Вступ до дисципліни

Марченко Анна Вікторівна

2015

Зміст

Вступ до дисципліни

Ми птиці інформаційного простору. Іншого у нас немає.
Мегабайти моєї пам’яті привалила інформація. Борсаюсь, як у пісках.
Ліна Костенко "*Записки українського самашедшого*"

Організація та структурованість даних дозволяє значно спростити сприйняття інформації та забезпечити адекватні та достовірні відповіді на безлічі запитань до інформаційного простору.

Загальнопрофесійна учбова дисципліна формує базові знання, що необхідні для засвоєння спеціальних дисциплін. Вивчення дисципліни «Організація баз даних та знань» відбувається у тісному зв’язку з іншими загальнопрофесійними курсами: «Алгоритмічні мови», «Програмування», «Методи та засоби комп'ютерних інформаційних технологій», «Операційні системи», «Експертні системи».

Вивчення дисципліни «Організація баз даних та знань» направлене на формування теоретичних знань та практичних навичок по основам проектування баз даних та знань, кваліфікованому використанню можливостей баз даних та знань. Для продуктивної праці рекомендується поступове вивчення матеріалів курсу: починаючи з теоретичної частини і завершуючи практичними завданнями. Закріпити отримані знання можна за допомогою тестів та тренажерів. Практичні завдання необхідно буде оформлювати у вигляді електронного звіту та надсилати викладачеві для перевірки.

Основна мета дисципліни:

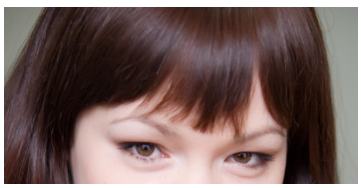
- оволодіння студентами основ концептуального та логічного проектування баз даних та знань, раціональної побудови корпоративних інформаційних систем, їх складових частин та їх взаємодії;
- випускники повинні знати об'єктно-орієнтований підхід при створенні систем електронного документообігу та можливості наявних на ринку систем управління базами даних як компонентів систем обробки інформації у системах електронного документообігу та САПР;
- випускники повинні самостійно розробляти архітектуру інформаційних систем та виконувати їх реалізацію та створювати технологічні програмні продукти обробки даних, забезпечуючи технологічну підтримку їх роботоспроможності.

Курс «Організація баз даних та знань» складається з 3 розділів, кожен з яких поділений на окремі теми. Для спрощення сприйняття інформації кожна тема поділена на питання. Для більш ефективного засвоєння матеріалів рекомендується вивчати курс по одній темі за заняття. По завершенню обробки теоретичного матеріалу теми раціональним є закріплення знань за допомогою тренажерів та практичних завдань та проходженням тестів. Для своєчасного виконання завдань рекомендується дотримуватись термінів звітування відповідно навчальному плану.

Завдання для дискусій направлені на отримання навичок вирішення питань вибору моделі даних та системи керування базами даних на початковому етапі проектування інформаційних систем.

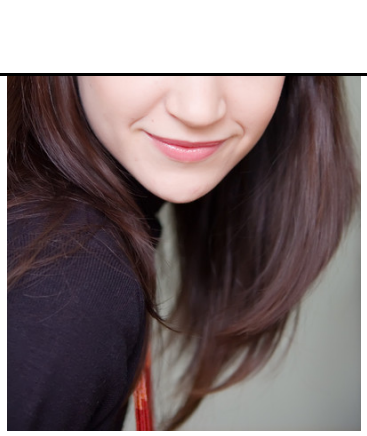
Індивідульне завдання реалізовано у вигляді курсової роботи по розробленню бази даних для предметної області, яка обирається згідно призначеного викладачем варіанту. Виконання роботи поділено на три тематичні етапи, які рекомендуєтсья виконувати поступово у визначені терміни. Для виконання курсової роботи достатньо буде знань та навичок, отриманих вами протягом вивчення теоретичних матеріалів та виконання практичних завдань.

П.І.П/б.	Марченко Анна Вікторівна
Науковий ступінь, вчене звання:	кандидат технічних наук
Посада та кафедра:	доцент кафедри комп'ютерних наук, секція ІТП.
Перелік дисциплін, що викладаються:	• «Архітектура ПК»; • «Експлуатація та обслуговування комп’ютерної техніки»; • «Вступ до спеціальності»;



Викладається:
- «Організація баз даних та знань».

Наукові інтереси:
- тривимірне геометричне моделювання робочих органів лопатевих насосів з метою автоматизації процесу проектування гідродинамічних машин.



Контактна інформація:
- електронна пошта: nenja_av@opm.sumdu.edu.ua
- персональна сторінка: http://cs.sumdu.edu.ua/staff/71-nenjaav
- робочий телефон: +38 (0542) 78-07-99
- місце роботи (аудиторія): СумДУ, корпус Г, аудиторія Г-1306.

Розділ 1 - Інформаційні системи з базами даних

Тема 1 - Інформаційні системи з базами даних

Ключові терміни:

Інформація, Системою керування базами даних, апаратне забезпечення, база даних, внутрішня модель, відображення «зовнішній-концептуальний», відображення «концептуальний-внутрішній», зовнішня модель, комунікаційне (мережне) забезпечення, концептуальна модель, концептуальна схема, концептуальне моделювання, лінгвістичне забезпечення, модель даних, організаційно-технологічне забезпечення, поліморфізм, предметна область, програмне забезпечення, сильно типізовані моделі, слабко типізовані моделі, спадкування, інкапсуляція, інформаційною системою

1.1. 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 рис. 1.1 наведені дві сторони визначення поняття інформації: функціональна й представницька. Перша загалом визначає коло дій над інформацією, а друга – результат виконання цих дій.

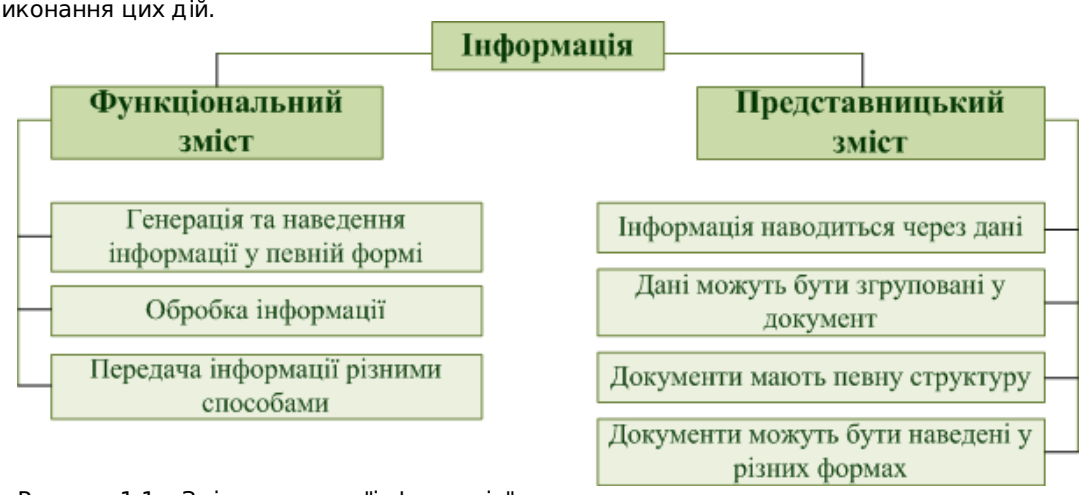


Рисунок 1.1 – Зміст поняття "інформація"

1.2. 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 рис. 1.2 наведені функції ІС через її основні структурні компоненти.

Рисунок 1.2 – Визначення інформаційної системи

1.3. 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 рис.1.3 наведені основні ознаки для кожної з зазначених вище концепцій. На рис. 1.4 проведене зіставлення основних методів обробки даних.

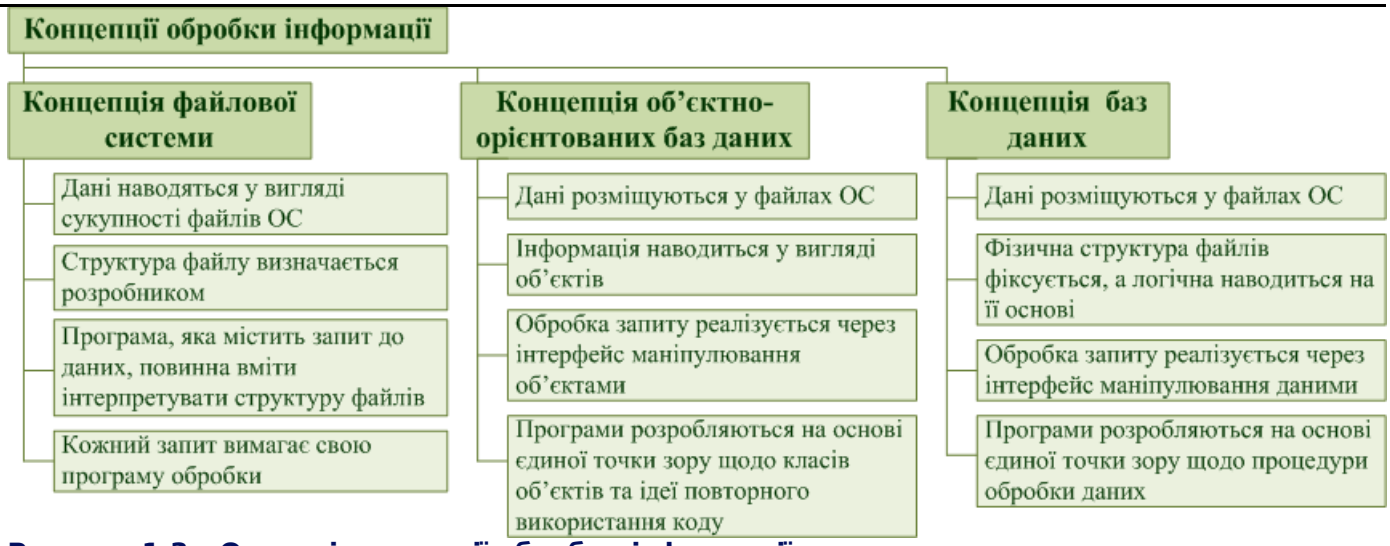


Рисунок 1.3 – Основні концепції обробки інформації

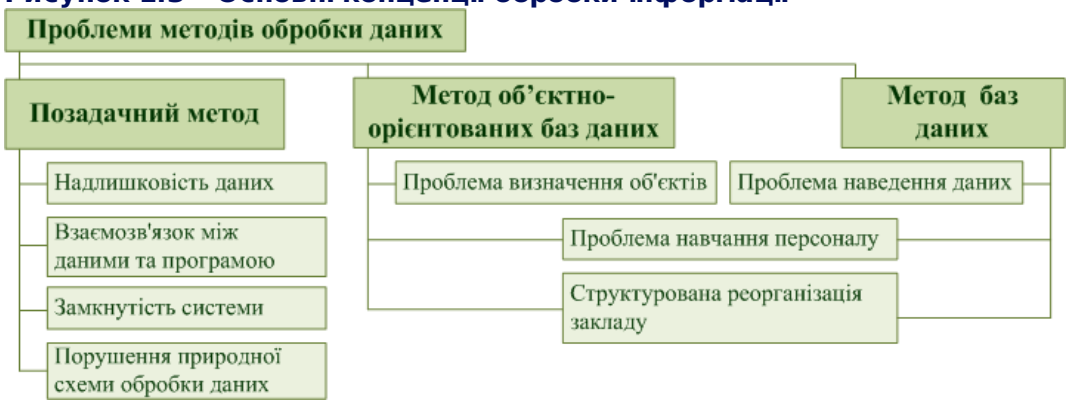


Рисунок 1.4 – Основні проблеми методів обробки інформації

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.

1.4. Бази даних і системи керування базами даних

1.4.1. Визначення поняття

Базу даних у загальному випадку можна визначити як уніфіковану сукупність збережених і

відтворених даних, що використовуються у рамках організації (EnglesR.A., 1972 р.). Однак поняття БД не ґрунтується в цей час на єдиній концепції, скоріше це ціле сімейство пов'язаних між собою понять з ПО, програмного й апаратного забезпечення, аналізу й моделювання даних і додатків. Існує кілька визначень БД.

База даних (за Дж. Мартіном) є сукупністю взаємозалежних даних, які спільно використовуються декількома додатками й зберігаються з мінімальною регульованою надлишковістю. Дані запам'ятовуються таким чином, щоб вони у міру можливості не залежали від програм. Для обробки даних застосовується загальний керуючий метод доступу. Якщо БД не перетинаються за структурою, то говорять про систему баз даних.

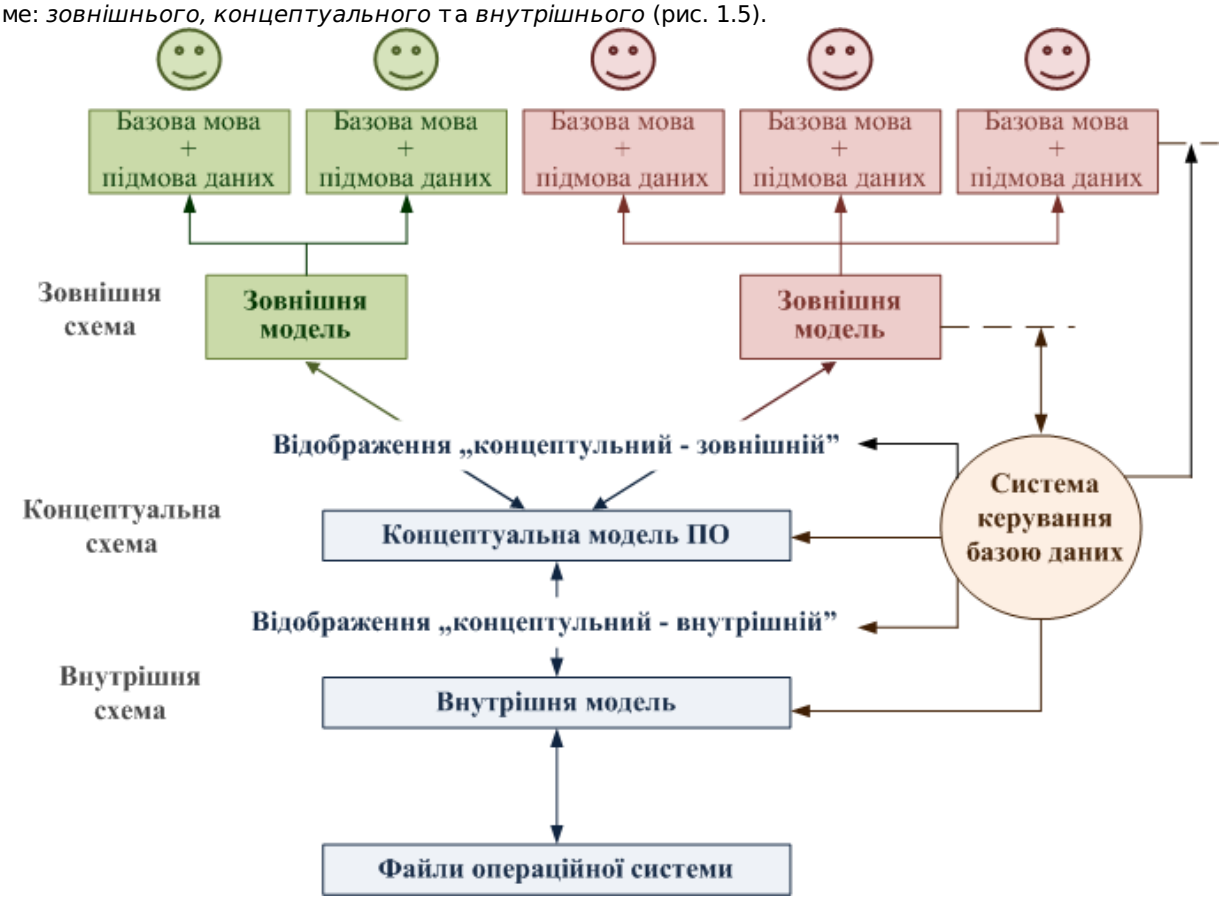
База даних (відповідно до матеріалів комітету КОДАСІЛ) складається зі всіх екземплярів записів, екземплярів наборів записів і областей, які контролюються конкретною схемою. Під схемою можна розуміти карту всієї логічної структури БД.

Для розроблювача ІС істотним моментом при використанні концепції баз даних (БД) є та обставина, що дані стають певним чином організовані, здобувають якусь упорядкованість і внутрішню структуру, а також те, що є деякий набір уніфікованих операцій обробки даних і декларативних засобів подання даних. До таких операцій варто віднести операції "Вставити" (Insert), "Додати" (Add), "Видалити" (Delete) і ряд інших. До декларативних засобів подання даних варто віднести мови визначення даних. Тобто використання даної концепції при створенні ІС припускає наявність мови визначення даних і мови маніпулювання даними, а також правил побудови інтерфейсів програм (додатків)із БД і користувачем.

Такий розподіл засобів маніпулювання даними і їхнього подання є деякою мірою умовним. Мова визначення даних служить для опису логічної структури (схеми) БД, а в деяких випадках і способів зберігання й доступу до даних. Мова маніпулювання даними надає алгоритмічні засоби побудови додатків для обробки елементів даних, які зберігаються в БД.

1.4.2 Архітектура БД

Основною ідеєю специфікації ANSI SPARC є виділення трьох архітектурних рівнів бази даних, а саме: *зовнішнього, концептуального* та *внутрішнього* (рис. 1.5).



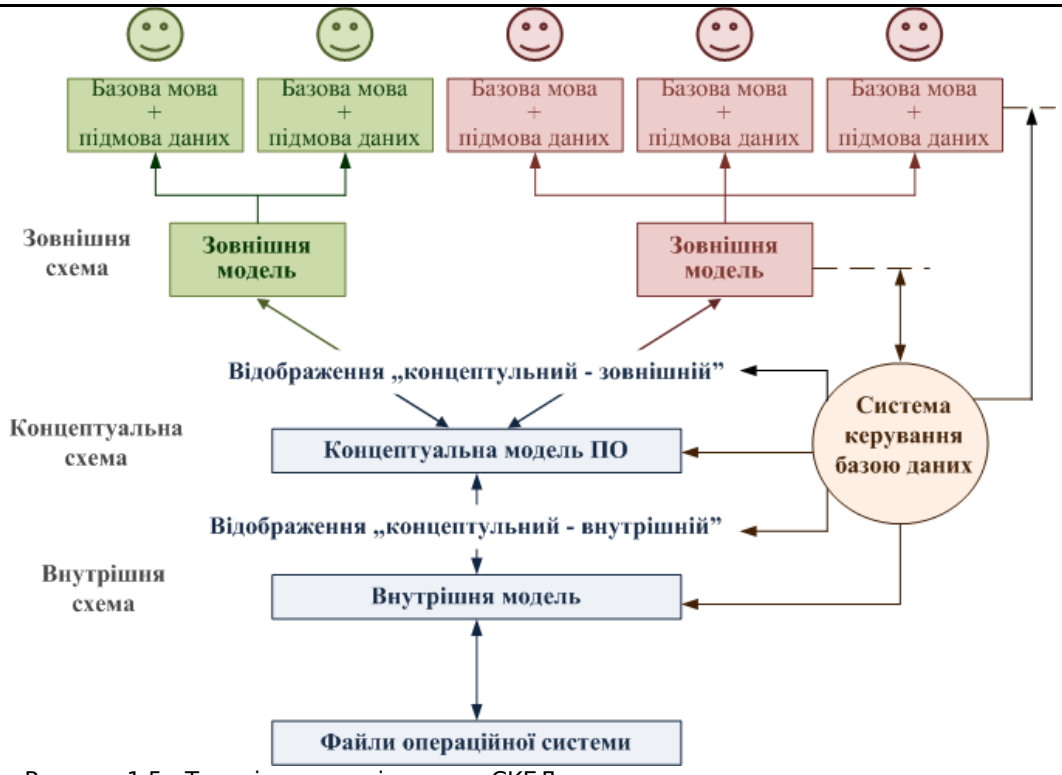


Рисунок 1.5 - Трьорівнева архітектура СКБД

4.2.1. 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 означення основних понять, що використовуються на концептуальному рівні.

Предметна область (ПО) - частина реального світу, для якої здійснюється концептуальне моделювання.

Концептуальна модель ПО формальне зображення сукупності думок, які характеризують можливі стани ПО, а також переходи з одного стану в 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 моделювання даних. У СКБД концептуальна модель подається у вигляді концептуальної схеми.

Опишемо властивості концептуальної моделі (схеми) й характерні особливості концептуального моделювання:

- Спільне та однозначне тлумачення предметної області всіма зацікавленими особами.
- Концептуальна схема відображує лише концептуально важливі аспекти ПО, виключаючи будь-які аспекти зовнішнього або внутрішнього відображення даних. Ця модель не повинна відображувати конкретні потреби окремих користувачів або додатків. Вона має фіксувати, чим є ПО в цілому, а не з точки зору інтересів або потреб користувачів.
- 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. В 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 «концепту альний-зовнішній». Аналогічно, будь-які зміни у внутрішній моделі не зачіпають концептуальну модель і моделі зовнішнього рівня, а тільки приводять до змін відображення «концептуальний-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.

1.4.2.2. Зовнішній рівень

Через зовнішній рівень користувачі та додатки отримують доступ до бази даних. Мета зовнішнього рівня - надати користувачу/додатку 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 додаток.

Зовнішня модель - це засоби зображення концептуальної моделі ПО з урахуванням інтересів конкретних користувачів або додатків. Кожна зовнішня модель подається в СКБД у вигляді зовнішньої схеми.

Зовнішній рівень виконує такі функції:

- Забезпечує зображення даних зручним для людини або додатку способом. Ступінь незалежності зовнішнього зображення від 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 даних Це досягається завдяки відображенню «концептуальний-зовнішній», що встановлює відповідність між концептуальною схемою і конкретною зовнішньою схемою. Потужність його засобів визначає ступінь логічної незалежності додатків від даних.

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

Загалом внутрішній рівень виконує такі функції:

- Забезпечує налаштування 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).

1.4.2.4. Відображення

Відображення зовнішнього рівня на концептуальний і концептуального рівня на внутрішній показані на рис. 1.5. Відображення «зовнішній-концептуальний» визначає відповідність між зовнішнім рівнем і концептуальним. Незалежність зовнішньої схеми, відтак і ступінь логічної незалежності даних, обумовлюються потужністю засобів опису цих відображень. Тобто можна описувати або змінювати зовнішню схему тільки в тих межах, які допускає це відображення.

Подібне можна сказати і про відображення «концептуальний-внутрішній», яке встановлює відповідність між концептуальною і внутрішньою моделями. Потужність його засобів визначає ступінь фізичної незалежності додатків від даних Будь-які зміни у фізичній структурі не повинні призводити до змін у концептуальній моделі – змінюється лише відображення «концептуальний-внутрішній».

За створення і ведення схем усіх рівнів (концептуальної, зовнішньої і внутрішньої), а також відображень відповідає адміністратор бази даних.

1.4.5 Системи керування базами даних

У разі застосування концепції БД для створення ІС природно виникає питання: хто або що повинне все це підтримувати? Таким чином, постає питання про Систему керування базою даних (СКБД). СКБД є складними програмними системами, що працюють на різних операційних платформах. Саме СКБД повинна надати засоби визначення й маніпулювання даними, зробивши дані незалежними від прикладних програм, що їх використовують.

До основних функцій СКБД слід віднести:

- забезпечення мовних засобів опису та маніпуляції даними;
- забезпечення підтримки логічної моделі даних;
- забезпечення взаємодії логічної та фізичної структур даних;
- забезпечення захисту та цілісності даних;
- забезпечення підтримки БД в актуальному стані.

Системою керування базами даних (Data-base Management System) називається сукупність програмних засобів, необхідних для використання БД і подання розробникам і користувачам безлічі різних подань даних.

1.5 Поняття моделі даних

Подання інформації за допомогою даних вимагає уніфікованого підходу до поняття даних як незалежного об'єкта моделювання. Тому для розробника ІС вибір відповідної моделі даних є однією з найважливіших проблем. Вибір моделі даних спричиняє вибір засобів аналізу ПО як області реального світу, що підлягає вивченню й обробці. Модель даних обмежує можливість вибору СКБД, тому що, як правило, окремо взята модель підтримує певну модель даних. Таким

чином, поняття моделі даних є одним з фундаментальних понять інформатики, від якого багато в чому залежать механізми реалізації ІС як програмно-апаратного комплексу.

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. У цьому випадку модельований світ начебто вміщується в гамівну сорочку. Наприклад, категорія «службовець» — строго фіксована, й усі її об’єкти повинні мати однакові властивості та структуру.

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 не має 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 (якщо вони використовуються) розглядається як додаткова. На відміну від сильно типізованих моделей, слабко типізовані забезпечують інтеграцію даних і категорій. Найкращі можливості такої інтеграції надаються численням предикатів, яке у багатьох моделях даних використовується для зображення знань, що не підтримуються базовими засобами моделювання.

Модель даних (Data Model) є логічною структурою даних, що становить притаманні цим даним властивості, незалежні від апаратного й програмного забезпечення й не пов'язані з функціонуванням комп'ютера.

Можна розглянути кілька аспектів моделювання в обробці даних:

- інформаційне моделювання:
- концептуальне моделювання (моделювання семантики ПО);
- логічне моделювання даних;
- фізичне моделювання:
- створення моделей доступу до даних;
- оптимізація фізичної організації даних в апаратному середовищі.

На рис. 1.6 ілюструється загальний зміст поняття моделі даних на цей час.

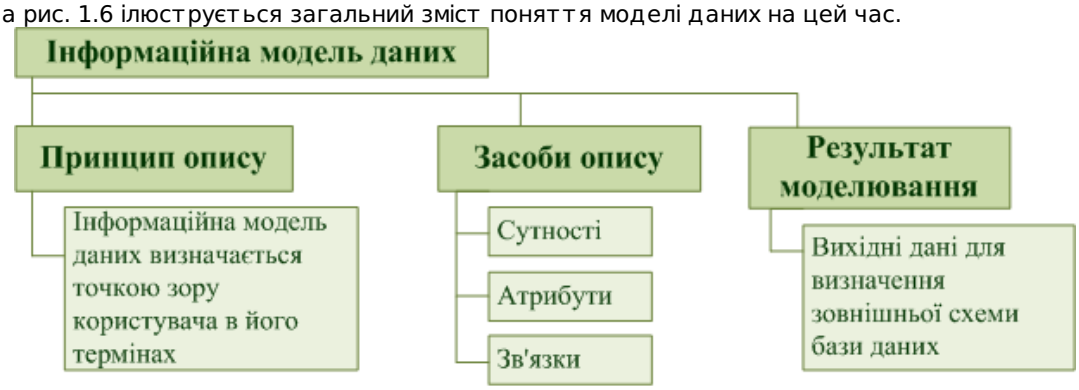


Рисунок 1.5 – Подання про інформаційну модель даних

Основні типи моделей та їх еквівалентність

Наявність у СКБД певної структури даних приводить до поняття баз структурованих даних, тобто дані в таких БД повинні бути представлені як сукупність взаємозалежних елементів. Слід мати на увазі, що для кожного типу БД використовуються відповідні моделі даних.

У цей час для баз структурованих даних розрізняють три основні типи логічних моделей даних залежно від характеру підтримуваних ними зв'язків між елементами даних - мережну, ієрархічну й реляційну. Ознаками класифікації у цих моделях є: ступінь твердості (фіксації) зв'язку, математичне подання структури моделі і припустимих типів даних (див. таблицю 1.1).

Рис. 1.7 ілюструє особливості кожної моделі даних. При зіставленні моделей варто пам'ятати, що всі вони теоретично еквівалентні. Еквівалентність моделей полягає в тому, що вони можуть бути зведені одна до одної шляхом формальних перетворень.

Таблиця 1.1 – Загальні характеристики моделей даних

одель даних	Характер зв'язків між об'єктами	Формальне подання
ережева	Напівтверді зв'язки	Довільний граф
рархічна	Тверді зв'язки	Деревоподібна структура
еляційна	Мінливі зв'язки	Плоский файл

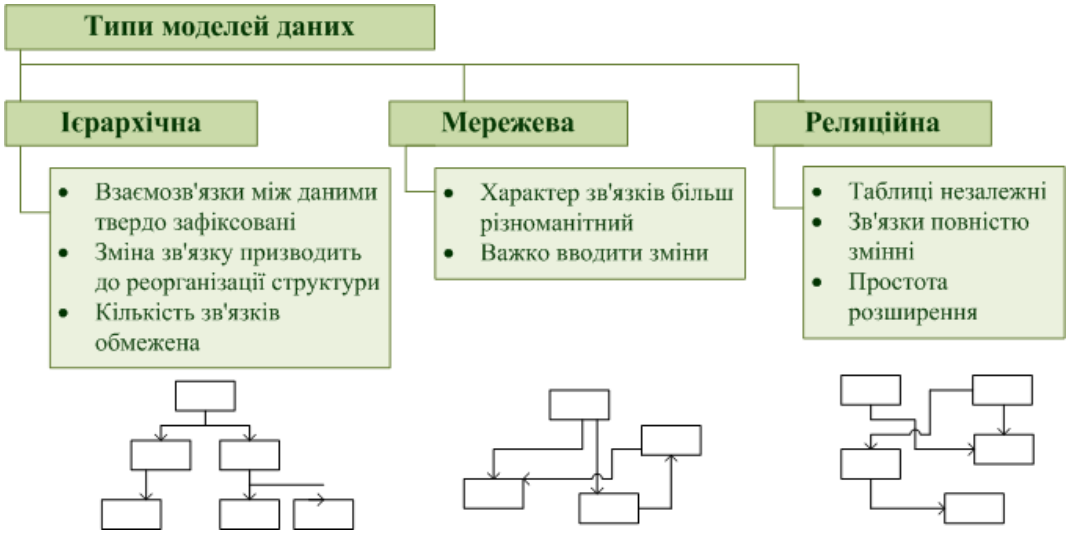


Рисунок 1.7 - Основні типи моделей даних

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

Структурно ІС містять у собі апаратне, програмне, комунікаційне, лінгвістичне й організаційно-технологічне забезпечення.

Існують три концепції обробки даних в ІС: файлової системи, баз даних, об’єктно-орієнтованих баз даних. Відповідно існують три методи обробки даних: позадачний метод, метод баз даних, об'єктно- орієнтований метод.

База даних визначається як сукупність взаємозалежних даних, які спільно використовуються декількома додатками й зберігаються з мінімальною регульованою надлишковістю. База даних складається зі всіх екземплярів записів, екземплярів наборів записів і областей, які контролюються конкретною схемою. Під схемою можна розуміти карту всієї логічної структури БД.

Системою керування базами даних називається сукупність програмних засобів, необхідних для використання БД і подання розробникам і користувачам безлічі різних подань даних. Загальноприйнятою є трьорівневі архітектура, яка має на увазі наявність трьої рівнів: зовнішнього, концептуального та внутрішнього.

1. В чому полягають основні функції інформаційної системи?
2. Як співвідносяться поняття «інформація» і «дані»?
3. Які перетворення даних ви знаєте?
4. В чому полягає функціональний зміст поняття «інформація»?
5. В чому полягає представницький зміст поняття «інформація»?
6. З яких елементів складається інформаційна система?
7. Які основні функції ІС ви знаєте?
8. Які властивості притаманні об’єкту ІС?
9. В чому полягає концепція незалежності прикладних програм від даних?
10. Основні аспекти концепції файлової системи?
11. Основні аспекти концепції баз даних?
12. Основні аспекти концепції об’єктно – орієнтованих баз даних?
13. Сформулюйте поняття «бази даних».
14. Що розуміється під системою керування базами даних?
15. Назвіть основні функції системи керування базами даних.

Тема 2 - Поняття предметної області

Ключові терміни:

бізнес-модель процесів, бізнес-правила, відношення, домен, діаграма "сутність-зв'язок", екземпляр сутності, модель життєвого циклу, модель потоку даних, набір специфікацій функцій, об'єктне ядро предметної області, предметна область, ситуації, стан предметної області, ступінь зв'язку, сутність, унікальний ідентифікатор сутності, функціональна модель, ім'я, інформаційна модель

2.1. 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.

Приклад визначення властивостей об'єкта.Сховати

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 рис. 2.1 наведений один із підходів до класифікації ситуацій у рамках ПО.

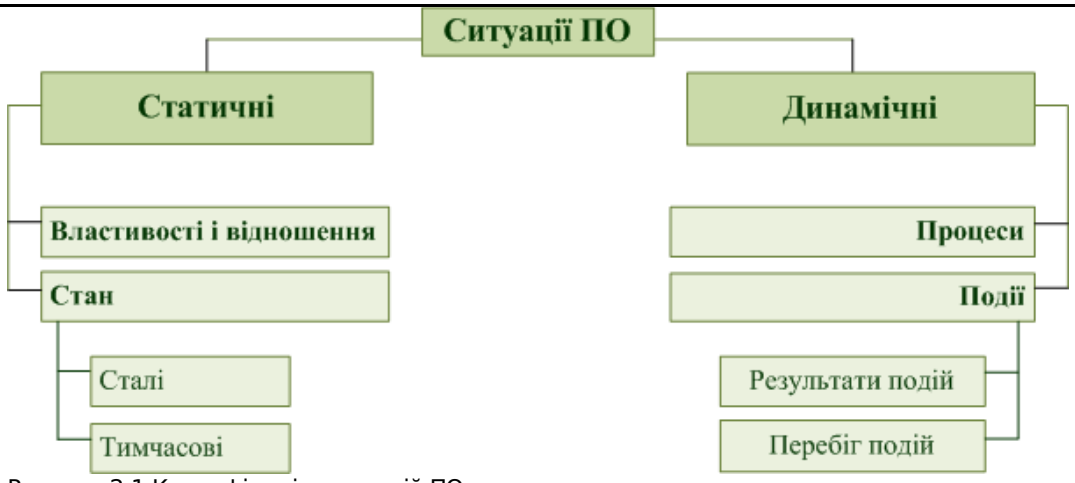


Рисунок 2.1 Класифікація ситуацій ПО

Приклади ситуаційСховати

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.

2.2 Інформаційна модель предметної області бази даних

Інформаційна модель даних призначена для подання семантики ПО у термінах суб'єктивних засобів опису - сутностей, атрибутів, ідентифікаторів сутностей, супертипів, підтипів і т.д.

Інформаційна модель ПО БД містить такі основні конструкції:

- діаграми "сутність-зв'язок" (Entity - Relationship Diagrams);
- визначення сутностей;
- унікальні ідентифікатори сутностей;
- визначення атрибутів сутностей;
- відношення між сутностями;
- супертипи й підтипи.

Елементи інформаційної моделі даних ПО є вхідними даними для вирішення завдання проектування БД - створення логічної моделі даних.

Предметом інформаційної моделі є абстрагування об'єктів або явищ реального світу у рамках ПО, у результаті якого виявляються сутності ПО. Як правило, вони позначаються іменником природної мови.

Сутність описується за допомогою даних, іменованих властивостями або атрибутами сутності. Як правило, атрибути є визначеннями у висловленні про сутності й позначаються іменниками природної мови. Сутності вступають у зв'язки один з одним через свої атрибути. Кожна група атрибутів, що описують один реальний прояв сутності, являє собою екземпляр сутності. Іншими словами, екземпляри сутності - це реалізації сутності, що відрізняються одне від одного й допускають однозначну ідентифікацію.

Одним з основних комп'ютерних засобів розпізнавання сутностей у базі даних є присвоєння сутностям ідентифікаторів. Часто ідентифікатор сутності називають ключем. Завдання вибору ідентифікатора сутності є суб'єктивним завданням. Оскільки сутність визначається набором своїх атрибутів, то для кожної сутності доцільно виділити таку підмножину атрибутів, що однозначно ідентифікує дану сутність.

Завдання розробника БД - забезпечити при збереженні екземплярів сутності у БД наявність у кожного її нового екземпляра унікального ідентифікатора. Унікальний ідентифікатор сутності - це атрибут сутності, що дозволяє відрізняти одну сутність від іншої. Якщо сутність має кілька унікальних ідентифікаторів, так званих можливих ключів, то розробник повинен обрати первинний ключ сутності.

Розрізняють однозначні й багатозначні атрибути. Однозначними є атрибути, які в межах конкретного екземпляра сутності мають тільки одне значення. У протилежному разі вони вважаються багатозначними.

Кожен атрибут сутності має домен. Домен - це вираз, який визначає значення, дозволені для даного атрибута. Іншими словами, домен - це область значень атрибута. Розробник БД повинен проконтролювати, щоб в інформаційній моделі ПО для кожного атрибута сутностей був визначений домен.

Сутності не існують окремо одна від одної. Між ними є реальні відношення, і вони повинні бути відбиті в інформаційній моделі ПО. При виділенні відношень акцент робиться на фіксацію зв'язків

і їх характеристик. Відношення (зв'язок) являє собою з'єднання (взаємовідношення) між двома або більше сутностями. Кожен зв'язок реалізується через значення атрибутів сутностей. Звичайно зв'язок позначається дієсловом. Кожен зв'язок також повинен мати свій унікальний ідентифікатор зв'язку.

Розробник БД повинен проконтролювати, щоб зв'язок між сутностями здійснювався через точно зазначені атрибути, які будуть визначати унікальний ключ зв'язку. Вибір ключів сутностей - одне з найважливіших проектних рішень, що має бути зробленим розробником при переході від інформаційної моделі ПО до логічної моделі БД.

Зв'язки характеризуються ступенем зв'язку і класом належності сутності до зв'язку. Ступінь (потужність) зв'язку - це відношення числа сутностей, що беруть участь в утворенні зв'язку. Існують такі типи: "один-до-одного", "один-до-множини", "множина-до-множини".

Типовою формою документування інформаційної моделі ПО є діаграми "сутність-зв'язок" (ER-діаграми), яка дозволяє графічно представити всі елементи інформаційної моделі згідно із простим, інтуїтивно зрозумілим, але чітко визначеним правилом - нотацією. Далі ми будемо користуватися умовними позначеннями, прийнятими в методології інформаційного проектування.

Сутність на ER-діаграмі наводиться прямокутником з ім'ям у верхній частині. Будемо використовувати англійські слова для іменування елементів моделі.

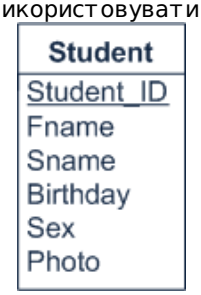


Рисунок 2.2 Подання сутності Student (студент) на ER-діаграмі з атрибутами й унікальним ідентифікатором сутності

У прямокутнику перераховуються атрибути сутності, при цьому атрибути, що становлять унікальний ідентифікатор сутності, підкреслюються.

Домени призначаються аналітиками й фіксуються в спеціальному документі - словнику даних (Data Dictionary). На стадіях розроблення логічної й фізичної моделей реляційної БД домени уточнюються у сутностях на ER-діаграмі.

Розробник БД повинен ретельно вивчити домени кожного атрибута з погляду на можливість їх реалізації у СКБД.

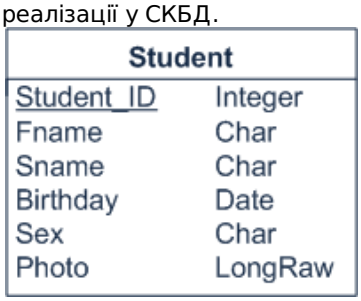


Рисунок 2.3 Візуалізація визначення доменів атрибутів на ER-діаграмі при створенні фізичної моделі реляційної БД

Відношення (зв'язок) сутностей на ER-діаграмі зображується лінією, що з'єднує ці сутності. Ступінь зв'язку зображується за допомогою символа "пташина лапка", що вказує на те, що у зв'язку бере участь багато (N) екземплярів сутності, і одинарною горизонтальною рискою, що вказує на те, що у зв'язку бере участь один екземпляр сутності.

Відношення читається вздовж лінії або ліворуч праворуч, або праворуч–ліворуч. На рис. 2.4 наведене таке відношення: кожен студент повинен бути зареєстрований за певною групою, в кожній групі може бути зареєстровано один або більше студентів.

Рисунок 2.4 Подання відношення між двома сутностями на ER-діаграмі

2.3 Концептуальна модель предметної області

Діаграма «сутності - зв’язки» виступає засобом опису схеми бази даних на концептуальному рівні проектування. Метод був запропонований у 1976 році Пітером Пін Шань Ченом. На діаграмах концептуального рівня сутності зображуються прямокутниками, атрибути – еліпсами, зв’язки –

ромбами.

Приведемо приклад концептуальної моделі предметної області обліку успішності студентів в межах кафедри. На рис. 2.5 наведений перший фрагмент концептуальної діаграми, який відображає приналежність студентів певній групі та вивчення студентами предметів.

Рисунок 2.5 Фрагмент концептуальної моделі ПО на першому етапі

Успішність засвоєння матеріалів дисциплін контролюється заліковими заходами, які визначають бал кожного студента з певної дисципліни. Отже необхідно виділити окремо сутність Statement (успішність), яка буде відображати зв'язок між сутностями Subject та Student (рис. 2.6).

