити виконання. Виконання усіх операторів транзакції відбувається послідовно в єдиному блоці;

ROLLBACK - відкат. Відбувається відміна дії усіх операторів транзакції, і база даних повертається в початковий стан, в якому вона знаходилася до початку транзакції.

Оператор commit застосовується тоді, коли усе SQL -операторы транзакції імовірно виконані (сприйняті СУБД і не викликали помилок) і вимагається підтвердити зміни, внесені до бази даних. Якщо при виконанні оператора commit станеться системний збій або помилка, можна виконати відкат транзакції, а потім спробувати виконати її знову.

Після ключового слова commit допускається необов'язкове ключове слово work (робота) : commit work.

Оператор rollback застосовується тоді, коли необхідно відмінити зміни, внесені транзакцією, і відновити базу даних в колишньому стані. Якщо при виконанні оператора rollback станеться системний збій, то після перезавантаження оператора rollback можна виконати знову, і він повинен відновити базу даних. Таким чином, rollback є відмовостійким оператором.

Застосування може складатися з декількох транзакцій. Якщо не вказаний явний оператор початку транзакції (наприклад, begin work), т про перша транзакція починається на самому початку застосування. Остання транзакція закінчується у кінці застосування. Для вказівки кінця транзакції використовується оператор commit або rollback. Кожен такий оператор завершує одну транзакцію і починає наступну. Застосування з декількома транзакціями має наступний вид:

Початок застосування
SQL -операторы транзакції 1;
COMMIT або ROLLBACK;
SQL -операторы транзакції 2;
COMMIT або ROLLBACK;
…
SQL -операторы транзакції N;
Кінець застосування

2 Визначення параметрів транзакції

Для транзакції можна встановити параметри - режим і рівень ізоляції. З цією метою застосовується оператор:

SET TRANSACTION
Режим
ISOLATION LEVEL рівень ізоляції;

Режим може набувати наступні два значення:

- read write (читання-запис) - значення за умовчанням;
- READ ONLY (тільки читання).

Рівень ізоляції транзакції може набувати чотири значення:

- serializable (послідовне виконання) - значення за умовчанням;

- REPEATABLE READ (читання, що повторюється);
- READ COMMITED (підтверджене читання);
- READ UNCOMMITED (непідтверджене читання).

Транзакція має параметри за умовчанням: read write і serializable. Вони зазвичай підходять більшості користувачів і засновані на допущеннях, що база даних оновлюватиметься і що краще якомога більше себе забезпечити. Режим read write дозволяє вносити зміни до бази даних, а рівень ізоляції serializable є найбільш безпечним. Якщо ж необхідно задати інші значення, то слід використовувати оператора settransaction. При цьому він записується або на початку застосування, визначаючи першу транзакцію, або після оператора commit або rollback, визначаючи наступну транзакцію. Якщо ж потім commit або rollback не викликати оператора set transaction, то наступна транзакція матиме параметри, прийняті за умовчанням.

Наприклад:

```
SET TRANSACTION
READ ONLY,
ISOLATION LEVEL READ COMMITED;
```

Лекція

Тема: ***«Представлення»***

План лекції

1 Визначення представлення
2 Оновлення даних в представленнях

Зміст лекції

1 Визначення представлення

Представлення, або перегляди (VIEW) є тимчасовими, похідними (інакше - віртуальні) таблицями і є об'єктами бази даних, інформація в яких не зберігається постійно, як в базових таблицях, а формується динамічно при зверненні до них. Звичайні таблиці відносяться до базових, тобто що містить дані і що постійно знаходиться на пристрої зберігання інформації. Представлення не може існувати само по собі, а визначається тільки в термінах однієї або декількох таблиць. Застосування представлень дозволяє розробникові бази даних забезпечити кожному користувачеві або групі користувачів найбільш відповідні способи роботи з даними, що вирішує проблему простоти їх використання і безпеки. Вміст представлень вибирається з інших таблиць за допомогою виконання запиту, причому при зміні значень в таблицях дані в представленні автоматично міняються. Представлення - це фактично той же запит, який виконується всякий раз при участі в якій-небудь команді. Результат виконання цього запиту в кожен момент часу стає змістом представлення. У користувача створюється враження, що він працює із справжньою, реально існуючою таблицею.

У СУБД є дві можливості реалізації представлень. Якщо його визначення просте, то система формує кожен запис представлення в міру необхідності, поступово прочитуючи початкові дані з базових таблиць. У разі складного визначення СУБД доводиться спочатку виконати таку операцію, як матеріалізація представлення, тобто зберегти інформацію, з якої складається представлення, в тимчасовій таблиці. Потім система приступає до виконання призначеної для користувача команди і формування її результатів, після чого тимчасова таблиця віддаляється.

Представлення - це зумовлений запит, що зберігається в базі даних, який виглядає подібно до звичайної таблиці і не вимагає для свого зберігання дисковій пам'яті. Для зберігання представлення використовується тільки оперативна пам'ять.

На відміну від інших об'єктів бази даних представлення не займає дискової пам'яті за винятком пам'яті, необхідної для зберігання визначення самого представлення.

Створення і зміни представлень представлені наступною командою:

{ CREATE | ALTER}
VIEW ім'я_перегляду [(ім'я_стовпця [,..n])]
[WITH ENCRYPTION]
AS SELECT_операТор
[WITH CHECK OPTION]

Розглянемо призначення основних параметрів.

За умовчанням імена стовпців в представленні відповідають іменам стовпців в початкових таблицях. Явна вказівка імені стовпця вимагається для обчислюваних стовпців або при об'єднанні декількох таблиць, що мають стовпці з однаковими іменами. Імена стовпців перераховуються через кому, відповідно до порядку їх дотримання в представленні.

Параметр WITH ENCRYPTION наказує серверу шифрувати SQL -код запиту, що гарантує неможливість його несанкціонованого перегляду і використання. Якщо при визначенні представлення необхідно приховати імена початкових таблиць і стовпців, а також алгоритм об'єднання даних, необхідно застосувати цей аргумент.

Параметр WITH CHECK OPTION наказує серверу виконувати перевірку змін, вироблюваних через представлення, на відповідність критеріям, визначеним в операторові SELECT. Це означає, що не допускається виконання змін, які приведуть до зникнення рядка з представлення. Таке трапляється, якщо для представлення встановлений горизонтальний фільтр і зміна даних призводить до невідповідності рядка встановленим фільтрам. Використання аргументу WITH CHECK OPTION гарантує, що зроблені зміни будуть відображені в представленні. Якщо користувач намагається виконати зміни, що призводять до виключення рядка з представлення, при заданому аргументі WITH CHECK OPTION сервер видасть повідомлення про помилку і усі зміни будуть відхилені.

Приклад 1. Показати в представленні клієнтів з Москви.

Створення представлення :

CREATE VIEW viewl
AS
SELECT КодКлиента, Прізвище, ГородКлиента
FROM Клієнт
WHERE ГородКлиента="Москва"

Вибірка даних з представлення:

SELECT * FROM viewl

Звернення до представлення здійснюється за допомогою оператора SELECT як до звичайної таблиці.

Представлення можна використовувати в команді так само, як і будь-кого дру¬гую таблицю. До представлення можна будувати запит, модифікувати його (якщо воно відповідає певним вимогам), сполучати з іншими таблицями. Зміст представлення не фіксований і оновлюється кожного разу, коли на нього посилаються в команді. Представлення значно розширюють можливості управління даними. Зокрема, це прекрасний спосіб дозволити доступ до інформації в таблиці, приховавши частину даних.

Так, в прикладі 8.1 представлення просто обмежує доступ користувача до даних таблиці Клієнт, дозволяючи бачити тільки частину значень.

Виконаємо команду:

INSERT INTO viewl VALUES (12,'Петров', 'Самара')

Це допустима команда в представленні, і рядок буде доданий за допомогою представлення viewl в таблицю Клієнт. Проте, коли інформація буде додана, рядок зникне з представлення, оскільки назва міста відмінна від Москви. Іноді такий підхід може стати проблемою, оскільки дані вже знаходяться в таблиці, але користувач їх не бачить і не в змозі виконати їх видалення або модифікацію. Для виключення подібних моментів служить WITH CHECK OPTION в опреде¬лении представлення. Фраза розміщується у визначенні представлення, і усі команди модифікації піддаватимуться перевірці.

Приклад 2. Створення представлення з перевіркою команд модифікації.

ALTER VIEW viewl
SELECT КодКлиента, Прізвище, ГородКлиента
FROM Клиенет
WHERE ГородКлиента='Москва' WITH CHECK OPTION

Для такого представлення вищезгадана вставка значень буде відхилена системою.

Таким чином, представлення може змінюватися командами модифікації DML, але фактично модифікація впливає не на само представлення, а на базову таблицю.

Представлення віддаляється командою:

DROP VIEW ім'я_перегляду [,..n]

2 Оновлення даних в представленнях

Не усі представлення в SQL можуть бути модифіковані. Представлення, що модифікується, визначається наступними критеріями:

грунтується тільки на одній базовій таблиці;

містить первинний ключ цієї таблиці;

не містить DISTINCT у своєму визначенні;

не використовує GROUP BY або HAVING у своєму визначенні;

по можливості не застосовує у своєму визначенні підзапити;

не використовує константи або вираження значень серед вибраних полів виводу;

у перегляд має бути включений кожен стовпець таблиці, атрибут NOT NULL, що має;

оператор SELECT перегляду не використовує агрегуючі (підсумкові) функції, з'єднання таблиць, процедури, що зберігаються, і функції, визначені користувачем;

грунтується на поодинокому запиті, тому об'єднання UNION не дозволене.

Якщо перегляд задовольняє цим умовам, до нього можуть застосовуватися оператори INSERT, UPDATE, DELETE. Відмінності між представленнями, що модифікуються, і представленнями, призначеними тільки для читання, не випадкові. Цілі, для яких їх використовують, різні. З представленнями, що модифікуються, в основному обходяться точно так, як і з базовими таблицями. Фактично, користувачі не можуть навіть усвідомити, чи являється об'єкт, який вони запрошують, базовою таблицею або представленням, тобто передусім це засіб захисту для приховання конфіденційних або таких, що не відносяться до потреб цього користувача частин таблиці. Представлення в режимі "тільки для читання" дозволяють отримувати і форматувати дані раціональніше.

Вони створюють цілий арсенал складних запитів, які можна виконати і повторити знову, зберігаючи отриману інформацію. Результати цих запитів можуть потім використовуватися в інших запитах, що дозволить уникнути складних предикатів і понизити вірогідність помилкових дій.

Приклад 3. Представлення, що не модифікується, з даними з різних таблиць.

```
CREATE VIEW view2 AS
SELECT Клієнт.Прізвище, Клієнт.Фирма
Угода.Кількість
FROM Клієнт INNER JOIN Угода
ON Клієнт.КодКлиента = Угода.КодКлиента
```

Приклад 4. представлення, що Не модифікується, з угрупуванням і підсумковими функціями.

```
CREATE VIEW view3 (Тип, Общ_залишок) AS
SELECT Тип, Sum(Залишок) FROM Товар GROUP BY Тип
```

Зазвичай в представленнях використовуються імена, отримані безпосередньо з імен полів основної таблиці. Проте іноді необхідно дати стовпцям нові імена, наприклад, у разі підсумкових функцій або обчислюваних стовпців.