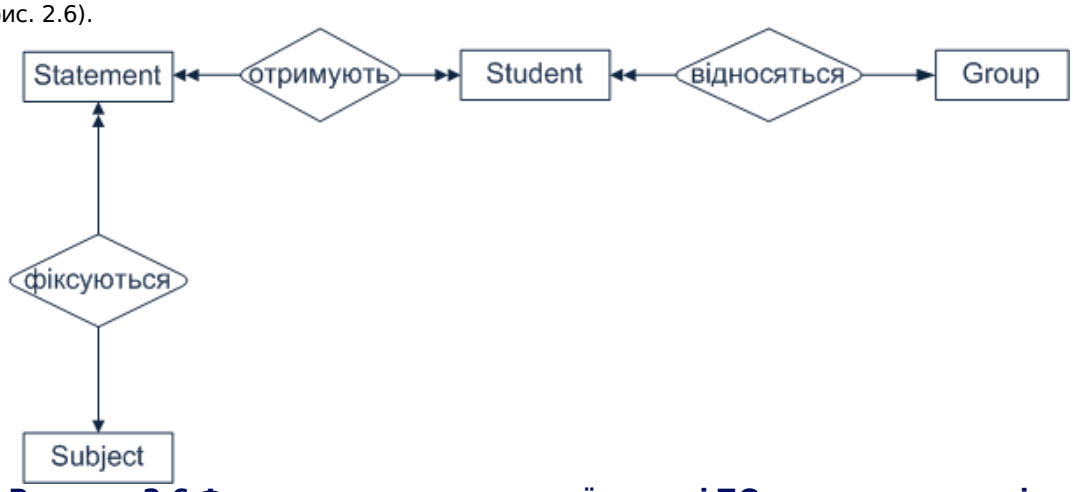


Рисунок 2.6 Фрагмент концептуальної моделі ПО на другому етапі

На діаграмі присутній один зв'язок «множина-до-множини», який є недопустимим в реляційній БД. Давайте проаналізуємо: кожен студент проходить контрольні заходи з декількох предметів і кожен предмет вивчається декількома студентами. Кожен студент навчається в одній групі і в кожній групі навчаються декілька студентів. Відомість на контрольний захід видається на групу студентів. Отже заміна сутності Student у зв'язку «отримують» на сутність Group вирішує конфліктну ситуацію. Кожен об'єкт БД має свої атрибути. Отже приходимо до такого вигляду фрагменту концептуальної діаграми ПО успішність, яка наведена на рис. 2.7.

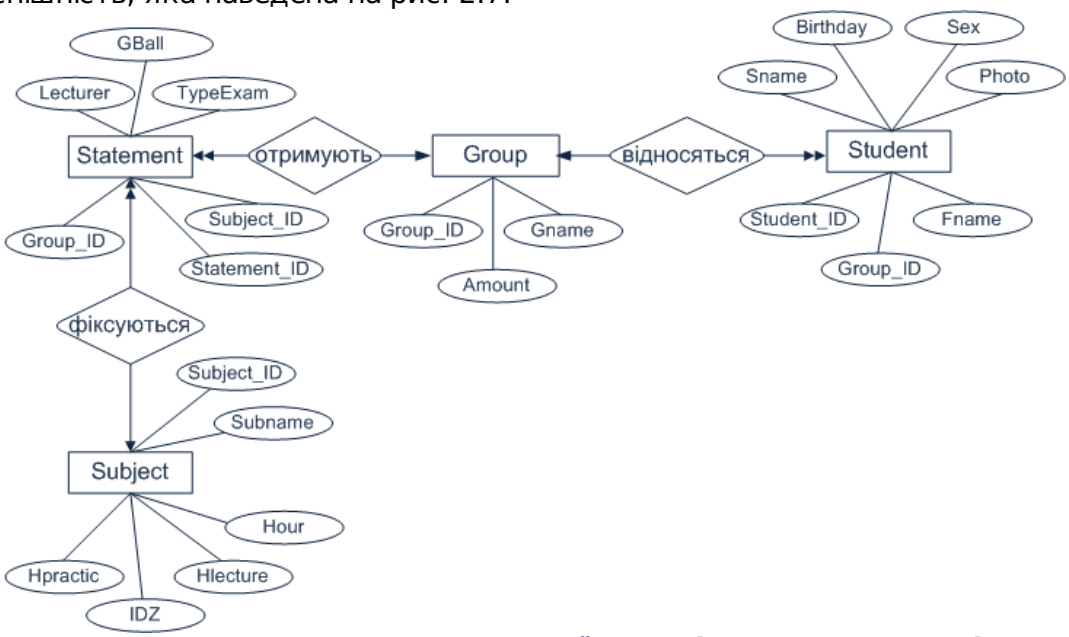


Рисунок 2.7 Фрагмент концептуальної моделі на третьому етапі

Взагалі, якщо досліджується достатньо об'ємна ПО, раціональним є розроблення декількох локальних концептуальних моделей при подальшому їх об'єднанні у більш складну загальну концептуальну модель. Локальні моделі формуються таким чином, щоб кількість об'єктів (сутностей) не перевищувала 6-7 штук.

2.4 Функціональна модель предметної області

Третім ключовим моментом створення ІС з метою автоматизації інформаційних процесів організації є аналіз функціональної взаємодії об'єктів автоматизації. Аналітики наводять результати у вигляді функціональної моделі ПО БД. Склад функціональної моделі істотно залежить від контексту конкретного ІТ-проекту і може бути представлений за допомогою досить широкого спектра документів у вигляді текстової і графічної інформації.

Визначимо функціональну модель ПО БД як сукупність деяких моделей, призначених для опису процесів обробки інформації. Будемо називати ці моделі конструкціями функціональної моделі. Нижче наведений перелік основних конструкцій функціональної моделі, які необхідні для виконання проектування реляційних БД.

Моделі процесів:

- бізнес-модель процесів (ієрархія функцій системи);
- модель потоку даних.

Моделі станів:

- модель життєвого циклу сутності;
- набір специфікацій функцій системи (вимоги);
- опис функцій системи через сутності й атрибути;
- бізнесу-правила, які реалізують функції.

2.4.1. Бізнес-модель процесів.

Бізнес-модель процесів призначена для опису процесів і функцій системи в ПО БД. Частіш за все, бізнес-модель документується відповідно нотаціям IDEF0 і подається у вигляді сукупності ієрархічно впорядкованих та взіємопов’язаних діаграм. Діаграми містять такі компоненти:

- контекстна діаграма;
- діаграма декомпозиції;
- діаграма дерева вузлів;
- діаграма «тільки для експозиції».

Контекстна діаграма є вершиною ієрархічної структури діаграм і є узагальненим описом системи та її взаємодії з зовнішнім середовищем.

Подальший опис системи будується послідовним розбиттям функціональної системи на більш детальні фрагменти – діаграми декомпозиції.

Діаграма дерева вузлів передає ієрархічну структуру функцій без відображення взаємозв’язку між ними.

Діаграми «тільки для експозиції» подають копії стандартних діаграм без синтаксису моделі. Вони призначені для демонстрації інших точок зору на роботи, для відображення деталей, які не підтримуються синтаксисом моделі у явному вигляді.

Основними елементами IDEF0-діаграм є роботи, вхідні ти вихідні параметри, керування, механізми та виклик. Приклади діаграм будуть наведені у наступній темі на прикладі аналізу процесу проектування бази даних.

2.4.2. Модель потоку даних.

Модель потоку даних призначена для опису процесів переміщення даних в ПО БД і подається у вигляді діаграми потоку даних (Data Flow Diagram). Основними елементами діаграми є:

- джерела даних (Data Source);
- процеси обробки даних (Data Process);
- сховища даних (Data Store);
- потоки даних (Data Flow).

Джерела даних вказують, хто користується або працює с даними. Процеси обробки вказують на операції, що відбуваються над даними. Сховища даних вказують на місця збереження даних. Потоки даних вказують на спосіб передачі даних між джерелами та сховищами даних.

Для подання діаграм потоку даних частіш за все використовують мережеві структури, які дозволяють дублювання сутностей та відсутність циклів. Потік зображується зліва направо. На діаграмах помічають допустимі та недопустимі напрямки переміщення даних без зазначення процесів керування потоком.

Діаграма потоку даних дозволяє:

- подати систему з точки зору джерел та користувачів даних;
- відобразити переміщення даних в процесі обробки;
- відобразити зовнішні механізми передачі даних;
- відобразити метод отримання даних.

Діаграма потоку даних надає проектувальнику інформацію про:

- сховища даних, що дозволить на наступних етапах проектування обґрунтовано визначити кількість БД для ІС;
- прийняті схеми перетворення інформації, що дозволить в подальшому скласти специфікації додатків.

2.4.3. Модель життєвого циклу.

Модель життєвого циклу (ЖЦ) сутності призначена для опису зміни станів сутності та переходів між ними. Подається у вигляді діаграм ЖЦ сутності (Entity Lifecycle Diagram), які відображають напрямки переходу сутності з деякого початкового стану до кінцевого стану і події, що ініціюють зміни станів сутності. Модель ЖЦ також може бути подана у текстовому вигляді опису.

2.4.4. Набір специфікацій функцій системи (вимог), опис функцій через сутності і атрибути, бізнес-правила.

Аналітики повинні подати проектувальникам БД набір специфікацій функцій – опис функціональність системи в формі чітко сформульованих бізнес-категорій, згрупованих за напрямками діяльності організації. Іноді подається перелік залежностей між функціями та подіями, що їх викликають.

Текстовий опис функції повинен містити виокремленні сутності та атрибути,а також однозначно інтерпретуватись при читанні.

Після отримання набору специфікацій функцій та опису через атрибути та сутності проектувальник БД може почати розроблення специфікацій модулів додатків БД.

Бізнес-правила подають конкретні вимоги та умови для функцій, які визначають поведінку даних, і використовуються для підтримки цілісності даних в БД.

Зробимо висновки:

- забезпечити користувача інформацією про зовнішній світ оперативно можна шляхом реалізації питально-відповідного відношення про предметну область. Існують декілька моделей ПО;
- інформаційна модель даних призначена для подання семантики ПО у термінах суб'єктивних засобів опису - сутностей, атрибутів, ідентифікаторів сутностей, супертипів, підтипів і т.д.;
- основні принципи реляційної БД на концептуальному рівні можна сформулювати так: всі дані подаються у вигляді впорядкованої структури рядків і стовпців, що називається відношенням; всі значення є скалярами, тобто для будь-якого рядка і стовпця будь-якого відношення існує одне і тільки одне значення; всі операції виконуються над цілим відношенням і результат операції є також ціле відношення. Цей принцип називається замиканням;
- функціональна модель призначена для опису процесів обробки даних у рамках виділеної ПО з різних точок зору.
- елементи інформаційної моделі ПО є вхідними даними для завдання створення логічної моделі даних. Елементи функціональної моделі ПО є вхідними даними для завдання проектування додатків БД і, частково, для завдання створення фізичної моделі БД.

1. В чому полягає призначення ІС?
2. Дайте визначення поняття «предметна область»?
3. Які моделі ПО ви знаєте?
4. Що містить у собі об’єктне ядро ПО?
5. Як формується ім’я об’єкту?
6. Що таке ситуації? Класифікація ситуацій ПО.
7. Дайте визначення інформаційній моделі ПО.
8. Які основні конструкції інформаційної моделі ви знаєте?
9. Дайте визначення сутності?
10. Як ідентифікується сутність в БД?
11. Що є відношенням БД?
12. Що таке зв'язок в БД? Чим характеризується зв'язок?
13. Які форми подання інформаційної моделі ви знаєте?
14. Що відображає концептуальна модель ПО?
15. Що таке функціональна модель ПО?
16. Які основні конструкції функціональної моделі ви знаєте?

Тема 3 - Проектування бази даних

Ключові терміни:

логічне проектування, методологія проектування баз даних, проектування бази даних, фізичне проектування

3.1 Життєвий цикл та методологія проектування

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 (рис. 3.1)

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 3.1 – Етапи життєвого циклу БД

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.

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.).

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).

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.

3.2 Етапи проектування БД

3.2.1 Визначення стратегії

Метою етапу визначення стратегії є формування спільно з замовником прикладних моделей, вироблення переліку рекомендацій і ухвалення узгодженого плану, складеного з урахуванням 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 розроблення перспективного плану створення системи, але витрати на його проведення навряд чи будуть економічно виправданими. Як правило, стратегія розроблення інформаційної системи визначається в результаті узагальненого аналізу, на підставі якого потім будується великомасштабна модель прикладної області. Стратегія має визначатися в достатньо стислі терміни з тим, щоб результати проектування не втрачали актуальності.

Результати цього етапу мають узгоджуватися одне з одним і бути достатньо чітко сформульовані, щоб замовник міг легко співвіднести запропоновану стратегію зі своїми завданнями і зрозуміти, які саме чинники обумовили ухвалення тих чи інших рішень. Окрім того, йому має бути викладена перспектива подальшого аналізу, уточнення й перегляду стратегічних рішень.

Результати. Основними результатами цього етапу мають бути

- опис напрямів прикладної діяльності, зокрема формулювання її цілей і завдань, визначення пріоритетів, обмежень, критичних чинників успіху та ключових показників ефективності;
- опис цілей і завдань автоматизації, витрат і можливого виграшу;
- узагальнена діаграма сутностей і зв’язків;
- узагальнена ієрархічна схема завдань (виробничих та управлінських);
- рекомендації щодо майбутньої реалізації та подолання можливих труднощів;
- визначення меж і окреслення сфери застосування системи баз даних;
- можлива архітектура системи;
- поетапний план проектування бази даних

Такий підхід до моделювання предметної області передбачає її відображення з трьох різних точок зору:

- загальний напрям прикладної 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 й завдання з часом можуть змінюватися, відтак неможливо запропонувати директивний метод моделювання стратегій. Тому важливо поєднувати неупередженість до нових рішень зі здатністю швидко оцінювати альтернативні напрямки діяльності з урахуванням заданих обмежень і пріоритетів.

Ключові чинники успіху:

- використання всіх можливих засобів, що дають змогу підвищити рівень знань про предметну область;
- активна участь у розробленні стратегії осіб, які добре розуміють справжні потреби організації;
- проведення плідних нарад із ретельним розглядом усіх питань

3.2.2 Аналіз предметної області

Підсумки етапу визначення стратегії є вихідними даними для етапу аналізу, де вони ретельно перевіряються, уточнюються і деталізуються, для того щоб забезпечити предметній області адекватність моделі, гарантувати можливість реалізації рішень і сформувати тверде підґрунтя для етапів концептуального моделювання, логічного й фізичного проектування.

Цей етап є найменше вивченим, найважчим і найтривалішим. Проте він найважливіший, оскільки саме на ньому формується більшість проектних рішень

Опис. Аналіз предметної області складається з аналізу даних та аналізу завдань. Аналіз даних передбачає документування всіх атрибутів. Аналіз завдань може потребувати застосування різноманітних методів побудови діаграм для дослідження зв'язків і способів використання даних, подій, станів даних, а також детального опису алгоритмів.

Вивчається потреба в заходах із контролю та захисту даних, їхньому резервному копіюванні та відновленні. Має бути проведений детальний аналіз наявних систем та інших чинників, що впливають на процес впровадження системи. Потрібно виявити всі обмеження і припущення, що можуть вплинути на подальше проектування, використання ресурсів і терміни проведення робіт.

Підхід. На цьому етапі аналітики й користувачі працюють пліч-о-пліч, встановлюючи й перевіряючи вимоги. Аналіз предметної області передбачає:

- проведення бесід з користувачами;
- перегляд усіх документів та бланків, які обробляються і формуються організацією;
- аналіз потоків документів;
- аналіз способів вирішення завдань організації;
- фіксація правил, обмежень та законів, що діють у предметній області.

Результати:

- узгоджена діаграма сутностей і зв’язків;
- відомості про обсяги даних, частоту виконання завдань, очікуваний користувачем рівень продуктивності;
- деталізовані й узгоджені описи завдань:
- первинний варіант стратегії впровадження;
- опис заходів з ревізії і контролю даних, резервного копіювання й відновлення;
- загальний опис процедур, що не автоматизуються;
- критерії прийнятності, якості, гнучкості та продуктивності;
- попереднє оцінювання обсягів системи;
- узгоджений підхід до здійснення етапу проектування й фази реалізації;
- уточнений план розроблення системи.

Ключові чинники успіху:

- активна участь користувачів;
- ретельна перевірка достовірності, повноти й несуперечності даних;
- виявлення всіх питань та припущень, що мають ключове значення для проектування і впровадження;
- встановлення точних характеристик ключових завдань і даних; жорсткий контроль за ходом робіт, концентрація зусиль на виконанні календарних планів і дотриманні запланованих термінів

3.2.3 Концептуальне моделювання предметної області

Етап концептуального моделювання полягає в побудові опису предметної області в термінах формальної мови, наприклад у термінах моделі сутностей і зв’язків. Ідеї побудови концептуальної моделі предметної області беруть свій початок із публікації робочої групи ANSI/SPARC, присвяченій архітектурі СКБД.

Опис. На базі змістовного опису предметної області, отриманого в результаті її аналізу, розроблюється строгий формальний опис її інформаційного забезпечення.

Результати:

- формальний опис інформаційного забезпечення предметної області:
- докладний і строгий опис сховищ даних;
- детальний опис потоків даних;
- детальний опис ієрархії й специфікація завдань, що вирішуються; детальний опис чинних у предметній області правил і обмежень.

Ключові чинники успіху :

- глибоке знання і практичний досвід використання мов опису концептуальної моделі,
- знання методів проектування реляційної моделі та/або інших моделей даних.

3.2.4 Логічне та фізичне моделювання даних

Етап проектування полягає в пошуку і визначенні якнайкращого способу реалізації та виконання вимог, сформульованих на етапі аналізу. При цьому має забезпечуватись належний рівень сервісу в умовах певного технологічного середовища, що відповідає ухваленим рішенням щодо рівня автоматизації.

Логічне проектування — це розроблення структур зберігання, методів доступу й логічної структури системи баз даних без прив'язки до конкретної СКБД.

Фізичне проектування — це проектування бази даних у конкретній СКБД.

Опис. Під час виконання цього етапу модель сутностей і зв'язків перетворюється на схему бази даних і специфікації зберігання даних на зовнішніх носіях. Прикладні задачі перетворюються на модулі й процедури з необхідними засобами ревізії/контролю та резервного копіювання й відновлення. Проектуються формати звітів, визначаються міжмодульні зв'язки. Виходячи із завдань, сформульованих на попередніх етапах, створюються проектні рішення з архітектури комунікаційної мережі. Для полегшення процесу пошуку проектних рішень можуть застосовуватися прототипи. Нарешті, на етапі проектування розроблюються програмні специфікації і план тестування системи, а отримана інформація й нові погляди на майбутню систему застосовуються для доопрацювання та уточнення стратегії її реалізації.

Підхід. Аналітики, розробники і проектувальники баз даних спілкуються з користувачами менше, ніж на етапі аналізу, проте вони повинні надати їм для перевірки результати своєї роботи або запропонувати на вибір різні варіанти проектних рішень. Корисним є створення прототипів, проте воно має розглядатися лише як засіб, а не самоціль.

Результати:

- архітектура системи:
- схеми системних модулів:
- логічна і фізична схеми;
- схема бази даних і файлів;
- детальні часові та ємнісні характеристики;
- програмні специфікації;
- специфікації неавтоматизованих процедур;
- чорновий варіант посібника для користувача;
- узгоджена стратегія впровадження, що складається з планів приймання і здачі системи, організаційної підготовки, заходів зі збирання даних, переходу на нову систему та встановлення обладнання;
- план випробувань системи;
- чорновий варіант експлуатаційної документації;
- уточнений план розроблення системи.

Ключові чинники успіху. В результаті виконання даного етапу має бути створений проект, що забезпечує задоволення прикладних вимог з урахуванням наявних технічних обмежень. Ключовими чинниками успіху в досягненні цієї мети є:

- знання можливостей апаратного й програмного забезпечення;
- розуміння прикладних потреб;
- ухвалення обґрунтованих компромісних рішень;
- виявлення і вирішення потенційних проблем.

Бізнес-модель процессу проектування бази даних

Процес проектування БД може бути представлений у вигляді моделі бізнес-процесів. Бізнес-модель процесу проектування дозволяє:

- відобразити суб'єктивну думку розробника БД на процес проектування конкретної БД;
- врахувати особливості IT-проекту, у рамках якого проектується БД;
- досить швидко скласти план проектування конкретної БД;
- прорахувати тривалість проектних робіт (створити тимчасову модель проектування).

3.3.1. Типова бізнес-модель процесу проектування бази даних

Значна частина проектів у сфері інформаційних технологій спрямована на розроблення й створення ІС, у рамках яких здійснюється обробка даних різної складності. Практично у всіх таких проектах вирішується завдання проектування БД певного типу.

В експлуатації БД повинна задовольняти набір вимог щодо ряду інтегрованих параметрів, таких, як:

- функціональність й адаптованість;
- продуктивність обробки трансакцій;
- пропускна здатність;
- час реакції;
- безпека.

Такі параметри іноді перебувають у протиріччі один з одним. Так, високі вимоги до функціональності на даному конкретному устаткуванні можуть вступати у конфлікт з високими вимогами до продуктивності. Наприклад, звіти можуть генеруватися протягом декількох годин і знизити в цей час реакції користувачів, що працюють із системою у діалоговому режимі.

Етап проектування БД вважається одним із найскладніших етапів створення БД, який не має явно вираженого початку й закінчення. У порівнянні з аналізом вимог до БД або розробленням 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.

Розглянемо типову бізнес-модель процесу проектування БД. На рис. 3.2 наведена контекстна діаграма процесу проектування БД.

Як видно з рисунка 3.2, на вхід процесу проектування БД подаються:

- інформаційна модель ПО БД: діаграми "сутність-зв'язок" (ER-діаграми);
- функціональна модель ПО БД: бізнес-модель процесів, діаграми потоку даних (DF-діаграми), діаграми станів, - діаграми життєвих циклів сутностей, специфікації на системи (вимоги), бізнес-правила;
- загальносистемні вимоги й обмеження;
- завдання зворотного впливу.

На виході процесу проектування БД формуються такі результати:

- фізична модель БД, що може бути перетворена у скрипт для створення БД;
- фізична БД;
- специфікація модулів додатків БД;
- план тестування БД.

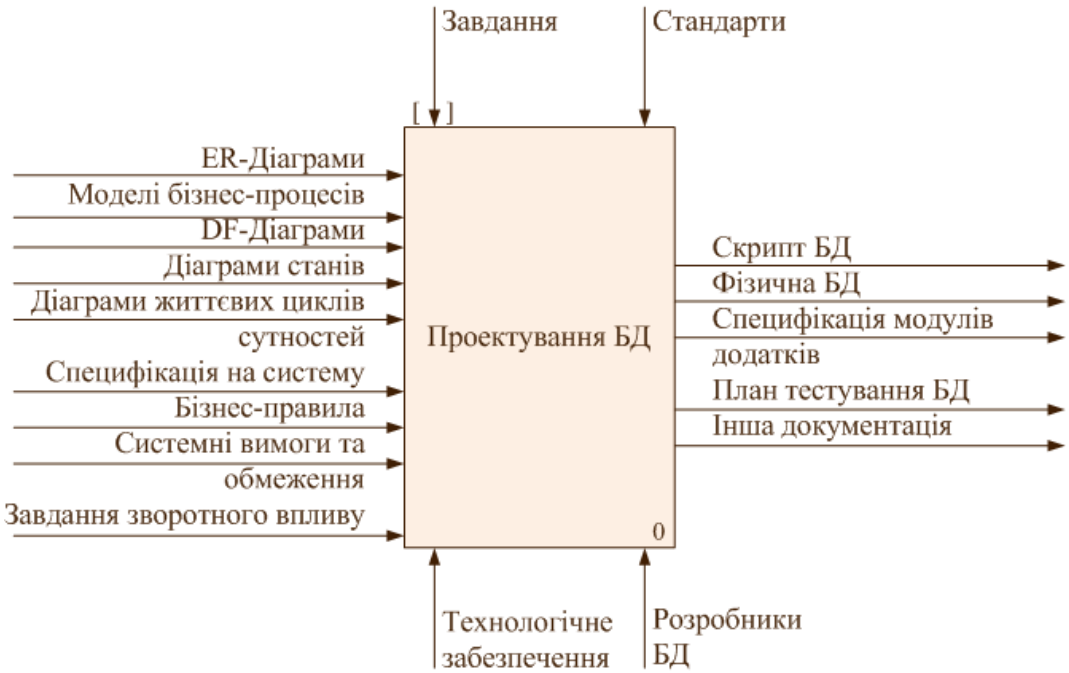


Рисунок 3.1 – Контекстна діаграма процесу проектування БД

3.3.2. Діаграма декомпозиції першого рівня

Починаючи функціональну декомпозицію процесу проектування БД, приходимо до діаграми декомпозиції процесу проектування БД першого рівня, яка відбиває основні, найбільш великі професійні завдання (етапи) проектування БД (рис. 3.3).

Такими завданнями (етапами) є:

- збір і аналіз вхідних даних – це початковий етап проектування, на якому здійснюються збір і контроль якості результатів аналізу ПО БД, готується план проектування БД;
- створення логічної моделі БД – це етап, на якому на підставі інформаційної моделі ПО БД створюється логічна структура БД, незалежна від її реалізації;
- створення фізичної моделі БД: внутрішня схема – це етап, на якому на підставі логічної моделі БД створюється фізична структура БД, залежна від її реалізації. На цьому етапі виконується перетворення відношення логічної моделі реляційної БД у команди створення об'єктів фізичної БД, у результаті чого створюється так звана внутрішня схема БД. Додатково може бути створена так звана зовнішня схема БД, останнє відбиває точку зору користувачів на дані в БД;
- створення фізичної моделі БД: урахування впливу транзакцій – це етап, на якому аналізуються можливі транзакції системи, за потреби виконується денормалізація відношення для забезпечення більш високої продуктивності БД;
- створення серверного коду – це етап, на якому на підставі функціональної моделі ПО БД створюється серверний код БД у вигляді тригерів, збережених процедур і пакетів. Ці модулі створюються розробником БД і виконуються сервером;
- проектування модулів додатків БД – це етап, на якому створюються специфікації модулів додатків, розробляються стратегії тестування БД і додатків, створюється план тестування додатків БД і готуються тестові дані;
- контроль якості проектування БД полягає в перевірці якості результатів проектування на кожному його етапі;
- урахування завдань зворотного впливу полягає у настроюванні деяких трансакцій до БД і локальному перепроектуванні БД відповідно до вимог, що надходять з інших етапів створення БД.

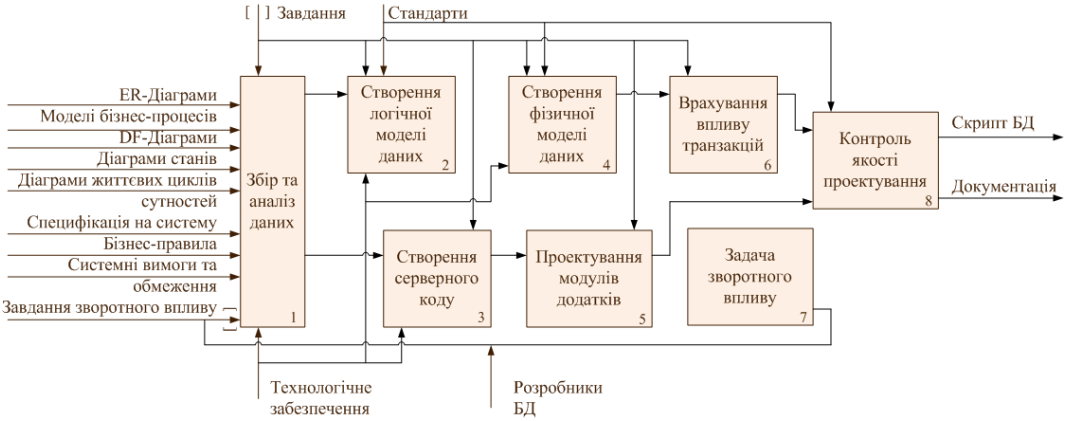


Рисунок 3.2 – Діаграма декомпозиції процесу проектування БД: перший рівень

Таким чином, можемо зробити висновок, що процес проектування БД полягає у досягненні компромісів між функціональними, інформаційними, апаратними, архітектурними й технологічними вимогами до БД і будується на інформованому прийнятті рішень за структурою БД. Проектування починається з моменту прийняття стратегічних рішень і триває на етапах реалізації й тестування.

Наведена у лекції бізнес-модель процесу проектування БД являє собою досить простий типовий приклад бізнесу-моделі проектування. У загальному випадку зміст бізнес-моделі проектування залежить від багатьох факторів: особистості менеджера й состава команди проекту, обсягу проекту, проектних ризиків і т.д.

Основною метою етапу створення логічної моделі БД є перетворення інформаційної моделі ПО БД у логічну модель реляційної БД, результатом є нормалізована схема відношень БД

Основна мета завдання розроблення внутрішньої схеми БД є перетворення логічної моделі реляційної БД у послідовність команд SQL для створення об'єктів реляційної БД. В результаті виконання обліку впливу транзакцій розробник БД створює фізичну модель БД, що враховує характер обробки даних у БД.

1. В чому полягає процес проектування БД?
2. Назвіть основні етапи процесу проектування БД.
3. В чому полягає мета процесу аналізу вхідних даних?
4. Назвіть основні етапи процесу аналізу вхідних даних.
5. В чому полягає мета процесу розроблення логічної структури БД?
6. Назвіть основні етапи процесу розроблення логічної структури.
7. В чому полягає мета процесу розроблення внутрішньої схеми БД?
8. Назвіть основні етапи процесу розроблення внутрішньої схеми БД.
9. Які основні етапи життєвого циклу бази даних вам відомі?

Тема 4 - Проектування модулів додатків

- Аналіз функціональної моделі предметної області бази даних
- Визначення функцій
- Відображення функцій у модулі
- Системні модулі
- Розміщення логіки обробки
- Загальні принципи розроблення специфікацій модулів

Ключові терміни:

деструктор, конструктор, системний модуль, таблиця "Функція-Сутність"

Аналіз функціональної моделі предметної області бази даних

Вхідними даними для вирішення завдання проектування модулів додатків БД є ієрархія функцій. На виході розробник повинен отримати опис (специфікацію) модулів додатків, а в процесі проектування модулів розробник будує відображення бізнес - вимог у специфікації модулів.

Алгоритм дій розробника БД полягає у наступному: спочатку розробник намагається сформулювати бізнес-вимоги (функції) у самому загальному вигляді, а потім виконує декомпозицію кожної такої бізнес-функції доти, поки не буде отримана деяка функція, яку можна вважати атомарною функцією.

Розглянемо методологію проектування на прикладі.Сховати

Розглянемо фрагмент ієрархії функцій для обробки заяв про виплату страхового відшкодування. На спрощеній схемі рис. 4.1 показана функція "2.Обробити заяву". Виконання цієї функції включає виконання чотирьох функцій наступного рівня: "2.1. Зареєструвати заяву", "2.2. Ухвалити рішення щодо заяви", "2.3. Здійснити платіж за заявою", "2.4. Закрити заяву".

На рис. 4.1 показана подальша декомпозиція функції "2.2. Ухвалити рішення щодо заяви". Отримана на цьому етапі функція "2.2.5. Дозволити ремонт" є атомарною функцією. Ремонт дозволяється або не дозволяється.

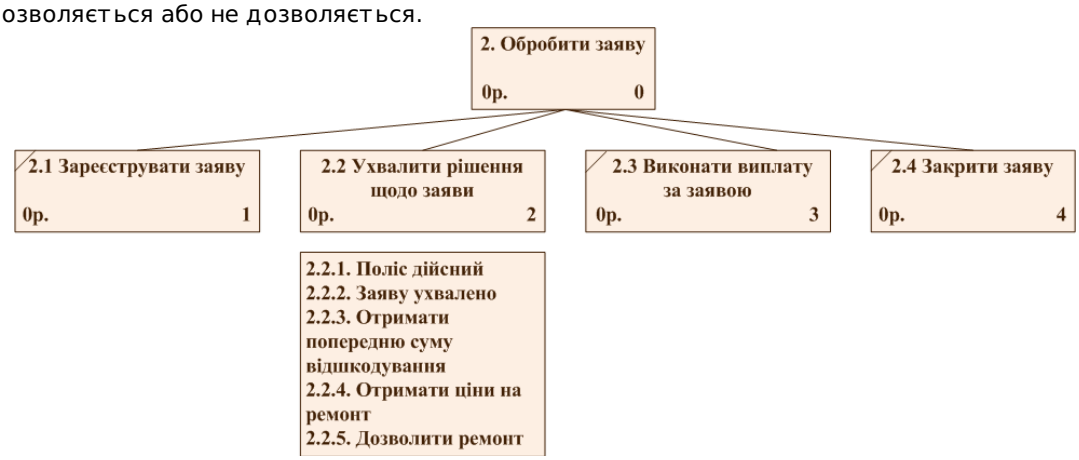


Рисунок 4.1 Ієрархія функції для обробки заяв про виплату страхового відшкодування

При розгляді ієрархії функцій розробникові БД варто звернути увагу на такі моменти:

- у функціональній моделі БД описуються бізнес-функції, і не всі вони будуть безпосередньо підтримуватися додатком БД;
- при розгляді ієрархій нерідко виникає ситуація, коли екземпляри однієї й тієї ж функції будуть мати різні номери.

Якщо в першому випадку додаткову інформацію про те, які бізнес-функції будуть реалізовані в системі, можна одержати від керівника проекту, то в другому випадку розробник БД, найімовірніше, має справу з помилкою аналітика у визначенні функції.

Визначення функцій

При розробці ієрархії функцій аналітик повинен надати текстовий опис до кожної функції, принаймні для верхнього й самого нижнього рівнів ієрархії. Бажано, щоб у цьому описі аналітики виділяли сутності ПО. Це важливо для того, щоб знати з якими сутностями ПО працює функція, тобто які потенційні об'єкти реляційної БД будуть використовуватися в кожній функції. Якщо це не зроблено, то розробникі БД будуть вимушені робити це самостійно.

ПрикладСховати

Визначення функції "2.2.2. Перевірити чи ухвалена заява".

"*Одержати й зареєструвати всі необхідні страховою компанією відомості про заяву (ВІДОМОСТІ ПРО ЗАЯВУ), включаючи всі докладні відомості про треті сторони (СТОРОННІ ЮРИДИЧНІ ОСОБИ) і свідках (ФІЗИЧНІ ОСОБИ).*

Вивчити страховий поліс (ПОЛІС) на предмет наявності виняткових ситуацій (ВИКЛЮЧЕННЯ) і визначити, чи діють ці ситуації у випадку даної заяви (ЗАЯВА).

Якщо є виключення, то закрити заяву й скласти стандартний лист заявникові про відмову у виплаті (ЛИСТ) заявникові (ЗАЯВНИК).

Якщо ніяких виключень ні, то змінити статус заяви на очікування оцінки, призначити й повідомити оцінювача (ОЦІНЮВАЧ)."

Із приклада видно, які сутності ПО беруть участь у виконанні функції (виділені в дужках), як міняється стан сутності (виділено курсивом) і який алгоритм роботи цієї функції.

Із приклада зрозуміло, що на цьому етапі розробник БД у якості вхідних даних використає також інформаційну модель ПО БД (опис сутностей).

При виконанні аналізу функцій корисно мати деяку таблицю (матрицю) "Функція-Сутність", яка повинна дати відповідь на наступні питання:

- чи має кожна сутність конструктор (функцію, що створює всі екземпляри сутності);
- чи має вона деструктор (функцію, що видаляє екземпляри сутності);
- чи є посилання на цю сутність (функції, які використають цю сутність й який образ).

Процес аналізу взаємодії функції й сутності прийнято позначати абревіатурою CRUD (Create, Reference, Update, Delete - створення, посилання, модифікація, видалення).

Корисними для розуміння розробником бази дані призначення функцій і того, як дані функції беруть участь у процесі обробки даних у системі, можуть бути діаграми потоку даних і діаграми життєвих циклів сутностей, які були розглянуті нами в другій лекції. Останні, зокрема, дають ясну картину зміни стану сутності, що важливо у визначенні атрибутів статусу сутності.

Відображення функцій у модулі

Одним з основних завдань проектування модулів додатків є побудова відображення функцій у модулі. При вирішенні цього завдання розробник БД повинен акцентувати увагу на структурі БД, що становить основу додатка.

Як правило, вирішення завдання відображення функцій у модулі виконується в чотири етапи:

1. Аналіз роботи функції.
2. Побудова моделі сутностей, що підтримує ці функції.
3. Почати проектування фізичної структури зі створення схеми, що підтримує розроблену модель сутностей.
4. Завершити проектування розробкою специфікацій модулів, які реалізують функції на запропонованій схемі БД.

Із запропонованого вище підходу видно, як тісно переплітаються у процесі проектування процеси розробки фізичної моделі БД і специфікацій модулів додатків. Таким чином, якщо розробником був розроблений чорновий варіант фізичної моделі БД по алгоритмах, розглянутим нами в попередніх лекціях, то на цьому етапі він повинен бути адаптований до реалізації функцій й, можливо, значно перероблений.

При відображенні функцій у модулі необхідно отримати схему, що ставить у відповідність кожної функції певний модуль.

ПрикладСховати

Розглянемо БД, що містить інформацію про співробітників, відділи й проекти організації. Припустимо, вона буде підтримувати бізнес-функцію "Керування проектами в організації". Функціональна модель ПО БД у термінах ієрархії функцій наведена на рис. 4.2, а на рис. 4.3 наведений перелік функцій керування проектами в організації.

Завдання полягає у відображенні функцій з переліку на рис. 4.3 у перелік модулів. Спочатку з переліку функцій повинні бути вилучені ті функції, які не будуть підтримуватися додатком БД. Розробник довідається в керівника проекту, що в додатку БД не будуть підтримуватися наступні функції:

- призначити куратора проекту; сповістити керівників підрозділів;
- сповістити співробітників; зібрати нарада;
- приступитися до виконання; скласти список робіт;
- визначити обсяг робіт; визначити вартість робіт;
- визначити час робіт; визначити виробничі потужності;
- розподілити виробничі потужності; розподілити роботи зі співробітників;

- контролювати хід виконання проекту.

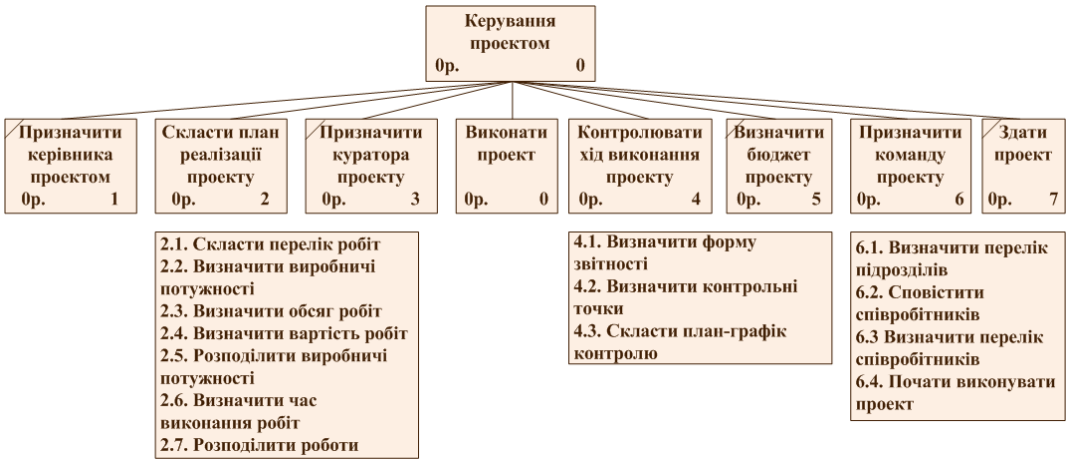


Рисунок 4.2 Ієрархія бізнес-функції "Керування проектами в організації"

Рисунок 4.3 Перелік функції керування проектами в організації

Таким чином, буде отриманий перелік функцій, що показаний у лівій колонці таблиці 4.1. Цьому переліку функцій повинен бути поставлений у відповідність перелік модулів додатка БД.

Таблиця 4.1 – Переліки функцій і модулів

Функції	Модуль
Призначити керівника проекту	Введення інформації про проект
Визначити бюджет проекту	Введення інформації про співробітників
Визначити список підрозділів	Пошук інформації про співробітників
Визначити список співробітників	Пошук інформації про проекти
Виконувати проект	Генерація звіту про виконані проекти
Здати проект	Генерація звіту про виконувані проекти

Керівник проекту передав розробникові БД характеристику додатки БД по керуванню виконанням проектів в організації. Цей додаток буде займатися обліком виконуваних і виконаних проектів в організації.

Розробник БД повинен встановити відображення функцій у модулі, як показано на рис. 4.4.

Наведений приклад показує загальний принцип побудови відображення бізнес-функцій у модулі.