Приклад 5. Представлення, що модифікується, з обчисленнями.

```
CREATE VIEW view4(Код, Назва, Тип, Ціна, Податок) AS
SELECT КодТовара, Назва, Тип, Ціна, Ціна*0.05 FROM Товар
```

Лекція

Тема: ***«Безпека даних»***

План лекції

1 Безпека даних
2 Реєстрація користувачів
3 Керування правами доступу
    3.1 Кому надаються права доступу
    3.2 Умови надання прав доступу
    3.3 Об’єкти, на які поширюються права доступу
    3.4 Операції, щодо яких специфікуються права доступу
    3.5 Можливість передавання прав доступу іншим особам

Зміст лекції

1 Безпека даних

Дані в системах баз даних мають зберігатися з гарантуванням конфіденційності та безпеки. Інформація не може бути загубленою або викраденою. Під безпекою даних у базі розуміють захист даних від випадкового або спланованого доступу до них осіб, які не мають на це права, від несанкціонованого розкриття, зміни або видалення.

Безпека даних підтримується комплексом заходів і засобів:

- організаційно-методичні заходи - передбачають розроблення інструкцій та правил, які регламентують доступ до даних та їхнє використання, а також створення відповідних служб і підрозділів, які стежать за дотриманням цих правил;
- правові та юридичні заходи - передбачають юридичне закріплення прав і обов’язків щодо зберігання, використання й передавання в електронному вигляді даних, які підлягають захисту, на рівні державних законів та інших нормативних документів;
- технічні засоби захисту - це комплекс технічних засобів, які сприяють вирішенню проблеми захисту даних;
- програмні засоби захисту - це комплекс математичних, алгоритмічних і програмних засобів, шо сприяють вирішенню проблеми захисту даних.

Далі йтиметься лише про програмні засоби захисту.

Система захисту – це сукупність заходів, що вживаються в системі баз даних для гарантування необхідного рівня безпеки.

У сучасних СКБД підтримується один з двох найбільш розповсюджених методів забезпечення захисту даних: вибірковий чи обов'язковий.

Вибірковий метод захисту передбачає, що користувачі мають різні права (привілеї, повноваження) доступу до різних або одних тих самих об'єктів бази даних.

Обов'язковий метод захисту передбачає, що кожному об'єкту бази даних надається певний рівень секретності, а кожному користувачу — певний рівень допуску. Доступ до об'єкта даних є лише в тих користувачів, які мають відповідний для цих даних рівень допуску.

Зазначимо, що вибірковий метод гнучкіший, аніж обов'язковий. Безпека даних може гарантуватися такими механізмами.

Реєстрація користувачів. Будь-який користувач для отримання доступу до бази даних має бути зареєстрований у системі під певним ім'ям і певним паролем.

Керування правами доступу. Адміністратор може визначити, яким користувачам до яких даних дозволяється доступ і які саме операції над цими даними (вибирання, введення, зміну чи видалення) він може виконувати

Ідентифікація та підтвердження автентичності всіх користувачів або додатків, що отримують доступ до бази даних. Будь-який користувач або додаток, звертаючись до системи баз даних, мають вказати своє ім'я і пароль Ім'я ідентифікує користувача, а пароль підтверджує автентичність імені. Ці два кроки - ідентифікація та підтвердження автентичності − виконуються лише один раз під час з'єднання з базою даних і залишаються чинними до завершення сеансу роботи з базою даних конкретного користувача чи застосування

Автоматичне ведення журналів доступу до даних. У цих журналах протоколюються операції, виконані над даними користувачами, з метою подальшого аналізу на випадок отримання доступу до бази в обхід системи захисту.

Шифрування даних на зовнішніх носіях. Здійснюється криптографічними методами на випадок несанкціонованого копіювання даних із зовнішніх носіїв. Припускається, що адміністратор бази даних має всі необхідні повноваження на виконання функцій, пов'язаних із захистом даних.

Довірче й адміністративне керування доступом

Довірче керування доступом до даних - це такий тип керування, коли система захисту дає змогу звичайним користувачам не лише отримувати доступ до певних даних, але й передавати повноваження на доступ до них іншим користувачам без адміністративного втручання.

Адміністративне керування доступом до даних − це таке керування, за якого система захисту дає змогу передавати повноваження на доступ до даних лише спеціально авторизованому користувачу (адміністратору).

2 Реєстрація користувачів

Будь-який користувач для отримання доступу до бази даних має бути зареєстрований у системі під певним ім’ям і паролем. Реєстрація необхідна для того, щоб знати, з ким має справу система у кожний момент часу. Зазначимо, що під користувачем розуміється не лише фізична особа, але й будь-яке джерело, що в змозі звернутися до бази даних (прикладна програма, операційна система, інтернет-додаток тощо).

Спрощений вигляд конструкції, що застосовується для означення користувача, є таким:

CREATE USER <користувач> UDENTIFIED <пароль>

Тут <користувач> – ім’я користувача, а <пароль> - пароль.

Спочатку користувач не має жодних повноважень. Щоб користувач міг виконувати ті чи інші операції над базою даних, йому необхідно передати відповідні повноваження.

3 Керування правами доступу

3.1 Кому надаються права доступу

Права доступу надаються всім, хто звертається до бази даних. Це можуть бути користувачі, прикладні програми, операційні системи тощо. Кожен, хто звертається до бази даних, має насамперед вказати своє ім’я і пароль. Для СКБД не важливо, хто саме звертається до бази даних, головне, щоб усі, хто хоче отримати можливість працювати з нею, були заздалегідь зареєстровані в системі

У деяких випадках може знадобитися виділити користувачів системи, які мають однакові повноваження. Наприклад, усі співробітники відділу кадрів можуть мати одні й ті ж права, а співробітники планового відділу — інші, причому також однакові. Для реалізації такого розподілу прав вводиться поняття ролі.

Роль – це сукупність повноважень, які можуть передаватися користувачам або іншим ролям. Можна присвоїти повноваження ролям, а згодом приписувати ролі користувачам. Коли користувачу присвоєна роль, він має ті повноваження, які приписані ролі.

Роль має всі повноваження, які присвоєні їй явно, й усі повноваження, які передані їй іншими ролями.

Спрощений синтаксис означення ролі є таким:

CREATE ROLE <роль> IDENTIFIED <пароль>

Тут <роль> — ім’я ролі.

Спочатку роль є порожньою, тобто не має жодного повноваження.

3.2 Умови надання прав доступу

Іноді доцільно специфікувати додаткові умови, за дотримання яких користувачам надаються певні права доступу. Йдеться про умови, які не визначаються іншими складовими прав доступу (кому надається доступ, до яких даних, які операції дозволяються).

часові характеристики (наприклад, «права доступу діють лише між 16 і 17 годинами першого понеділка кожного місяця»):

локалізація комп'ютерів у локальній мережі (наприклад, «права доступу діють лише для комп'ютерів, установлених у плановому відділі»).

У більшості СКБД немає засобів явного опису додаткових умов, що обмежують права доступу. За необхідності такі умови можуть бути специфіковані у прикладних системах.

3.3 Об'єкти, на які поширюються права доступу

Зазвичай у контексті прав доступу розрізняють об'єкти двох класів: системні й об'єкти бази даних. До системних об'єктів належать: база даних, кластери, тригери, транзакції тощо. До об'єктів бази даних належать таблиці, віртуальні таблиці та процедури. Крім того, в таблицях і віртуальних таблицях можуть додатково вказуватися стовпці, щодо яких специфікуються права доступу.

У деяких випадках виникає необхідність специфікувати рядки певної таблиці, що стосуються особи, для якої визначаються права доступу. Наприклад:

- будь-який користувач може змінювати у відношенні СЛУЖБОВЕЦЬ значення полів лише того рядка, який стосується його самого (тобто він може змінювати лише свої особисті лані), за винятком величини його зарплати
- у відношенні СЛУЖБОВЕЦЬ змінювати зарплату може лише той користувач, який є начальником відділу даного службовця

3.4 Операції, щодо яких специфікуються права доступу

До операцій, стосовно яких специфікуються права доступу, належать стандартні операції з маніпулювання об'єктами бази даних, а саме:

означення, переозначення й видалення таблиці або віртуальної таблиці;

вибірка, додавання, видалення, оновлення рядків у таблиці або віртуальній таблиці:

виконання збережених процедур.

У кожній конкретній СКБД наведений список операцій над об'єктами бази даних може розширюватися.

3.5 Можливість передавання прав доступу іншим особам

Інколи користувачу надаються не лише права на маніпулювання тими чи іншими об’єктами бази даних, але й можливість передавати ці права іншим. Наприклад, користувачу передається право створити таблицю, вводити в неї дані, змінювати і видаляти їх, навіть видалити всю таблицю. Він стає повноправним власником таблиці і може з нею робити що завгодно. Тому не дивно, що йому надають дозвіл на передавання будь-яких прав на цю таблицю іншим користувачам.

Лекція

Тема: ***«Розподілені бази даних»***

План лекції

1 Основні означення
2 Властивості розподілених баз даних
3 Логічна архітектура розподілених баз даних
4 Архітектура програмно-технічних засобів розподілених СКБД
    4.1 Властивості архітектури
    4.2 Різновиди архітектури

Зміст лекції

1 Основні означення

Розподілена база даних (РБД) – це множина логічно взаємозалежних баз даних, розподілених у комп'ютерній мережі.

Розподілена система керування базами даних (РСКБД) – це програмне забезпечення, яке керує РБД і надає такі механізми доступу до них, що їх застосування дає користувачу можливість працювати з РБД як з однією цілісною базою даних.

Розподілена система баз даних (РСБД) – це РБД разом із РСКБД.

Не слід плутати РСБД з централізованою базою даних, що використовується в мережі (рисунок 1). У цьому випадку база даних розташована на одному з комп'ютерів, а всі інші мають доступ до неї через комунікаційну мережу. Не є розподіленою також база даних, що працює в середовищі багатопроцесорних комп'ютерів. У цьому випадку ми маємо справу з паралельною БД.

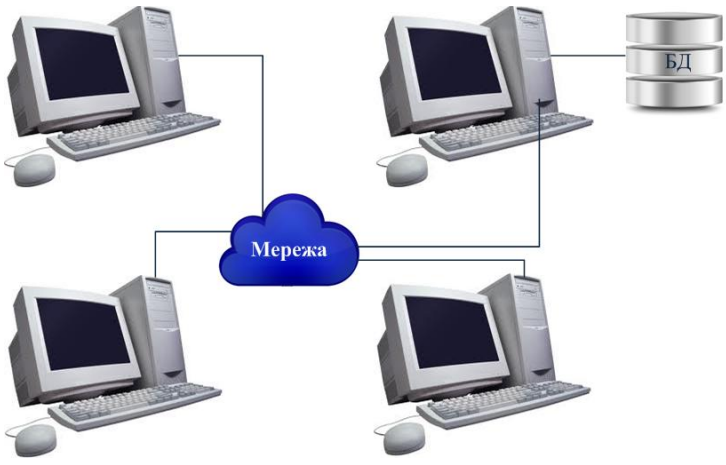


Рисунок 1 – Централізована база даних у комп'ютерній мережі

Архітектура розподіленої СКБД наведена на рисунок 2. Кожний з вузлів мережі містить свою базу даних, однак вони розглядаються як логічно єдина база, а не як сукупність розкиданих у мережі файлів. Усі дані є логічно взаємозалежними. Розподілена СКБД — це повноцінна СКБД, що виконує всі необхідні функції з керування даними.