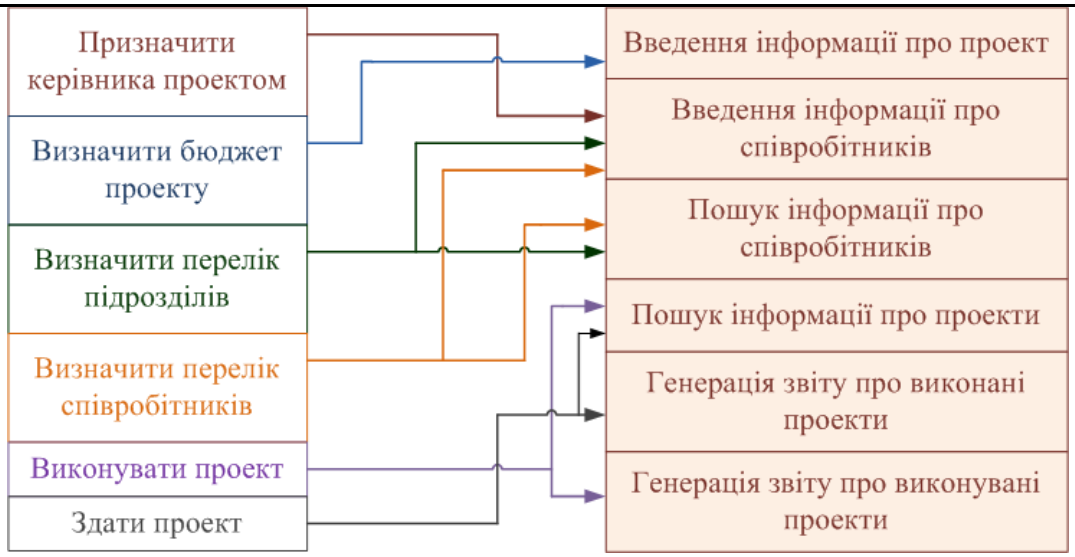


Рисунок 4.4 Відображення функції в модулі

Системні модулі

Під час реалізації функціональних можливостей необхідно розглянути достатню кількість процесів, які не є наявними у сформульованих бізнес-функціях. Наприклад, можливість видачі результатів роботи модуля в файл або на принтер.

Набір модулів додатків БД, який не слідує явно з бізнес-функції, але є необхідним для забезпечення роботи системи, називається системним. До таких модулів можна віднести процедури резервного копіювання та автоматичного відновлення, модулі зміни паролів, модулі друку та навігації по додаткам БД та інш.

Розробник БД самостійно вирішує питання розроблення специфікації системних модулів.

Розміщення логіки обробки

Під час розроблення серверного коду необхідно дотримуватись простих правил:

- задіювати найбільшу кількість обмежень на колонки реляційних таблиць для реалізації правил обробки даних без процедурного коду;
- задіювати тригери БД для підтримки цілісності даних під час процедурного введення правил обробки даних,;
- задіювати збережені процедури для інкапсуляції загальних бізнес-функцій;
- використовувати вимоги продуктивності під час остаточного вибору щодо розділення коду.

Як відмічають фахівці, переважна кількість недоліків в прикладних системах викликані через невизначеність різниці між правилами для даних, правилами для процесів та правилами для інтерфейсу. Розглянемо основні з них.

Правила для даних формулюють вимоги, якім повинні задовольняти дані, діють для кожного екземпляру даних та виводяться з моделі даних.

Приклади правил для данихСховати

Стать людини може бути лише чоловіча або жіноча; кожне замовлення призначене лише для одного покупця.

Правила для процесів визначають, що може або не може виконувати додаток, виводяться з моделі функцій.

Приклад правил для процесівСховати

Розмір стипендії не повинен перевищувати 1000 грн; тільки викладач може виставити оцінку за екзамен.

Правила для інтерфейсу визначають, яким повинен бачити додаток користувач. Ці правила не стосуються обробки, лише впливають на уяву користувача про додаток БД. Виводяться зі специфікації користувальницького інтерфейсу.

Приклад правил для інтерфейсуСховати

Всі коди оцінок повинні роз’яснюватись.

Виділення і аналіз цих правил приводить до формування трьох наборів документів: опис структури інтерфейсу, структури процесів (визначає реалізацію інтерфейсу) та структури даних (визначає основні об’єкти БД, з якими працюють процеси).

Коротко сформулюємо основні принципи розміщення бізнес-логіки в модулях додатків БД:

- правила для інтерфейсу реалізуються у зовнішній (клієнтській) частині системи незалежно від застосованих мов програмування або генераторів звітів;

- правила для процесів реалізуються у вигляді процедур, що викликаються з клієнтської частини і представляють серверний код;
- правила для даних реалізуються в самі БД у вигляді обмежень та тригерів.

Загальні принципи розроблення специфікацій модулів

Після розроблення схеми «функції-модулі» проектувальник починає до вирішення ємкої задачі написання специфікації модулів. Остання дозволить програмістам побудувати реальну систему з використанням БД.

Під час написання слід максимально виключити суб'єктивні думки щодо написання коду, оскільки код пишеться іншими фахівцями з тестуванням. Намагатись уникати формальних мов опису алгоритмів, використовувати формулювання у загальному вигляді. Не слід використовувати команди мови SQL, оскільки під час тестування адміністратор БД може змінити фізичну модель БД, що спричинить зміну команд. Слід уникати зайвих інструкцій.

Специфікація повинна обов'язково містити такі компоненти:

- умовну назву модуля;
- функції, виконання яких забезпечує модуль;
- перелік таблиць та колонок, до яких відбувається доступ;
- для кожної колонки – спосіб використання (запит, вставка, видалення або поновлення);
- перелік колонок;
- детальний опис всіх дій, що може виконувати модуль.

Приклад: типова специфікація модуля надання доступу до додатку БД.Сховати
Найменування: Сторінка для входу в додаток LogIn.

Мета: ідентифікація користувача та надання доступу до додатку БД.

Вхідні дані: Ім'я користувача. Пароль.

Таблиця БД: UserAccount

Колонки:

UserName – запит, пошук в предикаті пошуку;

UserPass – запит, пошук в предикаті пошуку.

Дії:

За умови відсутності користувача з таким ім'ям та паролем в БД відмовити в доступі та вивести запит на правильне введення даних не більш ніж 3 рази.

За умови наявності користувача з таким ім'ям та паролем в БД надати доступ до модулю «Головна сторінка», яка має різний вигляд відповідно повноваженням користувача.

Коментарі:

Залежно від типу модуля (екрана форма, звіт та інш.), специфікації можуть містити додаткову інформацію щодо розміщення кнопок або формат звіту. В цьому випадку специфікацію слід доповнити такими позиціями:

- дані про навігацію (який модуль викликає, які модулі викликаються);
- значення вхідних параметрів за замовчуванням;
- перелік подій, які оброблюються на екранній формі, яким чином оброблюються;
- перелік помилок і дій, пов'язаних з їх обробкою;
- дані щодо безпеки;
- макет екранної форми або шаблон звіту.

Ключові терміни:

деструктор, конструктор, системний модуль, таблиця "Функція-Сутність"

4.1 Аналіз функціональної моделі предметної області бази даних

Вхідними даними для вирішення завдання проектування модулів додатків БД є ієрархія функцій. На виході розробник повинен отримати опис (специфікацію) модулів додатків, а в процесі проектування модулів розробник будує відображення бізнес - вимог у специфікації модулів.

Алгоритм дій розробника БД полягає у наступному: спочатку розробник намагається сформулювати бізнес-вимоги (функції) у самому загальному вигляді, а потім виконує декомпозицію кожної такої бізнес-функції доти, поки не буде отримана деяка функція, яку можна вважати атомарною функцією.

При розгляді ієрархії функцій розробникові БД варто звернути увагу на такі моменти:

- у функціональній моделі БД описуються бізнес-функції, і не всі вони будуть безпосередньо підтримуватися додатком БД;

- при розгляді ієрархій нерідко виникає ситуація, коли екземпляри однієї й тієї ж функції будуть мати різні номери.

Якщо в першому випадку додаткову інформацію про те, які бізнес-функції будуть реалізовані в системі, можна одержати від керівника проекту, то в другому випадку розробник БД, найімовірніше, має справу з помилкою аналітика у визначенні функції.

4.2 Визначення функцій

При розробці ієрархії функцій аналітик повинен надати текстовий опис до кожної функції, принаймні для верхнього й самого нижнього рівнів ієрархії. Бажано, щоб у цьому описі аналітики виділяли сутності ПО. Це важливо для того, щоб знати з якими сутностями ПО працює функція, тобто які потенційні об'єкти реляційної БД будуть використовуватися в кожній функції. Якщо це не зроблено, то розробникі БД будуть вимушені робити це самостійно.

Із приклада видно, які сутності ПО беруть участь у виконанні функції (виділені в дужках), як міняється стан сутності (виділено курсивом) і який алгоритм роботи цієї функції.

Із приклада зрозуміло, що на цьому етапі розробник БД у якості вхідних даних використає також інформаційну модель ПО БД (опис сутностей).

При виконанні аналізу функцій корисно мати деяку таблицю (матрицю) "Функція-Сутність", яка повинна дати відповідь на наступні питання:

- чи має кожна сутність конструктор (функцію, що створює всі екземпляри сутності);
- чи має вона деструктор (функцію, що видаляє екземпляри сутності);
- чи є посилання на цю сутність (функції, які використають цю сутність й який образ).

Процес аналізу взаємодії функції й сутності прийнято позначати абревіатурою CRUD (Create, Reference, Update, Delete - створення, посилання, модифікація, видалення).

Корисними для розуміння розробником бази дані призначення функцій і того, як дані функції беруть участь у процесі обробки даних у системі, можуть бути діаграми потоку даних і діаграми життєвих циклів сутностей, які були розглянуті нами в другій лекції. Останні, зокрема, дають ясну картину зміни стану сутності, що важливо у визначенні атрибутів статусу сутності.

4.3 Відображення функцій у модулі

Одним з основних завдань проектування модулів додатків є побудова відображення функцій у модулі. При вирішенні цього завдання розробник БД повинен акцентувати увагу на структурі БД, що становить основу додатка.

Як правило, вирішення завдання відображення функцій у модулі виконується в чотири етапи:

1. Аналіз роботи функції.
2. Побудова моделі сутностей, що підтримує ці функції.
3. Почати проектування фізичної структури зі створення схеми, що підтримує розроблену модель сутностей.
4. Завершити проектування розробкою специфікацій модулів, які реалізують функції на запропонованій схемі БД.

Із запропонованого вище підходу видно, як тісно переплітаються у процесі проектування процеси розробки фізичної моделі БД і специфікацій модулів додатків. Таким чином, якщо розробником був розроблений чорновий варіант фізичної моделі БД по алгоритмах, розглянутим нами в попередніх лекціях, то на цьому етапі він повинен бути адаптований до реалізації функцій й, можливо, значно перероблений.

При відображенні функцій у модулі необхідно отримати схему, що ставить у відповідність кожної функції певний модуль.

4.4 Системні модулі

Під час реалізації функціональних можливостей необхідно розглянути достатню кількість процесів, які не є наявними у сформульованих бізнес-функціях. Наприклад, можливість видачі результатів роботи модуля в файл або на принтер.

Набір модулів додатків БД, який не слідує явно з бізнес-функції, але є необхідним для забезпечення роботи системи, називається системним. До таких модулів можна віднести процедури резервного копіювання та автоматичного відновлення, модулі зміни паролів, модулі друку та навігації по додаткам БД та інш.

Розробник БД самостійно вирішує питання розроблення специфікації системних модулів.

4.5 Розміщення логіки обробки

Під час розроблення серверного коду необхідно дотримуватись простих правил:

- задіювати найбільшу кількість обмежень на колонки реляційних таблиць для реалізації правил обробки даних без процедурного коду;
- задіювати тригери БД для підтримки цілісності даних під час процедурного введення правил обробки даних,;

- задіювати збережені процедури для інкапсуляції загальних бізнес-функцій;
- використовувати вимоги продуктивності під час остаточного вибору щодо розділення коду.

Як відмічають фахівці, переважна кількість недоліків в прикладних системах викликані через невизначеність різниці між правилами для даних, правилами для процесів та правилами для інтерфейсу. Розглянемо основні з них.

Правила для даних формулюють вимоги, якім повинні задовольняти дані, діють для кожного екземпляру даних та виводяться з моделі даних.

Правила для процесів визначають, що може або не може виконувати додаток, виводяться з моделі функцій.

Правила для інтерфейсу визначають, яким повинен бачити додаток користувач. Ці правила не стосуються обробки, лише впливають на уяву користувача про додаток БД. Виводяться зі специфікації користувальницького інтерфейсу.

Виділення і аналіз цих правил приводить до формування трьох наборів документів: опис структури інтерфейсу, структури процесів (визначає реалізацію інтерфейсу) та структури даних (визначає основні об'єкти БД, з якими працюють процеси).

Коротко сформулюємо основні принципи розміщення бізнес-логіки в модулях додатків БД:

- правила для інтерфейсу реалізуються у зовнішній (клієнтській) частині системи незалежно від застосованих мов програмування або генераторів звітів;
- правила для процесів реалізуються у вигляді процедур, що викликаються з клієнтської частини і представляють серверний код;
- правила для даних реалізуються в самі БД у вигляді обмежень та тригерів.

4.6 Загальні принципи розроблення специфікацій модулів

Після розроблення схеми «функції-модулі» проектувальник починає до вирішення ємкої задачі написання специфікації модулів. Остання дозволить програмістам побудувати реальну систему з використанням БД.

Під час написання слід максимально виключити суб'єктивні думки щодо написання коду, оскільки код пишеться іншими фахівцями з тестуванням. Намагатись уникати формальних мов опису алгоритмів, використовувати формулювання у загальному вигляді. Не слід використовувати команди мови SQL, оскільки під час тестування адміністратор БД може змінити фізичну модель БД, що спричинить зміну команд. Слід уникати зайвих інструкцій.

Специфікація повинна обов'язково містити такі компоненти:

- умовну назву модуля;
- функції, виконання яких забезпечує модуль;
- перелік таблиць та колонок, до яких відбувається доступ;
- для кожної колонки – спосіб використання (запит, вставка, видалення або поновлення);
- перелік колонок;
- детальний опис всіх дій, що може виконувати модуль.

Вхідними даними для вирішення завдання проектування модулів додатків БД є ієрархія функцій модуля додатка, на основі яких повинні бути сформульовані бізнес - вимоги (функції) у самому загальному вигляді. Другим етапом є декомпозиція кожної такої бізнес-функції доти, поки не буде отримана деяка функція, яку можна вважати атомарною функцією.

Із лекції видно, як тісно переплітаються у процесі проектування процеси розробки фізичної моделі БД і специфікацій модулів додатків. Наведені рекомендації визначають загальні принцип розроблення модулів БД, які можуть бути доповнені у кожному окремому випадку предметної області.

1. В чому полягає аналіз функціональних вимог до БД?
2. Назвіть основні етапи процесу розроблення додатка.
3. Які компоненти повинна мати таблиця «Функція - Сутність»?
4. Назвіть основні етапи процесу відображення функції в модулі.
5. Які модулі відносять до системних?
6. Сформулюйте основні правила логічного розроблення додатків БД.
7. Які компоненти повинні міститись у специфікації модуля БД?

1. К. Дж. Кейт Введення в системи баз даних Пер. с англ. 8-е изд. М.: Издательский дом «Вильямс», 2006.– 1328С.
2. Томас Коннолли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.– 3-е изд. М.: Издательский дом «Вильямс», 2003, 1436 с.
3. Джен Л. Харрингтон Проектирование реляционных баз данных — М.: Издательство «Лори», 2006 – 230 с.
4. Пасічник В.В.Організація баз даних та знань: підручник для ВНЗ/ В.В. Пасічник, В.А. Резніченко.– К.: Видавнича група BHV,2006.-384с.

5. В.Е. Туманов. Основы проектирования реляционных баз данных
6. Д.В. Кознов. Визуальное моделирование: теория и практика

Розділ 2 - Реляційна модель даних

Тема 5 - Реляційна модель даних

- Поняття відношення
- Форма подання відношення
- Операції над реляційними даними
- Функціональна залежність в даних

Ключові терміни:

атомарний ключ, відношення, домен, зовнішній ключ, ключ, обмежуюча умова, об'єднання, первинний ключ, перетин, подання, проекція, різниця, селекція, складений ключ, цілісність даних, частковий ключ

Поняття відношення

Широкому поширенню і популярністю реляційна модель даних була завдячує двом істотних перевагам:

1. Однорідність подання даних у моделі, що обумовлює простоту сприйняття її конструкцій користувачами БД;
2. Наявність розвиненої математичної теорії реляційних БД, що обумовлює коректність її застосування.

В основі реляційної моделі даних лежить поняття відношення, яке задається переліком своїх елементів і перерахуванням їх значень. Розглянемо приклад на рис.5.1 На ньому наведений розклад руху автобусів по маршруту "Москва - Черноголовка - Москва". Бачимо певну структуру. Кожен включений у розклад рейс має свій номер, час відправлення й час у дорозі. Розклад може бути представлено таблицею. Заголовки колонок таблиці звуться атрибутами. Перелік їх імен носить назви схеми відношення. Кожен атрибут визначає тип даних, що разом з областю його значень називається доменом. Вся таблиця цілком називається відношенням, а кожен рядок таблиці зветься кортежем відношення. Таким чином, відношення можна представити у вигляді двовимірної таблиці.

Підходи до визначення поняття відношення можуть бути різними. Математично відношення може бути визначене як безліч кортежів, що є підмножиною декартового добутку фіксованого числа областей (доменів). У результаті одержуємо, що у кожному кортежі повинне бути однакове число компонентів (атрибутів) і значення кожного з них вибирається з деякого певного домену.

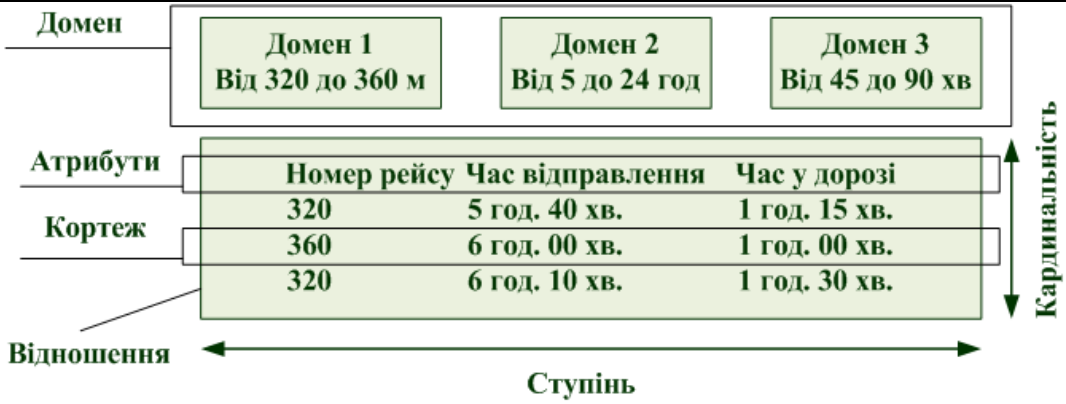


Рисунок 5.1 Розклад руху автобусів як відношення

Таблична форма подання відношення була введена з метою популяризації моделі серед непідготовлених користувачів БД. Трактування реляційної теорії на рівні таблиць приховують ряд визначень, важливих для розуміння як теорії реляційних БД, так і мови маніпулювання даними.

По-перше, атрибути різних відношень можуть бути визначені на одному домені, так само як й атрибути одного відношення. Це дуже важлива обставина, що дозволяє встановлювати зв'язки за значенням між відношеннями. По-друге, множина математично по своєму визначенню не може мати співпадаючих елементів, і, отже, кортежі у відношенні можна розрізнити лише за значенням їх компонентів. Це теж є дуже важливим для моделі: ніякі два кортежі не можуть мати повністю співпадаючих компонентів. **Таким чином, у реляційній моделі повністю виключається дублювання даних про сутності реального світу**. По-третє, відзначимо, що схема відношення також є множина, що дозволяє працювати з ними за допомогою теоретико-множинних операцій. Це є важливим моментом для побудови теорії проектування реляційних схем БД.

Існує певне розходження між математичним визначенням відношення й дійсне збереженим відношенням у пам'яті комп'ютера. За визначенням, відношення не може мати два ідентичних кортежі. Однак СКБД, що підтримують реляційну модель даних, зберігають відношення у файлах операційної системи комп'ютера. Розміщення відношень у файлах операційної системи допускає зберігання ідентичних кортежів. Якщо не використовується спеціальна техніка (контроль цілісності по первинному ключі), то звичайно більшість промислових СКБД допускають зберігання двох ідентичних кортежів у БД.

При створенні ІС сукупність відношень дозволяє зберігати дані про об’єкти предметної області і моделювати зв’язки між ними. У кожному зв’язку одне відношення може виступати як основне, а друге виступає в ролі похідного. Таким чином, один кортеж основного відношення може бути пов’язаним з декількома кортежами похідного відношення. Для підтримки цих зв’язків обидва відношення повинні містити атрибути, за якими вони пов’язані. В основному відношенню, це первинний ключ, а в підлеглому – набір атрибутів, що відповідає первинному ключу основного відношення.

Ключем або ключовим полем називається унікальне значення, що дозволяє тим чи іншим способом ідентифікувати сутність або частину сутності ПО, тобто ключ - це значення деякого атрибута або атрибутів у кортежі відношення, що представляє екземпляр сутності у реляційній моделі даних.

Прийнято розрізняти первинні ключі й часткові ключі. Математично первинним ключем відношення є підмножина звуження декартового добутку, що дозволяє однозначно ідентифікувати кортеж. Якщо первинний ключ містить кілька атрибутів, то він називається складеним ключем, у протилежному випадку - атомарним. Частковим ключем називається атрибут складеного ключа, якщо він однозначно визначає сукупність неключових атрибутів відношення. Атрибут кортежу, що є первинним ключем іншого відношення, називається зовнішнім (іноді стороннім) ключем. З визначення відношення випливає наступна важлива властивість реляційної моделі даних: кожне відношення повинне мати первинний ключ. Зазначимо, що ключ у контексті моделі ПО БД завжди відображає той або інший ступінь зв'язку між атрибутами сутностей ПО, тобто семантично ключ є засіб моделювання зв'язків у моделі.

ПрикладСховати

Розглянемо речення "Громадянин Іванов проживав у місті Москві 10 років". Можливими атрибутами у відношенні Місце_проживання є прізвище громадянина, назва міста проживання й час проживання. Прізвище громадянина може виступати як первинний ключ цього відношення, тому що особистість однозначно визначає час її проживання в конкретному місті. Таким чином, щодо цього моделюється зв'язок "проживав" між атрибутами "прізвище" й "місто".

Домен є семантичним поняттям, яке можна розглядати як підмножину значень деякого типу даних, що мають певний сенс. Домен характеризується такими властивостями:

- має унікальне ім’я в межах БД;
- визначений на деякому простому типу даних або на іншому домені;
- може мати деяку логічну умову, що дозволяє описати підмножину даних для цього домену;
- несе певний сенс.

Інші поняття ми розглядали у попередніх лекціях, тому зупинятись на них детально не будемо. До того ж, далі будуть наводитись приклади, які будуть пояснювати визначення та

властивості основних понять.

Обмежуючі умови підтримки цілісності.

Цілісність даних – узгодженість даних в БД. Обмежуюча умова – це правило, яке обмежує значення даних в БД. В реляційній моделі є декілька обмежуючих умов, що використовуються для перевірки даних в БД та надання сенсу структурі даних. Прийнято виділяти такі обмеження:

Категорійна цілісність. Рядки реляційної таблиці представляють в БД елементи певних об'єктів реального світу або категорій. Ключ реляційної таблиці однозначно визначає кожний рядок і, як наслідок, кожен елемент категорії. Таким чином, коли користувач вилучає дані певного рядку або маніпулює ними, повинні знати значення ключа цього рядка. Відповідно недопустимим є порожнє значення ключа або частини ключа.

Правило категорійної цілісності: ніякий ключовий атрибут рядка не може бути порожнім.

Цілісність на рівні посилання. При побудові реляційних таблиць для зв'язування рядків однієї таблиці з рядками іншої використовуються зовнішні ключі. БД, в якій всі не порожні зовнішні ключі посилаються на поточні значення ключів іншої таблиці, володіє цілісністю на рівні посилання.

Правило цілісності на рівні посилання: значення не порожнього зовнішнього ключа повинно бути рівним одному з поточних значень ключа другої таблиці.

Форма подання відношення

Відношення у реляційній моделі даних, як правило, представляються за допомогою функціональної форми запису (тому що ми записує функції декількох змінних у математичному аналізі), при цьому атрибути первинного ключа підкреслюються:

ІМ'Я_ВІДНОШЕННЯ (Атрибути первинного ключа, неключові атрибути).

ПрикладСховати

Подання зв'язку відношенням. Представимо зв'язок між особистістю й місцем її проживання через відношення (відношення наведене при розгляді попереднього питання лекції)

ПРОЖИВАЄ (Кл. особистість, Кл. населений_пункт, час)

Опис особистості:

ОСОБИСТІСТЬ (Кл. особистість, ФІО, вік, стать)

Опис населеного пункту:

НАСЕЛЕНИЙ_ПУНКТ (Кл.населений_пункт, географія, населення)

Однак найбільшого поширення одержало подання відношень у вигляді графічних діаграм, наприклад ER-діаграм, про які ми говорили раніш. Перевагами такого подання є наочність діаграм і можливість їх побудови у ряді CASE-засобів проектування БД.

Операції над реляційними даними

Множина операцій над реляційними даними становлять реляційну алгебру. Кожна операція задіює одну або дві таблиці. Основних операцій вісім, розбитих на дві групи. Розглянемо їх детальніше.

Традиційні операції.

Об'єднання двох відношень – це відношення, яке містить множину рядків, приналежних або першому, або другому вхідному відношенню, або обом відношенням одночасно (рис. 5.2).

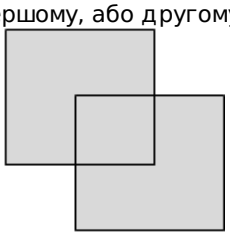


Рисунок 5.2 Операція об'єднання відношень

Перетин двох відношень – це відношення, яке містить множину рядків, приналежних одночасно і першому і другому відношенням (рис. 5.3).

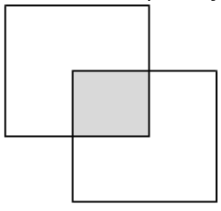


Рисунок 5.3 Операція перетину відношень

Різниця двох відношень – це відношення, що містить множину рядків, приналежних першому відношенню (рис. 5.4).

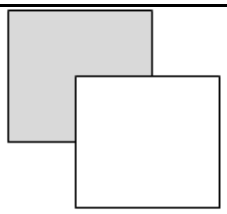


Рисунок 5.4 Операція різниці відношень
Декартовий добуток зображений на рис. 5.5.

$$\begin{array}{|c|}\hline a\\ b\\ c\\ \hline\end{array} \Xi \begin{array}{|c|}\hline x\\ y\\ z\\ \hline\end{array} = \begin{array}{|cc|}\hline a & x\\ a & y\\ b & x\\ b & y\\ c & x\\ c & y\\ \hline\end{array}$$

Рисунок 5.5 Операція декартового добутку відношень
Спеціальні операції.
Селекція – вибірка по критеріям, що виконується по рядкам.
Проекція – вибірка по колонкам.
Природнє поєднання зображене на рис. 5.6.

$$\begin{array}{|cc|}\hline a_1 & b_1\\ a_2 & b_2\\ a_3 & b_3\\ \hline\end{array} \Xi \begin{array}{|cc|}\hline b_1 & c_1\\ b_2 & c_2\\ b_3 & c_3\\ \hline\end{array} = \begin{array}{|ccc|}\hline a_1 & b_1 & c_1\\ a_2 & b_2 & c_2\\ a_3 & b_3 & c_3\\ \hline\end{array}$$

Рисунок 5.6 Операція природнього поєднання відношень
Ділення зображене на рис. 5.7.

$$\begin{array}{|cc|}\hline a & x\\ a & y\\ a & z\\ b & x\\ b & y\\ b & z\\ \hline\end{array} \Xi \begin{array}{|c|}\hline x\\ y\\ z\\ \hline\end{array} = \begin{array}{|c|}\hline a\\ b\\ \hline\end{array}$$

Рисунок 5.7 Операція ділення відношень
Результатом виконання операції є подання.

Функціональна залежність в даних

На стадії логічного проектування реляційної БД розробник визначає й вибудовує схеми відношення у рамках деякої ПО, а саме - представляє сутності, групує їх атрибути, виявляє основні зв'язки між сутностями. Так, у самому загальному змісті проектування реляційної БД полягає в обґрунтованому виборі конкретних схем відношення з безлічі різних альтернативних варіантів схем.

На практиці побудова логічної моделі БД, незалежно від моделі даних, виконується з урахуванням двох основних вимог: виключити надмірність і максимально підвищити надійність даних. Ці вимоги випливають із вимоги колективного використання даних групою користувачів.

Тому будь-яке апріорне знання про обмеження ПО, що накладають на взаємозв'язки між даними й значення даних, і знання про їх властивості і взаємини між ними може зіграти певну роль у дотриманні зазначених вище вимог. Формалізація таких апріорних знань про властивості даних ПО БД знайшла своє відображення у концепції функціональної залежності даних, тобто обмежень на можливі взаємозв'язки між даними, які можуть бути поточними значеннями схеми відношень.

Кортежі відношення можуть представляти екземпляри сутності ПО або фіксувати їх взаємозв'язок. Але навіть якщо ці кортежі відповідають схемі відношень й обрані з припустимих доменів, не кожен з них може бути поточним значенням деякого відношення. Наприклад, вік людини рідко буває більше 120 років, або той самий пілот не може одночасно виконувати два різних рейси. Такі обмеження семантики домену практично не впливають на вибір тієї або іншої схеми відношень. Вони являють собою обмеження на типи даних.

Оскільки функціональну залежність можна задати у вигляді таблиці, а таблиця є форма подання відношень, то стає очевидний зв'язок між функціональною залежністю і відношенням. Відношення може задавати функціональну залежність. Це твердження є першою конструктивною ідеєю, яка покладена в основу теорії проектування реляційних БД.

Приклад. Визначення функціональних залежностейСховати

Проілюструємо поняття функціональної залежності на прикладі графіка польотів аеропорту.

ГРАФІК_ПОЛЬОТІВ (Пілот, Рейс, Дата_вильоту, Час_вильоту)

Іванов	*100*	*8.07*	*10:20*
Іванов	*102*	*9.07*	*13:30*
Ісаєв	*90*	*7.07*	*6:00*
Ісаєв	*103*	*10.07*	*19:30*
Петров	*100*	*12.07*	*10:20*
Фролов	*90*	*8.07*	*6:00*
Фролов	*90*	*12.07*	*6:00*

Відомо: кожному рейсу відповідає певний час вильоту; для кожного пілота, дати й часу вильоту можливий тільки один рейс; на певний день і рейс призначається певний пілот.

Отже: "Час_вильоту" функціонально залежить від {"Рейс"}; "Рейс" функціонально залежить від {"Пілот", "Дата_вильоту", "Час_вильоту"}; "Пілот" функціонально залежний від {"Рейс", "Дата_вильоту"}.

Важливим завданням при виявленні функціональних залежностей на атрибутах відношень, що за визначенням є множиною, необхідно з'ясувати, який з атрибутів виступає як аргумент, а який - як значення функціональна залежність. Найбільш підходящими кандидатами в аргументи функціональної залежності є можливі ключі, тому що кортежі представляють екземпляри сутності, які ідентифікуються значеннями атрибутів свого ключа.

- 5.1 Поняття відношення
- 5.2 Форма подання відношення
- 5.3 Операції над реляційними даними
- 5.4 Функціональна залежність в даних

Ключові терміни:

атомарний ключ, відношення, домен, зовнішній ключ, ключ, обмежуюча умова, об’єднання, первинний ключ, перетин, подання, проекція, різниця, селекція, складений ключ, цілісність даних, частковий ключ

5.1 Поняття відношення

Широкому поширенню і популярністю реляційна модель даних була завдячує двом істотних перевагам:

1. Однорідність подання даних у моделі, що обумовлює простоту сприйняття її конструкцій користувачами БД;
2. Наявність розвиненої математичної теорії реляційних БД, що обумовлює коректність її застосування.

Підходи до визначення поняття відношення можуть бути різними. Математично відношення може бути визначене як безліч кортежів, що є підмножиною декартового добутку фіксованого числа областей (доменів). У результаті одержуємо, що у кожному кортежі повинне бути однакове число компонентів (атрибутів) і значення кожного з них вибирається з деякого певного домену.

По-перше, атрибути різних відношень можуть бути визначені на одному домені, так само як й атрибути одного відношення. Це дуже важлива обставина, що дозволяє встановлювати зв'язки за значенням між відношеннями. По-друге, множина математично по своєму визначенню не може мати співпадаючих елементів, і, отже, кортежі у відношенні можна розрізнити лише за значенням їх компонентів. Це теж є дуже важливим для моделі: ніякі два кортежі не можуть мати повністю співпадаючих компонентів. **Таким чином, у реляційній моделі повністю виключається дублювання даних про сутності реального світу**. По-третє, відзначимо, що схема відношення також є множина, що дозволяє працювати з ними за допомогою теоретико-множинних операцій. Це є важливим моментом для побудови теорії проектування реляційних схем БД.

Існує певне розходження між математичним визначенням відношення й дійсне збереженим відношенням у пам'яті комп'ютера. За визначенням, відношення не може мати два ідентичних кортежі. Однак СКБД, що підтримують реляційну модель даних, зберігають відношення у файлах операційної системи комп'ютера. Розміщення відношень у файлах операційної системи допускає зберігання ідентичних кортежів.

При створенні ІС сукупність відношень дозволяє зберігати дані про об’єкти предметної області і моделювати зв’язки між ними. У кожному зв’язку одне відношення може виступати як основне, а друге виступає в ролі похідного. Таким чином, один кортеж основного відношення може бути пов’язаним з декількома кортежами похідного відношення. Для підтримки цих зв’язків обидва відношення повинні містити атрибути, за якими вони пов’язані. В основному відношенню, це первинний ключ, а в підлеглому – набір атрибутів, що відповідає первинному ключу основного відношення.

Ключем або ключовим полем називається унікальне значення, що дозволяє тим чи іншим

способом ідентифікувати сутність або частину сутності ПО, тобто ключ - це значення деякого атрибута або атрибутів у кортежі відношення, що представляє екземпляр сутності у реляційній моделі даних.

Математично первинним ключем відношення є підмножина звуження декартового добутку, що дозволяє однозначно ідентифікувати кортеж. Якщо первинний ключ містить кілька атрибутів, то він називається складеним ключем, у протилежному випадку - атомарним. Частковим ключем називається атрибут складеного ключа, якщо він однозначно визначає сукупність неключових атрибутів відношення. Атрибут кортежу, що є первинним ключем іншого відношення, називається зовнішнім (іноді стороннім) ключем. З визначення відношення випливає наступна важлива властивість реляційної моделі даних: кожне відношення повинне мати первинний ключ. Зазначимо, що ключ у контексті моделі ПО БД завжди відображає той або інший ступінь зв'язку між атрибутами сутностей ПО, тобто семантично ключ є засіб моделювання зв'язків у моделі.

Домен є семантичним поняттям, яке можна розглядати як підмножину значень деякого типу даних, що мають певний сенс. Домен характеризується такими властивостями:

- має унікальне ім’я в межах БД;
- визначений на деякому простому типу даних або на іншому домені;
- може мати деяку логічну умову, що дозволяє описати підмножину даних для цього домену;
- несе певний сенс.

Обмежуючі умови підтримки цілісності.

Цілісність даних – узгодженість даних в БД. Обмежуюча умова – це правило, яке обмежує значення даних в БД. В реляційній моделі є декілька обмежуючих умов, що використовуються для перевірки даних в БД та надання сенсу структурі даних. Прийнято виділяти такі обмеження.

Категорійна цілісність. Рядки реляційної таблиці представляють в БД елементи певних об’єктів реального світу або категорій. Ключ реляційної таблиці однозначно визначає кожний рядок і, як наслідок, кожен елемент категорії. Таким чином, коли користувач вилучає дані певного рядку або маніпулює ними, повинні знати значення ключа цього рядка. Відповідно недопустимим є порожнє значення ключа або частини ключа.

Правило категорійної цілісності: ніякий ключовий атрибут рядка не може бути порожнім.

Цілісність на рівні посилання. При побудові реляційних таблиць для зв’язування рядків однієї таблиці з рядками іншої використовуються зовнішні ключі. БД, в якій всі не порожні зовнішні ключі посилаються на поточні значення ключів іншої таблиці, володіє цілісністю на рівні посилання.

Правило цілісності на рівні посилання: значення не порожнього зовнішнього ключа повинно бути рівним одному з поточних значень ключа другої таблиці.

5.2 Форма подання відношення

Відношення у реляційній моделі даних, як правило, представляються за допомогою функціональної форми запису (тому що ми записує функції декількох змінних у математичному аналізі), при цьому атрибути первинного ключа підкреслюються:

ІМ'Я_ВІДНОШЕННЯ (Атрибути первинного ключа, неключові атрибути).

Однак найбільшого поширення одержало подання відношень у вигляді графічних діаграм, наприклад ER-діаграм, про які ми говорили раніш. Перевагами такого подання є наочність діаграм і можливість їх побудови у ряді CASE-засобів проектування БД.

5.3 Операції над реляційними даними

Множина операцій над реляційними даними становлять реляційну алгебру. Кожна операція задіює одну або дві таблиці. Основних операцій вісім, розбитих на дві групи. Розглянемо їх детальніше.

Традиційні операції.

Об’єднання двох відношень – це відношення, яке містить множину рядків, приналежних або першому, або другому вхідному відношенню, або обом відношенням одночасно.

Перетин двох відношень – це відношення, яке містить множину рядків, приналежних одночасно і першому і другому відношенням.

Різниця двох відношень – це відношення, що містить множину рядків, приналежних першому відношенню.

Спеціальні операції.

Селекція – вибірка по критеріям, що виконується по рядкам.

Проекція – вибірка по колонкам.

Результатом виконання операції є подання.

5.4 Функціональна залежність в даних

На стадії логічного проектування реляційної БД розробник визначає й вибудовує схеми відношення у рамках деякої ПО, а саме - представляє сутності, групує їх атрибути, виявляє основні зв'язки між сутностями. Так, у самому загальному змісті проектування реляційної БД полягає в обґрунтованому виборі конкретних схем відношення з безлічі різних альтернативних варіантів схем.

На практиці побудова логічної моделі БД, незалежно від моделі даних, виконується з урахуванням двох основних вимог: виключити надмірність і максимально підвищити надійність даних. Ці вимоги випливають із вимоги колективного використання даних групою користувачів.

Формалізація знань про властивості даних ПО БД знайшла своє відображення у концепції функціональної залежності даних, тобто обмежень на можливі взаємозв'язки між даними, які можуть бути поточними значеннями схеми відношень.

Кортежі відношення можуть представляти екземпляри сутності ПО або фіксувати їх взаємозв'язок. Але навіть якщо ці кортежі відповідають схемі відношень й обрані з припустимих доменів, не кожен з них може бути поточним значенням деякого відношення. акі обмеження семантики домену практично не впливають на вибір тієї або іншої схеми відношень. Вони являють собою обмеження на типи даних.

Оскільки функціональну залежність можна задати у вигляді таблиці, а таблиця є форма подання відношень, то стає очевидний зв'язок між функціональною залежністю і відношенням. Відношення може задавати функціональну залежність. Це твердження є першою конструктивною ідеєю, яка покладена в основу теорії проектування реляційних БД.

Важливим завданням при виявленні функціональних залежностей на атрибутах відношень, що за визначенням є множиною, необхідно з'ясувати, який з атрибутів виступає як аргумент, а який - як значення функціональна залежність. Найбільш підходящими кандидатами в аргументи функціональної залежності є можливі ключі, тому що кортежі представляють екземпляри сутності, які ідентифікуються значеннями атрибутів свого ключа.

У підсумку сформулюємо основні властивості реляційної моделі даних, які випливають із поняття відношення як множини:

- всі кортежі одного відношення повинні мати ту саму кількість атрибутів.
- значення кожного з атрибутів повинне належати деякому певному домену.
- кожне відношення повинне мати первинний ключ.
- ніякі два кортежі не можуть мати повністю співпадаючих наборів значень.
- кожне значення атрибутів повинне бути атомарними, тобто не повинне мати внутрішньої структури й містити як компонент інше відношення.
- реляційна модель даних повинна бути несуперечливою, зокрема повинно виконуватися
    1. Принцип посилальної цілісності - зв'язки між відношеннями повинні бути замкнутими;
    2. Значення колонок повинні належати тому самому визначеному для них домену.
- порядок проходження кортежів у відношенні не має значення. Порядок є більшою мірою властивістю зберігання даних, чим властивістю безпосередньо самої реляційної моделі даних.

1. Назвіть переваги реляційної БД?
2. Поясніть поняття «відношення».
3. Які компоненти складають відношення?
4. Що таке ключове поле? Призначення та різновиди.
5. Поясніть поняття «цілісність даних».
6. Що таке категорійна цілісність?
7. Що таке цілісність по посиланню
8. Які основні операції реляційної алгебри відносяться до операцій над відношенням БД?
9. Назвіть основні властивості реляційної БД.
10. Назвіть основні принципи при визначені функціональних залежностей.

Тема 6- Теорія нормалізації реляційної моделі даних

- Нормальні форми відношення
- Перша нормальна форма відношення
- Друга нормальна форма відношення

- Третя нормальна форма відношення
- Нормальна форма Бойса-Кодда
- Четверта нормальна форма відношення

Ключові терміни:

аномалія відновлення, атомарне відношення, нормальна форма схеми відношень, нормалізація, нормалізація відношень, реляційна база даних, теорема Риссанена, теорема Фейджина

Нормальні форми відношення

Під реляційною БД прийнято розуміти сукупність екземплярів кінцевих відношень. Сукупність схем відношень утворює схему реляційної БД.

Схема реляційної БД є логічною моделлю реляційної БД. На основі інформаційної моделі у процесі проектування створюються логічна й фізична моделі даних. Інформаційна модель даних відбиває потреби системи в даних і зв'язку між даними з погляду їх споживачів - користувачів; логічна модель даних є незалежним логічним поданням даних; фізична модель даних містить визначення всіх реалізованих об'єктів у конкретній БД для конкретної СКБД.

Встановлення функціональної залежності й одержання найкращого з погляду мінімальності подання множини функціональних залежностей дозволять побудувати найбільш оптимальний варіант БД, що забезпечує надійність зберігання й обробки даних на основі методів еквівалентних перетворень схем відношень реляційної БД. Процес вирішення такого завдання називається нормалізацією відношень інформаційної моделі ПО й полягає у перетворенні її об'єктів у логічні таблиці БД. Основні вимоги наведені нижче:

- первинні ключі відношень повинні бути мінімальними;
- число відношень БД повинне по можливості давати найменшу надмірність даних – вимога надійності даних;
- число відношень БД не повинне приводити до втрати продуктивності системи;
- дані не повинні бути суперечливими, тобто при виконанні операцій включення, видалення й відновлення даних їх потенційна суперечливість повинна бути зведена до мінімуму;
- схема відношень БД повинна бути стійкою, здатною адаптуватися до змін при її розширенні додатковими атрибутами – вимога гнучкості структури БД;
- розкид часу реакції на різні запити до БД не повинен бути великим;
- дані повинні правильно відбивати стан ПО БД у кожен конкретний момент часу – вимога актуальності даних.

Створення системи, що одночасно задовольняє всім вищезгаданим вимогам, являє собою складну оптимізаційну задачу, що часом не має однозначного вирішення.

Теорія функціональних залежностей дозволяє встановити певні вимоги до схем відношень у реляційній БД. Ці вимоги формулюються у термінах властивостей відношень і називаються нормальними формами схем відношень. Кожна нормальна форма відношень пов'язана з певним класом функціональної залежності, які представлені у відношеннях. Одним з очевидних засобів усунення потенційної суперечливості даних у відношеннях логічної моделі реляційної БД є їх розбиття на двоє або більше відношень, у кожному з яких є присутньою тільки одна функціональна залежність.

Процес усунення потенційної суперечливості й надмірності даних у відношеннях реляційної БД називається нормалізацією вихідних схем відношень. Нормалізація відношень полягає у виконанні декомпозиції відношень, що перебувають у попередній нормальній формі, на двоє або більше відношень, які задовольняють вимогам наступної нормальної форми.

У теорії реляційних БД звичайно виділяється така послідовність нормальних форм: перша нормальна форма (1NF); друга нормальна форма (2NF); третя нормальна форма (3NF); нормальна форма Бойса-Кодда (BCNF); четверта нормальна форма (4NF); п'ята нормальна форма, або нормальна форма проекції-з'єднання (5NF або PJ/NF).

Основні властивості нормальних форм полягають у такому: кожна наступна нормальна форма у деякому змісті краще попередньої нормальної форми; при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

Перша нормальна форма відношення

Відношення перебуває у 1NF, якщо всі атрибути відношення є простими (вимогу атомарності атрибутів), тобто не мають компонентів. Іншими словами, домен атрибута повинен складатися з неподільних значень і не може містити в собі безліч значень із більше елементарних доменів. Інколи у відношеннях деякі функціональні залежності атрибутів від можливого ключа не є мінімальними. Це призводить до так званих аномалій відновлення. Під аномаліями відновлення розуміються труднощі, з якими зустрічаються при виконанні операцій додавання кортежів у відношення (INSERT), видалення кортежів (DELETE) і модифікації кортежів (UPDATE).

Приклад приведення відношення до 1НФСховати

Нехай є змінне відношення: Employer_Project_Task {Em_Number, Em_Degrees, Em_Pay, Pr_Number, Em_Task}. Атрибути містять відповідно дані про номер справи, розряд та заробітну платню службовця, номер проекту й про завдання, що виконує службовець у даному проекті. Припустимо, що розряд службовця визначає розмір його заробітної плати й що кожен

службовець може брати участь у декількох проектах за умови виконання тільки одного завдання. Тоді очевидно, що єдино можливим ключем відношення є складений атрибут {Em_Number, Pr_Number }. Діаграма мінімальної множини ER показана на рис.6.1.

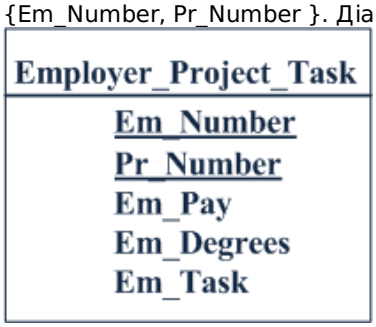


Рисунок 6.1 ER-діаграма відношення Employer_Project_Task

Стосовно нашого прикладу:

- додавання кортежів – ми не можемо доповнити відношення Employer_Project_Task даними про службовця, який ще не бере участь у жодному проекті (Em_Number є частиною первинного ключа й не може містити невизначених значень). Тим часом часто буває, що спочатку службовця беруть на роботу, встановлюють його розряд і розмір заробітної плати, а лише потім призначають для нього проект;
- видалення кортежів – ми не можемо зберегти у відношенні Employer_Project_Task дані про службовця, який завершив участь у своєму останньому проекті (з тієї причини, що значення атрибута Pr_Number для цього службовця стає невизначеним);
- модифікація кортежів – щоб змінити розряд службовця, ми будемо змушені модифікувати всі кортежі з відповідним значенням атрибута Em_Number. У іншому випадку буде порушений природний зв'язок Em_Number → Em_Degrees (в одного службовця є тільки один розряд).

Для подолання цих труднощів можна зробити декомпозицію змінного відношення Employer_Project_Task на два змінні відношення – Employer {Em_Number, Em_Degrees, Em_Pay} і Employer_Project_Task {Em_Number, Pr_Number, Em_Task}. На рис. 6.2 показані діаграми ER цих відношень. Тепер ми можемо легко впоратися з операціями відновлення.

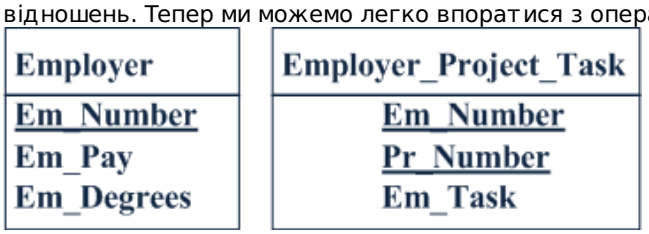


Рисунок 6.2 ER-діаграми у змінних відношеннях Employer і Employer_Project_Task

Друга нормальна форма відношення

Будемо вважати атрибут відношення ключовим, якщо він є елементом якого-небудь ключа відношення. В іншому випадку атрибут буде вважатися неключовим. Відношення перебуває у 2NF, якщо воно перебуває у 1NF, і всі неключові атрибути відношення функціонально мінімально залежать від первинного ключа. Іншими словами, 2NF вимагає, щоб відношення не містило часткових функціональних залежностей.

Приклад приведення відношення до 2НФСховати

Стосовно нашого прикладу: відношення Employer знаходиться у 2NF, а відношення Employer_Project_Task – ні, оскільки атрибут Em_Task функціонально залежить від двох ключових атрибутів: Em_Number та Pr_Number. Будь-яке змінне відношення, що перебуває у 1NF, але не перебуває у 2NF, може бути зведене до набору змінних відношень, що перебувають у 2NF. У результаті декомпозиції ми одержуємо набір проекцій вихідного змінного відношення, природне з'єднання значень яких відтворює значення вихідного змінного відношення (тобто це декомпозиція без втрат).

Третя нормальна форма відношення

Відношення перебуває у 3NF, якщо воно перебуває в 2NF, і всі неключові атрибути відношення залежать тільки від первинного ключа. Іншими словами, 3NF вимагає, щоб відношення не містило транзитивних функціонального зв'язку неключових атрибутів від ключа.

Приклад приведення відношення до 3 НФСховати

Функціональні залежності відношення Employer як і раніше породжують деякі аномалії відновлення. Вони викликаються наявністю транзитивного зв'язку Em_Number → Em_Pay (через зв'язок Em_Number → Em_Degrees і Em_Degrees → Em_Pay). Ці аномалії пов'язані з надмірністю зберігання значення атрибута Em_Pay у кожному кортежі, що характеризує службовців із тим самим розрядом:

- додавання кортежів – неможливо зберегти дані про новий розряд (і відповідному йому розміру зарплати), поки не з'явиться службовець із новим розрядом. Первинний ключ не може містити невизначені значення;
- видалення кортежів – при звільненні останнього службовця з даним розрядом ми втратимо інформацію про наявність такого розряду й відповідному розміру зарплати;
- модифікація кортежів – при зміні розміру зарплати, що відповідає деякому розряду, ми будемо змушені змінити значення атрибута Em_Pay у кортежах всіх службовців, яким призначений цей розряд (інакше не буде виконуватися зв'язок Em_Degrees → Em_Pay).

Можлива декомпозиція: для подолання цих труднощів зробимо декомпозицію змінного відношення Employer на два змінні відношення – Employer1 {Em_Number, Em_Degrees} й Degrees {Em_Degrees, Em_Pay}. На рис. 6.3 показані ER-діаграми цих змінних відношень.

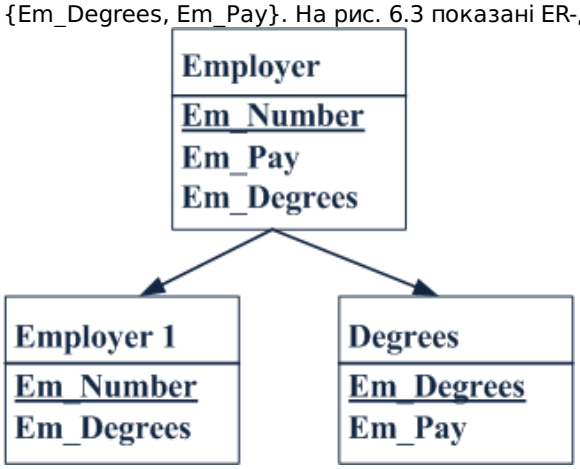


Рисунок 6.3 ER-діаграми у змінних відношеннях Employer1 і Degrees

Таким чином, процедура зведення відношення до 3NF складається у виконанні двох проекцій: по правій і по лівій частині транзитивного функціонального зв’язку.

Зрозуміло, що в процесі нормалізації декомпозиція відношення на незалежні проекції є кращою. Необхідні й достатні умови незалежності проекцій відношення забезпечує теорема Риссанена: проекції r_1 і r_2 відношення r є незалежними тоді й тільки тоді, коли кожний зв'язок у відношенні r логічно виходить зі зв'язку у r_1 і r_2; загальні атрибути r_1 і r_2 утворять можливий ключ хоча б для одного з цих відношень.

ПрикладСховати

Проілюструємо вірність цієї теореми на прикладі декомпозиції відношення Employer. У декомпозиції на проекції Employer1 і Degrees загальний атрибут Em_Degrees є можливим (і первинним) ключем відношення Degrees, а єдиний додатковий зв’язок відношення Employer (Em_Number→Em_Pay) логічно виходить зі зв'язку Em_Number→Em_Degrees і Em_Degrees→Em_Pay, які виконуються для відношення Employer1 й Degrees відповідно.

Атомарним відношенням називається відношення, яке неможливо декомпозувати на незалежні проекції. Далеко не завжди для неатомарних відношень потрібна декомпозиція на атомарні проекції. При виборі способу декомпозиції необхідно прагнути до одержання незалежних проекцій, але не обов'язково атомарних.

Нормальна форма Бойса-Кодда

Змінна відношення перебуває в нормальній формі Бойса-Кодда (BCNF) у тому і тільки в тому випадку, коли будь-який виконуваний для цього змінного відношення нетривіальний і мінімальний функціональний зв'язок має як детермінант деякий можливий ключ даного відношення.

Приклад приведення відношення до НФ Бойса-КоддаСховати

Наприклад, нехай є змінне відношення Employer_Project_Task1 { Em_Number Em_Name, Pr_Number, Em_Task} з множиною зв'язків, зображених на рис. 6.4.

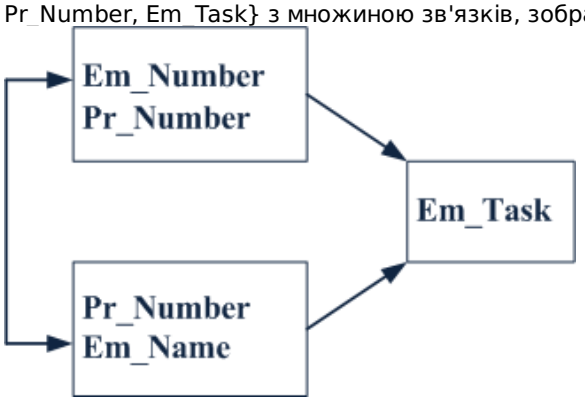


Рисунок 6.4 Діаграма функціонального зв’язку відношення Employer_Project_Task1

У відношенні Employer_Project_Task1 службовці унікально ідентифікуються як за номерами

справ, так і за іменами. Отже, існують зв'язки Em_Number→Em_Name й Em_Name→Em_Number. Але один службовець може брати участь у декількох проектах, тому можливими ключами є {Em_Number, Pr_Number} і {Em_Name, Pr_Number}.

Очевидно, що, хоча у відношенні Employer_Project_Task1 всі зв'язки неключових атрибутів від можливих ключів є мінімальними й транзитивні зв'язки відсутні, цьому відношенню властиві аномалії відновлення. Наприклад, у випадку зміни імені службовця необхідно обновити атрибут Em_Name у всіх кортежах відношення, що відповідають даному службовцеві. Інакше буде порушений зв'язок Em_Number→Em_Name, і БД виявиться у неузгодженому стані.

Причиною відзначених аномалій є те, що у вимогах 2NF і 3NF не була потрібна мінімальна функціональна залежність від первинного ключа атрибутів, що є компонентами інших можливих ключів. Проблему вирішує нормальна форма, що історично прийнята називати нормальною формою Бойса-Кодда і яка є уточненням 3NF у випадку наявності декількох можливих ключів, що перекриваються.

Відношення Employer_Project_Task1 може бути наведене до BCNF шляхом однієї з двох декомпозицій: Employer_Number_Name {Em_Number, Em_Name} і Employer_Number_Project_Task {Em_Number, Pr_Number, Pr_Task}, і Employer_Number_Name {Em_Number, Em_Name} і Employer_Name_Project_Task {Em_Name, Pr_Number, Pr_Task}.

Четверта нормальна форма відношення

У змінній відношення r з атрибутами A, B, C (у загальному випадку складовими) є багатозначна залежність B від A (AB) в тому і тільки в тому випадку, коли множина значень атрибута B, що відповідає парі значень атрибутів A й C, залежить від значення A і не залежить від значення C. Багатозначні залежності мають цікаву властивість "подвійності", що демонструє така лема Фейджина:

У відношенні r {A, B, C} виконується MVD A→→B у тому і тільки в тому випадку, коли виконується MVD A→→C.

Функціональний зв'язок є частковим випадком MVD, коли множина значень залежного атрибута обов'язково складається з одного елемента. Таким чином, якщо виконується зв'язок A→B, то виконується й MVD A→→B.

Теорема Фейджина. Нехай є змінна відношення r з атрибутами A, B, C (у загальному випадку, складовими). Відношення r декомпозується без втрат на проекції {A, B} й {A, C} тоді й тільки тоді, коли для нього виконується MVD A→→B | C.

Відношення перебуває у 4NF, якщо воно перебуває в 3NF або BCNF і всі незалежні багатозначні функціональні зв'язки рознесені в окремі відношення з тим самим ключем. Іншими словами, 4NF застосовується при наявності у відношенні більш ніж однієї MVD і вимагає, щоб відношення не містило незалежних багатозначних MVD.

Приклад приведення відношення до 4НФСховати

Розглянемо ще одну можливу інтерпретацію змінної відношення Employer_Project_Task. Припустимо, що кожен службовець може брати участь у декількох проектах, але в кожному проекті ним повинні виконуватися ті самі завдання. Можливе значення змінної відношення Employer_Project_Task показано на рис.6.5.

Додавання кортежу – якщо службовець, який вже бере участь у проектах, приєднується до нового проекту, то до тіла значення змінної відношення Employer_Project_Task необхідно додати стільки кортежів, скільки завдань виконує цей службовець.

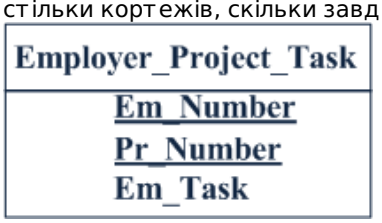


Рисунок 6.5 Можливе значення змінної відношення Employer_Project_Task

Видалення кортежів – якщо службовець припиняє участь у проектах, то відсутня можливість зберегти дані про завдання, які він може виконувати.

Модифікація кортежів – при зміні одного з завдань службовця необхідно змінити значення атрибута Em_Task у скількох кортежах, у скількох проектах бере участь службовець.

Труднощі, пов'язані з відновленням змінної відношення Employer_Project_Task, вирішуються шляхом його декомпозиції на два змінні відношення: Employer_Project_Number {Em_Number, Pr_Number} і Employer_Task {Em_Number, Em_Task}. Значення цих змінних відношень, що відповідають значенню змінної відношення Employer_Project_Task показані на рис. 6.6.

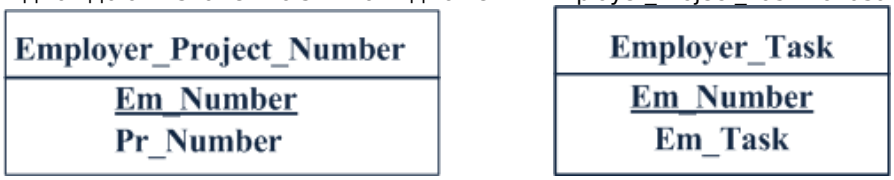


Рисунок 6.6 Діаграми відношень Employer_Project_Number і Employer_Task

Зверніть увагу, що останній варіант змінної відношення Employer_Project_Task в BCNF, оскільки всі атрибути заголовка відношення входять до складу єдино можливого ключа. Раніше обговорені принципи нормалізації тут не застосовані, але ми одержали корисну декомпозицію. Справа в тому, що у випадку останнього варіанта відношення, ми маємо справу з новим видом залежності, уперше виявленим Роном Фейджином у 1971р. Фейджин назвав залежності цього виду багатозначними (multi-valued dependency - MVD).

У відношенні Employer_Project_Task виконуються дві MVD: Em_Number→Pr_Number і Em_Number→Em_Task. Перша MVD означає, що кожному значенню атрибута Em_Number відповідає обумовлена тільки цим значенням множина значень атрибута Pr_Number. Аналогічно трактується друга MVD.

Ключові терміни:

аномалія відновлення, атомарне відношення, нормальна форма схеми відношень, нормалізація, нормалізація відношень, реляційна база даних, теорема Риссанена, теорема Фейджина

6.1 Нормальні форми відношення

Під реляційною БД прийнято розуміти сукупність екземплярів кінцевих відношень. Сукупність схем відношень утворює схему реляційної БД.

Встановлення функціональної залежності й одержання найкращого з погляду мінімальності подання множини функціональних залежностей дозволять побудувати найбільш оптимальний варіант БД, що забезпечує надійність зберігання й обробки даних на основі методів еквівалентних перетворень схем відношень реляційної БД. Процес вирішення такого завдання називається нормалізацією відношень інформаційної моделі ПО й полягає у перетворенні її об'єктів у логічні таблиці БД. Основні вимоги наведені нижче:

- первинні ключі відношень повинні бути мінімальними;
- число відношень БД повинне по можливості давати найменшу надмірність даних – вимога надійності даних;
- число відношень БД не повинне приводити до втрати продуктивності системи;
- дані не повинні бути суперечливими, тобто при виконанні операцій включення, видалення й відновлення даних їх потенційна суперечливість повинна бути зведена до мінімуму;
- схема відношень БД повинна бути стійкою, здатною адаптуватися до змін при її розширенні додатковими атрибутами – вимога гнучкості структури БД;
- розкид часу реакції на різні запити до БД не повинен бути великим;
- дані повинні правильно відбивати стан ПО БД у кожен конкретний момент часу – вимога актуальності даних.

Теорія функціональних залежностей дозволяє встановити певні вимоги до схем відношень у реляційній БД. Ці вимоги формулюються у термінах властивостей відношень і називаються нормальними формами схем відношень. Кожна нормальна форма відношень пов'язана з певним класом функціональної залежності, які представлені у відношеннях. Одним з очевидних засобів усунення потенційної суперечливості даних у відношеннях логічної моделі реляційної БД є їх розбиття на двоє або більше відношень, у кожному з яких є присутньою тільки одна функціональна залежність.

Процес усунення потенційної суперечливості й надмірності даних у відношеннях реляційної БД називається нормалізацією вихідних схем відношень. Нормалізація відношень полягає у виконанні декомпозиції відношень, що перебувають у попередній нормальній формі, на двоє або більше відношень, які задовольняють вимогам наступної нормальної форми.

У теорії реляційних БД звичайно виділяється така послідовність нормальних форм: перша нормальна форма (1NF); друга нормальна форма (2NF); третя нормальна форма (3NF); нормальна форма Бойса-Кодда (BCNF); четверта нормальна форма (4NF); п'ята нормальна форма, або нормальна форма проекції-з'єднання (5NF або PJ/NF).

Основні властивості нормальних форм полягають у такому: кожна наступна нормальна форма у деякому змісті краще попередньої нормальної форми; при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

6.2 Перша нормальна форма відношення

Відношення перебуває у 1NF, якщо всі атрибути відношення є простими (вимогу атомарності атрибутів), тобто не мають компонентів. Інколи у відношеннях деякі функціональні залежності атрибутів від можливого ключа не є мінімальними. Це призводить до так званих аномалій відновлення. Під аномаліями відновлення розуміються труднощі, з якими зустрічаються при виконанні операцій додавання кортежів у відношення (INSERT), видалення кортежів (DELETE) і модифікації кортежів (UPDATE).

Приклад приведення відношення до 1НФСховати
Нехай є змінне відношення: Employer_Project_Task {Em_Number, Em_Degrees, Em_Pay, Pr_Number, Em_Task}. Атрибути містять відповідно дані про номер справи, розряд та заробітну платню службовця, номер проекту й про завдання, що виконує службовець у даному проекті. Припустимо, що розряд службовця визначає розмір його заробітної плати й що кожен службовець може брати участь у декількох проектах за умови виконання тільки одного завдання. Тоді очевидно, що єдино можливим ключем відношення є складений атрибут {Em_Number, Pr_Number }. Діаграма мінімальної множини ER показана на рис.6.1.

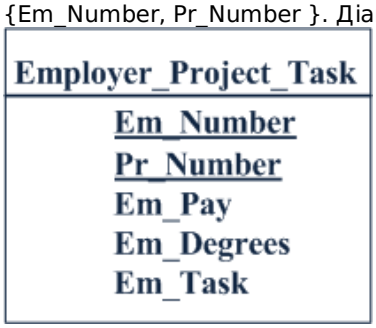


Рисунок 6.1 ER-діаграма відношення Employer_Project_Task

Для подолання цих труднощів можна зробити декомпозицію змінного відношення Employer_Project_Task на два змінні відношення – Employer {Em_Number, Em_Degrees, Em_Pay} і Employer_Project_Task {Em_Number, Pr_Number, Em_Task}. На рис. 6.2 показані діаграми ER цих відношень. Тепер ми можемо легко впоратися з операціями відновлення.

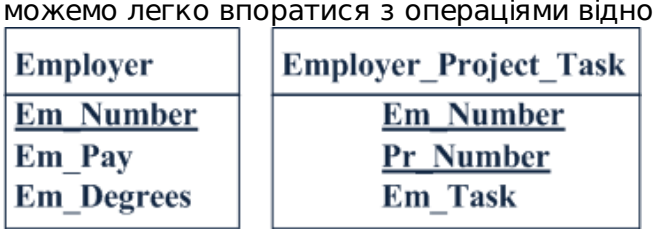


Рисунок 6.2 ER-діаграми у змінних відношеннях Employer і Employer_Project_Task

6.3 Друга нормальна форма відношення

В іншому випадку атрибут буде вважатися неключовим. Відношення перебуває у 2NF, якщо воно перебуває у 1NF, і всі неключові атрибути відношення функціонально мінімально залежать від первинного ключа. Іншими словами, 2NF вимагає, щоб відношення не містило часткових функціональних залежностей.
Приклад приведення відношення до 2НФСховати
Стосовно нашого прикладу: відношення Employer знаходиться у 2NF, а відношення Employer_Project_Task – ні, оскільки атрибут Em_Task функціонально залежить від двох ключових атрибутів: Em_Number та Pr_Number. Будь-яке змінне відношення, що перебуває у 1NF, але не перебуває у 2NF, може бути зведене до набору змінних відношень, що перебувають у 2NF. У результаті декомпозиції ми одержуємо набір проекцій вихідного змінного відношення, природне з'єднання значень яких відтворює значення вихідного змінного відношення (тобто це декомпозиція без втрат).

6.4 Третя нормальна форма відношення

Відношення перебуває у 3NF, якщо воно перебуває в 2NF, і всі неключові атрибути відношення залежать тільки від первинного ключа. Іншими словами, 3NF вимагає, щоб відношення не містило транзитивних функціонального зв’язку неключових атрибутів від ключа.
Приклад приведення відношення до 3 НФСховати
Функціональні залежності відношення Employer як і раніше породжують деякі аномалії відновлення. Вони викликаються наявністю транзитивного зв’язку Em_Number → Em_Pay (через зв'язок Em_Number → Em_Degrees і Em_Degrees → Em_Pay). Ці аномалії пов'язані з надмірністю зберігання значення атрибута Em_Pay у кожному кортежі, що характеризує службовців із тим самим розрядом.
Можлива декомпозиція: для подолання цих труднощів зробимо декомпозицію змінного відношення Employer на два змінні відношення – Employer1 {Em_Number, Em_Degrees} й Degrees

{Em_Degrees, Em_Pay}. На рис. 6.3 показані ER-діаграми цих змінних відношень.

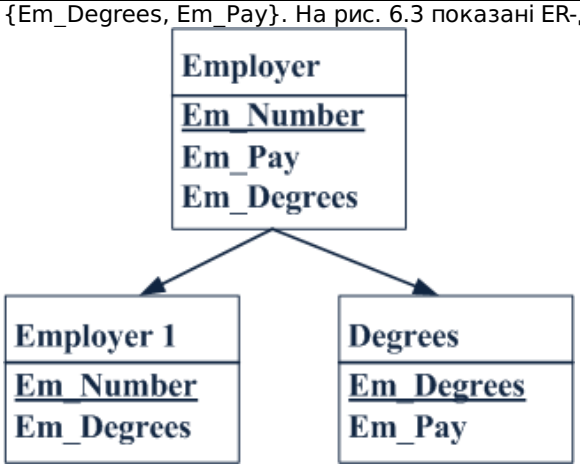


Рисунок 6.3 ER-діаграми у змінних відношеннях Employer1 і Degrees

Таким чином, процедура зведення відношення до 3NF складається у виконанні двох проекцій: по правій і по лівій частині транзитивного функціонального зв'язку.

Зрозуміло, що в процесі нормалізації декомпозиція відношення на незалежні проекції є кращою. Необхідні й достатні умови незалежності проекцій відношення забезпечує теорема Риссанена: проекції r_1 і r_2 відношення r є незалежними тоді й тільки тоді, коли кожний зв'язок у відношенні r логічно виходить зі зв'язку у r_1 і r_2; загальні атрибути r_1 і r_2 утворять можливий ключ хоча б для одного з цих відношень.

Атомарним відношенням називається відношення, яке неможливо декомпозувати на незалежні проекції. Далеко не завжди для неатомарних відношень потрібна декомпозиція на атомарні проекції. При виборі способу декомпозиції необхідно прагнути до одержання незалежних проекцій, але не обов'язково атомарних.

6.5 Нормальна форма Бойса-Кодда

Змінна відношення перебуває в нормальній формі Бойса-Кодда (BCNF) у тому і тільки в тому випадку, коли будь-який виконуваний для цього змінного відношення нетривіальний і мінімальний функціональний зв'язок має як детермінант деякий можливий ключ даного відношення.

Приклад приведення відношення до НФ Бойса-КоддаСховати

Наприклад, нехай є змінне відношення Employer_Project_Task1 { Em_Number Em_Name, Pr_Number, Em_Task} з множиною зв'язків, зображених на рис. 6.4.

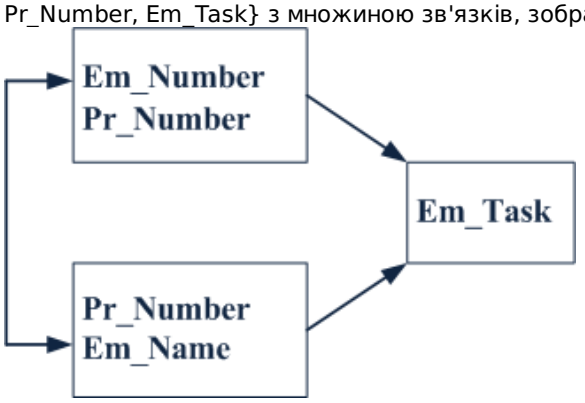


Рисунок 6.4 Діаграма функціонального зв'язку відношення Employer_Project_Task1

У відношенні Employer_Project_Task1 службовці унікально ідентифікуються як за номерами справ, так і за іменами. Отже, існують зв'язки Em_Number→Em_Name й Em_Name→Em_Number. Але один службовець може брати участь у декількох проектах, тому можливими ключами є {Em_Number, Pr_Number} і {Em_Name, Pr_Number}.

Очевидно, що, хоча у відношенні Employer_Project_Task1 всі зв'язки неключових атрибутів від можливих ключів є мінімальними й транзитивні зв'язки відсутні, цьому відношенню властиві аномалії відновлення. Наприклад, у випадку зміни імені службовця необхідно обновити атрибут Em_Name у всіх кортежах відношення, що відповідають даному службовцеві. Інакше буде порушений зв'язок Em_Number→Em_Name, і БД виявиться у неузгодженому стані.

Причиною відзначених аномалій є те, що у вимогах 2NF і 3NF не була потрібна мінімальна функціональна залежність від первинного ключа атрибутів, що є компонентами інших можливих ключів. Проблему вирішує нормальна форма, що історично прийнята називати нормальною формою Бойса-Кодда і яка є уточненням 3NF у випадку наявності декількох можливих ключів, що перекриваються.

Відношення Employer_Project_Task1 може бути наведене до BCNF шляхом однієї з двох декомпозицій: Employer_Number_Name {Em_Number, Em_Name} і Employer_Number_Project_Task {Em_Number, Pr_Number, Pr_Task}, і

Employer_Number_Name {Em_Number, Em_Name} і Employer_Name_Project_Task {Em_Name, Pr_Number, Pr_Task}.

6.6 Четверта нормальна форма відношення

У змінній відношення r з атрибутами A, B, C (у загальному випадку складовими) є багатозначна залежність B від A (AB) в тому і тільки в тому випадку, коли множина значень атрибута B, що відповідає парі значень атрибутів A й C, залежить від значення A і не залежить від значення C. Багатозначні залежності мають цікаву властивість "подвійності", що демонструє така лема Фейджина:

У відношенні r {A, B, C} виконується MVD A→→B у тому і тільки в тому випадку, коли виконується MVD A→→C.

Функціональний зв'язок є частковим випадком MVD, коли множина значень залежного атрибута обов'язково складається з одного елемента. Таким чином, якщо виконується зв'язок A→B, то виконується й MVD A→→B.

Теорема Фейджина. Нехай є змінна відношення r з атрибутами A, B, C (у загальному випадку, складовими). Відношення r декомпозується без втрат на проекції {A, B} й {A, C} тоді й тільки тоді, коли для нього виконується MVD A→→B | C.

Відношення перебуває у 4NF, якщо воно перебуває в 3NF або BCNF і всі незалежні багатозначні функціональні зв’язки рознесені в окремі відношення з тим самим ключем. Іншими словами, 4NF застосовується при наявності у відношенні більш ніж однієї MVD і вимагає, щоб відношення не містило незалежних багатозначних MVD.

Приклад приведення відношення до 4НФСховати

Розглянемо ще одну можливу інтерпретацію змінної відношення Employer_Project_Task. Припустимо, що кожен службовець може брати участь у декількох проектах, але в кожному проекті ним повинні виконуватися ті самі завдання. Можливе значення змінної відношення Employer_Project_Task показано на рис.6.5.

Employer_Project_Task
Em_Number
Pr_Number
Em_Task

Рисунок 6.5 Можливе значення змінної відношення Employer_Project_Task

Труднощі, пов'язані з відновленням змінної відношення Employer_Project_Task, вирішуються шляхом його декомпозиції на два змінні відношення: Employer_Project_Number {Em_Number, Pr_Number} і Employer_Task {Em_Number, Em_Task}. Значення цих змінних відношень, що відповідають значенню змінної відношення Employer_Project_Task показані на рис. 6.6.

Employer_Project_Number
Em_Number
Pr_Number

Employer_Task
Em_Number
Em_Task

Рисунок 6.6 Діаграми відношень Employer_Project_Number і Employer_Task

У відношенні Employer_Project_Task виконуються дві MVD: Em_Number→Pr_Number і Em_Number→Em_Task. Перша MVD означає, що кожному значенню атрибута Em_Number відповідає обумовлена тільки цим значенням множина значень атрибута Pr_Number. Аналогічно трактується друга MVD.

Виявлення всіх залежностей з'єднання є нетривіальним завданням, і для його вирішення немає загальних методів. Тому на практиці проектування реляйціних БД методом нормалізації звичайно завершується після досягнення 4NF.

Нормальні форми характеризуються такими властивостями:

- 1NF - всі атрибути відношення прості;
- 2NF - відношення перебуває в 1NF і не містить часткових залежностей;
- 3NF - відношення перебуває в 2NF і не містить транзитивних залежностей від ключа;
- BKNF - відношення перебуває в 3NF і не містить залежностей ключів від неключових атрибутів;
- 4NF - застосовується при наявності більш ніж одної багатозначної залежності - відношення перебуває в BKNF або 3NF і не містить незалежних багатозначних функціональних залежностей.

1. Назвіть вимоги до логічної моделі даних?

2. Поясніть поняття «нормалізації».
3. Які відношення називаються нормалізованими?
4. Вкажіть ознаки першої нормальної форми.
5. Вкажіть ознаки другої нормальної форми.
6. Вкажіть ознаки третьої нормальної форми.
7. Вкажіть ознаки четвертої нормальної форми.
8. Вкажіть ознаки нормальної форми Бойса-Кодда.
9. Відповідність якій нормальній формі завершує нормалізацію схеми БД?

Тема 7 - Введення в структуровану мову запитів SQL

Ключові терміни:

кореліруємий підзапит, оператор DDL, оператор DML, подання

Мова SQL оперує термінами, що трохи відрізняються від термінів реляційної теорії, наприклад, замість "відношення" використовуються "таблиці", замість "кортежів" - "рядки", замість "атрибутів" - "колонки" або "стовпці".

Стандарт мови SQL, хоча й заснований на реляційній теорії, але у багатьох місцях відходить він неї. Наприклад, відношення у реляційній моделі даних не допускає наявності однакових кортежів, а таблиці у термінології SQL можуть мати однакові рядки. Є й інші відмінності.

Мова SQL є реляційно повною. Це означає, що будь-який оператор реляційної алгебри може бути виражений підходящим оператором SQL.

7.1 Припустимі типи даних

Будь-який діалект SQL підтримують три загальних типи даних: строковий, числовий й тип для подання дати й часу. Завдання типу даних визначає значення й довжину даних, а також формат їхнього подання при візуалізації.

Для всіх типів даних визначено так зване нуль-значення, що вказує на відсутність даних у колонку зазначеного типу, тобто та обставина, що значення даних у сучасний момент часу невідомо.

Дані строкового типу являють собою послідовність рядків символів. Строкові дані можуть бути задані як з визначеною довжиною (ключові слова char або varchar (довжина рядка)), так і без вказівки довжини (ключове слово long varchar) для подання рядків довільної довжини.

Числові типи даних призначені для подання цілих чисел, чисел з десятковою крапкою й чисел із плаваючою крапкою. Будь-яке подання чисел задається своєю точністю й масштабом. Точність визначає припустиме подання кількосты значущих цифр числа, а масштаб - кількість значущих цифр після десяткової крапки.

Для подання цілих чисел використовуються типи interger (точність 10 значущих цифр) і smallint (точність 5 значущих цифр).

Для подання чисел з фіксованою десятковою крапкою використовуються типи number і decimal.

Для подання чисел із плаваючою крапкою в SQL передбачені такі типи даних:

- Double Precision - для чисел з точністю від 22 до 53 значущих цифр;
- Float (точність) - для подання чисел з точністю від 1 до 21 значущої цифри;

- Real - для чисел з точністю за замовчуванням (залежить від конкретної реалізації).

Звичайно в конкретних діалектах SQL використовуються три типи для подання таких даних:

- datestamp (timestamp) - для подання дати й часу;
- date - для подання дати;
- time - для подання часу.

Константи, вираження, системні змінні. Константи зазвичай визначають єдине значення й, відповідно до типу даних, що представляють, можуть бути строковими, числовими й представляти дату/час. Строкові константи повинні бути укладені в одинарні лапки.

В SQL існує ряд визначених системних змінних, які можна використовувати у виразах замість імен колонок і констант. До таких змінних відноять наступні:

- NULL - для подання невизначених значень;
- ROWID - (в SQLBase) внутрішній системний номер рядка в таблиці;
- USER - ім'я користувача, активного в цей момент;
- SYSDATETIME - системний поточний час і дата;
- SYSDATE - системна поточна дата;
- SYSTIME - системний поточний час;
- SYSTIMEZONE - часовий пояс, установлений у системі.

Виразом в SQL є ітем або комбінація ітемів з припустимими для них операціями, що дає єдине значення. У якості ітемів можуть виступати імена колонок, константи, зв'язані змінні, результати обчислень функцій, системні змінні й інші вирази. При цьому якщо один з ітемів має нуль-значення, то результат виразу також має нуль-значення.

7.2 Використання операторів мови SQL

7.2.1 Оператори SQL

Основу мови SQL становлять оператори, умовно розбиті на кілька груп за функціональним показником.

Оператори DDL (Data Definition Language) - оператори визначення об'єктів БД

- CREATE SCHEMA - створити схему БД
- DROP SHEMA - видалити схему БД
- CREATE TABLE - створити таблицю
- ALTER TABLE - змінити таблицю
- DROP TABLE - видалити таблицю
- CREATE DOMAIN - створити домен
- ALTER DOMAIN - змінити домен
- DROP DOMAIN - видалити домен
- CREATE COLLATION - створити послідовність
- DROP COLLATION - видалити послідовність
- CREATE VIEW - створити подання
- DROP VIEW - видалити подання

Оператори DML (Data Manipulation Language) - оператори маніпулювання даними

- SELECT - відібрати рядок з таблиць
- INSERT - додати рядок в таблицю
- UPDATE - змінити рядок в таблиці
- DELETE - видалити ряодк в таблиці
- COMMIT - зафіксувати внесені зміни
- ROLLBACK - відкотити внесені зміни

Оператори захисту й керування даними

- CREATE ASSERTION - створити обмеження
- DROP ASSERTION - видалити обмеження
- GRANT - надати привілею користувачеві або додатку на маніпулювання об'єктами
- REVOKE - скасувати привілею користувача або додатка

Найбільш важливими для користувача є оператори маніпулювання даними (DML).

7.2.2 Оператори маніпулювання даними

INSERT - вставка рядків у таблицю
Вставка одного рядка до таблиціСховати
INSERT INTO
P (PNUM, PNAME)
VALUES (4, "Іванов");

UPDATE - відновлення рядків у таблиці
Відновлення декількох рядківСховати
UPDATE P
SET PNAME = "Пушников"
WHERE P.PNUM = 1;

DELETE - видалення рядків у таблиці
Видалення рядків з таблиці...Сховати

DELETE FROM P;

7.2.3 Виборка даних

Оператор SELECT є фактично найважливішим для користувача й самим складним оператором SQL. Оператор SELECT завжди виконується над деякими таблицями, що входять у БД.