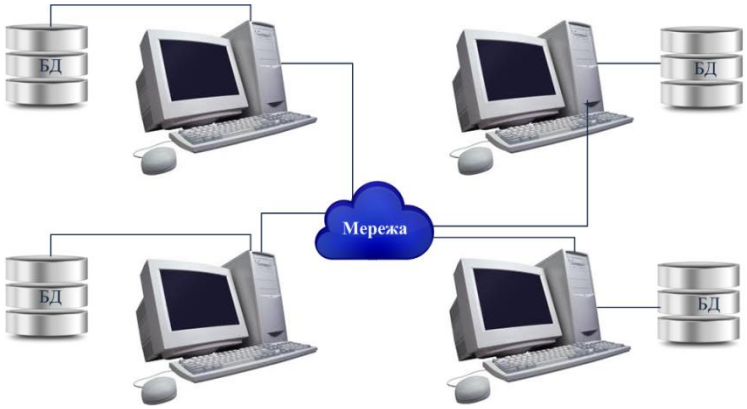


Рисунок 2 – Розподілена база даних

Залежно від типу програмного забезпечення розрізняють два типи РСКБД:

- однорідні;
- неоднорідні.

В однорідних РСКБД передбачається, як мінімум, що на всіх вузлах використовуються однотипні СКБД (наприклад, реляційні), які мають схожі функціональні можливості. Як максимум, припускається, що на всіх вузлах використовуються однакові технічні засоби, тобто однакові типи комп’ютерів і програмного забезпечення. Це стосується операційних систем, програмного забезпечення СКБД та моделей даних, що підтримуються.

У неоднорідних РСКБД вузли базуються на різних програмно-технічних платформах, які можуть містити різні типи СКБД. Окрім того, такі СКБД можуть підтримувати різні моделі даних. У цьому випадку ускладнюється вирішення проблеми їхньої взаємодії.

Однією з важливих проблем РСКБД є досягнення логічної незалежності даних від місця зберігання, тобто прозорість доступу до даних. Це означає, що користувач повинен мати можливість сприймати всі необхідні йому дані як єдине ціле, не зважаючи на те, в який спосіб вони розподілені у мережі.

2 Властивості розподілених баз даних

Вперше завдання про дослідження основ і принципів створення і функціонування розподілених інформаційних систем було поставлене відомим фахівцем в області баз даних Д. Дейтом.

Локальна автономія (local autonomy). Ця якість означає, що управління

даними на кожному з вузлів розподіленої системи виконується локально. База даних, розташована на одному з вузлів, є невід'ємним компонентом розподіленої системи. Будучи фрагментом загального простору даних, вона в той же час функціонує як повноцінна локальна база даних, а управління нею здійснюється локально, незалежно від інших вузлів системи.

Незалежність вузлів (no reliance on central site). Всі вузли рівноправні і незалежні, а розташовані на них БД є рівноправними постачальниками даних в загальний простір даних. База даних на кожному з вузлів самодостатньою – вона включає повний власний словник даних і повністю захищена від несанкціонованого доступу.

Безперервні операції (continuous operation). Це можливість безперервного доступу до даних в рамках розподіленої БД незалежно від їх розташування і незалежно від операцій, що виконуються на локальних вузлах.

Прозорість розташування (location independence). Користувач, що звертається до БД, нічого не повинен знати про реальне, фізичне розміщення даних у вузлах інформаційної системи.

Прозора фрагментація (fragmentation independence). Можливість розподіленого (тобто на різних вузлах) розміщення даних, логічно поєднаних в єдине ціле. Існує фрагментація двох типів: горизонтальна і вертикальна. Перша означає, що рядки таблиці зберігаються на різних вузлах. Друга означає розподіл стовпців логічної таблиці по декількох вузлах.

Прозоре тиражування (replication independence). Тиражування даних - це асинхронний процес перенесення змін об'єктів вихідної бази даних в бази, розташовані на інших вузлах розподіленої системи

Обробка розподілених запитів (distributed query processing). Можливість виконання операцій вибірки даних з розподіленої БД, за допомогою запитів, сформульованих на мові SQL

Обробка розподілених транзакцій (distributed transaction processing). Можливість виконання операцій оновлення розподіленої бази даних, які не порушують цілісність і узгодженість даних. Ця мета досягається вживанням двофазного протоколу фіксації транзакцій.

Незалежність від устаткування (hardware independence). Ця властивість означає, що як вузли розподіленої системи можуть виступати ПК будь-яких моделей і виробників

Незалежність від операційних систем (operationg system independence). Ця якість витікає з попереднього і означає різноманіття операційних систем, керуючих вузлами розподіленої системи

Прозорість мережі (network independence). Доступ до будь-яких баз даних відбувається по мережі. Спектр підтримуваних конкретною СУБД мережевих протоколів не має бути обмеженням системи, заснованої на розподіленій БД

Незалежність від баз даних (database independence). Ця якість означає, що в розподіленій системі можуть працювати СУБД різних виробників, і можливі операції пошуку і оновлення в базах даних різних моделей і

форматів.

3 Логічна архітектура розподілених баз даних

Логічна архітектура розподілених баз даних − це архітектура логічно взаємозалежних даних. Графічне зображення логічної архітектури розподілених баз даних наведене на рисунок 3. Особливість архітектури РБД полягає в тому, що виникає ще один рівень, глобальний концептуальний, завданням якого (як і в моделі ANSI/SPARC) є зображення концептуальної моделі предметної області в цілому. На цьому рівні описується характер розподілу даних.

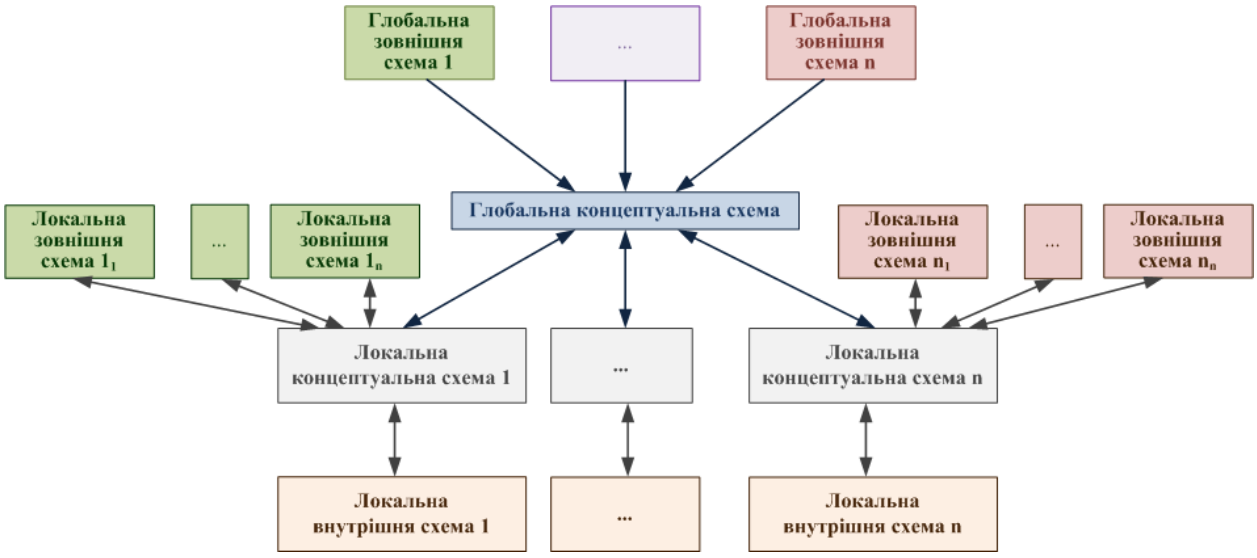


Рисунок 3 − Логічна архітектура розподіленої СКБД

На локальному концептуальному рівні здійснюється локальний опис ПО. Тобто схема цього рівня містить опис тільки тієї частини предметної області, що є специфічною для конкретного вузла розподіленої ПО; дані інших вузлів розподіленої ПО в схемі не вказуються. Відображення «глобальний-концептуальний/локальний-концептуальний» дають змогу зобразити глобальну концептуальну схему у вигляді сукупності локальних концептуальних схем, і навпаки.

Глобальна зовнішня схема зображує дані, необхідні користувачам і додаткам, у бажаному для них вигляді, незалежно від способу розподілу даних за вузлами. Це завдання вирішується завдяки тому, що глобальні зовнішні схеми базуються на глобальній концептуальній схемі.

Локальні зовнішні схеми базуються на локальних концептуальних схемах, відтак вони можуть посилатися лише на ті дані, що розташовані на відповідному вузлі розподіленої ПО.

Локальні внутрішні схеми − це схеми зберігання даних на конкретних вузлах розподіленої ПО. Вони пов’язані з відповідними локальними концептуальними схемами.

4 Архітектура програмно-технічних засобів розподілених СКБД

4.1 Властивості архітектури

Архітектура програмно-технічного комплексу розподілених СКБД має три головні характеристики:

- розподіленість;
- неоднорідність;
- автономність.

Розподіленість. Спосіб розподілу компонентів системи баз даних за комп'ютерами мережі визначається тим, чи є в мережі єдиний комп'ютер з повноваженнями розподіленої СКБД, або ж їх кілька. Якщо таких комп'ютерів кілька, то чи є між ними взаємодія тощо.

Неоднорідність. Важливою характеристикою є міра однорідності програмно-технічних засобів СКБД. Ідеться про технічне забезпечення (типи комп'ютерів), комунікаційні засоби зв'язку, операційні системи і типи баз даних (моделі даних, мови запитів, алгоритми керування транзакціями тощо).

Автономність. Характеризує, наскільки самостійно компоненти СКБД можуть виконувати свої функції. До питань автономності належать:

- автономність проектних рішень (наскільки самостійно компоненти СКБД можуть розв'язувати задачі, які були спроектовані для них); автономність вирішенням комунікаційних проблем (наскільки самостійно компоненти СКБД можуть встановлювати зв'язки з іншими компонентами);
- автономність обчислень (наскільки самостійно компоненти СКБД можуть приймати рішення щодо виконання локальних операцій).

4.2 Різновиди архітектури

Основними різновидами архітектури програмно-технічних засобів розподіленої СКБД є:

- клієнт-серверна архітектура;
- архітектура з багатьма незалежними серверами;
- архітектура із взаємодіючими серверами;
- архітектура однорангової мережі.

Клієнт-серверна архітектура. Така архітектура передбачає наявність єдиного комп'ютера-сервера і багатьох комп'ютерів-клієнтів, що взаємодіють між собою через канали зв'язку. На сервері розташована СКБД та інтегрована (централізована) база даних. Ніякого розподілу баз даних за вузлами мережі немає. На клієнтських комп'ютерах виконуються додатки, які працюють із серверною базою даних, а також розміщене програмне забезпечення для зв'язку з віддаленою СКБД. Як клієнти, так і сервер оснащені комунікаційним програмним забезпеченням.