Насправді в БД можуть бути не тільки постійно збережені таблиці, а також тимчасові таблиці й так називані подання. Подання - це SELECT-вираз, що просто зберігається в БД. З погляду користувачів подання - це таблиця, яка не зберігається постійно в БД, а "виникає" у момент звертання до неї. З погляду оператора SELECT і постійно збережені таблиці, і тимчасові таблиці й подання виглядають зовсім однаково.

Результатом виконання оператора SELECT завжди є таблиця. Таким чином, за результатами дій оператор SELECT схожий на оператори реляційної алгебри. Будь-який оператор реляційної алгебри може бути виражений певним чином сформульованим оператором SELECT. Складність оператора SELECT визначається тим, що він містить у собі всі можливості реляційної алгебри, а також додаткові можливості, яких у реляційній алгебрі немає

7.2.3.1. Відбір даних з однієї таблиці

Ключові слова, що використовуються: ***SELECT***...***FROM***...
Вибірка всіх даних з таблиціСховати
SELECT *
FROM P;
В результаті одержимо нову таблицю, що містить повну копію даних з вихідної таблиці P.

Виборка даних з таблиць, які задовольняють певній умові: ключове слово ***WHERE.*** Як умову в розділі WHERE можна використовувати складні логічні вирази, що використовують поля таблиць, константи, порівняння (>, <, = і т.д.), дужки, союзи AND й OR, заперечення NOT.
Вибірка рядків за умовоюСховати
SELECT*
FROM P
WHERE P...PNUM>2;

Вибірка деяких колонок з вихідної таблиці реалізується вказівкою списку потрібних колонок
Вибірка визначених колонок з таблиціСховати
SELECT P.NAME
FROM P;
В результаті одержимо таблицю з однією колонкою, що містить всі найменування постачальників.

Якщо у вихідній таблиці було присутнє кілька рядків з різними номерами, але однаковими найменуваннями, то в результуючій таблиці будуть рядки з повтореннями - дублікати рядків автоматично не відкидаються.

Вибірка даних без дублювання даних досягається використанням ключового слова ***DISTINCT.*** Використання ключового слова DISTINCT приводить до того, що у результуючій таблиці будуть вилучені всі повторювані рядки.
Вибірка даних без дублювання інформаціїСховати
SELECT DISTINCT P.NAME
FROM P;

Використання ключового слова ***AS*** дозволяє проіменувати колонки в результуючій таблиці
Приклад використання скалярних виразів і перейменувань колоноСховати
SELECT TOVAR...TNAME, TOVAR.KOL,

```
TOVAR.PRICE, "="AS EQU,
TOVAR.KOL*TOVAR.PRICE AS SUMMA
FROM TOVAR;
```

В результаті одержимо таблицю з колонками, яких не було у вихідній таблиці TOVAR:

TNAME	KOL	PRICE	EQU	SUMMA
Болт	10	100	=	1000
Гайка	20	200	=	4000
Гвинт	30	300	=	9000

Існує можливість упорядкувати результати запитів по полю за допомогою використання ключового слова ***ORDER BY*** та по декількох полях за зростанням або зменшенням за допомогою ключових слів ***ASC***, ***DESC***). Якщо явно не зазначені ключові слова ASC або DESC, то за замовчуванням приймається впорядкування по зростанню (ASC).

Приклади вибірки даних з упорядкуванням результатів запитівСховати

```
SELECT PD...PNUM, PD.DNUM, PD.VOLUME
FROM PD
ORDER BY DNUM;
```

В результаті одержимо наступну таблицю, впорядковану по полю DNUM:

PNUM	DNUM	VOLUME
1	1	100
2	1	150
1	2	200
1	3	300

```
SELECT PD.PNUM, PD.DNUM, PD.VOLUME
FROM PD
ORDER BY DNUM ASC, VOLUME DESC;
```

В результаті одержимо таблицю, у якій рядки йдуть у порядку зростання значення поля DNUM, а рядки, з однаковим значенням DNUM, йдуть в порядку зменшення значення поля VOLUME:

PNUM	DNUM	VOLUME
2	1	150
1	1	100
1	2	200
1	3	300

7.2.3.2. Відбір даних з декількох таблиць

Існує декілька способів отримати дані з декількох таблиць.

Природне з'єднання таблиць дозволяє отримати дані шляхом явної вказівки умов з'єднання. Таблиці, що з'єднують, перераховані у розділі ***FROM*** оператора, умова з'єднання наведена у розділі ***WHERE***. Розділ ***WHERE***, крім умови з'єднання таблиць, може також містити й умови відбору рядків.

Приклад синтаксису запитаСховати

```
SELECT P.PNUM, P.PNAME, PD.DNUM,
PD.VOLUME
FROM P, PD
WHERE P.PNUM = PD.PNUM;
```

В результаті одержимо нову таблицю, у якій рядки з даними про постачальників з'єднані з рядками з даними про поставки деталей:

PNUM	PNAME	DNUM	VOLUME
1	Іванов	1	100
1	Іванов	2	200
1	Іванов	3	300
2	Петров	1	150
2	Петров	2	250
3	Сидоров	1	1000

Другий спосіб - природне з'єднання таблиць (спосіб 2 - ключові слова ***JOIN... USING...***). Ключове слово USING дозволяє явно вказати, по яким із загальних колонок таблиць буде вироблятися з'єднання.

Приклад синтаксису запитаСховати

```
SELECT PNUM, P.PNAME, PD.DNUM,
PD.VOLUME
```

```
FROM P JOIN PD USING PNUM;
```

Третій спосіб - природне з'єднання таблиць (спосіб 3 - ключове слово ***NATURAL JOIN***). У розділі FROM не зазначається, по яких полях виробляється з'єднання. NATURAL JOIN автоматично з'єднує по всіх однакових полях у таблицях.

Приклад синтаксису запитаСховати

```
SELECT P.PNUM, P.PNAME, PD.DNUM,
   PD.VOLUME
   FROM P NATURAL JOIN PD;
```

7.2.4 Вбудовані функції

Арифметичні функції та функції обробки дати й часу

SQL підтримує повний набір арифметичних операцій і математичних функцій для побудови арифметичних виражень над колонками БД (+, -, *, /, ABS, LN, SQRT і т.д.). Перелік основних вбудованих математичних функцій та функцій по роботі з даними типу дата/час ви можете знайти в літературних джерелах, перелік яких наданий наприкінці модуля.

Арифметичні вирази необхідні для одержання даних, які безпосередньо не зберігаються в колонках таблиць БД, але значення яких необхідні користувачеві.

Приклад використання арифметичних функцій та функцій обробки датиСховати

Якщо вам потрібен був список нових службовців, що надійшли за останній квартал в організацію, то ви можете написати запит у наступному виді:

```
SELECT ENAME, HIREDATE,
HIREDATE + 92 DAYS
FROM EMPLOYEE
WHERE HIREDATE + 92 DAYS > SYSDATE
AND DEPNO=30;
```

Ключове слово ***SYSDATE*** завжди повертає поточну дату. У цьому прикладі також показано, як використається арифметичний оператор додавання зі змінними типу "дата". До змінного типу "дата" можна додавати й віднімати з нього ціле число днів, місяців, років, годин, хвилин, секунд, мікросекунд. Для цього використаються відповідні ключові слова (DAY, MONTH і т.д.), що випливають за цілою константою (дробова частина ігнорується, якщо ви вказуєте число з десятковою крапкою). Є обмеження на використання дужок у таких вираженнях (так, висновок у дужки вираження 1 DAYS + 1 YEARS приведе до помилки).

Використання агрегатних функцій у запитах

В мові SQL передбачені такі оператори агрегатних функцій:

AVG(X) = AVG(ALL X) AVG(DISTINCT X) Обчислює середнє значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати.

COUNT(*) COUNT(X) = COUNT(ALL X) COUNT(DISTINCT X) Обчислює числа ітемів. При вказівці * завжди повертається число рядків у таблиці. Вказівка DISTINCT придушує дублікати.

MAX(X) = MAX(ALL X) MAX (DISTINCT X) Обчислює максимальне значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

MIN(X) = MIN(ALL X) MIN (DISTINCT X) Обчислює мінімальне значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

SUM(X) = SUM(ALL X) SUM (DISTINCT X) Обчислює суму значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

STDDEV([DISTINCT|ALL]X) Обчислює стандартне відхилення на безлічі значень аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

VARIANCE([DISTINCT|ALL]) Обчислює квадрат дисперсії

Приклади синтаксису запитівСховати

Запит: одержати загальну кількість постачальників (ключове слово ***COUNT***):

```
SELECT COUNT(*) AS N
FROM P;
```

В результаті одержимо таблицю з одним стовпцем й одним рядком, що містить кількість рядків з таблиці P.

Запит: одержати загальну, максимальну, мінімальну й середню кількості деталей, що поставляють, (ключові слова SUM, MAX, MIN, AVG):

```
SELECT SUM(PD.VOLUME) AS SM,
MAX(PD.VOLUME) AS MX,
MIN(PD.VOLUME) AS MN,
AVG(PD.VOLUME) AS AV
FROM PD;
```

В результаті одержимо наступну таблицю з одним рядком:

SM	MX	MN	AV
2000	1000	100	333,33

Використання агрегатних функцій з угрупованнями

Для виведення результатів даних, сгрупованих за певною умовою, слід використовувати два ключових слова: ***GROUP BY*** та ***HAVING.***

В переліку полів оператора SELECT, який містить розділ GROUP BY можна включати тільки агрегатні функції й поля, які входять в умову групування.

Умови, що використовують агрегатні функції повинні бути розміщені у спеціальному розділі HAVING. В одному запиті можуть зустрітися як умови відбору рядків у розділі WHERE, так й умови відбору груп у розділі HAVING. Умови відбору груп не можна перенести з розділу HAVING у розділ WHERE. Аналогічно й умови відбору рядків не можна перенести з розділу WHERE у розділ HAVING, за винятком умов, що включають поля зі списку угруповання GROUP BY.

Приклади синтаксису запитівСховати

Запита: для кожної деталі одержати сумарну кількість, що поставляється (ключове слово ***GROUP BY***...):

```
SELECT..DNUM, SUM(PD.VOLUME) AS SM
GROUP BY PD.DNUM;
```

Цей запит буде виконуватися в такий спосіб. Спочатку рядки вихідної таблиці будуть згруповані так, щоб у кожну групу потрапили рядки з однаковими значеннями DNUM. Потім усередині кожної групи буде просумовано поле VOLUME. Від кожної групи до результуючої таблиці буде включений один рядок.

Запит: одержати номери деталей, сумарна кількість поставки яких перевершує 400 (ключове слово ***HAVING***...):

```
SELECT PD.DNUM, SUM(PD.VOLUME) AS SM
GROUP BY PD.DNUM
HAVING SUM(PD.VOLUME) > 400;
```

Умова, що сумарна кількість поставки повинна бути більше 400, не може бути сформульована у розділі WHERE, тому що в цьому розділі не можна використовувати агрегатні функції. Умови, що використовують агрегатні функції повинні бути розміщені у спеціальному розділі HAVING.

7.2.5 Використання підзапитів

Дуже зручним засобом, що дозволяє формулювати запити більш зрозумілим чином, є можливість використання підзапитів, вкладених в основний запит.

Самий простий спосіб виконання запиту прямою вказівкою в розділі WHERE вкладеного запиту.

Приклад синтаксису запитаСховати

Запит: одержати список постачальників, статус яких менше максимального статусу у таблиці постачальників (порівняння з підзапитом):

```
SELECT *
FROM P
WHERE P.STATYS < (SELECT MAX(P.STATUS) FROM P);
```

Тоді як поле P.STATUS рівняється з результатом підзапиту, то підзапит повинен бути сформульований так, щоб повертати таблицю, що складається рівно з одного рядка й однієї колонки.

Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Виконати один раз вкладений підзапит й одержати максимальне значення статусу.
2. Просканувати таблицю постачальників P, щоразу порівнюючи значення статусу постачальника з результатом підзапиту, і відібрати тільки ті рядки, в яких статус менше максимального.

Другий спосіб формування підзапиту - використання предиката ***IN***.

Приклад синтаксису запитаСховати

Запит: одержати перелік постачальників, що поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE P.PNUM IN (SELECT DISTINCT PD.PNUM
                 FROM PD WHERE PD.DNUM = 2);
```

У цьому випадку вкладений підзапит може повертати таблицю, що містить кілька рядків.

Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Виконати один раз вкладений підзапит й одержати список номерів постачальників, що поставляють деталь номер 2.
2. Просканувати таблицю постачальників P, щораз перевіряючи, чи втримується номер постачальника в результаті підзапиту.

Існує ще один спосіб - використання предиката ***EXIST***.
Приклад синтаксису запитаСховати
Запит: одержати перелік постачальників, що поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE EXIST (SELECT *
             FROM PD
             WHERE PD.PNUM = P.PNUM AND PD.DNUM = 2);
```

Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Просканувати таблицю постачальників P, щоразу виконуючи підзапит з новим значенням номера постачальника, взятим з таблиці P.
2. В результат запиту включити тільки ті рядки з таблиці постачальників, для яких вкладений підзапит повернув непусту множину рядків.

Коли вкладений підзапит містить параметр (зовнішнє посилання), переданий з основного запиту, такі підзапити називаються кореліруємими (*correlated*). Зовнішнє посилання може приймати різні значення для кожного рядка-кандидата, оцінюваного за допомогою підзапиту, тому підзапит повинен виконуватися заново для кожного рядка, який відбирається в основному запиті. Такі підзапити характерні для предиката EXIST, але можуть бути використані й в інших підзапитах.

Може здатися, що запити, які містять кореліруєми підзапити будуть виконуватися повільніше, ніж запити з некореліруємими підзапитами. Насправді це не так, тому що те, як користувач сформулював запит, не визначає, як цей запит буде виконуватися. Мова SQL є не процедурною, а декларативною. Це значить, що користувач, який формулює запит, просто описує, яким повинен бути результат запиту, а як цей результат буде отриманий - за це відповідає сама СКБД.

Приклад синтаксису запита з використанням предиката ***NOT EXIST***Сховати
Запит: одержати перелік постачальників, що не поставляють деталь номер 2:

```
SELECT *
FROM P
WHERE NOT EXIST (SELECT *
                 FROM PD
                 WHERE PD.PNUM = P.PNUM AND PD.DNUM = 2);
```

Також як й у попередньому прикладі, тут використовується кореліреємий підзапит. Відмінність у тому, що в основному запиті будуть відібрані ті рядки з таблиці постачальників, для яких вкладений підзапит не видасть жодного рядка.

7.2.6. Використання об'єднання, перетинання й різниці

Результуючі таблиці поєднуваних запитів повинні бути сумісні, тобто мати однакову кількість стовпців й однакові типи стовпців у порядку їхнього перерахування. Не потрібно, щоб поєднувані таблиці мали б однакові імена колонок. Це відрізняє операцію об'єднання запитів в SQL від операції об'єднання у реляційній алгебрі. Найменування колонок у результуючому запиті будуть автоматично взяті з результату першого запиту в об'єднанні.

Для виконання об'єднання двох підзапитів використовуються ключові слова ***UNION*** (об'єднання) та ***INTERSECT*** (перетин) та ***EXCEPT*** (різниця).
Приклади синтаксису запитівСховати
Запит: одержати імена постачальників, що мають статус, більший 3-ого або що поставляють хоча б одну деталь номер 2

```
SELECT P.PNAME
FROM P
WHERE P.STATUS > 3 UNION SELECT P.PNAME
                         FROM P, PD
                         WHERE P.PNUM = PD.PNUM AND PD.DNUM = 2;
```

Запит: одержати імена постачальників, що мають статус, більший 3-ого й одночасно поставляють хоча б одну деталь номер 2:

```
SELECT P.PNAME
FROM P
WHERE P.STATUS > 3 INTERSECT SELECT P.PNAME
                             FROM P, PD
                             WHERE P.PNUM = PD.PNUM AND PD.DNUM = 2;
```

Запит: одержати імена постачальників, що мають статус, більший 3-ого, за винятком тих, хто поставляє хоча б одну деталь номер 2:

```
SELECT P.PNAME
FROM P
WHERE P.STATUS > 3 EXCEPT SELECT P.PNAME
                          FROM P, PD
                          WHERE P.PNUM = PD.PNUM AND PD.DNUM = 2;
```


Ключові терміни:

кореліруємий підзапит, оператор DDL, оператор DML, подання

Мова SQL оперує термінами, що трохи відрізняються від термінів реляційної теорії, наприклад, замість "відношення" використовуються "таблиці", замість "кортежів" - "рядки", замість "атрибутів" - "колонки" або "стовпці". Мова SQL є реляційно повною. Це означає, що будь-який оператор реляційної алгебри може бути виражений підходящим оператором SQL.

7.1 Припустимі типи даних

Будь-який діалект SQL підтримують три загальних типи даних: строковий, числовий й тип для подання дати й часу. Завдання типу даних визначає значення й довжину даних, а також формат їхнього подання при візуалізації.

Для всіх типів даних визначено так зване нуль-значення, що вказує на відсутність даних у колонку зазначеного типу, тобто та обставина, що значення даних у сучасний момент часу невідомо.

Дані строкового типу являють собою послідовність рядків символів. Строкові дані можуть бути задані як з визначеною довжиною (ключові слова char або varchar (довжина рядка)), так і без вказівки довжини (ключове слово long varchar) для подання рядків довільної довжини.

Числові типи даних призначені для подання цілих чисел, чисел з десятковою крапкою й чисел із плаваючою крапкою. Будь-яке подання чисел задається своєю точністю й масштабом. Точність визначає припустиме подання кількосты значущих цифр числа, а масштаб - кількість значущих цифр після десяткової крапки.

Для подання цілих чисел використовуються типи interger (точність 10 значущих цифр) і smallint (точність 5 значущих цифр).

Для подання чисел з фіксованою десятковою крапкою використовуються типи number і decimal.

Для подання чисел із плаваючою крапкою в SQL передбачені такі типи даних:

- Double Precision - для чисел з точністю від 22 до 53 значущих цифр;
- Float (точність) - для подання чисел з точністю від 1 до 21 значущої цифри;
- Real - для чисел з точністю за замовчуванням (залежить від конкретної реалізації).

Звичайно в конкретних діалектах SQL використовуються три типи для подання таких даних:

- datestamp (timestamp) - для подання дати й часу;
- date - для подання дати;
- time - для подання часу.

Константи, вираження, системні змінні. Константи зазвичай визначають єдине значення й, відповідно до типу даних, що представляють, можуть бути строковими, числовими й представляти дату/час. Строкові константи повинні бути укладені в одинарні лапки.

В SQL існує ряд визначених системних змінних, які можна використовувати у виразах замість імен колонок і констант. До таких змінних відноять наступні:

- NULL - для подання невизначених значень;
- ROWID - (в SQLBase) внутрішній системний номер рядка в таблиці;
- USER - ім'я користувача, активного в цей момент;

- SYSDATETIME - системний поточний час і дата;
- SYSDATE - системна поточна дата;
- SYSTIME - системний поточний час;
- SYSTIMEZONE - часовий пояс, установлений у системі.

Виразом в SQL є ітем або комбінація ітемів з припустимими для них операціями, що дає єдине значення. У якості ітемів можуть виступати імена колонок, константи, зв'язані змінні, результати обчислень функцій, системні змінні й інші вирази. При цьому якщо один з ітемів має нуль-значення, то результат виразу також має нуль-значення.

7.2 Використання операторів мови SQL

7.2.1 Оператори SQL

Основу мови SQL становлять оператори, умовно розбиті на кілька груп за функціональним показником.

Оператори DDL (Data Definition Language) - оператори визначення об'єктів БД

- CREATE SCHEMA - створити схему БД
- DROP SHEMA - видалити схему БД
- CREATE TABLE - створити таблицю
- ALTER TABLE - змінити таблицю
- DROP TABLE - видалити таблицю
- CREATE DOMAIN - створити домен
- ALTER DOMAIN - змінити домен
- DROP DOMAIN - видалити домен
- CREATE COLLATION - створити послідовність
- DROP COLLATION - видалити послідовність
- CREATE VIEW - створити подання
- DROP VIEW - видалити подання

Оператори DML (Data Manipulation Language) - оператори маніпулювання даними

- SELECT - відібрати рядок з таблиць
- INSERT - додати рядок в таблицю
- UPDATE - змінити рядок в таблиці
- DELETE - видалити ряодк в таблиці
- COMMIT - зафіксувати внесені зміни
- ROLLBACK - відкотити внесені зміни

Оператори захисту й керування даними

- CREATE ASSERTION - створити обмеження
- DROP ASSERTION - видалити обмеження
- GRANT - надати привілею користувачеві або додатку на маніпулювання об'єктами
- REVOKE - скасувати привілею користувача або додатка

7.2.2 Оператори маніпулювання даними

INSERT - вставка рядків у таблицю
Вставка одного рядка до таблиціСховати

```
INSERT INTO
   P (PNUM, PNAME)
   VALUES (4, "Іванов");
```

UPDATE - відновлення рядків у таблиці
Відновлення декількох рядківСховати

```
UPDATE P
   SET PNAME = "Пушников"
   WHERE P.PNUM = 1;
```

DELETE - видалення рядків у таблиці
Видалення рядків з таблиці...Сховати

```
DELETE FROM P;
```

7.2.3 Виборка даних

Оператор SELECT завжди виконується над деякими таблицями, що входять у БД.

Подання - це SELECT-вираз, що просто зберігається в БД. З погляду користувачів подання - це таблиця, яка не зберігається постійно в БД, а "виникає" у момент звертання до неї.

Результатом виконання оператора SELECT завжди є таблиця.

7.2.3.1. Відбір даних з однієї таблиці

Ключові слова, що використовуються: ***SELECT***...***FROM***...

Виборка даних з таблиць, які задовольняють певній умові: ключове слово ***WHERE.*** Як умову в розділі WHERE можна використовувати складні логічні вирази, що використовують поля таблиць, константи, порівняння (>, <, = і т.д.), дужки, союзи AND й OR, заперечення NOT.

Вибірка деяких колонок з вихідної таблиці реалізується вказівкою списку потрібних колонок.

Якщо у вихідній таблиці було присутнє кілька рядків з різними номерами, але однаковими найменуваннями, то в результуючій таблиці будуть рядки з повтореннями - дублікати рядків автоматично не відкидаються.

Вибірка даних без дублювання даних досягається використанням ключового слова ***DISTINCT.*** Використання ключового слова DISTINCT приводить до того, що у результуючій таблиці будуть вилучені всі повторювані рядки.

Вибірка даних без дублювання інформаціїСховати

```
SELECT DISTINCT P.NAME
  FROM P;
```

Використання ключового слова ***AS*** дозволяє проіменувати колонки в результуючій таблиці

Приклад використання скалярних виразів і перейменувань колоноСховати

```
SELECT TOVAR...TNAME, TOVAR.KOL,
  TOVAR.PRICE, "="AS EQU,
  TOVAR.KOL*TOVAR.PRICE AS SUMMA
  FROM TOVAR;
```

В результаті одержимо таблицю з колонками, яких не було у вихідній таблиці TOVAR:

TNAME	KOL	PRICE	EQU	SUMMA
Болт	10	100	=	1000
Гайка	20	200	=	4000
Гвинт	30	300	=	9000

Існує можливість упорядкувати результати запитів по полю за допомогою використання ключового слова ***ORDER BY*** та по декількох полях за зростанням або зменшенням за допомогою ключових слів ***ASC***, ***DESC***). Якщо явно не зазначені ключові слова ASC або DESC, то за замовчуванням приймається впорядкування по зростанню (ASC).

Приклади вибірки даних з упорядкуванням результатів запитівСховати

```
SELECT PD...PNUM, PD.DNUM, PD.VOLUME
  FROM PD
  ORDER BY DNUM;
```

В результаті одержимо наступну таблицю, впорядковану по полю DNUM:

PNUM	DNUM	VOLUME
1	1	100
2	1	150
1	2	200
1	3	300

7.2.3.2. Відбір даних з декількох таблиць

Існує декілька способів отримати дані з декількох таблиць.

Природне з'єднання таблиць дозволяє отримати дані шляхом явної вказівки умов з'єднання. Таблиці, що з'єднують, перераховані у розділі ***FROM*** оператора, умова з'єднання наведена у розділі ***WHERE***. Розділ ***WHERE***, крім умови з'єднання таблиць, може також містити й умови відбору рядків.

Другий спосіб - природне з'єднання таблиць (спосіб 2 - ключові слова ***JOIN... USING...***). Ключове слово USING дозволяє явно вказати, по яким із загальних колонок таблиць буде вироблятися з'єднання.

Третій спосіб - природне з'єднання таблиць (спосіб 3 - ключове слово ***NATURAL***

JOIN). У розділі FROM не зазначається, по яких полях виробляється з'єднання. NATURAL JOIN автоматично з'єднує по всіх однакових полях у таблицях.

7.2.4 Вбудовані функції

Арифметичні функції та функції обробки дати й часу

SQL підтримує повний набір арифметичних операцій і математичних функцій для побудови арифметичних виражень над колонками БД (+, -, *, /, ABS, LN, SQRT і т.д.). Перелік основних вбудованих математичних функцій та функцій по роботі з даними типу дата/час ви можете знайти в літературних джерелах, перелік яких наданий наприкінці модуля.

Арифметичні вирази необхідні для одержання даних, які безпосередньо не зберігаються в колонках таблиць БД, але значення яких необхідні користувачеві.

Приклад використання арифметичних функцій та функцій обробки датиСховати

Якщо вам потрібен був список нових службовців, що надійшли за останній квартал в організацію, то ви можете написати запит у наступному виді:

```
SELECT ENAME, HIREDATE,
HIREDATE + 92 DAYS
FROM EMPLOYEE
WHERE HIREDATE + 92 DAYS > SYSDATE
AND DEPNO=30;
```

Використання агрегатних функцій у запитах

В мові SQL передбачені такі оператори агрегатних функцій:

AVG(X) = AVG(ALL X) AVG(DISTINCT X) Обчислює середнє значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати.

COUNT(*) COUNT(X) = COUNT(ALL X) COUNT(DISTINCT X) Обчислює числа ітемів. При вказівці * завжди повертається число рядків у таблиці. Вказівка DISTINCT придушує дублікати.

MAX(X) = MAX(ALL X) MAX (DISTINCT X) Обчислює максимальне значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

MIN(X) = MIN(ALL X) MIN (DISTINCT X) Обчислює мінімальне значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

SUM(X) = SUM(ALL X) SUM (DISTINCT X) Обчислює суму значення аргументу, що може бути виразом будь-якого типу. Нуль-значення ігноруються, ключове слово DISTINCT придушує дублікати

Використання агрегатних функцій з угрупованнями

Для виведення результатів даних, сгрупованих за певною умовою, слід використовувати два ключових слова: ***GROUP BY*** та ***HAVING.***

В переліку полів оператора SELECT, який містить розділ GROUP BY можна включати тільки агрегатні функції й поля, які входять в умову групування.

Умови, що використовують агрегатні функції повинні бути розміщені у спеціальному розділі HAVING. В одному запиті можуть зустрітися як умови відбору рядків у розділі WHERE, так й умови відбору груп у розділі HAVING. Умови відбору груп не можна перенести з розділу HAVING у розділ WHERE. Аналогічно й умови відбору рядків не можна перенести з розділу WHERE у розділ HAVING, за винятком умов, що включають поля зі списку угруповання GROUP BY.

Приклади синтаксису запитівСховати

Запита: для кожної деталі одержати сумарну кількість, що поставляється (ключове слово ***GROUP BY***...):

```
SELECT..DNUM, SUM(PD.VOLUME) AS SM
GROUP BY PD.DNUM;
```

Цей запит буде виконуватися в такий спосіб. Спочатку рядки вихідної таблиці будуть згруповані так, щоб у кожну групу потрапили рядки з однаковими значеннями DNUM. Потім усередині кожної групи буде просумовано поле VOLUME. Від кожної групи до результуючої таблиці буде включений один рядок.

Запит: одержати номери деталей, сумарна кількість поставки яких перевершує 400 (ключове слово ***HAVING***...):

```
SELECT PD.DNUM, SUM(PD.VOLUME) AS SM
GROUP BY PD.DNUM
HAVING SUM(PD.VOLUME) > 400;
```

Умова, що сумарна кількість поставки повинна бути більше 400, не може бути сформульована у розділі WHERE, тому що в цьому розділі не можна використовувати агрегатні функції. Умови, що використовують агрегатні функції повинні бути розміщені у спеціальному розділі HAVING.

7.2.5 Використання підзапитів

Дуже зручним засобом, що дозволяє формулювати запити більш зрозумілим чином, є можливість використання підзапитів, вкладених в основний запит.

Самий простий спосіб виконання запиту прямою вказівкою в розділі WHERE вкладеного запиту.

Приклад синтаксису запитаСховати

Запит: одержати список постачальників, статус яких менше максимального статусу у таблиці постачальників (порівняння з підзапитом):

```
SELECT *
FROM P
WHERE P.STATYS < (SELECT MAX(P.STATUS) FROM P);
```

Тоді як поле P.STATUS рівняється з результатом підзапиту, то підзапит повинен бути сформульований так, щоб повертати таблицю, що складається рівно з одного рядка й однієї колонки.

Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Виконати один раз вкладений підзапит й одержати максимальне значення статусу.
2. Просканувати таблицю постачальників P, щоразу порівнюючи значення статусу постачальника з результатом підзапиту, і відібрати тільки ті рядки, в яких статус менше максимального.

Другий спосіб формування підзапиту - використання предиката ***IN***. Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Виконати один раз вкладений підзапит й одержати перелік значень.
2. Просканувати таблицю запиту, щораз перевіряючи, чи втримується значення відповідного атрибута в результаті підзапиту.

Існує ще один спосіб - використання предиката ***EXIST***. Результат виконання запиту буде еквівалентний результату такої послідовності дій:

1. Просканувати таблицю запиту, щоразу виконуючи підзапит з новим значенням відповідного атрибута з таблиці запиту.
2. В результат запиту включити тільки ті рядки з таблиці запиту, для яких вкладений підзапит повернув непусту множину рядків.

Коли вкладений підзапит містить параметр (зовнішнє посилання), переданий з основного запиту, такі підзапити називаються кореліруємими (*correlated*). Зовнішнє посилання може приймати різні значення для кожного рядка-кандидата, оцінюваного за допомогою підзапиту, тому підзапит повинен виконуватися заново для кожного рядка, який відбирається в основному запиті. Такі підзапити характерні для предиката EXIST, але можуть бути використані й в інших підзапитах.

7.2.6. Використання об'єднання, перетинання й різниці

Результуючі таблиці поєднуваних запитів повинні бути сумісні, тобто мати однакову кількість стовпців й однакові типи стовпців у порядку їхнього перерахування. Не потрібно, щоб поєднувані таблиці мали б однакові імена колонок. Це відрізняє операцію об'єднання запитів в SQL від операції об'єднання у реляційній алгебрі. Найменування колонок у результуючому запиті будуть автоматично взяті з результату першого запиту в об'єднанні.

Для виконання об'єднання двох підзапитів використовуються ключові слова ***UNION*** (об'єднання) та ***INTERSECT*** (перетин) та ***EXCEPT*** (різниця).

Приклади синтаксису запитівСховати

Запит: одержати імена постачальників, що мають статус, більший 3-ого або що поставляють хоча б одну деталь номер 2

```
SELECT P.PNAME
FROM P
WHERE P.STATUS > 3 UNION SELECT P.PNAME
                         FROM P, PD
                         WHERE P.PNUM = PD.PNUM AND PD.DNUM = 2;
```

Запит: одержати імена постачальників, що мають статус, більший 3-ого й одночасно поставляють хоча б одну деталь номер 2:

```
SELECT P.PNAME
FROM P
```

```
WHERE P.STATUS > 3 INTERSECT SELECT P.PNAME
                             FROM P, PD
                             WHERE P.PNUM = PD.PNUM AND PD.DNUM = 2;
```

Запит: одержати імена постачальників, що мають статус, більший 3-ого, за винятком тих, хто поставляє хоча б одну деталь номер 2:

```
SELECT P.PNAME
FROM P
WHERE P.STATUS > 3 EXCEPT SELECT P.PNAME
                          FROM P, PD
                          WHERE P.PNUM = PD.PNUM AND PD.DNUM = 2;
```

Фактично стандартною мовою доступу до БД у цей час стала мова SQL.

Стандарт мови SQL, хоча й заснований на реляційній теорії, але в багатьох місцях відходить він неї.

Основу мови SQL становлять оператори, умовно розбиті на кілька груп по виконуваних функціях:

- Оператори DDL (Data Definition Language) - оператори визначення об'єктів БД.
- Оператори DML (Data Manipulation Language) - оператори маніпулювання даними.
- Оператори захисту й керування даними, і ін.

Одним з основних операторів DML є оператор SELECT, що дозволяє витягати дані з таблиць й одержувати відповіді на різні запити. Оператор SELECT містить у собі всі можливості реляційної алгебри. Це означає, що будь-який оператор реляційної алгебри може бути виражений за допомогою підходящого оператора SELECT. Цим доводиться реляційна повнота мови SQL.

Розрізняють концептуальну схему виконання оператора SELECT і фактичну схему його виконання. Концептуальна схема описує, у якій логічній послідовності повинні виконуватися операції, щоб одержати результат. При реальному виконанні оператора SELECT на перший план виступає досягнення максимальної швидкості виконання запиту. Для цього використовується оптимізатор, який, аналізуючи різні плани виконання запиту, вибирає найкращий з них

1. Назвіть основні типи даних мови SQL?
2. На які групи поділені оператори мови SQL?
3. Назвіть основні оператори групи визначення об’єктів БД.
4. Назвіть основні оператори групи маніпулювання даними в БД
5. Назвіть основні оператори групи захисту і керування даними.
6. Яке ключове слово дозволяє уникнути дублювання рядків у результуючій таблиці?
7. Яким чином можна впорядкувати записи у результуючій таблиці?
8. Як реалізувати з’єднання таблиць?
9. Чи можна впорядкувати дані за декількома полями?

Тема 8 - Цілісність та безпека даних

Ключові терміни:

безпека даних, вибірковий метод захисту, декларативне обмеження цілісності, динамічне обмеження цілісності, користувач бази даних, обмеження цілісності, обов’язковий метод захисту, організаційно-методичний захід, програмний засіб, процедурне обмеження цілісності, роль, семантичне обмеження цілісності, система захисту, структурне обмеження цілісності, технічний засіб, цілісність, юридичний захід

Цілісність даних

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.

8.1.1. 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.

8.1.2 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).

8.1.2.1. 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

ПрикладСховати

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

8.1.2.2. 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.

ПрикладСховати

```
CREATE TABLE ГРУПА ( #GNUMBER(3),
  #D NUMBER(3).
  Курс NUMBER(I) 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.

8.1.2.3. 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.

8.1.2.4. 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.

ПрикладСховати

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
```

8.1.3 Динамічні обмеження цілісності

Динамічні обмеження цілісності - це обмеження, які встановлюють залежність між різними частинами бази даних у різні моменти часу. Виділяють такі різновиди динамічних обмежень: ситуаційно орієнтовані; операційно орієнтовані.

Ситуаційно - орієнтовані обмеження задаються у вигляді вимог, яким мають відповідати послідовні стани бази даних, тобто завдяки таким обмеженням фіксуються допустимі переходи бази даних з одного стану в інший.

ПрикладСховати

Приклад можливих переходів значення атрибута Сімейний стан можна побачити на рис. 8.1.

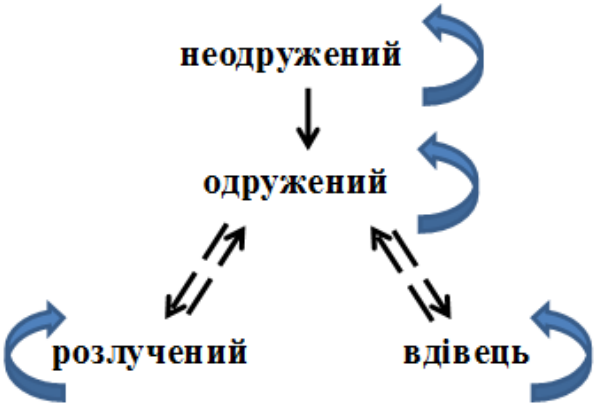


Рисунок 8.1 - Можливі переходи значення атрибута «Сімейний стан»

Для специфікації ситуаційно-орієнтованих обмежень застосовуються ті самі засоби, що й для специфікації структурних обмежень. Окрім того, використовується можливість посилатися на значення бази даних до її оновлення й після.

Це досягається за допомогою уточнюючих фраз OLD та NEW, що вживаються в посиланнях на значення, які зберігалися в базі даних до й після зміни її стану.

Зазвичай для опису динамічних обмежень використовуються тригери, які ініціюються під час зміни стану бази даних командами INSERT, UPDATE та DELETE.

Операційно-орієнтовані обмеження цілісності задаються у вигляді допустимих послідовностей операцій. Допустимість операції або послідовності операцій може залежати від поточного стану бази даних.

ПрикладСховати

«Додавання в базу даних інформації про те, що х є чоловіком у допустиме лише в тому випадку, коли і х, і у в даний момент не є одруженими». Зазвичай подібні обмеження реалізуються за допомогою означення тригерів для відповідних операцій маніпулювання таблицями. Наприклад, якщо є відношення ПОДРУЖЖЯ (Чоловік, Дружина) та ОСОБА(Прізвище, Сімейний стан), то описане вище обмеження можна специфікувати так:

```
CREATE TRIGGER Подружжя_Перевірка
BEFORE INSERT ON ПОДРУЖЖЯ
WHEN (ПОДРУЖЖЯ.Чоловік=ОСОБА.Прізвище AND
ОСОБА.Сімейний_Стан = 'Одружений') AND NEW
(ПОДРУЖЖЯ.Жінка = ОСОБА.Прізвище AND
ОСОБА.СімейнийСтан = ‘Одружена’)
BEGIN
ROLLBACK TRANSACTION
END
```

8.1.4. Семантичні обмеження цілісності

Семантичні обмеження цілісності, або прикладні правила цілісності, — це правила, які характеризують обмеження, що діють у предметній області.
ПрикладСховати
Прикладами семантичних обмежень можуть бути:

- описане вище правило зміни стану атрибута Сімейний стан;
- «лист, що надійшов, вважається обробленим, коли з ним ознайомлені всі зацікавлені особи і щодо нього ухвалене відповідне рішення»:
- «розмір посадових окладів має варіюватися в межах від 3000 до 5000 грн.»;
- «студент може перейти на наступний курс або залишитися на тому самому, але не на попередньому»;
- «одна й та ж особа не може бути завідувачем двох кафедр».

Для специфікації семантичних обмежень використовують фразу C0NSTRAINT і тригери. У деяких випадках для підтримки семантичних обмежень цілісності пишуться спеціальні процедури, які зберігаються в СКБД, або спеціальні процедури, що входять до складу зовнішнього застосування.

8.1.5. Підтримка цілісності у разі виникнення перебоїв

Нагадаємо, що цілісність бази даних - це її відповідність модельованій предметній області у будь-який момент часу. Механізми опису обмежень цілісності забезпечують підтримку цілісності з «логічної» точки зору. Проте перебої в програмному або апаратному забезпеченні також можуть призвести до порушення цілісності (а в деяких випадках до повного руйнування бази даних). Для забезпечення цілісності бази даних на цей випадок пропонуються такі механізми:

- періодичне створення резервної копії бази даних;
- ведення журналу всіх змін стану бази даних.

Загальна схема підтримки цілісності на випадок перебоїв є такою. У певний момент часу створюється резервна копія бази даних. Починаючи з цього моменту в журналі фіксуються всі зміни, що виконуються в базі. Якщо в якийсь момент часу база даних виявляється настільки зіпсованою, що її неможливо відновити, то береться резервна копія і до неї застосовуються всі зафіксовані в журналі операції. У такий спосіб резервна копія стає актуальною.

8.2 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).

8.2.1. 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.

8.2.3. Керування правами доступу

8.2.3.1. 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.

8.2.3.2. 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).

ПрикладСховати

- 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.

8.2.3.3. 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.

ПрикладСховати

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

8.2.3.4. 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.

8.2.3.5. 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.

8.2.3 Специфікація повноважень в СКБД Oracle

Розглянемо, як означуються права доступу до даних в СКБД Oracle. Інструкція, що дозволяє означити користувача бази даних. Щойно створений користувач не має жодних повноважень. Надання повноважень користувачу здійснюється командою GRANT.

Команда GRANT дає змогу передати повноваження користувачам або ролям. Спрощений синтаксис команди є таким:

GRANT { <операція> | ALL} [(<список стовпців>)] ON <список об'єктів> TO {<користувач> I <роль> | PUBLIC} WITH GRANT OPTION

Специфікатор <операція> вказує, щодо яких операцій описується повноваження. Такими операціями можуть бути: SELECT, INSERT, UPDATE, DELETE та інші. Фраза ALL вказує, що повноваження передаються щодо всіх операцій.

У списку об'єктів зазначаються всі об'єкти бази даних, щодо яких специфікуються повноваження. Цей список містить посилання на таблиці й віртуальні таблиці. За допомогою параметра <список стовпці в> можна додатково вказати стовпці таблиць.

Повноваження може бути передане користувачу, ролі або всім, хто вказується у фразі TO.

Якщо повноваження передається користувачу, він одержує право виконувати операції згідно з цим повноваженням. Коли повноваження передається ролі, вона ним поповнюється.

Фраза WITH GRANT OPTION означає, що одержувач повноваження отримує також право передавати це повноваження іншому користувачу або ролі Розглянемо кілька прикладів надання й передавання прав доступу:

Надати користувачу на ім'я Джон права на виконання будь-яких операцій над таблицею ФАКУЛЬТЕТ і дозвіл на передавання цих прав іншим користувачам. GRANT ALL ON ФАКУЛЬТЕТ TO Джон WITH GRANT OPTION

ПрикладСховати

Надати всім користувачам право переглядати дані з таблиці Лекція.

GRANT SELECT ON ЛЕКЦІЯ TO PUBLIC

Надати користувачу на ім'я Джон право змінювати стовпець фонд у таблиці КАФЕДРА.

GRANT UPDATE (Фонд) ON КАФЕДРА TO Джон

8.2.4. Обов’язкові методи захисту

Обов’язкові методи захисту, або методи обов’язкового керування доступом застосовуються до баз, в яких дані мають статичну або жорстку структуру, характерну, наприклад, для урядових або військових організацій. Як уже наголошувалося, основна ідея полягає в тому, що кожний об’єкт даних має певний рівень секретності, а кожний користувач — певний рівень доступу. Передбачається, що ці рівні утворюють строгий ієрархічний порядок (наприклад, цілком таємно, таємно, для службового користування тощо).

8.2.4.1 Ведення журналів доступу

Не буває невразливих систем захисту. Настирливий зловмисник завжди може знайти спосіб подолати всі системи контролю і захисту. Тому під час роботи з дуже важливими даними необхідно реєструвати у відповідному журналі всі події, що стосуються функціонування системи захисту. У зв’язку з цим система захисту має надавати можливість виконувати наведені далі дії:

- реєструвати всі спроби отримання доступу до системи баз даних, у тому числі й безуспішні. Ця реєстрація має бути максимально повного (повинні реєструватися відомості про те, хто отримував доступ або намагався його отримати, з якого термінала або вузла мережі, о котрій годині тощо);
- реєструвати дії користувачів з використання всіх ресурсів системи, зокрема й даних, а також інші дії та події, які можуть вплинути на захист даних;
- надавати користувачам, які мають адміністративні повноваження, можливість переглядати й аналізувати результати реєстрації, виявляти небезпечні ситуації, встановлювати причини їхнього виникнення та знаходити користувачів, відповідальних за порушення політики безпеки.

Навіть сама лише констатація факту, що в системі підтримується реєстрація всіх дій користувачів, у деяких випадках має суттєве значення для запобігання несанкціонованому проникненню в систему.

8.2.4.2. Обхід системи захисту

До носіїв, на яких записані дані, можна отримати доступ в обхід системи захисту. Якщо отримати доступ до файлів операційної системи, які містять дані з бази, то можна прочитати вміст цих файлів, оскільки фірми-розробники переважної більшості СКБД створюють формати зберігання даних такими, що вони є доступними широкому колу користувачів і застосувань. Більше того, той, хто отримав доступ до файлів бази, може змінити їхній вміст або навіть видалити їх.

Найефективнішими засобами боротьби з такою загрозою є використання методів криптографії, тобто шифрування інформації. Коли дані записані в базі у зашифрованому вигляді, тоді той, хто отримав доступ до них в обхід системи захисту, стикається з проблемою дешифрування. В ідеальному випадку метод шифрування має бути таким, щоб витрати на дешифрування перевищували виграш від володіння інформацією або щоб час дешифрування перевищував час. протягом якого дані розглядаються як секретні.

У системах розподілених баз даних, коли інформація пересилається каналами зв’язку, небезпечними є різні форми перехоплення даних, що передаються. Єдиним контрзаходом, який дає змоіу запобігти подібним перехопленням, є шифрування даних перед їхнім передаванням каналами зв’язку.

Головними аспектами проектування і функціонування бази даних є забезпечення цілісності та захисту даних. Під цілісністю розуміється достовірність даних у будь-який момент часу, під безпекою - захист від несанкціонованого доступу.

Цілісність даних забезпечується обмеженнями цілісності, безпека - розподілом прав доступу між групами користувачів даними.

Обмеження цілісності діють на такі основні об’єкти бази даних, як відношення, атрибути, зв’язки між відношеннями та між атрибутами. Механізми підтримки цілісності відношень реалізує СКБД.

Особливим аспектом функціонування бази даних є підтримка цілісності у разі виникнення перебоїв, для чого використовуютсья такі механізми, як періодичне створення резервної копії бази даних та ведення журналу всіх змін стану бази даних.

Безпека даних підтримується цілим комплексом заходів і засобів: організаційно-методичні та юридичні заходи, технічні та програмні засоби захисту. В сучасних СКБД підтримується один з двох найбільш розповсюджених методів забезпечення захисту даних: вибірковий чи обов’язковий.

Слід акцентувати увагу на двох типах керування доступом до даних, які відрізняються один

від одного можливістю передавати повноваження на доступ до даних: довірче та адміністративне керування доступом до даних

Визначити права доступу користувачів – це означає зафіксувати інформацію стосовно:

- осіб, яким надаються права доступу;
- умов надання прав доступу;
- об'єкта, на які поширюються права доступу;
- операцій, щодо яких специфікуються права доступу;
- можливості передавання прав доступу іншим особам.

1. Що таке цілісність даних?
2. Якими засобами можна забезпечити цілісність даних?
3. За якими показниками класифікують обмеження цілісності?
4. За допомогою яких засобів підтримується безпека даних?
5. Чим відрізняється довірче керування доступом від адміністративного?
6. Як специфікуються повноваження в Oracle?
7. В чому полягає ідея обов'язкових методів захисту?
8. В який спосіб ведення журналів доступу може підвищити безпеку даних?
9. Як захистити дані, доступ до яких можливий поза системою захисту СКБД?

1. К. Дж. Кейт Введення в системи баз даних Пер. с англ. 8-е изд. М.: Издательский дом «Вильямс», 2006.– 1328С.
2. Томас Коннолли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика.– 3-е изд. М.: Издательский дом «Вильямс», 2003, 1436 с.
3. Джен Л. Харрингтон Проектирование реляционных баз данных — М.: Издательство «Лори», 2006 – 230 с.
4. Пасічник В.В.Організація баз даних та знань: підручник для ВНЗ/ В.В. Пасічник, В.А. Резніченко.– К.: Видавнича група BHV,2006.-384с.
5. В.Е. Туманов. Основы проектирования реляционных баз данных
6. Д.В. Кознов. Визуальное моделирование: теория и практика
7. Пушников А.Ю. Введение в системы управления базами данных. Часть 2. Нормальные формы отношений и транзакции: Учебное пособие/Изд-е Башкирского ун-та. - Уфа, 1999. - 138 с
8. В.В. Кириллов, Г.Ю. Громов. Учебное пособие по SQL: Структурированный язык запросов (SQL)

Розділ 3 - Знайомтсво з іншими видами баз даних

Тема 9 - Розподілені бази даних

- Основні означення
- 9.2 Властивості розподілених баз даних
- 9.3 Логічна архітектура розподілених баз даних
- 9.4 Архітектура програмно-технічних засобів розподілених СКБД
  - 9.4.1. Властивості архітектури
  - 9.3.2. Різновиди архітектури

Ключові терміни:

видавець, дистриб’ютор, передплатник, реплікація, розподілена база даних, розподілена система баз даних, розподілена система керування базами даних, транзакція, фрагментація

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 (рис. 9.1). 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.

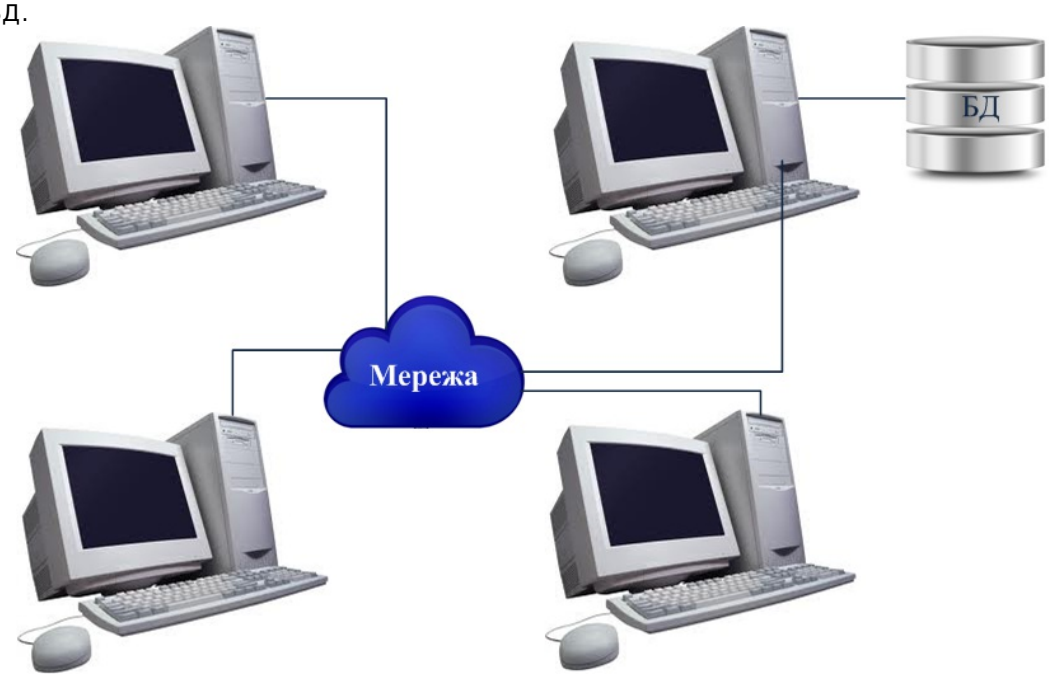


Рисунок 9.1 – Централізована база даних у комп’ютерній мережі

Архітектура розподіленої СКБД наведена на рис. 9.2. 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.

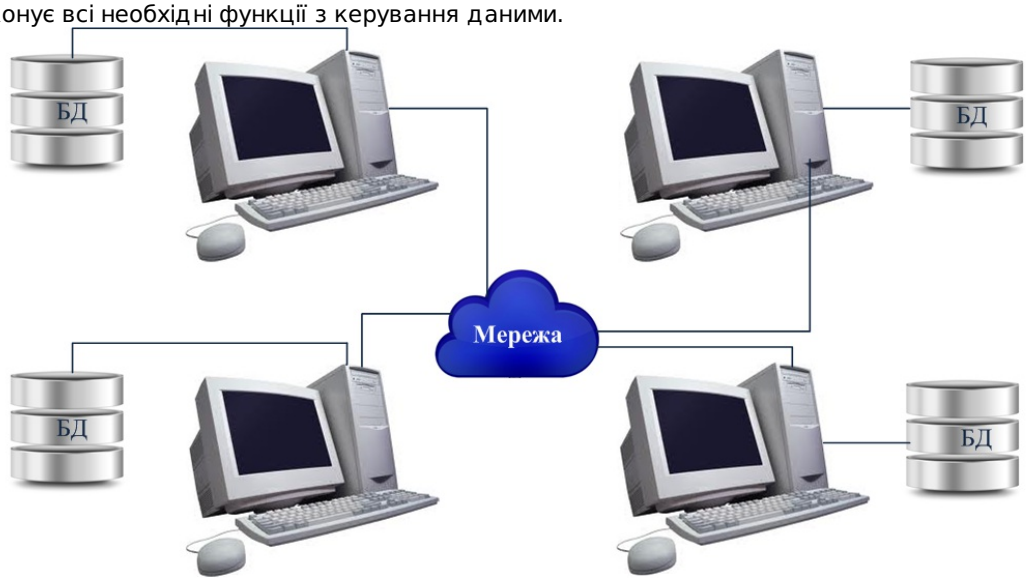


Рисунок 9.2 - Розподілена база даних

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.

9.2 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.

9.3 Логічна архітектура розподілених баз даних

Логічна архітектура розподілених баз даних – це архітектура логічно взаємозалежних даних. Графічне зображення логічної архітектури розподілених баз даних наведене на рис. 9.3. 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.

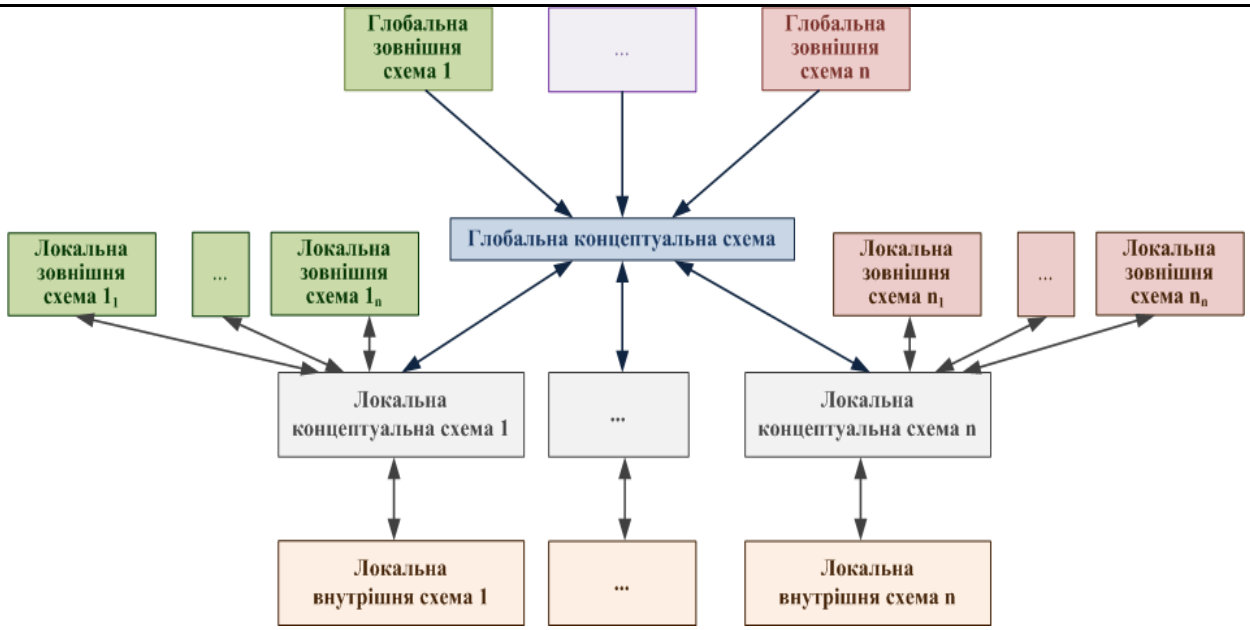


Рисунок 9.3 – Логічна архітектура розподіленої СКБД

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.

9.4 Архітектура програмно-технічних засобів розподілених СКБД

9.4.1. 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).

9.3.2. 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.

9.4 Розподілене зберігання даних

У розподіленій СКБД дані розподіляються за вузлами мережі. Існують два основні механізми розподіленого зберігання даних:

- фрагментація;
- реплікація

9.4.1. Фрагментація

Суть фрагментації полягає в тому, щоб поділити логічну базу даних на фрагменти з метою зберігання кожного фрагмента на певному вузлі мережі. Одиницями фрагментації можуть бути відношення та складені відношення. У випадку, коли одиницею фрагментації є відношення, вирішується проблема, яке відношення в якій базі даних має зберігатися. За іншого підходу допускається, що будь-яке відношення може бути зображене у вигляді сукупності фрагментів, що розподіляються за різними базами даних.

Фрагментацію, що здійснюється розподілом відношень за базами даних, теоретично здійснити нескладно, тому розглянемо проблему фрагментації власне відношень.

Фрагментація відношень. Завдання фрагментації відношень формулюється в такий спосіб. Нехай задане відношення R. Його потрібно зобразити у вигляді сукупності відношень R_1, ..., R_n так, щоб ця сукупність відповідала критеріям ефективності (за часом доступу, пам'яттю, завантаженістю комп'ютерів тощо).

Фрагментація є коректною, якщо вона повна, не містить перетинів і може бути реконструйована. Пояснимо ці терміни.

Декомпозиція відношення R на фрагменти R_1, R_2, ... R_n є повною тоді й лише тоді, коли кожен елемент даних з R належить якомусь із відношень R_i.

Декомпозиція відношення R на фрагменти R_1, R_2, ... R_n може бути реконструйована, якщо існує такий реляційний вираз $\varphi(R_1, R_2, ... R_n)$, що $R=\varphi(R_1, R_2, ... R_n)$.

Декомпозиція відношення R на фрагменти R_1, R_2, ... R_n не містіть перетинів, якщо будь-який елемент даних з R міститься не більш ніж в одному фрагменті.

Є три типи фрагментації відношень:

- горизонтальна;

- вертикальна;
- змішана

Горизонтальна фрагментація

Горизонтальна фрагментація полягає в розподілі кортежів відношення за фрагментами. Формально горизонтальну фрагментацію можна визначити в такий спосіб. Нехай задане відношення R і на ньому визначений предикат F_i. Тоді горизонтальний фрагмент Ri відношення визначається так:

$$R_i = \sigma_{F_i}(R)$$

Тобто горизонтальний фрагмент Ri - це множина кортежів R, що задовольняють умову F_i.

ПрикладСховати

Нехай задане відношення

ПРОЕКТ (Рном, Назва, Тип, Вартість).

Очевидно, що множина предикатів

{ Вартість < 300000. Вартість = 300000. Вартість > 300000}

породжує повну фрагментацію відношення ПРОЕКТ, яка не містить перетинів. Предикати

{ Вартість < 300000. Вартість > 200000}

породжують повну фрагментацію, що містить перетин. А предикати

{ Вартість < 200. Вартість > 300}

породжують неповну фрагментацію, яка не містить перетинів.

Переваги горизонтальної фрагментації:

- дає змогу паралельно обробляти фрагменти відношення;
- дає можливість поділяти відношення так, щоб кортежі розташовувалися там, де вони будуть використовуватися найчастіше.

Вертикальна фрагментація

Суть вертикальної фрагментації полягає в тому, що відношення поділяється на дві чи більше проекцій, тобто схема відношення поділяється на певну множину підсхем.

Для відновлення вихідного відношення з фрагментів необхідно, щоб усі підсхеми містили первинний ключ. Існує інший підхід, коли під час поділу схеми до кожного фрагмента автоматично додається поле з ідентифікатором кортежу. Значення цього ідентифікатора зазвичай встановлюються системою автоматично.

ПрикладСховати

Нехай задано відношення

ПОСТАЧАННЯ(#Рном, Постачальник, Обладнання, Проект).

ПОСТАЧАННЯ

#Рном	Постачальник	Обладнання	Проект
Р1	Іванов	Двигун	АН-24
Р2	Іванов	Двигун	ЯК-40
Р3	Іванов	Шасі	АН-24
Р4	Петров	Двигун	АН-24
Р5	Петров	Електрообладнання	ЯК-40
Р6	Петров	Електрообладнання	АН-70

Вертикальна фрагментація з використанням первинного ключа може виглядати так:

ПОСТАЧАННЯ-1

#Рном	Обладнання	Проект
Р1	Двигун	АН-24
Р2	Двигун	ЯК-40
Р3	Шасі	АН-24
Р4	Двигун	АН-24
Р5	Електрообладнання	ЯК-40
Р6	Електрообладнання	АН-70

ПОСТАЧАННЯ-2

#Рном	Постачальник
Р1	Іванов
Р2	Іванов
Р3	Іванов
Р4	Петров
Р5	Петров
Р6	Петров

Переваги вертикальної фрагментації:

- дає змогу поділяти кортежі відношення так, щоб їхні частини розташовувалися там, де вони використовуватимуться найчастіше;
- дає змогу проводити паралельну обробку відношень;
- наявність ідентифікатора кортежу дає можливість здійснювати ефективне з'єднання вертикальних фрагментів.

Змішана фрагментація. Змішана фрагментація передбачає послідовне застосування вертикальної і горизонтальної фрагментацій.

Розподіл даних за вузлами мережі

Після отримання усіх необхідних фрагментів відношень постає проблема розподілу цих фрагментів за вузлами мережі. Єдиних рекомендацій стосовно того, як це робити, немає. Потрібно знайти оптимальний розподіл фрагментів F за вузлами мережі S за умови, що відомий розподіл додатків Q за вузлами мережі.

Визначаючи оптимальність, слід враховувати різні параметри, зокрема:

- вартість передавання, зберігання й обробки даних;
- часові характеристики;
- продуктивність;
- обмеження (наприклад, ємнісні характеристики вузлів мережі).

Для того щоб розподілити фрагменти за вузлами мережі, необхідна додаткова інформація, яка стосується

- бази даних та розмірів фрагментів;
- додатків (місць їхнього розташування, частоти використання тих чи інших фрагментів для вибирання даних, частоти використання фрагментів для відновлення даних);
- вузлів мережі (вартості зберігання й обробки даних у вузлах);
- мережі (вартості та часових характеристик передавання даних між двома вузлами).

9.4.2. Реплікація

Реплікація — це механізм розподілу даних за вузлами, що дозволяє зберігати копії тих самих даних на різних вузлах мережі з метою прискорення пошуку і підвищення стійкості до відмов. Відношення чи фрагмент є реплікованим, якщо його копії зберігаються на двох або більше вузлах (копії ще називають репліками). За повної реплікації відношення його копії зберігаються на всіх вузлах мережі. Допускається ситуація, коли вся база даних зберігається на всіх вузлах мережі — це називається поєною реплікацією бази даних.

Переваги реплікації:

- доступність (у разі перебою в роботі вузла, що містить відношення H, його доступність на інших вузлах зберігається);
- паралелізм (виконання запитів до відношення R може бути розпаралелено за всіма репліками відношення);
- зниження вартості передавання даних (відношення R доступне локально в усіх вузлах, де є його репліки).

Недоліки реплікації:

- підвищується вартість зберігання, створення і відновлення даних; підвищуються вимоги до ресурсів;
- ускладнюється підтримання цілісності даних, наприклад одночасне відновлення різних реплік одного й того ж відношення

Механізми реплікації

Для реалізації реплікації використовуються три сервери: видавець, дистриб'ютор і передплатник.

Видавцем називають сервер, що надає розміщені на ньому дані для копіювання на інші сервери. Окрім створення копії даних, видавець відстежує внесені до його бази даних зміни і готує нову копію.

Дистриб'ютором називається сервер, що підтримує розподілену базу даних. Він виконує роль посередника, копіює всі публікації, підготовлені видавцем, і пересилає їх передплатникам. Дистриб'ютором може бути виділений сервер або сервер, сконфігурований як видавець чи передплатник. Конкретні функції, що їх виконує дистриб'ютор, залежать від методів реплікації.

Передплатником називається сервер, що отримує копії даних, надані видавцем. Механізми зміни даних передплатником відрізняються від механізмів зміни даних видавцем.

Є два методи відновлення даних передплатників:

- Реплікація за запитом. Передплатник періодично звертається до дистриб'ютора із запитом про зміни, що відбулися з моменту останнього з'єднання.
- Примусова реплікація. Дистриб'ютор сам встановлює з'єднання з передплатником і пересилає йому необхідні дані.

Залежно від методу реплікації, передплатники можуть чи не можуть вносити зміни в репліковані дані. У найпростішому випадку змінювати дані може тільки видавець, у складніших моделях реплікації - передплатники і видавці. Змінені дані, отримані від усіх передплатників, синхронізуються і поєднуються з даними видавця, а потім розсилаються передплатникам.

Моделі реплікації

Є такі моделі реплікації:

- реплікація моментальних знімків;
- реплікація транзакцій.

Реплікація моментальних знімків є найпростішою моделлю реплікації. Моментальний знімок це повна копія даних, обраних для реплікації; вона розсилається передплатникам.

Під час реплікації транзакцій використовується журнал транзакцій бази даних. Обрані транзакції копіюються в базу даних дистриб'ютора зі збереженням інформації про послідовність їхнього виконання, потім розсилаються серверам- передплатникам і виконуються на них у тому ж порядку, в якому виконувалися на сервері-видавці.

Цей механізм зменшує завантаження мережі, його рекомендується використовувати у великих базах даних з невеликою кількістю змін.

Топологія реплікацій

Топологія реплікацій описує характер взаємозв'язків між учасниками реплікації:

- реплікація «один-до-багатьох» передбачає наявність одного видавця і кількох передплатників;
- реплікація «багато-до-одного» має місце, коли дані від кількох видавців пересилаються одному передплатнику;
- реплікація «багато-до-багатьох» означає, що дані від кількох видавців пересилаються кільком передплатникам.

9.5 Обробка розподілених транзакцій

Транзакція — набір команд, що виконується як єдине ціле. У транзакції або всі команди будуть виконані, або жодна з них не виконається. Якщо хоча б одна з команд транзакції не може бути виконана, здійснюється відкочування (відновлюється стан системи, в якому вона перебувала до початку виконання транзакції).

Транзакції мають задовольняти вимоги ACID (Atomicity, Consistency, Isolation, Durability — атомарність, несуперечність, ізольованість, довговічність), що гарантують правильність і надійність роботи системи.

Атомарність передбачає таке:

- виконуються всі операції транзакції або жодна з них не виконується;
- якщо виконання транзакції було перерване, то всі зроблені транзакцією зміни мають бути скасовані

Дії, спрямовані на підтримку атомарності транзакції під час її аварійного завершення у зв'язку з помилками введення/виведення, перевантаженнями системи чи блокуваннями, називаються відновленням транзакції.

Дії, спрямовані на забезпечення атомарності під час виходу з ладу системи, називаються відновленням у разі виникнення перебоїв.

Несуперечність означає, що транзакція, яка працює з несуперечною базою даних, після завершення роботи залишає її також у несуперечному стані. Транзакція не повинна порушувати цілісності бази даних

Для вирішення проблем одночасного доступу інститут ANSI розробив спеціальний стандарт, який визначає чотири рівні блокування (кожний вищий рівень передбачає виконання умов усіх нижчих рівнів)

- Рівень 0. Заборона «забруднення» даних. На цьому рівні вимагається, щоб змінювати дані могла лише одна транзакція. Якщо іншій транзакції необхідно змінити ці ж дані, то вона має очікувати завершення першої транзакції.
- Рівень 1. Заборона некоректного зчитування. Якщо певна транзакція розпочала змінювати дані, то жодна інша транзакція не зможе зчитати ці дані доти, доки перша не завершиться.
- Рівень 2. Заборона неповторюваного зчитування. Якщо певна транзакція зчитує дані, то жодна інша транзакція не зможе їх змінити. Отже, під час повторного зчитування дані перебуватимуть у початковому стані.
- Рівень 3. Заборона «фантомів». Якщо транзакція звертається до даних, то жодна інша транзакція не зможе додати чи видалити рядки, що можуть бути зчитані під час виконання транзакції. Реалізація цього рівня блокування виконується блокуванням діапазону ключів. Це блокування накладається не на конкретні рядки таблиці, а на рядки, що відповідають певній логічній умові.

Властивість **ізольованості** означає, що на роботу транзакції не мають впливати інші транзакції. Транзакція «бачить» дані в тому стані, в якому вони перебували до початку роботи

іншої транзакції, або в тому стані, в якому вони перебувають після її завершення. Одна транзакція не може переглядати проміжні стани даних, що використовуються іншими транзакціями. Якщо транзакція зчитує кілька разів ті самі дані, то вона повинна отримувати їх щоразу в тому стані, в якому вони були під час першого зчитування. Ще одним аспектом ізольованості є неможливість роботи з неповними результатами — незавершена транзакція не може передавати свої результати іншим транзакціям до підтвердження свого успішного завершення.

Після того, як було підтверджено успішне завершення роботи транзакції (Commit), система має гарантувати, що її результати не будуть втрачені, незважаючи на можливі перебої. Це й називається **довговічністю**. Для забезпечення довговічності застосовуються механізми відновлення бази даних.

Ключові терміни:

видавець, дистриб’ютор, передплатник, реплікація, розподілена база даних, розподілена система баз даних, розподілена система керування базами даних, транзакція, фрагментація

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.

Архітектура розподіленої СКБД наведена на рис. 9.1. 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.

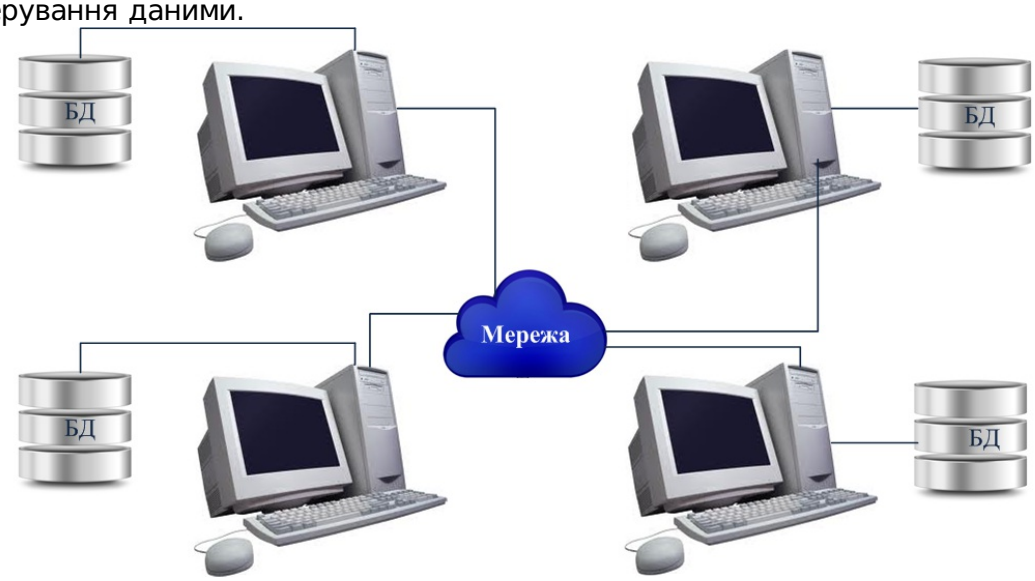


Рисунок 9.1 - Розподілена база даних

Залежно від типу програмного забезпечення розрізняють два типи РСКБД.

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.

9.2 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. Серед властивостей розподілених баз даних виділяють такі:

Локальна автономія - управління даними на кожному з вузлів розподіленої системи виконується локально.

Незалежність вузлів - всі вузли рівноправні і незалежні, а розташовані на них БД є рівноправними постачальниками даних в загальний простір даних.

Безперервні операції - можливість безперервного доступу до даних в рамках розподіленої БД незалежно від їх розташування і незалежно від операцій, що виконуються на локальних вузлах.

Прозорість розташування - користувач, що звертається до БД, нічого не повинен знати про реальне, фізичне розміщення даних у вузлах інформаційної системи.

Прозора фрагментація - можливість розподіленого (тобто на різних вузлах) розміщення даних, логічно поєднаних в єдине ціле. Існує фрагментація двох типів: горизонтальна і вертикальна.

Прозоре тиражування - тиражування даних - це асинхронний процес перенесення змін об'єктів вихідної бази даних в бази, розташовані на інших вузлах розподіленої системи

Обробка розподілених запитів - можливість виконання операцій вибірки даних з розподіленої БД, за допомогою запитів, сформульованих на мові SQL

Обробка розподілених транзакцій - можливість виконання операцій оновлення розподіленої бази даних, які не порушують цілісність і узгодженість даних.

Незалежність від устаткування - як вузли розподіленої системи можуть виступати ПК будь-яких моделей і виробників

Незалежність від операційних систем - різноманіття операційних систем, керуючих вузлами розподіленої системи

Прозорість мережі - спектр підтримуваних конкретною СУБД мережевих протоколів не має бути обмеженням системи, заснованої на розподіленій БД

Незалежність від баз даних - в розподіленій системі можуть працювати СУБД різних виробників, і можливі операції пошуку і оновлення в базах даних різних моделей і форматів.

9.3 Логічна архітектура розподілених баз даних

Логічна архітектура розподілених баз даних – це архітектура логічно взаємозалежних даних. Графічне зображення логічної архітектури розподілених баз даних наведене на рис. 9.2. 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 9.2 – 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.

9.4 Архітектура програмно-технічних засобів розподілених СКБД

9.4.1. Властивості архітектури

Архітектура програмно-технічного комплексу розподілених СКБД має три головні характеристики.

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.

Неоднорідність. Важливою характеристикою є міра однорідності програмно-технічних засобів СКБД.

Автономність. Характеризує, наскільки самостійно компоненти СКБД можуть виконувати свої функції. До питань автономності належать:

- автономність проектних рішень;
- автономність обчислень.

9.3.2. Різновиди архітектури

Основними різновидами архітектури програмно-технічних засобів розподіленої СКБД є:

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.

9.4 Розподілене зберігання даних

9.4.1. Фрагментація

Суть фрагментації полягає в тому, щоб поділити логічну базу даних на фрагменти з метою зберігання кожного фрагмента на певному вузлі мережі. Одиницями фрагментації можуть бути відношення та складені відношення. У випадку, коли одиницею фрагментації є відношення, вирішується проблема, яке відношення в якій базі даних має зберігатися. За іншого підходу допускається, що будь-яке відношення може бути зображене у вигляді сукупності фрагментів, що розподіляються за різними базами даних.

Фрагментацію, що здійснюється розподілом відношень за базами даних, теоретично здійснити нескладно, тому розглянемо проблему фрагментації власне відношень.

Фрагментація відношень. Завдання фрагментації відношень формулюється в такий спосіб. Нехай задане відношення R. Його потрібно зобразити у вигляді сукупності відношень R_1, ..., R_n так, щоб ця сукупність відповідала критеріям ефективності (за часом доступу, пам'яттю, завантаженістю комп'ютерів тощо).

Фрагментація є коректною, якщо вона повна, не містить перетинів і може бути реконструйована. Пояснимо ці терміни.

Декомпозиція відношення R на фрагменти R_1, R_2, ... R_n є повною тоді й лише тоді, коли кожен елемент даних з R належить якомусь із відношень R_i.

Декомпозиція відношення R на фрагменти R_1, R_2, ... R_n може бути реконструйована, якщо існує такий реляційний вираз $\varphi(R_1, R_2, ... R_n)$, що $R=\varphi(R_1, R_2, ... R_n)$.

Декомпозиція відношення R на фрагменти R_1, R_2, ... R_n не містіть перетинів, якщо будь-який елемент даних з R міститься не більш ніж в одному фрагменті.

Є три типи фрагментації відношень:

- горизонтальна;
- вертикальна;
- змішана

Горизонтальна фрагментація

Горизонтальна фрагментація полягає в розподілі кортежів відношення за фрагментами. Формально горизонтальну фрагментацію можна визначити в такий спосіб. Нехай задане відношення R і на ньому визначений предикат F_i. Тоді горизонтальний фрагмент Ri відношення визначається так:

$$R_i = \sigma_{F_i}(R)$$

Тобто горизонтальний фрагмент Ri - це множина кортежів R, що задовольняють умову F_i.

Вертикальна фрагментація

Суть вертикальної фрагментації полягає в тому, що відношення поділяється на дві чи більше проекцій, тобто схема відношення поділяється на певну множину підсхем.

Для відновлення вихідного відношення з фрагментів необхідно, щоб усі підсхеми містили первинний ключ. Існує інший підхід, коли під час поділу схеми до кожного фрагмента автоматично додається поле з ідентифікатором кортежу. Значення цього ідентифікатора зазвичай встановлюються системою автоматично.

Змішана фрагментація. Змішана фрагментація передбачає послідовне застосування вертикальної і горизонтальної фрагментацій.

Розподіл даних за вузлами мережі

Після отримання усіх необхідних фрагментів відношень постає проблема розподілу цих фрагментів за вузлами мережі. Єдиних рекомендацій стосовно того, як це робити, немає. Потрібно знайти оптимальний розподіл фрагментів F за вузлами мережі S за умови, що відомий розподіл додатків Q за вузлами мережі.

9.4.2. Реплікація

Реплікація — це механізм розподілу даних за вузлами, що дозволяє зберігати копії тих самих даних на різних вузлах мережі з метою прискорення пошуку і підвищення стійкості до відмов. Відношення чи фрагмент є реплікованим, якщо його копії зберігаються на двох або більше вузлах (копії ще називають репліками). За повної реплікації відношення його копії зберігаються на всіх вузлах мережі. Допускається ситуація, коли вся база даних зберігається на всіх вузлах мережі — це називається поєною реплікацією бази даних.

Механізми реплікації

Видавцем називають сервер, що надає розміщені на ньому дані для копіювання на інші сервери.

Дистриб'ютором називається сервер, що підтримує розподілену базу даних.

Передплатником називається сервер, що отримує копії даних, надані видавцем.

Є два методи відновлення даних передплатників:

- Реплікація за запитом. Передплатник періодично звертається до дистриб'ютора із

запитом про зміни, що відбулися з моменту останнього з'єднання.
- Примусова реплікація. Дистриб'ютор сам встановлює з'єднання з передплатником і пересилає йому необхідні дані.

Залежно від методу реплікації, передплатники можуть чи не можуть вносити зміни в репліковані дані. У найпростішому випадку змінювати дані може тільки видавець, у складніших моделях реплікації - передплатники і видавці. Змінені дані, отримані від усіх передплатників, синхронізуються і поєднуються з даними видавця, а потім розсилаються передплатникам.

Моделі реплікації

Є такі моделі реплікації:

- реплікація моментальних знімків;
- реплікація транзакцій.

Топологія реплікацій

Топологія реплікацій описує характер взаємозв'язків між учасниками реплікації:

- реплікація «один-до-багатьох» передбачає наявність одного видавця і кількох передплатників;
- реплікація «багато-до-одного» має місце, коли дані від кількох видавців пересилаються одному передплатнику;
- реплікація «багато-до-багатьох» означає, що дані від кількох видавців пересилаються кільком передплатникам.

9.5 Обробка розподілених транзакцій

Транзакція — набір команд, що виконується як єдине ціле. У транзакції або всі команди будуть виконані, або жодна з них не виконається. Якщо хоча б одна з команд транзакції не може бути виконана, здійснюється відкочування (відновлюється стан системи, в якому вона перебувала до початку виконання транзакції).

Транзакції мають задовольняти вимоги ACID (Atomicity, Consistency, Isolation, Durability — атомарність, несуперечність, ізольованість, довговічність), що гарантують правильність і надійність роботи системи.

Атомарність передбачає таке:

- виконуються всі операції транзакції або жодна з них не виконується;
- якщо виконання транзакції було перерване, то всі зроблені транзакцією зміни мають бути скасовані

Несуперечність означає, що транзакція, яка працює з несуперечною базою даних, після завершення роботи залишає її також у несуперечному стані. Транзакція не повинна порушувати цілісності бази даних

Для вирішення проблем одночасного доступу інститут ANSI розробив спеціальний стандарт, який визначає чотири рівні блокування (кожний вищий рівень передбачає виконання умов усіх нижчих рівнів)

- Рівень 0. Заборона «забруднення» даних.
- Рівень 1. Заборона некоректного зчитування.
- Рівень 2. Заборона неповторюваного зчитування.
- Рівень 3. Заборона «фантомів».

Властивість **ізольованості** означає, що на роботу транзакції не мають впливати інші транзакції. Транзакція «бачить» дані в тому стані, в якому вони перебували до початку роботи іншої транзакції, або в тому стані, в якому вони перебувають після її завершення.

Після того, як було підтверджено успішне завершення роботи транзакції (Commit), система має гарантувати, що її результати не будуть втрачені, незважаючи на можливі перебої. Це й називається **довговічністю**.

На відміну від мережі з централізованою базою даних кожний з вузлів мережі з розподіленою базою даних містить свою базу даних. Кожна розглядається як логічно єдина база. Розподілена СКБД — це повноцінна СКБД, що виконує всі необхідні функції з керування даними. Залежно від типу програмного забезпечення розрізняють однорідні та неоднорідні РСКБД

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.

До основних характеристик РСКБД слід віднести такі, як: розподіленість, неоднорідність та автономність. Вони визначають взаємодію комп'ютерів, засоби зв'язку та самостійність

кожного окремого вузла.

Різновиди архітектури програмно-технічних засобів розподіленої СКБД визначають розподіл баз даних між вузлами мережі, місце збереження інформації в мережі.

Визначають два основні механізми розподіленого зберігання даних: фрагментація та реплікація. Відмінність між ними полягає у розподілу логічної бази даних на фрагменти: фрагментація визначає поділ бази на окремі блоки, які зберігаються на певному вузлі мережі, а реплікація - збереження копій бази даних на кодному вузлі.

Транзакції до розподілених баз даних мають задовольняти вимогам атомарності, несуперечності, ізольованості та довговічності.

1. Що таке однорідні й неоднорідні розподілені бази даних?
2. Що таке розподіленість, неоднорідність й автономність баз даних?
3. Які механізми розподіленого зберігання даних ви знаєте?
4. Які різновиди фрагментації баз даних ви знаєте?
5. Що таке реплікація? Які існують механізми й моделі реплікації?
6. Що таке правила ACID виконання транзакцій?