Архітектура з багатьма незалежними серверами. Ця архітектура

передбачає існування багатьох серверів, що мають доступ до своїх локальних баз даних. У такому випадку на комп'ютерах клієнтів має зберігатись інформація стосовно того, які дані на яких серверах розташовані, а також має бути розміщене програмне забезпечення, що дає змогу декомпозувати запити для їхнього виконання на різних серверах і потім об'єднати результати.

Архітектура із взаємодіючими серверами. У цьому випадку передбачається, що кожний із серверів містить повну інформацію стосовно того, які дані на яких серверах зберігаються, а також здатний обробляти розподілені запити. У разі потреби один сервер може звернутися до іншого для одержання необхідних даних. Кожен клієнт має свій сервер, до якого звертається для виконання операцій над розподіленою базою даних.

Архітектура однорангової мережі. Усі комп'ютери мережі є серверами. На кожному комп'ютері розміщено розподілену СКБД і базу даних, з кожного комп'ютера можна надіслати до іншого запит на отримання необхідних даних.

Лекція

Тема: ***«Паралельні бази даних»***

План лекції

1 Основні поняття паралельної обробки даних
2 Архітектура багатопроцесорних систем
    2.1 Архітектура зі спільною па м 'я ттю
    2.2 Архітектура без спільної пам’яті
    2.3 Архітектура зі спільною зовнішньою пам'яттю

Зміст лекції

1 Основні поняття паралельної обробки даних

Під час паралельн ої обробки даних вели ка задача поділяється на кілька менших, які одночасно виконую ться на кілько х вузлах комп’ютерної системи. Відтак вихідна задача виріш ується зн ачно швидше.

Ефективність реалізації паралельн ої обробки дан их залежить:

- від способу поділу задачі на підзадачі, які можна виконувати водночас (паралельно);
- від можливості ви явлен ня підзадач. які мають виконуватися послідовно, та підтримки послідовної схеми їхн ього виконання.

Система паралельної обробки д ан и х має такі властивості:

- кожний вузол системи може виконувати задачі одночасно з іншими вузлами;
- система може синхронізувати вико нання задач (підзадач),
- вузли системи спільн о використовую ть ресурси, тобто дані, а також ди ски та інші пристрої.

До ключових аспектів паралельн ої обробки даних належать наступні.

- прискорення обчислень та масштабованість – це два основні критерії ефективності паралельн ої обробки даних. Обчислювальні задачі в паралельний спосіб мають виконуватись швидше, ніж у послідовний. Отже, можна казати про прискорення обчислень завдяки паралелізму. Терміном «масштабованість» позначають наявн ість легкого способу підвищ увати потужність системи так, щоб вона була здатна обробити зростаю чі обсяги даних. Рівень масіптабовапості можна виміряти за допомогою коеф іцієнту масштабу вашім,що показує, у скільки разів прискорення обчислень перевищує збільшення апаратних витрат;

- синхронізація – не координація обробки задач, що виконуються одночасно, вона необхідна для отримання коректного результату обчислень. Запорукою у спіш ної паралельн ої обробки є таки й поділ на підзадачі, коли потрібна незначна синхропізація. Внаслідок зменш ен н я син хронізації зростає швидкодія та збільш ується коефіцієнт масш табування. Потреба в син хронізації залежить від можливості спільного використання ресурсів, обсягів і кількості задач;
- блокування – це механізм керуван ня доступом до ресурсів, за допомогою якого досягається синхронізація виконання задач;
- передавання повідомлень це один із ключових аспектів паралельної обробки даних, оскільки вона виконується ефективно лише в тому випадку, коливузли системи з'єднані високоп родукти вн ими каналами зв'язку. Технологія паралельних баз даних надає н изку переваг пр икладним системам, які їх використовують;
- високапродуктивність. Що більше вузлів, то швидше виконуються задачі;
- підвищена працездатність. За відмови одного з вузлів його функції може взяти на себе інший, і система залишиться працездатною. Отже, порівняно з однопроцесорною системою, дан і будуть більш доступними;
- підвищена гнучкість. Можна настрою вати конфігурацію технічних засобів відповідно до потреб прикладних задач, нарощ увати обчислю вальні потужності системи або, навпаки, зменшувати їх;
- збільшення кількості користувачів. Технологія паралельних баз даних надає можливість подолати наслідки обмеженості ресурсів пам’яті, що дозволяє суттєво збільшити кількість користувачів, які обслуговуються водночас.

Дані можуть бути розподілені за різними дисками з метою розпаралелювання операцій введення/виведення, що виконуються різними процесорами. Окремі операції (наприклад, сортування чи з'єднання) можна виконувати в паралельн ому режимі. Високорівневість мови, на якій формулюються запити (напр иклад, мови SQL), також сприяє розпаралєлюванню її виразів.

2 Архітектура багатопроцесорних систем

Існує кілька типів архітектури багатопроцесорних систем. Ця архітектура визначає спосіб взаємодії між процесором, оперативною

пам'яттю і дисковим простором. Н айбільш загальними її р ізно видами с архітектури:

- зі спільною пам'яттю;
- зі спільною зовніш ньою пам’яттю;
- без спільної пам’яті.

2.1 Архітектура зі спільною па м 'я ттю

Багато процесорів мають спільну оперативну й дискову пам’ять, до якої вони здійснюють спільний доступ.

Основною перевагою архітектури зі спільною пам’яттю є те, що повідомлення та дані не передаються комунікаційними каналами. Процесори обмінюються даними через спільну пам'ять. Це також полегшує синхронізацію процесів. Ще однією перевагою такої архітектури є те, що нерівномірне розподілення навантаження процесорів можна усунути з мінімальними додатковими витратами. Недоліки таких систем пов’язані з проблемами спільного використання ресурсів. Лише за незначної кількості процесорів кожен з них може отримати доступ до бажаної ділянки пам'яті.

2.2 Архітектура без спільної пам’яті

Ця архітектура протилежна попередній. Кожен процесор мас свою власну оперативну й дискову пам 'ять

В архітектурі без спільної пам'яті процесори передають один одному повідомлення та дані через комунікаційну мережу. Основна перевага подібної архітектури полягає в можливості масштабування до сотень і тисяч процесорів без втручання кожного з них у роботу всіх інших.

Недоліком таких систем є нерівномірне розподілення навантаження процесорів та їхня залежність від пропускної здатності комунікаційної мережі. Якщо дані розподілені асиметрично, то паралельне виконання операцій сортування та з'єднання с малоефективним, оскільки залежить віл способу координації обміну даними між вузлами. Ніс один недолік подібної архітектури пов'язаний з ситуацією, коли виходить з ладу один із процесорів, адже дані, якими цей процесор володіє, стають недоступними.

Архітектура бел спільної пам'яті може бути побудована з використанням комп'ютерів з'єднаних комунікаційною мережею. У цьому випадку така архітектура є найдешевшим варіантом реалізації паралельних систем баз даних. У більшості досліджень у галузі паралельних баз даних за основу береться саме такий варіант архітектури

2.3 Архітектура зі спільною зовнішньою пам'яттю

У цій архітектурі кожний процесор має власну оперативну нам'ять, але всі вони працюють зі спільним дисковим простором.

У деяких випадках за допомогою архітектури зі спільною зовнішньою пам’яттю можна вирішувати проблеми, пов'язан із недоліками архітектури інших типів.

Наприклад, у цій архітектурі можна вирішити проблему відмови вузла, що виникає в системах без спільної пам'яті, оскільки диски, на яких зберігаються дані, доступні всім процесорам

Лекція

Тема: ***«Основи XML»***

План лекції

1 Базові поняття XML

Зміст лекції

Мова XML (eXtensible Markup Language – розширювана мова розмітки) була розроблена і підтримується консорціумом W3C. Вона розроблялася як мова розмітки документів, а не як мова опису баз даних. Розширюваність є головною відмінністю XML від іншої популярної мови розмітки - HTML Спочатку фахівці вважали, що ця мова замінить HTML як мову публікації веб-документів, проте наявність у мові засобів визначення нових тегів, а також можливість створювати вкладені структури тегів дозволила використовувати XML для зображення даних складної структури, а не тільки документів. У зв'язку з цим мова стала інтенсивно використовуватися в застосуваннях, що здійснюють обмін даними, а не просто як замінни к HTML Завдяки відкритості і розширюваності XML стала основою для нового покоління форматів зображення даних в Інтернеті.

Формати попередніх поколінь базувалися на «плоских» текстах, що складалися з рядків. XML дає змогу описувати структурован і дані, структура яких може бути довільною або фіксу вати ся за допомогою схем XML-даних. Окрім того, розроблені мови з організації пош уку в документах, записаних мовою XML.

1 Базові поняття XML

Елементи

Будь-який XML-документ складається з елементів. Елемент починається тегом

<ім`я тега> і закінчується тегом </ім`я тега>, що називаються початковим тегом і кінцевим тегом відповідно. Між цими двома тегами розташовується вміст елементу. В елементі можуть міститися інші елементи. Вкладеність елементів має бути коректною, тобто внутрішній елемент повшіен закінчуватися раніше зовнішнього.

Наприклад, із двох вказаних нижче записів перш ий є коректним, а другий - ні.

<факультет>...<кафедрз>...</кафедра>...</факультет>

<факультет>.. <кафедра> «/факультет».. </кафедра»

Окрім інших елементів, елемент може містити текст, наприклад.

«кафедра>

Кафедру було створено в жовтні 2000 року Першим завідувачем кафедри був професор П.І Іванов

<назва_кафедри>комл'ютерних наук</назва_кафедри>

<завідувач> П. Петров</завідувач>

<фонд>2000.00</фонд>

</кафедра >

Атрибути

Елементи можуть мати атрибути,зн ачення яки х вказую ться всередині початкового тега елементу згідно з таким форматом: ік'я_атрибута-знйчекня. Елемсит може мати кілька атрибутів, але кожне ім ’я атрибута мас бути унікальним у межах елементу. Наведемо приклад означення атрибутів:

<факультет корпус="6/1" назва="інформагики">

Зазначимо, що атрибути і вкладені елементи взаємозамінні. Наприклад, наведена щойно кон струкція рівнозначн а такій:

<факультет>

<корпус>5/1</корпус>

<назва>інформатики</назва>

</факультет>

Атрибути і вкладені елементи можуть мати різне семантичне навантаження: в контексті документа атрибути ідентифікують елементи й відображують властивості елементу в цілому, а за допомогою вкладених елементів визначається структура документа

Порожні елементи

Порожнійназивається елемент, що не має вмісту, і зокрема не містить інших елементів. Є два еквівалентних способи позначення порожнього елементу: <ім`я_елементу [атрибути]> </ім`я_елементу>, або <ім`я_елементу [атрибути]/> Наприклад;

<кафедра назва_кафедри = "конп`ютерних_наук" завідувач = "І.П. Петров" фонд = "2000.00"/>

Розділи CDATA

Щоб символи текстового рядка, який визначає вміст елементу, не розпізнавалися як символи розмітки, вони записуються в так званому CDATA -розділі Він може розміщ уватися всюди, де допускається записувати рядкові дані. CDATA-розділ починається із запису <|[CDATA] і закінчується символами]]>. Всередині можуть вказуватись довільні символи, разом із тими, які використовую ться для розмітки.