Тема 10 - Обєктно-орієнтовані бази даних

Ключові терміни:

OQL ODMG, клас, літерал, об’єкт, поведінка об’єкта, поліморфізм, реалізаційна частина, стан об’єкта, успадкування, інтерфейсна частина

10.1 Об’єктно-орієнтована модель ODMG

В об’єктно-орієнтованій моделі дані та методи, що їх обробляють, об’єднуються в структури, які називаються об’єктами. Типи об’єктів називаються класами. З точки зору баз даних є такі важливі особливості об'єктно-орієнтованої моедлі (далі ООМ):

- підтримка структур даних, що мають довільний рівень складності;
- ідентифікованість та унікальність об’єктів;
- належність об’єктів класам;
- інкапсуляція;
- успадкування та ієрархії класів;
- поліморфізм.

Складні структури даних. Складні об’єкти будуються з простіших за допомогою конструкторів. Найпростішими об’єктами є: числа, символи, символьні рядки довільної довжини, булеві змінні тощо. Існують різні конструктори складних об’єктів (кортежів, множин, мультимножин, списків та масивів). Мінімальний

набір конструкторів, який повинна мати система, - це конструктори множин, списків і кортежів. Множини необхідні, оскільки завдяки їм набори об'єктів реального світу зображуються в природний спосіб. Кортежі надають спосіб зображення властивостей сутностей. Списки та масиви потрібні для відображення впорядкованості об'єктів реального світу, а також для зображення широкого кола математичних об'єктів.

Будь-який конструктор має бути застосовним до будь-якого об'єкту (наприклад, повинна надаватися можливість побудови множини з масивів або масиву з множин). Конструктори реляційної моделі не мають такої властивості, оскільки конструкція множини може бути застосована лише до кортежів, а конструкція кортежу - лише до атомарних значень. Навіть реляційна модель, що підтримує ненормалізовані відношення (тобто відношення, які не перебувають у першій нормальній формі), не має вказаної властивості, оскільки конструкцією верхнього рівня завжди має бути відношення.

Маніпулювання складними об'єктами забезпечується відповідними операціями, які часто розповсюджуються на всі компоненти таких об'єктів. Прикладом може бути вибирання чи видалення складного об'єкту або створення його копії. Існує можливість визначати додаткові операції над складними об'єктами.

Ідентифікованість, унікальність і стан об'єктів. Кожний об'єкт є унікальним, тобто забезпечується унікальна ідентифікація об'єктів (для мов програмування унікальними ідентифікаторами можуть бути адреси пам'яті, за якими зберігаються об'єкти).

Стан об'єкта — це поточне значення, приписане об'єкту. Об'єкт може мати єдиний стан протягом свого життєвого циклу або переходити з одного стану в інший. Оскільки об'єкти мають властивість інкапсуляції (що буде розглянута нижче), то стан об'єкта є абстракцією, яка визначається лише через його поведінку (методи).

Унікальність об'єкта не залежить від його стану. Два об'єкти, що перебувають в одному й тому ж стані, є рівними, але не ідентичними. Не може існувати двох або більше екземплярів одного об'єкта (двох копій одного й того самого об'єкта з одним і тим самим ідентифікатором) У моделі з ідентифікованістю об'єктів об'єкт існує незалежно від свого значення. Отже, є два поняття еквівалентності об'єктів: об'єкти можуть бути ідентичними (бути одним і тим самим об'єктом) або вони можуть бути рівними (перебувати в одному й тому самому стані). У цьому контексті слід розглянути такі два аспекти: розрізнення та змінення об'єктів.

Розрізнення об'єктів. У моделі з об'єктами, що ідентифікуються, два об'єкти можуть спільно використовувати компоненти (зокрема інші об'єкти). Отже, схематичним відображенням складного об'єкта є граф, натомість у системі без ідентифікованості об'єктів - це дерево.

ПрикладСховати

Розглянемо такий приклад: людина має ім'я, вік і дітей. Припустимо, що Іван і Марія мають сина на ім'я Петро. У реальному житті можуть мати місце дві ситуації: Іван і Марія є батьками однієї і тієї самої дитини або кожний із них має по сину. У моделі з ідентифікованістю об'єктів адекватно відображуються обидві ситуації.

Зміна об'єктів. Припустимо, що Іван і Марія є батьками хлопчика на ім'я Петро. Тоді зміни, що стосуються сина Марії, будуть виконуватися з об'єктом Петро, відтак і з сином Івана.

Поведінка об'єктів. Поведінка об'єкта це сукупність операцій (методів), які він надає. Лише через ці операції розкривається семантика об'єкта. Виконання операцій є єдиним способом взаємодії між об'єктами. Всі можливі операції об'єкта утворюють його інтерфейс. Лише використовуючи операції, можна змінити стан об'єкта.

Класи об'єктів. В об'єктно-орієнтованій моделі клас узагальнює спільні риси об'єктів, що мають однакові властивості, й відповідає поняттю абстрактного типу даних. Клас означує спосіб реалізації множини об'єктів, встановлюючи їхню структуру, поведінку та інтерфейс, тобто спосіб запам'ятовування інформації про їхні стани. Проте власне стан має запам'ятовувати сам об'єкт.

Клас є водночас фабрикою та сховищем об'єктів. Як фабрика об'єктів клас використовується для створення нових об'єктів. Термін сховище об'єктів означає, що до класу приєднується набір об'єктів, які є його екземплярами. Отже, класи використовуються для створення об'єктів і маніпулювання ними.

Однією з основних властивостей класу, відтак і його об'єктів, є інкапсуляція.

Інкапсуляція. Інкапсуляція вимагає, щоб дані та програмні коди для маніпулювання даними були приховані. З цієї точки зору об'єкт поділяється на інтерфейсну й реалізаційну частини. Інтерфейсна частина є специфікацією набору операцій, допустимих над об'єктом. Лише ця частина об'єкта видима для методів інших об'єктів. Реалізаційна частина складається з даних, що описують

стан об'єкта, і процедур, що реалізують операції над об'єктом. Інкапсуляція специфікується на рівні оголошення класу.

Успадкування. Успадкування є механізмом, що дає змогу створювати нові класи з використанням даних і методів інших класів. Це дає можливість деякі властивості, спільні для багатьох класів, описувати в базовому класі.

ПрикладСховати

Наприклад, якщо необхідно додати новий клас Одяг, що повністю збігається з класом Товар, за винятком наявності додаткових відомостей про розмір, мовою Java це можна записати так:

```
public class Одяг extends Товар {
int розмір:
}
```

У такий спосіб ми вказали, що Одяг — це Товар, тому перший має всі дані й методи другого, а також власні дані, що зберігаються в змінній розмір. Успадкування дає змогу будувати ієрархію класів.

Поліморфізм. Принцип поліморфізму є розширенням принципу успадкування й дає змогу переозначувати методи в успадкованих класах.

ПрикладСховати

Наприклад, є кілька різновидів товарів, для яких властиві специфічні правила обчислення ціни. Зокрема товарами можуть бути продукти, на які не діють націнки, та одяг, на який націнка встановлена. Всі товари успадковують метод Сумарна ціна() з базового класу Товар, але правила обчислення ціни можуть бути різними. Це означає, що кожний клас, який є нащадком класу Товар, може мати власну реалізацію методу Сумарна_ціна(). Тому вираз х. Сумарна ціна () може означати різні правила обчислення ціни залежно від того, екземпляром якого класу є об'єкт х.

10.2 Мова опису об'єктів ODL ODMG

Будь-яка СКБД має мову опису даних (МОД), що використовується для опису схем баз даних. Мова опису об'єктів ODL ODMG розглядається як розширення МОД, призначене для опису об'єктів, їхніх атрибутів, зв'язків та операцій. Основою цієї мови стала мова IDL (Interface Definition Language), розроблена групою OMG.

Мова ODL є абстрактною в тому розумінні, що згенерована ODL-схема має бути незалежною від мов програмування та конкретної СКБД. У зв'язку з цим в ODL розглядаються лише аспекти означення об'єктів різних типів і повністю ігноруються питання реалізації методів. Згенерована ODL-схема може вільно переміщуватися між СКБД, що підтримують концепцію ODMG, використовуватися програмами, записаними різними мовами програмування, і навіть транслюватися в конкретні МОД, наприклад ті, які базуються на стандарті SQL-1999.

10.2.1. Основні положення

ODL - це мова, призначена насамперед для специфікації класів. Вона підтримує об'єктну модель ODMG і не є мовою програмування. Більше того, ODL незалежна від мов програмування. Основна мета розробки цієї мови - створити єдину основу для опису об'єктів і тим самим забезпечити перенесення схем об'єктних даних між різними ООСКБД.

Основні положення об'єктної моделі даних ODMG:

- базовим поняттям моделі є об'єкт;
- поведінка об'єкта визначається за допомогою множини його операцій;
- стан об'єкта визначається за допомогою множини його властивостей:
- об'єкти належать до класів, саме через класи вони специфікуються;
- клас має інтерфейсну та реалізаційну частини;
- клас є об'єктом;
- ODL специфікує класи.

В об'єктній моделі ODMG мова йде насамперед про типи, а вже потім про класи. Клас розглядається як різновид типу, що має одну реалізацію. У загальному випадку допускається, що тип має кілька реалізацій. Множинність реалізацій типу необхідна для підтримки неоднорідних баз даних, розподілених у мережі, та середовищ з різними мовами програмування й компіляторами. Нас не пікавить реалізація, тому далі основна увага приділятиметься класам.

10.2.2. Система типів ODL

Базовими типами мови є: integer, float, string, boolean, перелічувані типи, що створюються за допомогою ключового слова enum, та класи. Похідні типи створюються за допомогою конструкторів типів. Є конструктор Struct для структур і чотири конструктори для типів колекцій: Set, Bag, List, Array. Опис структур і колекцій наведено далі.

10.2.3. Об’єкти

Основні характеристики об’єктів:

- OID унікально ідентифікує об’єкт, відрізняючи його від інших об’єктів тієї предметної області, де він був створений. Будь-який об’єкт має лише один OID, але може мати більше одного імені. Об’єкти можуть ідентифікуватися предикатами, визначеними на їхніх властивостях.
- Видалення об’єкта не призводить до рекурсивного видалення пов’язаних із ним об’єктів.

На рис. 10.1 наведена класифікація об’єктів.

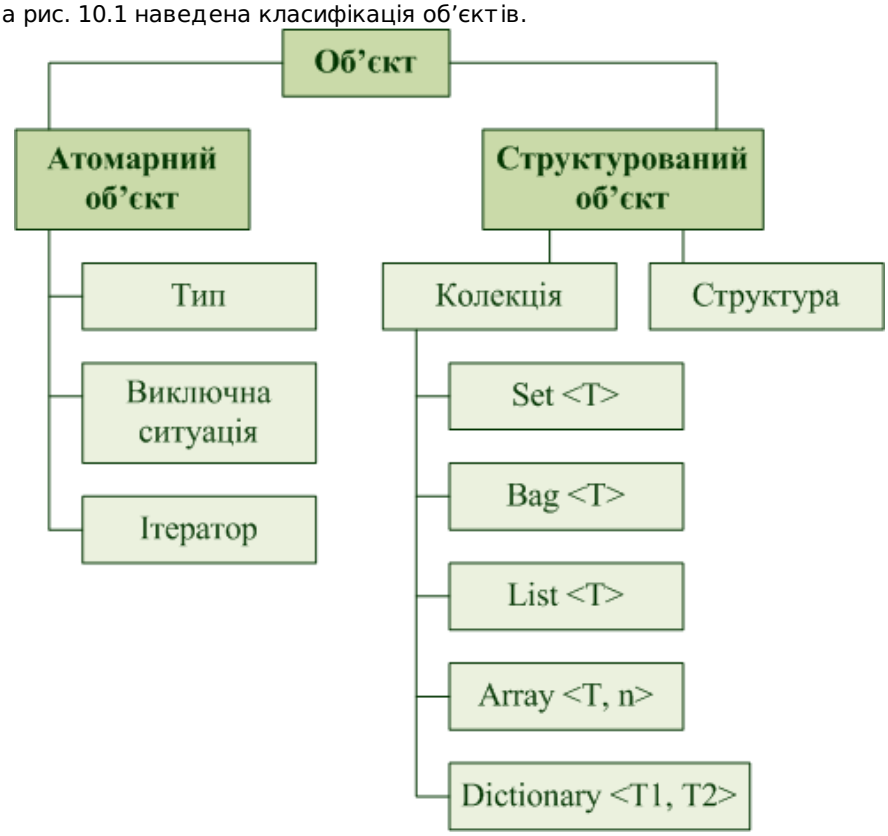


Рисунок 10.1 – Класифікація об’єктів у ODL

10.2.4. Літерали

Літерали — це об’єкти, екземпляри яких не можна змінювати. Для наперед визначених типів літералів не можна змінювати операції. На рис. 10.2 наведена класифікація літералів.

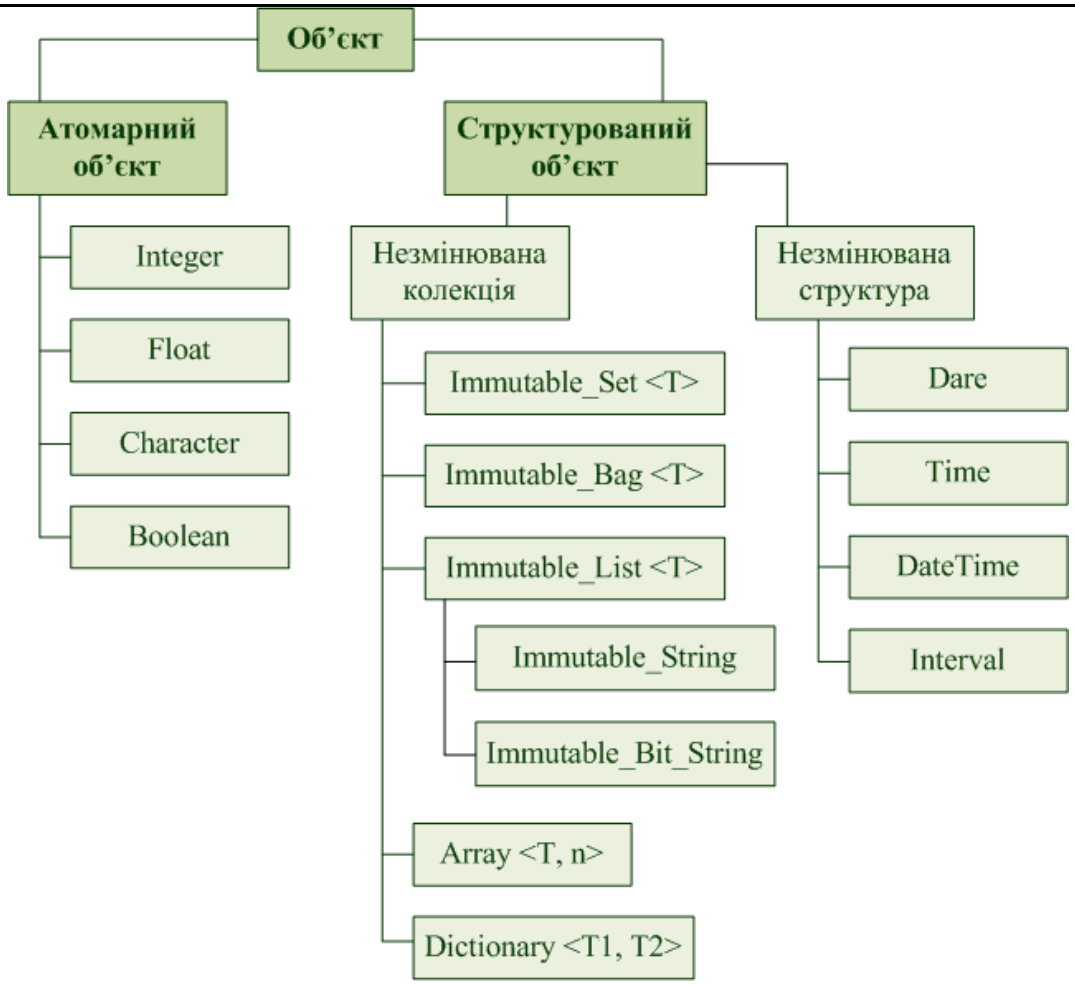


Рисунок 10.2 – Класифікація літералів

10.3 Об'єктна мова запитів OQL ODMG

OQL ODMG - це незалежна мова запитів до об'єктної моделі даних ODMG, синтаксис якої базується на мові SQL. Окрім того, передбачається можливість її використання в мовах програмування.

Мова запитів орієнтована на побудову виразів, її конструкції мають такі властивості:

- будь-який запит є виразом, що має тип — об'єкт або літерал;
- вирази та операції над ними можуть вкладатися одне в одне;
- результатом виконання запиту є об'єкти, що належать типам, означеним у моделі ODMG, і можуть брати участь у формуванні виразів.

Мова має високорівневі примітиви для маніпулювання множинами, об'єктами, структурами, масивами і списками. У ній відсутні оператори оновлення, замість них використовуються операції, визначені для об'єктів. Передбачається, що всі створювані об'єкти мають OID, а літерали унікально ідентифікуються своїм значенням.

10.3.1. Запити OQL

В OQL запитом може бути будь-який вираз, що повертає об'єкт, колекцію об'єктів, літерал або колекцію літералів. Оскільки запит — це вираз, а будь-який вираз має тип, то й будь-якому запиту ставиться у відповідність тип.

Найпоширенішим різновидом запиту, як і в SQL, є select-запит.

Приклад..Сховати

Приведений нижче запит знаходить множину осіб чоловічої статі, й при цьому типом значення, що повертається, буде Bag<Person>:

```
SELECT x
FROM persons x
WHERE x.sex = 'M'
```

Зазначимо, що select-вираз повертає колекцію типу Bag, тобто мультимножину (множина зі значеннями, що повторюються). Використання слова DISTINCT приводить до того, що тип колекції, яка повертається, - Set, тобто множина. Наведений нижче запит повертає список номерів факультетів для людей на ім'я Pete, тобто літерал типу Set<integer>.

```
SELECT DISTINCT x.faculty_id FROM persons x WHERE x.name = 'Pete'
```

Зауважте, що в запитах слід звертатися саме до екстентів, а не до класів.

10.3.2. Обчислення проміжних результатів

Запит можна не лише сформулювати для виконання, але й визначити для подальшого використання в інших запитах (певний аналог віртуальних таблиць, створюваних командою CREATE VIEW у мові SQL). Це досягається за допомогою означення запиту, що має такий вигляд:

define ім'я_запиту as вираз

ПрикладСховати

```
define Joe as element (
        SELECT x
        FROM students x
        WHERE x.name = 'Joe')
```

Тут element є перетворювачем типу, що зводить тип одноелементної колекції до типу елементу. Отже, Joe є іменем об'єкта класу students із значенням атрибута name рівним 'Joe'. Тепер до об'єкта Joe можна звернутися, наприклад, для отримання значень його атрибутів. Так, вирази Joe.Birthdate. Joe.faculty_id. Joe.passport повертають дату народження, номер факультету і номер паспорта студента на ім'я Joe.

Ім'я запиту розглядається як елементарний вираз, як і змінні, атоми та пойменовані об'єкти. Приклади елементарних виразів: 27, nil, students, Joe.

10.4 Архітектура ООСКБД

Є різні підходи до створення ООСКБД. Крім розробки власне об'єктно- орієнтованих СКБД, є численні способи поєднання об'єктно-орієнтованої та реляційної моделей даних. Усі ці підходи ми розглянемо далі.

10.4.1. Розширення реляційних СКБД

Реляційні СКБД надають можливість звертатися до них програмам, написаним різними, зокрема об'єктно-орієнтованими, мовами програмування. У цьому випадку об'єктно-орієнтовані прикладні програми виконують усі функції, пов'язані з відображенням об'єктної моделі в реляційну, тобто перетворюють об'єкти на структури даних, які можуть бути безпосередньо записані в табличні БД, підтримують властивості успадкування, інкапсуляції, зв'язування з об'єктами їхніх методів.

РСКБД бере на себе єдину функцію - зберігання даних, які пов'язані з об'єктами, причому зберігання у вигляді реляційних таблиць, усе інше виконує прикладна програма.

Даний підхід передбачає включення до складу РСКБД засобів, які полегшують процес відображення об'єктів у базі даних і маніпулювання ними. Тобто сама РСКБД удосконалюється, полегшуючи обробку об'єктів, але залишається при цьому реляційною. До можливих розширень РСКБД належать такі.

- Надання можливості автоматичного створення унікальних ідентифікаторів кортежів відношень. Наприклад, в Oracle є тип rowid, який виконує цю функцію. Такий тип сприяє вирішенню проблеми підтримки OID, хоча й не вирішує її повністю, оскільки OID має бути унікальним у всій базі даних, а не в межах одного відношення, як rowid.
- Створення механізмів означення нових типів даних (такі механізми є в мовах програмування). Введення нових типів, навіть якщо вони відповідають досить складним структурам даних, не суперечить реляційній моделі. Інша річ, що реляційна модель в «чистому» вигляді не надає можливості працювати зі складеними структурами даних, вони розглядаються як атомарні.

Перевага підходу, який базується на розширенні реляційних СКБД, полягає в тому, що надається можливість використовувати всю потужність реляційних систем баз даних. Недолік - слабка розвиненість засобів зображення об'єктів і маніпулювання ними, багато з цих функцій виконують прикладні програми.

10.4.2. Створення самостійних ООСКБД

Об'єктно-орієнтовані СКБД реалізують гнучку модель даних, яка базується на тій же парадигмі, що й об'єктно-орієнтовані мови програмування. ООБД забезпечують глибшу інтеграцію з об'єктно-орієнтованими додатками, ніж реляційні бази даних, і мінімізують обсяг роботи з програмування збереження і вибирання об'єктно-орієнтованих даних.

Переваги використання однакових моделей у додатках та базі даних виявляються тоді, коли об'єктно-орієнтовані моделі є складними. Якщо ієрархія успадкування є багаторівневою, колекції перетворюють граф об'єкта на павутину, застосуванням складно використовувати поліморфізм та посилання, об'єктна база даних робить такі застосування меншими і їхня продуктивність підвищується. Це знижує витрати на написання і налагодження програмного коду й підвищує загальну продуктивність застосувань

Самостійні ООСКБД забезпечують повну підтримку об'єктно-орієнтованої парадигми. Це передбачає безпосередню інтеграцію з об'єктно-орієнтованими мовами програмування,

підтримку об'єктних типів, зв'язків між об'єктами та операцій бази даних, які інтерпретують об'єкти (найчастіше об'єкти інтерпретуються як записи) Прикладами об'єктно-орієнтованих операцій бази даних можуть бути: створення посилань на об'єкти і завантаження з бази даних об'єктів, на які є посилання, забезпечення блокування на об'єктному рівні й повернення об'єктів як результатів запитів або операцій з курсорами.

Інкапсуляція. Стабільна (persistent) поведінка класу є незалежною від проектування бази даних, яке не потребує змінення реалізацій класів.

Успадкування. Об'єкт інтерпретується одним записом незалежно від того, наскільки глибокою є ієрархія успадкування. Всі дані об'єкту запам'ятовуються в єдиному записі. Коли об'єкт вибирається з бази даних, усі дані базового класу та всі дані похідних класів завжди доступні.

Поліморфізм. Концепція поліморфізму тісно пов'язана з успадкуванням. Стосовно об'єктно- орієнтованих баз даних це означає, що можна оперувати різними класами за допомогою спільного базового класу, одержуючи дані з об'єктів необхідного похідного класу. Тобто об'єкт базового класу може мати тип похідного класу. Якщо ви послідовно переглядаєте всі екземпляри базового класу, то будете одержувати всі об'єкти цього класу, незалежно від того, якого вони типу. Видалення екземпляра базового класу призводить до видалення всього об'єкту, включаючи дані похідних об'єктів.

Ідентифікованість об'єкта. ООСБД поєднують ідентифікованість об'єкта в базі даних з ідентифіковашстю об'єкта в оперативній пам'яті. Якщо виконується збереження об'єкта, то ООСБД «знає», чи відповідає він об'єкту з бази даних (чи є він у базі даних), а під час вибирання об'єкта з бази даних - чи є він у пам'яті. Програмісту не потрібно власноруч підтримувати відповідність між об'єктами бази даних і об'єктами в пам'яті.

Посилання на об'єкти. Самостійні ООСКБД дають можливість створювати в пам'яті посилання на об'єкти, а потім відображувати їх у базі даних і навпаки.

Переваги та недоліки. Перевагою ООСКБД є їхня повна узгодженість із об'єктно-орієнтованою парадигмою програмування, що знімає всі проблеми, пов'язані зі зберіганням і маніпулюванням об'єктами у базі даних.

Основний недолік пов'язаний з тим, що для самостійних ООСКБД слід вирішувати весь комплекс проблем, пов'язаних із СКБД, які вже вирішені в наявних реляційних СКБД.

10.4.3. Об'єктно-реляційні СКБД

Особливість даного підходу полягає в тому, що на базі наявних реляційних СКБД реалізується об'єктно-орієнтований інтерфейс. Робота з цим інтерфейсом здійснюється так само, як і в ООСКБД, але всі проблеми, пов'язані зі створенням і веденням баз даних, вирішуються в реляційній СКБД.

Основна проблема, пов'язана зі створенням такого інтерфейсу, — відображення об'єктно-орієнтованої моделі в реляційну. Є кілька способів інтеграції об'єктного і реляційного підходів, що будуть розглянуті далі.

Об'єктно-реляційний шлюз

Об'єктно-реляційний шлюз автоматично виділяє об'єкти програми й зберігає їх у реляційній базі даних.

Об'єктно-орієнтований додаток працює як звичайний користувач СКБД (рис. 10.3). Такий варіант дає змогу програмістам повністю сконцентруватися на об'єктно-орієнтованому проектуванні

Рисунок 10.3 – Використання об'єктно-реляційного шлюзу

Об'єктно-реляційний прошарок між об'єктною та реляційною СКБД

У разі використання об’єктно-орієнтованого прошарку програма взаємодіє з БД за допомогою мови ООСКБД, а прошарок замінює всі об’єктно-орієнтовані елементи цієї мови на їхні реляційні еквіваленти (рис. 10.4).

Рисунок 10.4 – Використання об’єктно-реляційного прошарку

За це доводиться розплачуватися продуктивністю. Окрім іншого, прошарок має перетворювати об’єкти на набори зв’язаних відношень, генерувати унікальні OID об’єктів і передавати ці дані до реляційної БД.

Уніфікована об’єктно-реляційна СКБД

Цей підхід передбачає створення гібридних об’єктно-реляційних СКБД, що можуть зберігати як табличні дані, так і об’єкти. Вважається, що майбутнє саме за гібридними СКБД. Сьогодні розробники реляційних СКБД починають додавати до своїх продуктів об’єктно-орієнтовані засоби.

Ключові терміни:

OQL ODMG, клас, літерал, об’єкт, поведінка об’єкта, поліморфізм, реалізаційна частина, стан об’єкта, успадкування, інтерфейсна частина

10.1 Об’єктно-орієнтована модель ODMG

В об’єктно-орієнтованій моделі дані та методи, що їх обробляють, об’єднуються в структури, які називаються об’єктами. Типи об’єктів називаються класами. З точки зору баз даних є такі важливі особливості об'єктно-орієнтованої моедлі (далі ООМ), які ми розглянемо далі.

Складні структури даних. Складні об’єкти будуються з простіших за допомогою конструкторів. Найпростішими об’єктами є: числа, символи, символьні рядки довільної довжини, булеві змінні тощо. Будь-який конструктор має бути застосовним до будь-якого об’єкту (наприклад, повинна надаватися можливість побудови множини з масивів або масиву з множин). Маніпулювання складними об’єктами забезпечується відповідними операціями, які часто розповсюджуються на всі компоненти таких об’єктів. Прикладом може бути вибирання чи видалення складного об’єкту або створення його копії. Існує можливість визначати додаткові операції над складними об’єктами.

Ідентифікованість, унікальність і стан об’єктів. Кожний об’єкт є унікальним, тобто

забезпечується унікальна ідентифікація об'єктів (для мов програмування унікальними ідентифікаторами можуть бути адреси пам'яті, за якими зберігаються об'єкти).

Стан об'єкта — це поточне значення, приписане об'єкту. Об'єкт може мати єдиний стан протягом свого життєвого циклу або переходити з одного стану в інший. Оскільки об'єкти мають властивість інкапсуляції (що буде розглянута нижче), то стан об'єкта є абстракцією, яка визначається лише через його поведінку (методи). Унікальність об'єкта не залежить від його стану.

Розрізнення об'єктів. У моделі з об'єктами, що ідентифікуються, два об'єкти можуть спільно використовувати компоненти (зокрема інші об'єкти). Отже, схематичним відображенням складного об'єкта є граф, натомість у системі без ідентифікованості об'єктів - це дерево.

Поведінка об'єктів. Поведінка об'єкта це сукупність операцій (методів), які він надає. Лише через ці операції розкривається семантика об'єкта.

Класи об'єктів. Клас означує спосіб реалізації множини об'єктів, встановлюючи їхню структуру, поведінку та інтерфейс, тобто спосіб запам'ятовування інформації про їхні стани. Проте власне стан має запам'ятовувати сам об'єкт. Однією з основних властивостей класу, відтак і його об'єктів, є інкапсуляція.

Інкапсуляція. Інкапсуляція вимагає, щоб дані та програмні коди для маніпулювання даними були приховані. З цієї точки зору об'єкт поділяється на інтерфейсну й реалізаційну частини. Інтерфейсна частина є специфікацією набору операцій, допустимих над об'єктом. Лише ця частина об'єкта видима для методів інших об'єктів. Реалізаційна частина складається з даних, що описують стан об'єкта, і процедур, що реалізують операції над об'єктом. Інкапсуляція специфікується на рівні оголошення класу.

Успадкування. Успадкування є механізмом, що дає змогу створювати нові класи з використанням даних і методів інших класів. Це дає можливість деякі властивості, спільні для багатьох класів, описувати в базовому класі.

Поліморфізм. Принцип поліморфізму є розширенням принципу успадкування й дає змогу переозначувати методи в успадкованих класах.

10.2 Мова опису об'єктів ODL ODMG

Будь-яка СКБД має мову опису даних (МОД), що використовується для опису схем баз даних. Мова опису об'єктів ODL ODMG розглядається як розширення МОД, призначене для опису об'єктів, їхніх атрибутів, зв'язків та операцій. Основою цієї мови стала мова IDL (Interface Definition Language), розроблена групою OMG.

10.2.1. Основні положення

ODL - це мова, призначена насамперед для специфікації класів. Вона підтримує об'єктну модель ODMG і не є мовою програмування. Більше того, ODL незалежна від мов програмування. Основна мета розробки цієї мови - створити єдину основу для опису об'єктів і тим самим забезпечити перенесення схем об'єктних даних між різними ООСКБД.

Основні положення об'єктної моделі даних ODMG:

- базовим поняттям моделі є об'єкт;
- поведінка об'єкта визначається за допомогою множини його операцій;
- стан об'єкта визначається за допомогою множини його властивостей:
- об'єкти належать до класів, саме через класи вони специфікуються;
- клас має інтерфейсну та реалізаційну частини;
- клас є об'єктом;
- ODL специфікує класи.

В об'єктній моделі ODMG мова йде насамперед про типи, а вже потім про класи. Клас розглядається як різновид типу, що має одну реалізацію.

10.2.2. Система типів ODL

Базовими типами мови є: integer, float, string, boolean, перелічувані типи, що створюються за допомогою ключового слова enum, та класи. Похідні типи створюються за допомогою конструкторів типів. Є конструктор Struct для структур і чотири конструктори для типів колекцій: Set, Bag, List, Array. Опис структур і колекцій наведено далі.

10.2.3. Об'єкти

Основні характеристики об’єктів:

- OID унікально ідентифікує об’єкт, відрізняючи його від інших об’єктів тієї предметної області, де він був створений. Будь-який об’єкт має лише один OID, але може мати більше одного імені. Об’єкти можуть ідентифікуватися предикатами, визначеними на їхніх властивостях.
- Видалення об’єкта не призводить до рекурсивного видалення пов’язаних із ним об’єктів.

На рис. 10.1 наведена класифікація об’єктів.

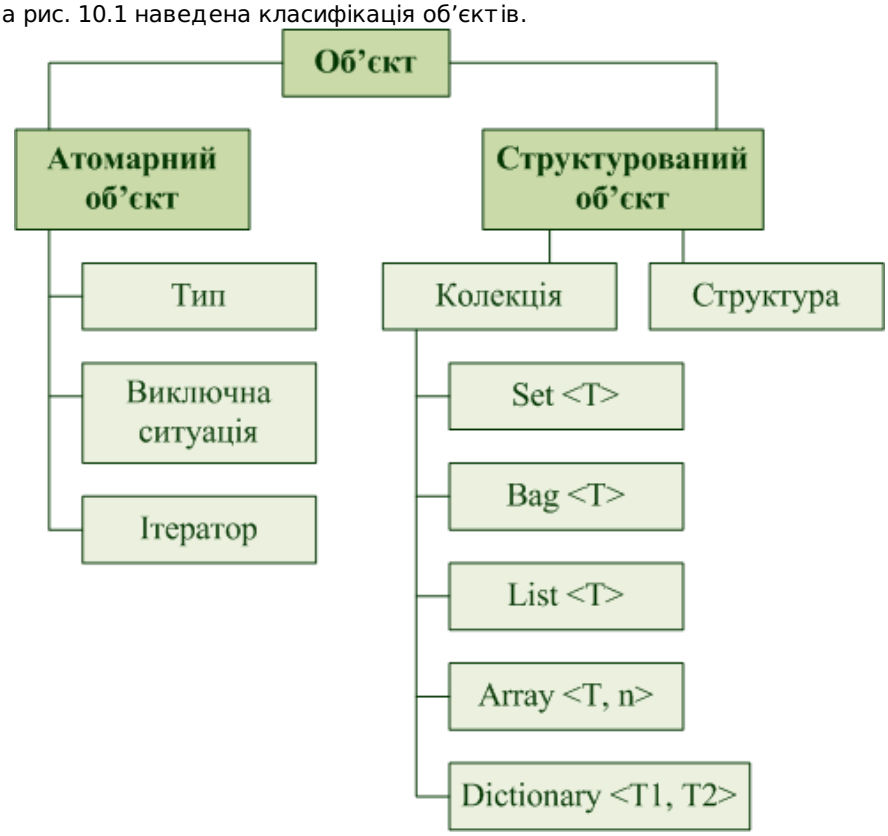


Рисунок 10.1 – Класифікація об’єктів у ODL

10.2.4. Літерали

Літерали — це об’єкти, екземпляри яких не можна змінювати. Для наперед визначених типів літералів не можна змінювати операції. На рис. 10.2 наведена класифікація літералів.

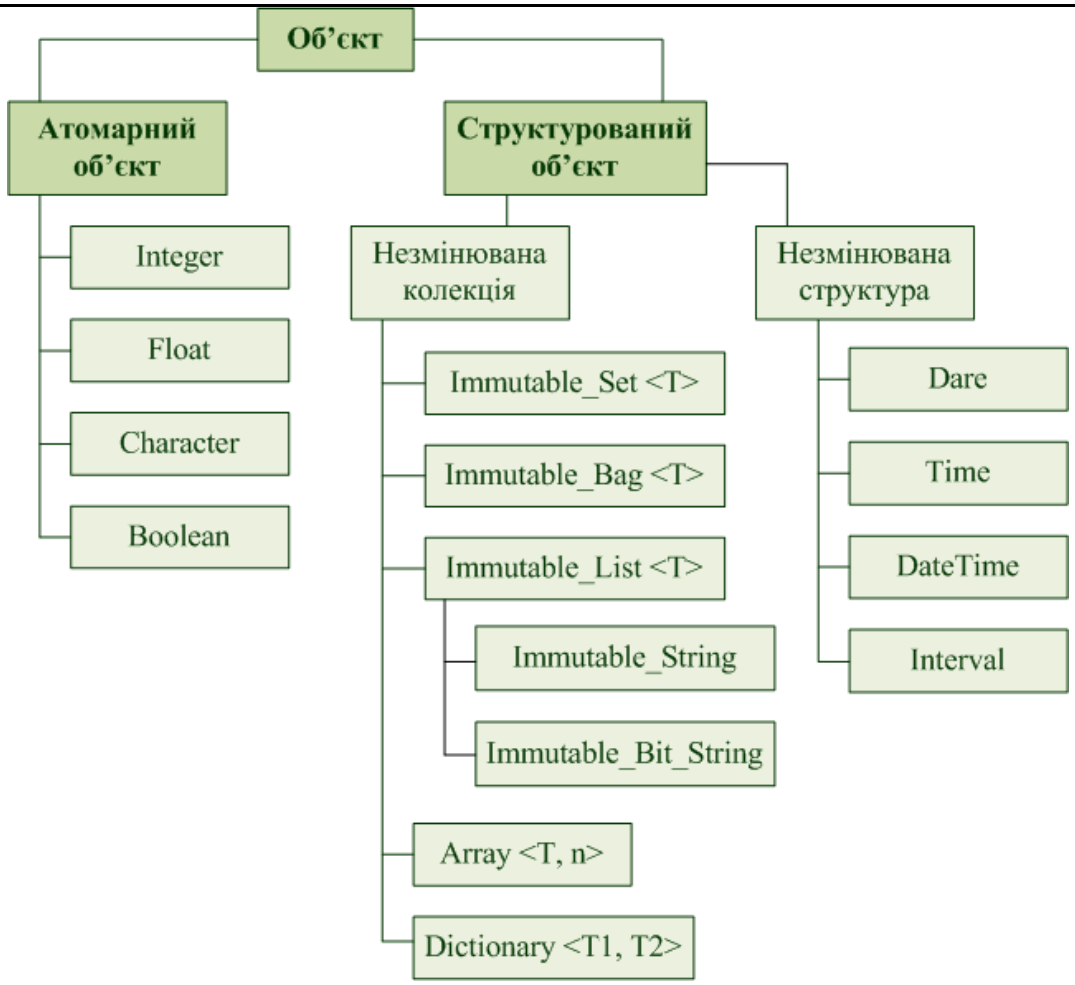


Рисунок 10.2 – Класифікація літералів

10.3 Об'єктна мова запитів OQL ODMG

OQL ODMG - це незалежна мова запитів до об'єктної моделі даних ODMG, синтаксис якої базується на мові SQL. Окрім того, передбачається можливість її використання в мовах програмування.

Мова запитів орієнтована на побудову виразів, її конструкції мають такі властивості:

- будь-який запит є виразом, що має тип — об'єкт або літерал;
- вирази та операції над ними можуть вкладатися одне в одне;
- результатом виконання запиту є об'єкти, що належать типам, означеним у моделі ODMG, і можуть брати участь у формуванні виразів.

10.3.1. Запити OQL

В OQL запитом може бути будь-який вираз, що повертає об'єкт, колекцію об'єктів, літерал або колекцію літералів. Оскільки запит — це вираз, а будь-який вираз має тип, то й будь-якому запиту ставиться у відповідність тип.

Найпоширенішим різновидом запиту, як і в SQL, є select-запит. В запитах слід звертатися саме до екстентів, а не до класів.

10.3.2. Обчислення проміжних результатів

Запит можна не лише сформулювати для виконання, але й визначити для подальшого використання в інших запитах (певний аналог віртуальних таблиць, створюваних командою CREATE VIEW у мові SQL). Це досягається за допомогою означення запиту, що має такий вигляд:

define ім'я_запиту as вираз

10.4 Архітектура ООСКБД

10.4.1. Розширення реляційних СКБД

Реляційні СКБД надають можливість звертатися до них програмам, написаним різними, зокрема

об'єктно-орієнтованими, мовами програмування. У цьому випадку об'єктно-орієнтовані прикладні програми виконують усі функції, пов'язані з відображенням об'єктної моделі в реляційну, тобто перетворюють об'єкти на структури даних, які можуть бути безпосередньо записані в табличні БД, підтримують властивості успадкування, інкапсуляції, зв'язування з об'єктами їхніх методів.

РСКБД бере на себе єдину функцію - зберігання даних, які пов'язані з об'єктами, причому зберігання у вигляді реляційних таблиць, усе інше виконує прикладна програма.

Даний підхід передбачає включення до складу РСКБД засобів, які полегшують процес відображення об'єктів у базі даних і маніпулювання ними. Тобто сама РСКБД удосконалюється, полегшуючи обробку об'єктів, але залишається при цьому реляційною. До можливих розширень РСКБД належать такі.

- Надання можливості автоматичного створення унікальних ідентифікаторів кортежів відношень.
- Створення механізмів означення нових типів даних (такі механізми є в мовах програмування).

Перевага підходу - можливість використовувати всю потужність реляційних систем баз даних. Недолік - слабка розвиненість засобів зображення об'єктів і маніпулювання ними.

10.4.2. Створення самостійних ООСКБД

Об'єктно-орієнтовані СКБД реалізують гнучку модель даних, яка базується на тій же парадигмі, що й об'єктно-орієнтовані мови програмування. ООБД забезпечують глибшу інтеграцію з об'єктно-орієнтованими додатками, ніж реляційні бази даних, і мінімізують обсяг роботи з програмування збереження і вибирання об'єктно-орієнтованих даних.