Посилання на об'єкти

Документи XML можуть містити посилання на різноманітні об’єкти, зокрема на розділи тексту в тому ж самому або в іншому документі. Посилання є рядком, що починається з символу амперсанда (&) і закінчується крапкою з комою. Наприклад, наведені нижче посилання доцільно записати на початку журнальної статті

<стаття>
&Вступ;
&тіло;
&висновок;
&ресурси;
</ стаття>

Простори імен

XML-дані можуть перелапатися а однієї оргапіаації до іншої. Імена деяких елементів в різних організаціях можуть інтерпретуватися по-різному, внаслідок чого документ буде розтлумачено некоректно. Використання певного унікального рядка як доповнення до імені елементу допомагає вирішити цю проблему. Такий рядок позначає простір імені вказується перед іменем елементу згідно з таким синтаксисом: ім`я_простору: ім`я елементу. Передбачається, що ім’я xml є зарезервованим, і користувач не може йото вживати для власник потреб. Наведемо приклад використання простору імен;

<NAU:кафедра>
<NAU: назва_кафедри> комп’ютерних наук </NAU: назва_кафедри>
<NAU: завідувачі> І.П. Петров </NAU: завідувачі>
<NAU: фонд> 2000,00 </NAU: фонд>
</NAU:кафедра>

Зазначимо, що останнім часом почали створюватися служби реєстрації і підтримки просторів імен для різних співтовариств розробників і користувачів XML-ресурсів

Лекція

Тема: ***«Бази даних із вбудованою підтримкою XML»***

План лекції

1 Різновиди баз даних із вбудованою підтримкою XML
- 1.1 Текстові бази даних із вбудованою підтримкою XML
- 1.2 Модельні бази даних із вбудованою підтримкою XML

2 Огляд функцій і можливостей БД із вбудованою підтримкою XML
- 2.1 Колекції документів
- 2.2 Мови запитів
- 2.3 Транзакції, блокування й одночасний доступ
- 2.4 Інтерфейс прикладного програмування
- 2.5 Ідентичність логічного документа його оригіналу, що зберігається в базі даних
- 2.6 Індексування
- 2.7 Збереження зовнішніх об’єктів

Зміст лекції

Бази даних з вбудованою підтримкою XML розробляються спеціально для збереження і маніпулювання документами XML. Для таких баз даних існує поняття логічної моделі документа XML в межах якої документи означуються й зберігаються, а також здійснюється їхній пошук і маніпулю вання ними. Модель документа має містити елементи, атрибути, а також описувати впорядкованість дан их у документі. БД із вбудованою підтримкою XML не висувають вимог до моделі фізичного зображення документів, тому можуть реалізовуватись як незалежно, так і на основі наянних реляційних, об'єктно-орієнтованнх, ієрархічних або інших баз даних. Головне, щоб зовнішній інтерфейс був орієнтований саме на документи XML, а не на модель, яка підтримується СКБД.

Системи баз даних із вбудованою підтримкою XML, подібно до інших систем баз даних, забезпечують виконання усіх властивих СКБД функцій, таких як захист даних, підтримка їхньої цілісності, резервного копіювання та відновлення, мов запитів, транзакцій, тригерів, доступу багатьох кори стувачів тощо.

Використання БД із вбудованою підтримкою XML є найефективнішим у разі збереження й обробки документів XML, орієнтован их на зображення документів.

Саме такі БД підтримують упорядкованість усередині документа, коментарі, розділи CDATA, що звичайно не реалізується в СКБД з іншими моделями даних, розширеними для підтримки XML БД із вбудованою підтримкою XML дають змогу використовувати мови запитів на ослові XML

й формулювати запити типу «Вивести документи, де третій абзац після початку кожного розділу містить текст, виділений жирним шрифтом». Очевидно, що подібні запити важко формулювати в SQL.

БД із вбудованою підтримкою XML можуть зберігати слабкоструктуровані дані, даючи при цьому змогу підвищувати швидкість пошуку та обробляти документи, що не мають DTD або інших схем XML (документи без схем). Тобто такі бази даних дозволяють зберігати й обробляти будь-які документи XML, що не містять опису відображень. Під час використання реляційних або об'єктно-орієнтованнх баз даних для збереження документів XML необхідно попередньо описати відображення схеми XML на схему бази даних. Відсутність або необов’язковість попереднього настроювання на схему XML у БД із вбудованою підтримкою XML виявляється досить корисною у застосуваннях типу пошукових веб-систем.

Пошуковий механізм цих застосувань обробляє документи XML, що можуть мати найрізноманітніш і схеми або навіть зовсім не супроводжуватися схемами.

1 Різновиди баз даних із вбудованою підтримкою XML

Є два різновиди БД із вбудованою підтримкою XML:

- орієнтовані на зберігання тексту (текстові XML-БД);
- орієнтовані на модель (модельні XML-БД).

1.1 Текстові бази даних із вбудованою підтримкою XML

У текстових XML-БД документи XML зберігаються у вигляді текстів. Сховищем таких текстів можуть бути файли операційної системи, поля таблиці реляційної бази даних, що дають змогу зберігати рядкові об’єкти великого розміру, або спеціальні репозиторії текстової інформації

Характерною ознакою всіх орієнтованих на зберігання тексту БД із вбудованою підтримкою XML є наявність індексів, що дають можливість легко й швидко виконувати пошук у XML-документах. У БД на основі XML цього типу пошук здійснюється одноразовим переглядом індексу. Коли індекс переглянуто, необхідний фрагмент документа можна вибирати одноразовим зчитуванням, якщо цей фрагмент зберігається на диску у вигляді безперервної послідовності байтів.

З іншого боку, збирання документа з його складових частин, яке здійснюється в реляційних системах баз даних і в деяких орієнтованих на модель БД із вбудованою підтримкою XML, вимагає багаторазового перегляду індексу і багаторазових звернень до диска.

Текстові XML-БД подібні до ієрархічних БД тим, що обидві значно перевершують реляційну БД за продуктивністю в тому випадку, коли логіка пошуку даних відповідає попередньо визначеній ієрархічній структурі. Якщо логіка пошукового виразу і форма виведення даних не відповідають

ієрархічній структурі документа, то текстові XML-БД, як і ієрархічні, у швидкодії значно програють реляційннм БД.

1.2 Модельні бази даних із вбудованою підтримкою XML

Іншою категорією є БД, орієнтовані на модельне зображення документів XML

У таких БД документ XML зображується у внутрішній об'єктній моделі й саме в такому вигляді зберігається. Спосіб збереженн я моделі документів XML залежить від специфіки реалізації XML-БД.

У деяких XML-БД моделі документів зберігаються в реляційних чи об’єктно-орієнтованих базах даних. Наприклад, збереження моделі DOM у реляційній базі даних може привести до побудови таких таблиць, як Elements, Attributes, PCDATA, Entities та EntityReferences. В інших орієнтованих на модель базах даних із вбудованою підтримкою XML використовуються фізичні структури, спеціально розроблені для оптимального збереження і маніпулювання моделлю документа XML. Модельні XML-БД, побудовані на основі баз даних іншого типу, мають гірші характеристики продуктивності, особливо щодо пошуку даних, аніж ті, які використовують власні, спеціально розроблені підобрану модель, формати збереження даних.

Модельні XML-БД, у яких для збереження даних використовуються спеціальні фізичні структури, характеризуються продуктивністю пошуку даних, порівнянною з продуктивністю текстових XML-БД під час пошуку за ієрархічною структурою. Це пояснюється тим, що більшість модельних XML-БД використовують покажчики для навігації структурою документа. Показники продуктивності текстових і модельних XML- БД залежать від форми подання результатів.

Текстові XML-БД набагато швидше виводять документи в текстовому вигляді, а модельні скоріше виводять документи, наприклад, у вигляді DOM-дерев (за умови, що їхня модель документа легко відображується в DOM).

Як і текстові XML-БД, модельні XML-БД працюють менш ефективно підчас пошуку й виведення даних у форматах, що відрізняються від того, який обраний для збереження документів

2 Огляд функцій і можливостей БД із вбудованою підтримкою XML

2.1 Колекції документів

У багатьох БД із вбудованою підтримкою XML дані зберігаються у вигляді колекцій. Колекції відіграють ту саму роль, що й таблиці в реляційній базі даних або каталоги у файловій системі. Припустимо, що БД із вбудованою підтримкою XML використовується для зберігання документів, які містять відомості про викладачів вузу. У цьому випадку можна визначити колекцію викладачів для того, щоб запити, які стосуються викладачів, обробляли лише документи цієї колекції.

Інший приклад: необхідно зберігати документацію щодо всіх програмних продуктів, які використовуються на підприємстві. У цьому випадку можна визначити ієрархію колекцій. На першому рівні можна побудувати колекцію із загальною інформацією про програмні продукти, а на другому – підколекції, кожна з яких містить документацію, що стосується програмних продуктів певного типу.

2.2 Мови запитів

БД із вбудованою підтримкою XML мають підтримувати принаймні одну спеціалізовану мову запитів. Найбільш популярними мовами є XPath (з можливим розширенням для побудови запитів за багатьма документами) і XQuery. Деякі БД із вбудованою підтримкою XML оснащені власними спеціалізованими мовами запитів. Вельми ймовірно, що в майбутньому всі БД із вбудованою підтримкою XML будуть підтримувати мову XQuery.

О новлення й видален ня документів

У БД із вбудованою підтримкою XML застосовується багато різних стратегій оновлення і видалення документів. До них належать:

- просте видалення або заміна наявних документів;
- модифікація документів з використанням певної моделі DOM;
- використання спеціалізованих мов з метою модифікації документів або їхніх фрагментів.

Багато з цих методів маніпулювання документами є специфічними для конкретних XML-БД. Проте існують кілька стандартних мов маніпулювання документами XML. Наприклад, мова XUpdate, в якій для ідентифікації вершин використовуються вирази мови XPath, призначена для оновлення XML-документів. Крім того, членами робочої групи W3C XQuery було запропоновано кілька розширень мови XQuery засобами оновлення і видалення документів XML та їхніх фрагментів. Незабаром мова XQuery має бути офіційно розширена можливостями з оновлення документів.

2.3 Транзакції, блокування й одночасний доступ

Практично всі БД із вбудованою підтримкою XML дозволяють використовувати транзакції з можливістю відкочування. Однак блокування здійснюється на рівні всього документа, а не на рівні окремих вершин у його структурі. У зв'язку з цим багатокористувацькнй режим підтримується на досить низькому рівні. Чи справді це впливає на ефективність роботи в даному режимі, залежить від конкретного застосування і від того, що розуміти під поняттям «документ». Розглянемо кілька прикладів.

1. Документом є розділ технічної документації. Під час написання документації обов'язки між її розробниками були розподілені так, що кожний працював над своїм розділом. Блокування на рівні документа (тобто розділу) навряд чи призведе до виникнення проблем, пов'язаних з одночасним доступом, оскільки

малоймовірно, що один і той самий розділ оновлюватиметься кількома авторами водночас.

2. Документ містить інформацію про резервування авіаквитків на один рейсовий політ. Великою є ймовірність того, що за наявності багатьох авіакас попереднього резервування і продажу квитків блокування на рівні документа призведе до виникнення проблеми одночасного доступу.
3. Документ містить інформацію, що використовується в послідовному технологічному процесі, наприклад під час укладання фінансової угоди. На кожному кроці цього процесу здійснюється зчитування даних із документа і додавання до нього нових даних. Наприклад, один із кроків може полягати в перевірці платоспроможності клієнта і введенні у документ даних про нове кредитування. Інший крок може полягати в перевірці заборгованості клієнта за іншими контрактами і введенні або зміні значення сумарної заборгованості. Якщо блокування здійснюється на рівні вершин структури документа, то багато операцій, подібних до згаданих вище, можуть виконуватися паралельно. Якщо блокування здійснюється на рівні документа, то операції можуть виконуватися лише послідовно. Це може призвести до неприпустимих затримок у роботі застосувань, що інтенсивно використовуються.

Блокування на рівні вершини XML-структури зазвичай вимагає одночасного блокування «батька» цієї вершини, а також «батька» цього «батька» і так до самого кореня. Щоб пояснити, чому потрібне багаторівневе блокування, розглянемо такий приклад. Нехай транзакція виконує оновлення листкової вершини. Якщо не було заблоковано її батьківську вершитту, то інша транзакнія може в цей час видалити дану батьківську вершину, що, у свою чергу, призведе до видалення листкової вершини, яка оновлюється першою транзакцією. Таким чином, кожного разу під час оновлення вершини слід заблокувати її саму, всі її дочірні вершини та всіх її предків аж до кореневої вершини. Отже, буде заблоковано певний шлях віл кореня до листка, а вершини, розташовані на інших шляхах в ієрархічній структурі документа XML. залишаться доступними для інших транзакцій.

2.4 Інтерфейс прикладного програмування

Майже всі БД із вбудованою підтримкою XML надають інтерфейс прикладного програмування. Це, як правило, ODBC-подібний інтерфейс, що містить методи для з’єднання з базою даних, аналізу метаданих, виконання запитів та перегляду результату. Результат виконання запиту може повертатися у вигляді XML-тексту дерева DOM або в інших форматах. Якщо запити можуть повертати багато документів, то надаються методи їхнього ітеративного перебирання.

Багато БД із вбудованою підтримкою XML надають можливості виконувати запити і повертати результати з використанням протоколу HTTP.

2.5 Ідентичність логічного документа його оригіналу, що зберігається в базі даних

Характерною ознакою БД із вбудованою підтримкою XML є ідентичність документа, що вибирається з БД, його оригіналу. Це дуже важлива властивість для застосувань, що оперують поняттям документа, і для яких розділи CDATA, використання сутностей, коментарі й команди обробки становлять невід'ємну частину документа.

Ідентичність логічних і збережених документів є менш важливою для застосувань, що орієнтуються на обробку не самих документів, а наявних у них даних.

Для таких застосувань у документі XML важливими є: атрибути, текст та ієрархічна структура. У цьому випадку під час передавання даних між документами XML і базами даних важливо зберегти ідентичність не всього документа, а лише зазначених вище складових. В окремих випадках для орієнтованих на дані застосувань важливо зберігати порядок дочірніх елементів і даних PCDATA у межах батьківського елементу. У загальному випадку застосування, що орієнтован і на обробку даних, не підтримують впорядкованість елементів документа, команди обробки, коментарі й фізичні структури зберігання даних (посилання на сутності, розділи CDATA і т. д.), тому їх не можна використовувати для збереження й обробки документів.

Усі БД із вбудованою підтримкою XML можуть підтримувати ідентичність елементів документів, порядку їхнього розташ ування, атрибутів і даних PCDATA

Більш повне забезпечення ідентичності збережених і логічних документів залежить від реалізації конкретних XML-БД. Як правило, текстові XML-БД підтримують повну ідентичність документів, а модельні XML-БД-ідентичність на рівні моделі документа.

2.6 Індексування

У БД із вбудованою підтримкою XML індекси використовуються для прискорення пошуку. Існують три типи індексів. Індекси значень дають можливість індексувати значення атрибутів або тексту. Вони дозволяють здійснювати швидкий пошук за запитами типу «Знайти всі елементи або атрибути, що містять значення база даних XML». Індекси структури дають можливість індексувати розташування елементів і атрибутів та прискорю вати пошук за запитами на кшталт «Знайти всі елементи <заголовок>». Разом індекси значень і структури дають змогу відповідати на запити типу «Знайти всі елементи <заголовок>, що містять значення база даних XML». Нарешті, повнотекстові індекси дають можливість індексувати всі слова в тексті та значеннях атрибутів і використовуються при реалізації запитів типу «Знайти

документи, що містять усі слова з фрази база даних XML», або спільно зі структурними індексами для обробки запитів типу «Знайти документи, що містять усі слова з фрази база даних XML в елементах <заголовон>». Як правило, в базах даних із вбудованою підтримкою XML застосовуються індекси значень і структури, а інколи й повнотекстові індекси.

2.7 Збереження зовнішніх об’єктів

Під час збереження документів XML у базі даних принциповим є спосіб обробки зовнішніх об'єктів. Можна вибрати зовнішній об'єкт і зберегти його разом з основним документом, або ж залишити в документі, що зберігається, тільки посилання на зовнішній об'єкт. У кожному окремому випадку кращим може виявитися як перший, так і другий спосіб.

Розглянемо випадок, коли документ містить посилання на CGI-програму, що надає інформацію про погоду на поточний день. Якщо документ, що зберігається, використовується як веб-сторінка з інформацією про сьогоднішню погоду, то було б помилковим, зберігаючи документ, замінити посилання на зовнішній об'єкт його значенням, оскільки в цьому випадку збережена сторінка вже не міститиме реальних даних наступного дня. З іншого боку, якщо документ є частиною колекції хронологічних даних про погоду і якщо він має містити дані про погоду на день його збереження, то буде помилковим зберігати документ разом із посиланням на зовнішній об'єкт, оскільки в цьому випадку документ буде містити прогноз погоди не на той день, коли він був збережений, а на той день, коли до нього звертаються.

БД із вбудованою підтримкою XML не можуть самостійно обирати спосіб збереження зовнішніх об’єктів, вони лише надають можливість робити такий вибір у кожному конкретному випадку

Лекція

Тема: ***«Загальна характеристика баз знань»***

План лекції

1 Базові поняття
2 Виведення на знаннях

Зміст лекції

1 Базові поняття

Знання пов'язані з даними, ґрунтуються на них, але представляють результат розумової діяльності людини, узагальнюють його досвід, отриманий в ході виконання якої-небудь практичної діяльності. Вони є результатом емпіричного досвіду.

Знання - це виявлені закономірності наочної області (принципи, зв'язки, закони), що дозволяють вирішувати завдання в цій області. При обробці на ЕОМ знання трансформуються аналогічно даним:

- знання в пам'яті людини як результат мислення;
- матеріальні носії знань (підручники, методичні посібники);
- поле знань - умовний опис основних об'єктів наочної області, їх атрибутів і закономірностей, що їх зв'язують;
- знання, описані на мовах подання знань (продукційні мови, семантичні мережі, фрейми);
- бази знань.

Часто використовуються такі визначення знань: знання - це добре структуровані дані, або дані про дані, або метадані.

Існує безліч способів визначати поняття. Один з широко вживаних способів, заснований на ідеї інтенсионала.

Інтенсионал поняття - це визначення через поняття більш високого рівня абстракції з вказівкою специфічних властивостей. Цей спосіб визначає знання. Інший спосіб визначає поняття через перерахування понять нижчого рівня ієрархії або фактів, що відносяться до визначуваного. Це є визначення через дані, або екстенсионал поняття.

Знання можуть бути класифіковані по наступних категоріях:

- поверхневі - знання про видимі взаємозв'язки між окремими подіями і фактами в наочної області;
- глибинні - абстракції, аналогії, схеми, що відображують структуру і процеси в наочної області.

Знання можна розділити на процедурні і декларативні. Історично первинними були процедурні знання, тобто знання, "розчинені" в алгоритмах. Вони управляли даними. Для їх зміни необхідно було змінювати

програми. Проте з розвитком штучного інтелекту пріоритет даних поступово змінювався, і все більша частина знань зосереджувалася в структурах даних (таблиці, списки, абстрактні типи даних), тобто збільшувалася роль декларативних знань.

Сьогодні знання набули чисто декларативної форми, тобто знаннями вважаються речення, записані на мовах подання знань, наближених до природніх і зрозумілих неспеціалістам. Існують десятки моделей (або мов) подання знань для різних наочних областей. Більшість з них може бути зведені до наступних класів: продукційні; семантичні мережі; фрейми; формальні логічні моделі і багато інших.

Ці і інші моделі ми розглянемо в наступній лекції розділу

Стратегії здобуття знань

Існує декілька стратегій здобуття знань. Найбільш поширені:

- придбання;
- витягання;
- формування.

Під придбанням знань розуміється спосіб автоматизованої побудови бази знань за допомогою діалогу експерта і спеціальної програми (при цьому структура знань заздалегідь закладається в програму). Ця стратегія вимагає істотного попереднього опрацювання наочної області. Систем придбання знань дійсно набувають готові фрагменти знань відповідно до структур, закладених розробників систем. Більшість цих інструментальних засобів орієнтована на конкретні експертні системи з жорстко визначеною предметною областю і моделлю подання знань, тобто не є універсальними.

Термін витягання знань стосується безпосереднього живого контакту інженера по знаннях і джерела знань. Автори схильні використовувати цей термін як більш ємкий і такий, що більш точно виражає сенс процедури перенесення компетентності експерта через інженера по знаннях в базу знань експертної системи.

Термін формування знань традиційно закріпився за надзвичайно перспективною областю інженерії знань, яка займається розробкою моделей, методів і алгоритмів аналізу даних для здобуття знань і вчення, що активно розвивається. Ця область включає індуктивні моделі формування гіпотез на основі повчальних вибірок, вчення аналогічно і інші методи.

2 Виведення на знаннях

База знань – сукупність знань, що відносяться до деякої предметної області, формально представлених так, щоб на їх основі можна було здійснювати міркування. Бази знань найчастіше використовуються в контексті експертних систем, де з їх допомогою подаються навики і досвід експертів, зайнятих практичною діяльністю у відповідної області (наприклад, в медицині або в математиці). Зазвичай база знань є сукупністю правил виводу.

Не дивлячись на всі недоліки, найбільшого поширення набула

продукційна модель подання знань. При використанні продукційної моделі база знань складається з набору правил. Програма, що управляє перебором правил, називається машиною виведення. Машина виведення (інтерпретатор правил) виконує дві функції: по-перше, перегляд існуючих фактів з робочої пам'яті (бази даних) і правил з бази знань і додавання (в міру можливості) в робочу пам'ять нових фактів, а по-друге, визначення порядку перегляду і вживання правил. Цей механізм управляє процесом консультації, зберігаючи для користувача інформацію про отримані висновки, і запрошує у нього інформацію, коли для спрацьовування чергового правила в робочій пам'яті виявляється недостатньо даних.

У переважній більшості систем, заснованих на знаннях, механізм виведення є невеликою за об'ємом програмою і включає два компоненти — один реалізує власне виведення, інший управляє цим процесом. Дія компонента виведення заснована на вживанні правила, яке називається modus ponens.

Правило modus ponens. Якщо відомо, що достеменне твердження А, то існує правило вигляду – «Якщо А, то В», тоді твердження В теж істинно. Правила спрацьовують, коли знаходяться факти, що задовольняють їх лівій частині: якщо достеменна посилання, то має бути достеменний і висновок. Компонент виведення повинен функціонувати навіть при недоліку інформації. Отримане рішення може і не бути точним, проте, система не повинна зупинятися через те, що відсутня будь-яка частина вхідної інформації. Компонент, що управляє, визначає порядок вживання правив і виконує чотири функції:

Зіставлення — зразок правила зіставляється з наявними фактами.

Вибір — якщо в конкретній ситуації може бути застосовано відразу декілька правив, то з них вибирається одне, найбільш відповідне по заданому критерію (вирішення конфлікту).

Спрацьовування — якщо зразок правила при зіставленні збігся з якими-небудь фактами з робочої пам'яті, то правило спрацьовує.

Дія — робоча пам'ять піддається зміні шляхом додавання в неї висновку що спрацював, правила. Якщо в правій частині правила міститься вказівка на будь-яку дію, то вона виконується (як, наприклад, в системах забезпечення безпеки інформації).

Інтерпретатор продукцій працює циклічно. У кожному циклі він переглядає всі правила, щоб виявити ті, посилання яких збігаються з відомими на даний момент фактами з робочої пам'яті, Після вибору правило спрацьовує, його висновок заноситься в робочу пам'ять, і потім цикл повторюється спочатку. В одному циклі може спрацювати лише одне правило. Якщо декілька правил успішно зіставлені з фактами, то інтерпретатор виконує вибір по певному критерію єдиного правила, яке спрацьовує в даному циклі. Цикл роботи інтерпретатора схематично представлений на рисунку 1.

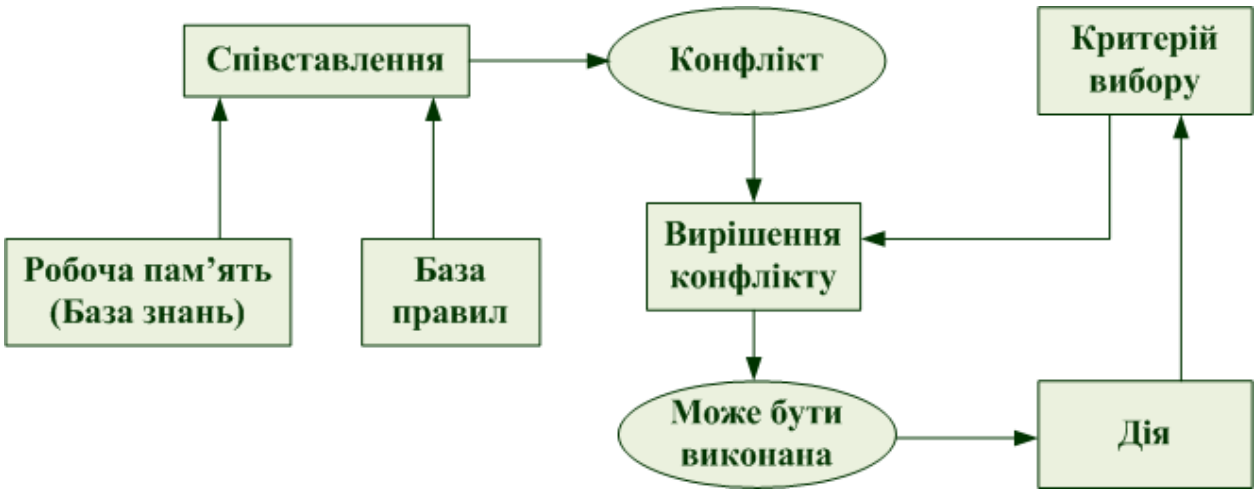


Рисунок 1 – Цикл роботи інтерпретатора

Інформація з робочої пам'яті послідовно зіставляється з посиланнями правил для виявлення успішного співставлення. Сукупність відібраних правил складає так звану конфліктну безліч. Для вирішення конфлікту інтерпретатор має критерій, за допомогою якого він вибирає єдине правило, після чого правило спрацьовує. Це виражається в занесенні фактів, які створюють висновок правила, в робочу пам'ять або в зміні критерію вибору конфліктуючих правил. Якщо ж по закінченню правила згадується назва якої-небудь дії, то воно виконується. Робота машини виведення залежить лише від стану робочої пам'яті і від складу бази знань. На практиці зазвичай враховується історія опрацювання, тобто поведінка механізму випаду в попередніх циклах. Інформація про поведінку механізму виведення запам'ятовується в пам'яті станів. Зазвичай пам'ять станів містить протокол системи.

Від вибраного методу пошуку, тобто стратегії виведення, залежатиме порядок вживання і спрацьовування правил. Процедура вибору зводиться до визначення напряму пошуку і способу його здійснення. Процедури, що реалізовують пошук, зазвичай закладені в механізм виведення, тому в більшості систем інженери знань не мають до них доступу і, отже, не можуть в них нічого змінювати за власним бажанням. При розробці стратегії управління виводом поважно визначити дна питання:

Яку точку в просторі станів прийняти як початкову? Від вибору цієї точки залежить і метод здійснення пошуку — в прямому або зворотному напрямі.

Якими методами можна підвищити ефективність пошуку рішення?

Ці методи визначаються обраною стратегією перебору – глибину, ширину, по підзадачах або інш. При зворотному порядку виведення спочатку висувається деяка гіпотеза, а потім механізм виведення як би повертається назад, переходячи до фактів, намагаючись знайти ті, які підтверджують гіпотезу (рисунок 2, зліва). Якщо вона виявилася правильною, то вибирається наступна гіпотеза, що деталізує першу і що є по відношенню до неї підціллю.

Далі відшукуються факти, підтверджуючі істинність підпорядкованої гіпотези. Виведення такого типа називається керованим цілями, або керованим консеквентами. Зворотний пошук застосовується в тих випадках, коли цілі відомі і їх порівняно небагато.

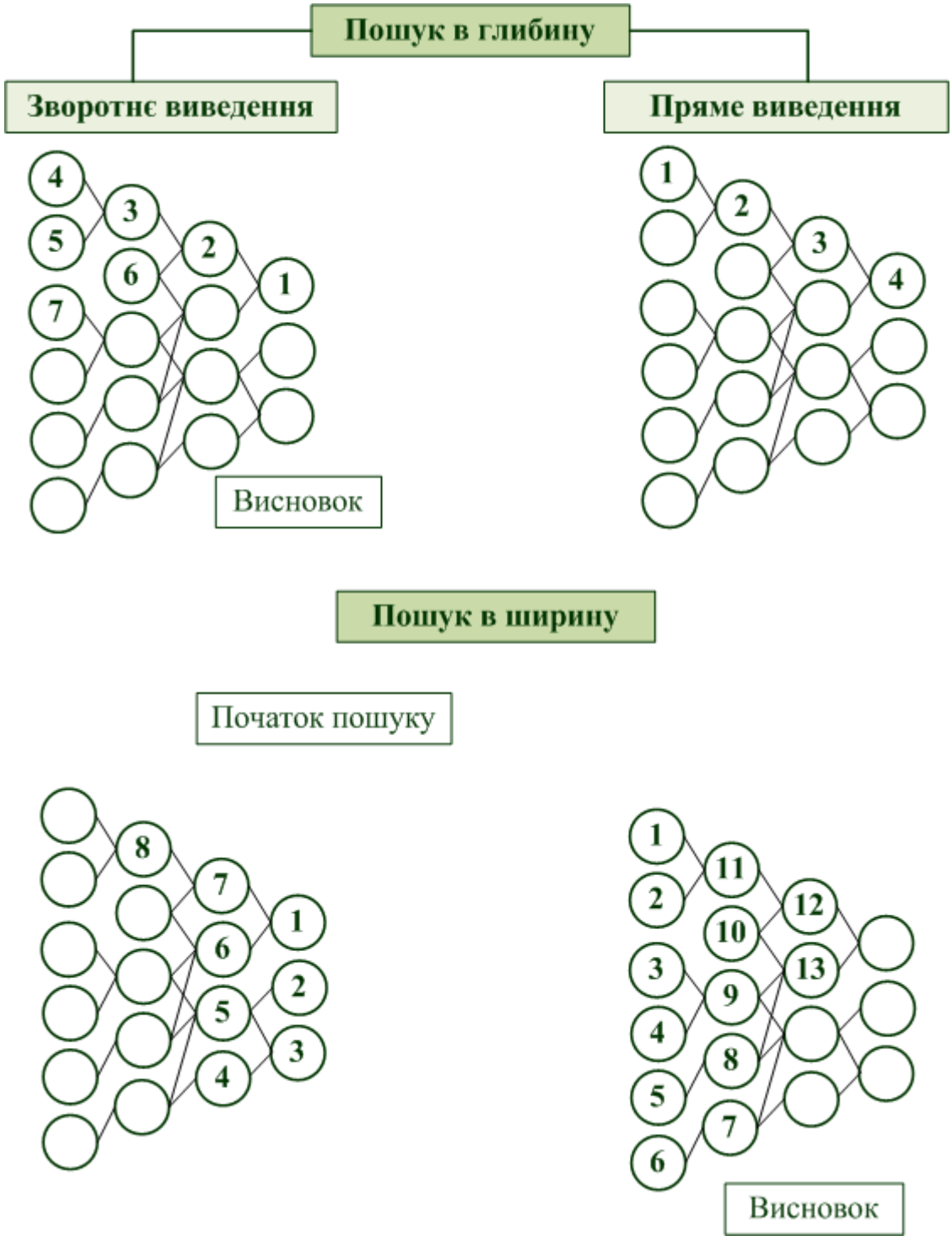


Рисунок 2 – Стратегії виведення

У системах з прямим виведенням по відомих фактах відшукується висновок, який з цих фактів виходить (див. рисунок 2, справа). Якщо такі висновки вдається знайти, то вони заносяться в робочу пам'ять. Пряме

виведення часто називають виведенням, керованими даними або керованими антецедентами. Існують системи, в яких вивід грунтується на поєднанні згаданих вище методів, — зворотного і обмеженого прямого. Такий комбінований метод отримав назву циклічного.

Є фрагмент бази знань з двох правил:

П1. ЯКЩО «відпочинок — влітку» і «людина — активний», ТО «їхати в гори».

П2. Якщо «любить сонце», ТО «відпочинок влітку».

Припустимо, що до системи поступили факти – «людина активна» і «любить сонце».

ПРЯМЕ ВИВЕДЕННЯ - виходячи з фактичних даних, отримати рекомендацію.

1-й прохід: Крок 1. Пробуємо П1, не працює (не вистачає даних «відпочинок – влітку») –

Крок 2. Пробуємо П2, працює, в базу поступає факт «відпочинок — влітку».

2-й прохід: Крок 2. Пробуємо П2, працює, активується мета «їхати в гори», яка і виступає як порада, яку пропонує експертна система.

ЗВОРОТНЕ ВИВЕДЕННЯ - підтвердити вибрану мету за допомогою наявних правил і даних.

1-й прохід. Крок 1. Мета — «їхати в гори»:

пробуємо П1 — даних «відпочинок — влітку» – немає, вони стають новою метою і пишеться правило, де мета в лівій частині.

Крок 2. Мета «відпочинок — влітку»:

правило П2 - підтверджує мету і активує її.

2-й прохід: Крок 3. Пробуємо П1, підтверджується шукана мета.

У системах, база знань яких налічує сотні правил, бажаним є використання стратегії управління виведенням, що дозволяє мінімізувати час пошуку рішення і тим самим підвищити ефективність виведення. До таких стратегій належать:

- пошук в глибину;
- пошук в ширину;
- розбиття на під задачі;
- альфа-бета.

При пошуку в глибину як чергова підмета вибирається та, яка відповідає наступному, детальнішому рівню опису завдання. Наприклад, діагностуюча система, зробивши на основі відомих симптомів припущення про наявність певного захворювання, продовжуватиме запрошувати уточнюючі ознаки і симптоми цієї хвороби до тих пір, поки повністю не спростує висунуту гіпотезу.

При пошуку в ширину, навпаки, система спочатку проаналізує всі симптоми, що знаходяться на одному рівні простору станів, навіть якщо вони відносяться до різних захворювань, і лише потім перейде до симптомів наступного рівня детальності.

Розбиття на підзадачі виконує виділення підзадач, вирішення яких

розглядається як досягнення проміжних цілей на шляху до кінцевої мети. Прикладом, підтверджуючим ефективності розбиття на підзадачі, є пошук несправностей в комп'ютері – спочатку виявляється підсистема (живлення, пам'ять і т. д.), що відмовила, що значно звужує простір пошуку. Якщо вдається правильно зрозуміти суть завдання і оптимально розбити її на систему ієрархічно зв'язаних цілей–підцілей, то можна домогтися мінімального шляху розв’язання в просторі пошуку.

Альфа-бета алгоритм дозволяє зменшити простір станів шляхом видалення гілок, неперспективних для успішного пошуку. Тому є видимими лише ті вершини, в які можна потрапити в результаті наступного кроку, після чого неперспективні напрями виключаються. Альфа-бета алгоритм знайшов широке вживання в основному в системах, орієнтованих на різні ігри, наприклад в шахових програмах.

Лекція

Тема: ***«Моделі знань»***

План лекції

1 Загальні поняття
2 Продукційна модель знань
3 Семантична модель знань

Зміст лекції

1 Загальні поняття

Моделювання – процес представлення досліджуваного об'єкту деякою послідовністю інших об'єктів або подань, що реалізовують ті або інші сторони об'єкту, що вивчається, з необхідною точністю. Модель завжди переслідує певну мету, і залежно від мети змінюється сама модель. Модель ніколи не відображає всю глибину об'єкту, що вивчається. Розрізняють такі види моделей:

- Модель предметної області.
- Модель бази даних.
- Модель бази знань.

Кожна модель зберігає знання про модельований фрагмент предметної області (інформаційній моделі) і виконує мовну функцію. Цей мовний компонент має велике значення при активізації моделі на ЕОМ. Перший етап побудови моделі пов'язаний з виявленням структури моделі на основі апріорної інформації, а другий етап пов'язаний з введенням в неї емпіричної інформації і є результатом узагальнення спостережень за реальним об'єктом.

При моделюванні знань і даних існує два аспекти: математичний і інструментальний. З точки зору математики модель може бути представлена безліччю відношень. Виникнення і розвиток інструментального аспекту обумовлене тим, що через відсутність в теорії банків інформації формальних методів побудови моделей ЕОМ прийняла на себе функції побудови моделей і стала інструментом побудови моделей. Ця модель призначена для опису на семантичному (смисловому) рівні ПО і навколишнього середовища. Вона є засобом інтерпретації вмісту бази знань. З точки зору інструментального аспекту моделювання знань, інтерес представляють мовні засоби опису моделі. Мови представлення знань можна розбити на три групи: логічні, реляційні, продукційні.

До логічних відносяться мови, засновані на численні висловів і предикатів. Основою реляційних мов є облік зв'язків (відношень) між предметами реального світу і їх властивостями.

Найбільш перспективною з мов даного типа є мова підстав на фреймах. Фрейм виступає як одиниця інформації, яка розглядає мінімальне необхідне число ознак в описі предмету, без яких цей предмет не існує.

Основою мови продукційного типа є поняття продукції, що складається з двох частин: умови і дії. Умовою є опис деякої ситуації, а дію визначає набір операцій, які необхідно здійснити, якщо вхідні параметри відповідають описаній ситуації.

2 Продукційна модель знань

Продукційна модель – модель, заснована на правилах, дозволяє представити знання у вигляді пропозицій типа – «Якщо (умова), то (дія)».

Під «умовою» (антецедент) розуміється деяке речення-зразок, по якому здійснюється пошук в базі знань, а під «дією» (консеквентом) - дії, що виконуються при успішному результаті пошуку (вони можуть бути проміжними, такими, що наступають далі як умови, і термінальними або цільовими, які завершують роботу системи). Найчастіше виведення на такій базі знань буває прямим (від даних до пошуку мети) або зворотним (від мети для її підтвердження - до даних).

Дані – це вихідні факти, що зберігаються в базі фактів, на підставі яких запускається інтерпретатор правил, що перебирає правила з продукційної бази знань.

Продукційна модель найчастіше застосовується в промислових експертних системах. Вона приваблює розробників своєю наочністю, високою модульністю, легкістю внесення доповнень і змін і простотою механізму логічного виведення. Існує велика кількість програмних засобів, що реалізовують продукційний підхід (мова OPS 5; «оболонки» або «порожні» ЕС — EXSYS Professional, Kappa, ЕКСПЕРТ і ін.).

3 Семантична модель знань

Термін семантична означає «смислова», а сама семантика – це наука, що встановлює відношення між символами і об'єктами, які вони позначають, тобто наука, що визначає сенс знаків. Мережа – це орієнтований граф, вершини якого – поняття, а дуги – відношення між ними. Як поняття зазвичай виступають абстрактні або конкретні об'єкти, а відношення – це зв'язки типа: «це» («АКО — A-kind-of» «is»), «має частиною» («has part»), «належить», «любить».

Характерною особливістю семантичних мереж є обов'язкова наявність трьох типів відношень:

- клас – елемент класу (квітка – троянда);
- властивість – значення (колір – жовтий);
- приклад елементу класу (троянда – чайна).

Можна запропонувати декілька класифікацій семантичних мереж, пов'язаних з типами відношень між поняттями. За кількістю типів відношень:

- Однорідні (з єдиним типом відношень).
- Неоднорідні (з різними типами відношень).

За типами відношень:

- Бінарні (у яких відношення зв'язують два об'єкти).
- N-арні (у яких є спеціальні відношення, що зв'язують більше двох понять).

Найчастіше в семантичних мережах використовуються такі відношення:

- зв'язки типа «частина – ціле» («клас – підклас», «елемент – безліч» і т.д.);
- функціональні зв'язки, які визначуються зазвичай дієсловами – виконує, кількісні (більше, менше, дорівнює);
- просторові (далеко від, близько від, за, під, над);
- часові (раніше, пізніше, протягом);
- атрибутивні зв'язки (мати властивість, мати значення);
- логічні зв'язки (І, АБО, НЕ);
- лінгвістичні зв'язки і ін.

Проблема пошуку рішення в базі знань типа семантичної мережі зводиться до завдання пошуку фрагмента мережі, що відповідає деякій підмережі, що відображає поставлений запит.

На рисунку 1 подана семантична мережа. Як вершини тут виступають поняття «ЕОМ – ПЕОМ - Ноутбук», «Toshiba Satellite C660-1EM», «Процесор» і «Материнська плата».

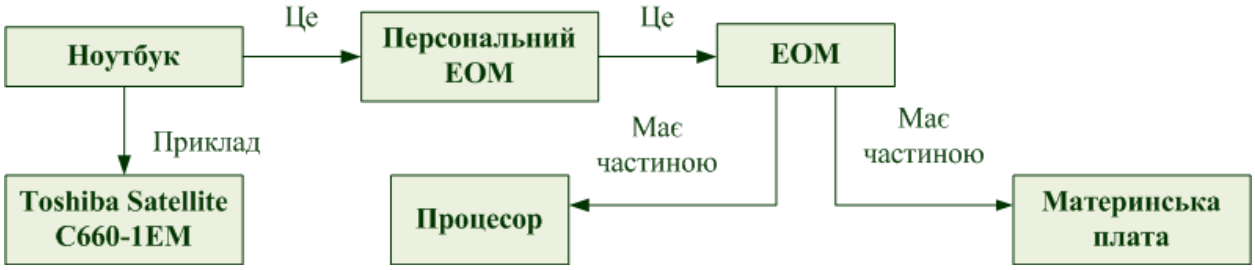


Рисунок 1 – Семантична мережа

Дана модель представлення знань була запропонована американським психологом Кілліаном. Основною її перевагою є те, що вона більш за інших відповідає сучасним уявленням про організацію довготривалої пам'яті людини. Недоліком цієї моделі є складність організації процедури пошуку виведення на семантичній мережі. Для реалізації семантичних мереж існують спеціальні мережеві мови, наприклад NET [Цейтін, 1985], мова реалізації систем Simer+mir [Осипов, 1Э97] і ін. Широко відомі експертні системи, що використовують семантичні мережі як мову представлення знань, - PROSPECTOR, CASNET, TORUS [Хейес-рот і ін. 19S7; Durkin, 1998].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Пасічник В.В., Резніченко В.А. Організація баз даних та знань. - К.: Видавнича група BHV, 2006. - 384с.: іл.

2 Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 1. Організація баз даних та знань: підручник. – Львів: «Магнолія-2006», 2015.–440с.

3 Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань: навч. посібник. – Львів: «Магнолія-2006», 2012.–584с.

4 Марченко А.В. Організація баз даних та знань. Електронний курс. – Суми: СумДУ. URL: https://ocw.sumdu.edu.ua/content/811 (дата звернення: 01.11.2017).

5 MySQL 5 / М.В. Кузнецов. И.В. Симдянов. – СПб.: БХВ-Петербург, 2010.–1024 с.: ил.

6 Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. – СПб.: КОРОНА принт, 2000. – 416с.

7 Дж. Грофф, П. Вайнберг SQL: Полное руководство: Пер. с англ. – 2-е изд., перераб. и доп. – К.: Издательская группа BHV, 2001. – 816с., ил.

8 Базы даннях: учеб пособие для студ. высш. учеб. заведений / А.В. Кузин, С.В. Левонисова. – 2-е изд., стер. – М.: Издательский центр «Академия», 2008. – 320 с.