Переваги використання однакових моделей у додатках та базі даних виявляються тоді, коли об'єктно-орієнтовані моделі є складними. Якщо ієрархія успадкування є багаторівневою, колекції перетворюють граф об'єкта на павутину, застосуванням складно використовувати поліморфізм та посилання, об'єктна база даних робить такі застосування меншими і їхня продуктивність підвищується. Це знижує витрати на написання і налагодження програмного коду й підвищує загальну продуктивність застосувань

Самостійні ООСКБД забезпечують повну підтримку об'єктно-орієнтованої парадигми. Це передбачає безпосередню інтеграцію з об'єктно-орієнтованими мовами програмування, підтримку об'єктних типів, зв'язків між об'єктами та операцій бази даних, які інтерпретують об'єкти (найчастіше об'єкти інтерпретуються як записи) Прикладами об'єктно-орієнтованих операцій бази даних можуть бути: створення посилань на об'єкти і завантаження з бази даних об'єктів, на які є посилання, забезпечення блокування на об'єктному рівні й повернення об'єктів як результатів запитів або операцій з курсорами.

Перевагою ООСКБД є їхня повна узгодженість із об'єктно-орієнтованою парадигмою програмування, що знімає всі проблеми, пов'язані зі зберіганням і маніпулюванням об'єктами у базі даних.

Основний недолік пов'язаний з тим, що для самостійних ООСКБД слід вирішувати весь комплекс проблем, пов'язаних із СКБД, які вже вирішені в наявних реляційних СКБД.

10.4.3. Об'єктно-реляційні СКБД

Особливість даного підходу полягає в тому, що на базі наявних реляційних СКБД реалізується об'єктно-орієнтований інтерфейс. Робота з цим інтерфейсом здійснюється так само, як і в ООСКБД, але всі проблеми, пов'язані зі створенням і веденням баз даних, вирішуються в реляційній СКБД.

Основна проблема, пов'язана зі створенням такого інтерфейсу, — відображення об'єктно-орієнтованої моделі в реляційну. Є кілька способів інтеграції об'єктного і реляційного підходів, що будуть розглянуті далі.

Об'єктно-реляційний шлюз автоматично виділяє об'єкти програми й зберігає їх у реляційній базі даних. Об'єктно-орієнтований додаток працює як звичайний користувач СКБД (рис. 10.3). Такий варіант дає змогу програмістам повністю сконцентруватися на об'єктно-орієнтованому проектуванні

Рисунок 10.3 – Використання об'єктно-реляційного шлюзу

Об'єктно-реляційний прошарок між об'єктною та реляційною СКБД

У разі використання об'єктно-орієнтованого прошарку програма взаємодіє з БД за допомогою мови ООСКБД, а прошарок замінює всі об'єктно-орієнтовані елементи цієї мови на їхні реляційні еквіваленти (рис. 10.4).

Рисунок 10.4 – Використання об'єктно-реляційного прошарку

За це доводиться розплачуватися продуктивністю. Окрім іншого, прошарок має перетворювати об'єкти на набори зв'язаних відношень, генерувати унікальні OID об'єктів і передавати ці дані до реляційної БД.

Уніфікована об'єктно-реляційна СКБД

Цей підхід передбачає створення гібридних об'єктно-реляційних СКБД, що можуть зберігати як табличні дані, так і об'єкти. Вважається, що майбутнє саме за гібридними СКБД. Сьогодні розробники реляційних СКБД починають додавати до своїх продуктів об'єктно-орієнтовані засоби.

Нині ООСКБД приділяється багато уваги як з теоретичної, так і з практичної точок зору. Можна виділити три характерні риси сучасного стану досліджень у цій галузі:

- відсутність загальноприйнятої моделі даних;
- відсутність єдиної формальної теорії;
- активна експериментаторська діяльність.

Реляційна та об'єктна моделі відрізняються одна від одної і тому в додатках необхідно підтримувати дві різні моделі даних. Створивши ієрархічну структуру класів, слід спроектувати під них реляційну схему бази даних. Після створення схеми бази даних необхідно повернутися до проектування класів і визначити, яким чином реалізувати стабільну поведінку кожного з них з урахуванням створеної схеми бази даних

Створення відображення між об'єктно-орієнтованим додатком і реляційною схемою бази даних — це не тривіальне завдання. Реляційна модель може лише апроксимувати об'єктно-орієнтовану ієрархію, тому багато її переваг втрачаються під час збереження і вибірки даних. Необхідно розробити складний програмний продукт для збереження об'єктної моделі в таблицях, подальшої вибірки даних із них та відновлення об'єктної моделі в оперативній пам'яті Виникає також проблема продуктивності, пов'язана з необхідністю постійного реконструювання складних зв'язків між об'єктами

Мова маніпулювання та запитів має високорівневі примітиви для маніпулювання

множинами, об’єктами, структурами, масивами і списками. У ній відсутні оператори оновлення, замість них використовуються операції, визначені для об’єктів. Передбачається, що всі створювані об’єкти мають OID, а літерали унікально ідентифікуються своїм значенням

Є різні підходи до створення ООСКБД. Крім розробки власне об’єктно- орієнтованих СКБД, є численні способи поєднання об’єктно-орієнтованої та реляційної моделей даних, серед яких об’єктно-реляцій ний підхід має значні переваги порівняно з іншими.

1. Перелічіть основні складові об’єктно-орієнтованої моделі даних.
2. Назвіть основні положення об’єктної моделі даних ODMG.
3. Які типи даних означені в мові опису об’єктів ODL ODMG?
4. Які є різновиди об’єктів у ODL ODMG?
5. Що таке «колекція»? Які є вбудовані типи колекцій?
6. Що таке «структура»? Які стандартні операції означені для структури?
7. Що таке «клас»? Які є характеристики класу?
8. Як специфікуються зв’язки між двома класами? Які є типи зв’язків?
9. Що таке «вирази конструювання?
10. Які є підходи до реалізації об’єктно-орієнтованих СКБД?

1. Пасічник В.В.Організація баз даних та знань: підручник для ВНЗ/ В.В. Пасічник, В.А. Резніченко.– К.: Видавнича група BHV,2006.-384с.
2. Никитина Т.П. Базы данных и знаний: Учебное пособие / НикитинаТ.П.– Ярославль: МУБиНТ, 2002.– 58 с.
3. Ломакин В.В. Базы данных и базы знаний: учебное пособие/ Сост. Ломакин В.В. – Белгород: Изд-во БелГУ, 2010. – 216с.

Розділ 4 – Знайомство з базами знань

Тема 11 - Загальна характеристика баз знань

- 11.1 Базові поняття
- 11.2 Виведення на знаннях
- 11.3 Елементи експертних систем

Ключові терміни:

база знань, витягання знань, глибинне знання, експертна система, екстенсионал поняття, знання, керована ціль, керований антецедент, керований консеквент, машина виведення, поверхневе знання, придбання знань, формування знань, інтенсионал поняття

11.1 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.

11.2 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 *A*, *то* існує правило вигляду – «*Якщо A, то B*», тоді твердження *B* 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:

1. Зіставлення — зразок правила зіставляється з наявними фактами.
2. Вибір — якщо в конкретній ситуації може бути застосовано відразу декілька правив, то з них вибирається одне, найбільш відповідне по заданому критерію (вирішення конфлікту).
3. Спрацьовування — якщо зразок правила при зіставленні збігся з якими-небудь фактами з робочої пам'яті, то правило спрацьовує.
4. 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 рис. 11.1.

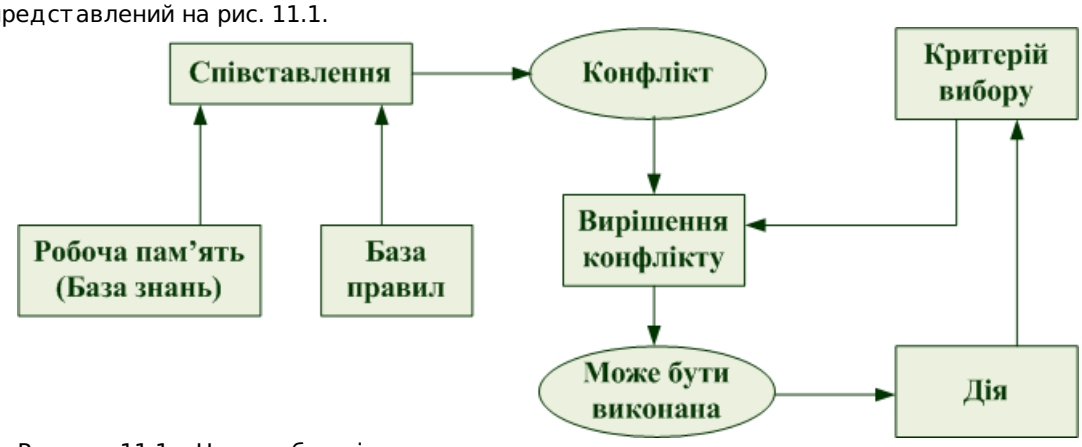


Рисунок 11.1 – Цикл роботи інтерпретатора

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:

1. Яку точку в просторі станів прийняти як початкову? Від вибору цієї точки залежить і метод здійснення пошуку — в прямому або зворотному напрямі.
2. 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 (рис. 11.2, 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.

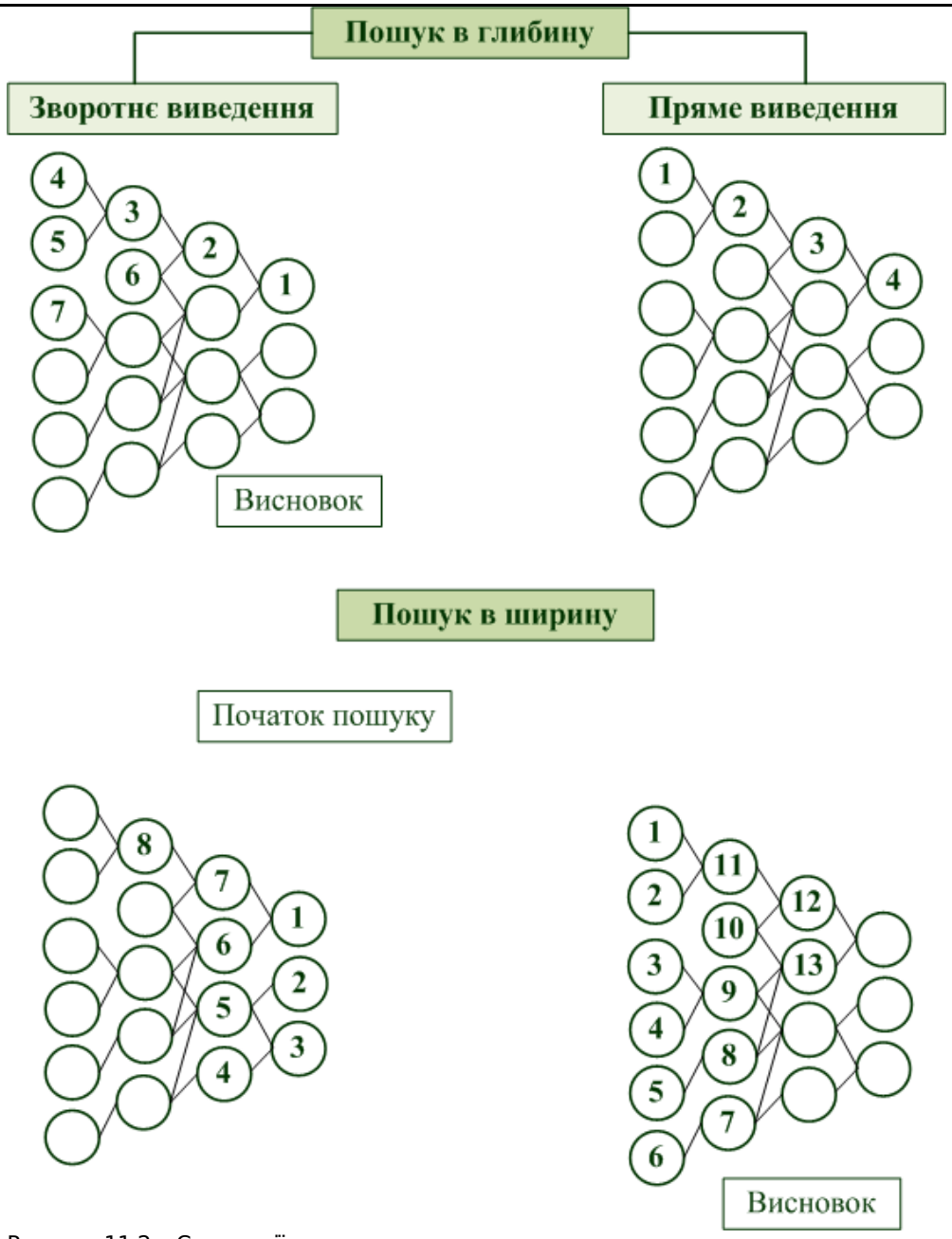


Рисунок 11.2 – Стратегії виведення

У системах з прямим виведенням по відомих фактах відшукується висновок, який з цих фактів виходить (див. рис.11.2, 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.

ПрикладСховати

Є фрагмент бази знань з двох правил:

1. П1. ЯКЩО «відпочинок — влітку» і «людина — активний», ТО «їхати в гори».
2. 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.

11.3 Елементи експертних систем

Експертна система – це комплекс комп'ютерного програмного забезпечення, що допомагає людині приймати обґрунтовані рішення. Експертні системи використовують інформацію, отриману заздалегідь від експертів – людей, які в якій-небудь області є кращими фахівцями. Всі експертні системи включають, принаймні, три основні елементи: базу знань, машину виводу і інтерфейс користувача.

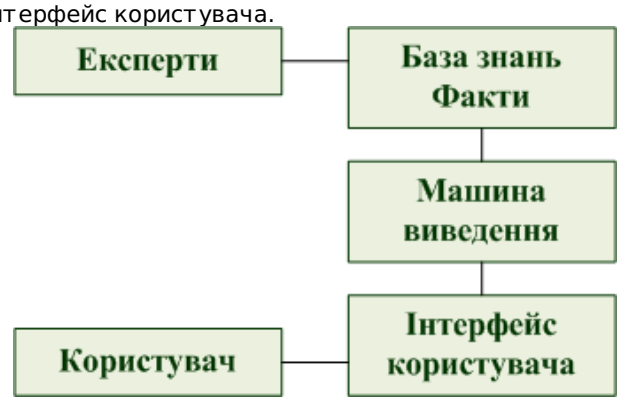


Рисунок 9.3 – Елементи експертної системи

База знань. База знань містить відомі факти, виражені у вигляді об'єктів, атрибутів і умов. Окрім описових уявлень про дійсність, вона включає вирази невизначеності – обмеження на достовірність факту. В цьому відношенні вона відрізняється від традиційної бази даних внаслідок свого символьного, а не числового або буквеного вмісту. При обробці інформації бази даних користуються заздалегідь визначеними логічними правилами Відповідно, база знань, що представляє вищий рівень абстракції, має справу з класами об'єктів, а не з самими об'єктами. База знань створюється людьми – консультантами.

Машина виведення. Головним в експертній системі є механізм, що здійснює пошук в базі знань за правилами раціональної логіки для здобуття рішень. Ця машина виведення приводиться в дію при здобутті запиту користувача і виконує такі завдання:

- порівнює інформацію, що міститься в запиті користувача, з інформацією бази знань;
- шукає певну мету або причинні зв'язки;
- оцінює відносну визначеність фактів, ґрунтуючись на відповідних коефіцієнтах довіри, пов’язаних з фактом.

Як випливає з її назви, машина виведення призначена для побудови висновків. Її дія аналогічно міркуванням експерта-людини, яка оцінює проблему і пропонує гіпотетичні рішення. У пошуку

цілей на основі запропонованих правил, машина виведення звертається до бази знань до тих пір, поки не знайде вірогідну дорогу до здобуття прийнятного результату.

Інтерфейс користувача. Завдання інтерфейсу користувача полягає в організації обміну інформацією між оператором і машиною виводу. Інтерфейс з використанням природної мови створює видимість довільної бесіди, застосовуючи повсякденні вирази в правильно побудованих пропозиціях. Очевидно, що ніж природніший такий інтерфейс, тим вище за вимогу до зовнішньої і оперативної пам'яті.

Отже, системи, що надають користувачеві максимум зручностей, витрачають більше ресурсів основної машини, доступних з видалених робочих станцій. Інструментальні засоби, призначені для роботи на персональних комп'ютерах, що мають обмежені можливості, неминуче приносять "дружність" в жертву ефективності.

Інтерфейс прямого введення повинен уміти розпізнавати мову, або, принаймні, достатню кількість ключових слів і фраз, щоб уловлювати їх зв'язок з даною проблемою і її пропонованими рішеннями.

Аспект людський. У роботі з експертними системами беруть участь як мінімум три групи людей. По-перше, адміністрація встановлює призначення експертної системи, обмежує предметну область, яку повинна охоплювати система, і точно визначає, які вигоди організація зможе отримувати з її використання. По-друге, фахівець із збору знань (інженер - когнітолог) збирає інформацію, необхідну для бази знань, порівнює відповідні дані і евристично організовує інформацію. По-третє, потенційний користувач вказує, як використовуватиметься система, якого роду проблеми належить вирішувати, і яким чином здійснюватиметься взаємодія програми з оператором. І, нарешті, системі потрібний експерт (частіше група експертів) у встановленої наочної області, для здобуття від нього знань, як у формі фактичної інформації, так і відносно аналітичних методів, які застосовуються для вирішення проблем в цій області.

Машинний аспект. Комп'ютерну частину системи представляють компоненти програмного забезпечення, які обробляють отриману інформацію про дійсність, закладену в символьному вигляді в базу знань. У базу знань поступають факти. Зв'язок між фактами представлений евристичними правилами – виразами декларативного знання про стосунки між об'єктами. Кожне таке правило має складову "якщо і компонент "те" (висновок-дія), які визначають прямий або зворотний причинно-наслідковий зв'язок. Дійсні твердження лише вірогідні, іншими словами, міра їх визначеності не завжди абсолютна.

Кількісні коефіцієнти визначеності збільшують точність міркування експертних систем. Загальноприйнята схема полягає в тому, щоб варіювати рівень довіри від 0, що представляє мінімальну міру визначеності, до 100 – вища міра.

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.

Бази знань найчастіше використовуються в контексті експертних систем, де з їх допомогою подаються навики і досвід експертів, зайнятих практичною діяльністю у відповідної області. Сучасні експертні системи працюють в основному з поверхневими знаннями. Це пов'язано з тим, що на даний момент немає адекватних моделей, які дозволяють працювати з глибинними знаннями.

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

Машина виведення є основним робочим компонентом експертних систем, вона застосовує логіку, чіткі міркування, встановлені правилами, для перевірки висновку і здобуття виводів.

1. Дайте визначення поняття «Знання».
2. Назвіть основні перетворення знань при машинній обробці?
3. Поясніть поняття інтенсионал та екстенсионал поняття.
4. Наведіть класифікацію знань.
5. Які способи здобуття знань ви знаєте?
6. Дайте визначення бази знань.
7. Як ви розумієте правило modus ponens?
8. Які методи виведення ви знаєте?
9. Які стратегії управління виведенням ви знаєте?
10. Дайте визначення поняття «Експертна система».
11. Вкажіть склад експертної системи?
12. Назвіть основні функції експертної системи?

Тема 12- Моделі знань

- Загальні поняття
- Продукційна модель знань
- Семантична модель знань
- Фрейм
- Логічна модель формальна

Ключові терміни:

дані, дія, логіко - лінгвістічна модель, мережа, моделювання, предикат, продукційна модель, пропозиціональна функція, семантика, терм, умова, фрейм

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:

1. Модель предметної області.
2. Модель бази даних.
3. 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.

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.).

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» («AKO — 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:

1. Однорідні (з єдиним типом відношень).
2. Неоднорідні (з різними типами відношень).

За типами відношень:

1. Бінарні (у яких відношення зв'язують два об'єкти).
2. 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.

ПрикладСховати

На рис. 12.1 подана семантична мережа. Як вершини тут виступають поняття «ЕОМ – ПЕОМ - Ноутбук», «Toshiba Satellite C660-1EM», «Процесор» і «Материнська плата».

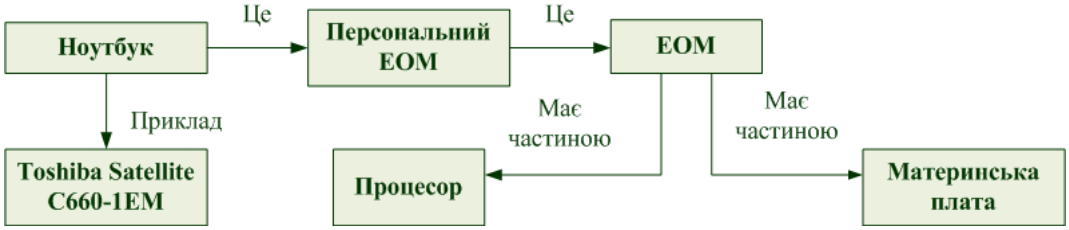


Рисунок 12.1 Семантична мережа

Дана модель представлення знань була запропонована американським психологом Килліаном. 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].

Фрейм

Термін фрейм (від англійською «frame», що означає «каркас» або «рамка») був запропонований Марвіном Лі Мінським в 70-і роки як позначення структури знань для сприйняття просторових сцен. Ця модель, як і семантична мережа, має глибоке психологічне обґрунтування. Фрейм – це абстрактний образ для представлення деякого стереотипу.

ПрикладСховати

Наприклад, вимовляння вголос слова «кімната» породжує в тих, що слухають образ кімнати: «приміщення з чотирма стінами, підлогою, стелею, вікнами і двері, певної площі». З цього опису нічого не можна прибрати (наприклад, прибравши вікна, ми отримаємо вже комірку, а не кімнату), але в нім є «дірки» або «слоти» - це незаповнені значення деяких атрибутів – наприклад, кількість вікон, колір стін, висота стелі, покриття підлоги і ін. У теорії фреймів такий образ кімнати називається фреймом кімнати.

Фреймом також називається і формалізована модель для відображення образу

Розрізняють фрейми-зразки, або прототипи, що зберігаються в базі знань, і фрейми- екземпляри, які створюються для відображення реальних фактичних ситуацій на основі даних, що поступають.

Модель фрейму є досить універсальною, оскільки дозволяє відображувати все різноманіття знань про світ через такі елементи:

- фрейми-структури, що використовуються для позначення об'єктів і понять (заїм, застава, вексель);
- фрейми - ролі (менеджер, касир, клієнт);
- фрейми - сценарії (банкротство, збори акціонерів, святкування іменин);
- фрейми - ситуації (тривога, аварія, робочий режим пристрою) і ін.

Традиційно структура фрейму може бути представлена як список властивостей:
(ІМ'Я ФРЕЙМА;
(ім'я 1-го слота; значення 1-го слота),
(ім'я 2-го слота: значення 2-го слота),
(ім'я N-го слота: значення N-го слота)).

При використанні табличної форми подання фрейму можна використовувати додаткові стовпці призначені для опису способу здобуття слотом його значення і можливого приєднання до того або іншого слота спеціальних процедур, що допускається в теорії фреймів.

Як значення слота може виступати ім'я іншого фрейму, так утворюються мережі фреймів. Існує декілька способів здобуття слотом значень у фреймі-екземплярі:

- за замовчуванням від фрейма-зразка (Default-значення);
- через спадковість властивостей від фрейму, вказаного в слоті AKO;
- по формулі, вказаній в слоті;
- через приєднану процедуру;
- явно з діалогу з користувачем;
- з бази даних.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж — так зване спадковість властивостей. Спадковість відбувається по AKO-зв'язках (A Kind of = це). Слот AKO вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

ПрикладСховати

На рис. 12.2 наведений приклад фреймової моделі ієрархічного типу. Фрейми утворюють ієрархію, остання породжує багаторівневу структуру, що описує або об’єкт за умови опису слотами лише властивостей об’єкту, або ситуацію чи процес за умови опису окремими слотами процедур, що під’єднані до фрейму та викликаються після його актуалізації.

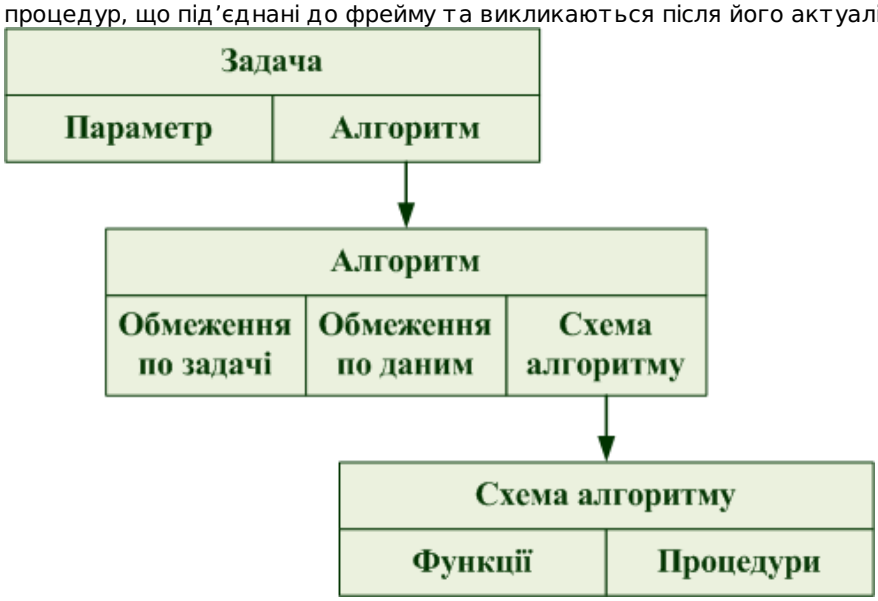


Рисунок 12.2 Фреймова модель

Формально фрейм – це тип даних вигляду:
$F=<N, S_1, S_2, S_3>$,
де N – ім’я об’єкту;
S_1 – множина слотів, що містять факти, визначаючі декларативну семантику фрейму;
S_2 – множина слотів, які забезпечують зв’язки з іншими фреймами (каузальні, семантичні та інш.);
S_3 – множина слотів, які забезпечують перетворення, що визначають процедурну семантику фрейму.

Основні переваги фреймів як моделі подання знань є здатність відображати концептуальну основу організації пам'яті людини. Спеціальні мови представлення знань в мережах фреймів FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймова «оболонка»

Карра і інші програмні засоби дозволяють ефективно будувати промислові ЕС. Широко відомі такі орієнтовані для фрейму експертні системи, як ANALYST, МОДІС, TRISTAN, Alteftld.

Логічна модель формальна

Основна ідея при побудові логічних моделей знань полягає в наступному – вся інформація, необхідна для вирішення прикладних завдань, розглядається як сукупність фактів і тверджень, що представляються як формули в деякій логіці. Знання відображуються сукупністю таких формул, а здобуття нових знань зводиться до реалізації процедур логічного виведення.

У основі логічних моделей знань лежить поняття формальної теорії, картежем, що задається у вигляді:

$S=<A, F, A_x, R>$,

де A – множина базових символів (алфавіт);
F – множина, звана формулами;
A_x – виділена підмножина апріорі дійсних формул (аксіом);
R – кінцева множина відношень між формулами, так звані правила виведення.
Основні переваги логічних моделей знань:

- як «фундамент» тут використовується класичний апарат математичної логіки, методи якої досить добре вивчені і формально обґрунтовані;
- існують досить ефективні процедури виведення, в тому числі реалізовані в мові логічного програмування «Пролог»;
- у базах знань можна зберігати лише безліч аксіом, а всі останні знання отримувати з них за правилами виведення.

У логічних моделях знань слова, що описують суть предметної області, називаються термами (константи, змінні, функції), а слова, що описують відношення суті, – ***предикатами***. Предикат – логічна N-арна пропозиціональна функція, визначена для предметної області і така, що набуває значень або істинності, або помилковості. Пропозиціональною називається функція, яка ставить у відповідність об'єктам з області визначення одне з значень істина/брехня. Предикат набуває значень «істина» або «брехня» залежно від значень вхідних в нього термів.

Спосіб опису предметної області, який використовується в логічних моделях знань, приводить до втрати деяких нюансів, властивих природному сприйняттю людини, і тому знижує описову можливість таких моделей. Складнощі виникають при описі «багатосортних» світів, коли об'єкти не є однорідними.

Так, вислови: «2 + 2 = 4» «Москва – столиця Росії» мають одне і те ж значення «істина», але різний сенс. З метою подолання складнощів і розширення описових можливостей логічних моделей знань розробляються псевдофізичні логіки, логіки, що оперують з нечіткими знаннями, емпіричними кванторами, забезпечуючі індуктивні (від приватного до загального), дедуктивні (від загального до приватного) і традуктівні (на одному рівні спільності) виведення. Такі розширені моделі, що об'єднують можливості логічного і лінгвістичного підходів, прийнято називати логіко - лінгвістічними моделями предметної області.

Ключові терміни:

дані, дія, логіко - лінгвістічна модель, мережа, моделювання, предикат, продукційна модель, пропозиціональна функція, семантика, терм, умова, фрейм

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 наступні види моделей:

1. Модель предметної області.
2. Модель бази даних.
3. Модель бази знань.

Кожна модель зберігає знання про модельований фрагмент предметної області (інформаційній моделі) і виконує мовну функцію.

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. 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.

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).

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.

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 стосунків:

1. Однорідні (з єдиним типом відношень).
2. Неоднорідні (з різними типами відношень).

За типами відношень:

1. Бінарні (у яких відношення зв'язують два об'єкти).
2. 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.

Фрейм – це абстрактний образ для представлення деякого стереотипу.

Фреймом також називається і формалізована модель для відображення образу Розрізняють фрейми-зразки, або прототипи, що зберігаються в базі знань, і фрейми- екземпляри, які створюються для відображення реальних фактичних ситуацій на основі даних, що поступають.

Модель фрейму є досить універсальною, оскільки дозволяє відображувати все різноманіття знань про світ через такі елементи:

- фрейми-структури, що використовуються для позначення об'єктів і понять (заїм, застава, вексель);
- фрейми - ролі (менеджер, касир, клієнт);
- фрейми - сценарії (банкротство, збори акціонерів, святкування іменин);
- фрейми - ситуації (тривога, аварія, робочий режим пристрою) і ін.

Традиційно структура фрейму може бути представлена як список властивостей:
(ІМ'Я ФРЕЙМА;
(ім'я 1-го слота; значення 1-го слота),
(ім'я 2-го слота: значення 2-го слота),
(ім'я N-го слота: значення N-го слота)).

Як значення слота може виступати ім'я іншого фрейму, так утворюються мережі фреймів. Існує декілька способів здобуття слотом значень у фреймі-екземплярі:

- за замовчуванням від фрейма-зразка (Default-значення);
- через спадковість властивостей від фрейму, вказаного в слоті АКО;
- по формулі, вказаній в слоті;
- через приєднану процедуру;
- явно з діалогу з користувачем;
- з бази даних.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж — так зване спадковість властивостей. Спадковість відбувається по АКО-зв'язках (A Kind of = це). Слот АКО вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються, тобто

переносяться, значення аналогічних слотів.

Основна ідея при побудові логічних моделей знань полягає в наступному – вся інформація, необхідна для вирішення прикладних завдань, розглядається як сукупність фактів і тверджень, що представляються як формули в деякій логіці. Знання відображуються сукупністю таких формул, а здобуття нових знань зводиться до реалізації процедур логічного виведення.

У основі логічних моделей знань лежить поняття формальної теорії, картежем, що задається у вигляді:

$S=<A, F, A_X, R>$,

де A – множина базових символів (алфавіт);

F – множина, звана формулами;

A_X – виділена підмножина апріорі дійсних формул (аксіом);

R – кінцева множина відношень між формулами, так звані правила виведення.

У логічних моделях знань слова, що описують суть предметної області, називаються термами (константи, змінні, функції), а слова, що описують відношення суті, – ***предикатами***. Предикат – логічна N-арна пропозиціональна функція, визначена для предметної області і така, що набуває значень або істинності, або помилковості. Пропозиціональною називається функція, яка ставить у відповідність об'єктам з області визначення одне з значень істина/брехня.

З метою подолання складнощів і розширення описових можливостей логічних моделей знань розробляються псевдофізичні логіки, логіки, що оперують з нечіткими знаннями, емпіричними кванторами, забезпечуючі індуктивні (від приватного до загального), дедуктивні (від загального до приватного) і традуктівні (на одному рівні спільності) виведення. Такі розширені моделі, що об'єднують можливості логічного і лінгвістичного підходів, прийнято називати логіко - лінгвістічними моделями предметної області.

Система штучного інтелекту – це система, що оперує знаннями про проблемні області. Без бази знань систем штучного інтелекту не існує. Для формалізації і представлення знань розробляються спеціальні моделі представлення знань і мови для опису знань, виділяються різні типи знань.

Моделі подання знань – це одне з найважливіших напрямів досліджень в галузі штучного інтелекту, оскільки без знань останній просто не може існувати. Наразі розроблено вже достатня кількість моделей. Кожна з них володіє своїми перевагами та недоліками, тому для кожної окремої задачі необхідно обирати саме власну модель. Від цього вибору буде залежати не лише ефективність виконання поставленої задачі, а й можливість її розв’язання взагалі.

1. Поясніть поняття «Моделювання»? Які типи моделей ви знаєте?
2. Які мови подання знань ви знаєте?
3. Дайте визначення та основні характеристики продукційної мови.
4. Дайте визначення та основні характеристики семантичної мови.
5. Дайте визначення та основні характеристики мови фреймів.
6. Назвіть основні характеристики логічної мови подання знань.
7. Від чого залежить вибір мови подання знань?

1. Пасічник В.В.Організація баз даних та знань: підручник для ВНЗ/ В.В. Пасічник, В.А. Резніченко.– К.: Видавнича група BHV,2006.-384с.
2. Никитина Т.П. Базы данных и знаний: Учебное пособие / НикитинаТ.П.– Ярославль: МУБиНТ, 2002.– 58 с.
3. Ломакин В.В. Базы данных и базы знаний: учебное пособие/ Сост. Ломакин В.В. – Белгород: Изд-во БелГУ, 2010. – 216с.

Курсова робота

Доброго дня, шановні студенти!

Пропоную вашій увазі курсову роботу з дисципліни "Організація баз даних та знань".

Метою виконання курсової роботи є закріплення та поглиблення теоретичних знань, оволодіння практичними навичками розроблення інформаційних систем для вирішення конкретних задач в області обробки даних, а також використання сучасних систем управління базами даних, та закріплення навиків самостійної роботи.

При виконанні курсової роботи необхідно вирішити такий ряд задач:

- дослідження особливостей предметної області бази даних, що розроблюється;
- вивчення додаткової літератури по розробленню бази даних в рамках обраної предметної області (далі ПО);
- розроблення бази даних (далі БД): концептуальне, логічне та фізичне моделювання даних;

- аналіз отриманих результатів роботи БД шляхом реалізації розробленої моделі засобами обраної СКБД.

Теми курсових робіт визначені в обсязі, достатньому для самостійного розроблення БД в рамках обраних ПО. Перелік тем із детальним описом ПО приведений у додаткових файлах до першої частини курсової роботи "Аналіз предметної області".

До теми та обсягу курсової роботи можуть бути внесені зміни після узгодження з керівником роботи. Допустимим є вибір теми роботи поза межами встановленого переліку, якщо нова тема пов'язана з виконуваною студентом науково-практичною діяльністю.

Варіанти завдання на курсову роботу призначаються викладачем відповідно номеру за списком групи.

Методичні вказівки до виконання розділів курсової роботи приведені в додаткових файлах до відповідних модулів індивідуального завдання.

Робота над курсовою роботою оцінюється за 100-бальною шкалою. Розподіл балів по частинах курсової роботи приведений в таблиці 1. При дотриманні термінів виконання, вказаних в таблиці, студент отримує максимальну кількість балів за відповідний етап роботи за умови відсутності помилок. При відсутності на вказаний термін етапу роботи, кількість балів зменшується на 10% за кожен пропущений тиждень.

Таблиця 1 – Рекомендовані терміни виконання розділів курсової роботи з дисципліни «ОБДЗ»

Номер розділу	Назва розділу роботи	Кількість балів за розділ	Сумарна кількість балів за розділи	Термін кінцевої сдачі розділу
1	Аналіз предметної області	10	10	30 січня
2	Моделювання даних	50	60	30 березня
3	Реалізація в СКБД	40	100	30 квітня

Бажаю успіхів в роботі!

- Зміст та обсяг пояснювальної записки до курсової роботи
  - Структура роботи
  - Короткий опис розділів курсової роботи
- Вимоги до оформлення пояснювальної записки

ЗМІСТ ТА ОБСЯГ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ ДО КУРСОВОЇ РОБОТИ

Структура роботи

Результати курсової роботи для перевірки повинні бути подані у вигляді пояснювальної записки в електронному варіанті та додатку (сформованого в архів), робота якого перевіряється викладачем. Пояснювальна записка в друкованій формі та носій з результатами виконаної роботи повинні бути здані викладачеві під час консультації до іспиту при очному спілкуванні.

Обсяг пояснювальної записки до курсової роботи повинен становити 25-35 аркушів. Пояснювальна записка повинна містити текстових опис роботи та графічні ілюстрації.

Курсова робота повинна містити:

Титульний аркуш

1.Завдання

2.Реферат

3.Вступ.

4.Аналіз предметної області.

5.Розроблення бази даних.

6.Реалізація бази даних.

7. Висновки.

8.Список літератури.

9.Додатки

Титульний аркуш повинен бути оформлений згідно зразку, приведеного у додаткових файлах до методичних вказівок.

Аркуш завдання є вхідним документом для розроблення проекту і повинен бути заповнений по затвердженій формі (при роздруківці вааріанту пояснювальної записик завдання слід друкувати з вдох сторін аркушу). В завданні приводиться графік виконання етапів роботи. Зразок оформлення завдання та реферату приведені також у додаткових файлах.

Короткий опис розділів курсової роботи

У *вступі* студент повинен відобразити основні аспекти сучасного стану проблем проектування бази даних, основні етапи проектування бази даних. Визначити актуальність роботи, мету і задачі дослідження, описати порядок та засоби реалізації задач проектування. Обсяг вступу повинен складати 2-3 сторінки.

Зміст розділів «*Аналіз предметної області*» , «*Розроблення бази даних*»

та «Реалізація бази даних» буде описаний у методичних вказівках до відповідних розділів індивідуального завдання.

Розділ «*Висновки*» повинен містити стислі висновки за підсумками проведеної роботи. Висновки повинні бути сформульовані таким чином, щоб надати вичерпну інформацію про основний зміст роботи за методи, які були використані для реалізації поставлених завдань.

Розділ «*Список літератури*» є обов'язковим і повинен становити від 10 до 15 джерел. По тексту повинні бути виконані посилання на літературне джерело за номером у переліку (в квадратних дужках, наприклад [11]) з використання засобу створення посилань.

ВИМОГИ ДО ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальну записку виконують на сторінці формату А4 (210х297) з основним написом за формою 2 на аркуші з розділом «Зміст» та за формою 2а на подальших аркушах.

Текст пояснювальної записки виконують за допомогою одного з текстових редакторів з однієї сторони аркуша формату А4 з наступними полями: зверху – 15 мм, знизу – 30 мм, справа – 10 мм, зліва – 25 мм.

Рекомендовано застосовувати шрифт Times New Roman, розмір 14 пунктів, міжрядний інтервал 1,5; абзацний відступ 1,27 см, вирівнювання тексту - за шириною.

Обовязковим є використання стилів заголовків та абзаців підписів рисунків та таблиць. Рекомендованим є створення шаблону пояснювальної записки, який доцільним буде використовувати в рамках оформлення результатів кваліфікаційних робіт бакалавра та спеціаліста, а також комплексних курсових проектів з спеціалізованих дисциплін. Формати стилів заголовків та абзаців приведені в табл. 2.

Новий розділ слід начинати з нової сторінки з використанням вставки розриву сторінки після попереднього розділу.

Нумерація наскрізна: титул, завдання та реферат не нумеруються; нумерація починається з аркушу змісту з номеру «4».

Таблиця 2 – Параметри стилів

ю	Параметри шрифту	Параметри абзацу
	Times New Roman, 16 пт, напівжирний, Регістр - прописні	Вирівнювання: по центру Відступи: без абзацу Інтервал міжрядковий:1,5 Інтервали: перед – 12 пт, після – 12 пт
	Times New Roman, 16 пт, напівжирний, Регістр – як в реченні	Вирівнювання: по центру Відступи: без абзацу Інтервал міжрядковий 1,5 Інтервали: перед – 6 пт, після – 6 пт
	Times New Roman, 14 пт, напівжирний, Регістр – як в реченні	Вирівнювання: по ширині Відступи: 1,27 см Інтервал міжрядковий 1,5 Інтервали: перед – 3 пт, після – 3 пт
	Times New Roman, 14 пт, звичайний, Регістр – як в реченні	Вирівнювання: по ширині Відступи: 1,27 см Інтервал міжрядковий 1,5 Інтервали: перед – 0 пт, після – 0 пт
ий	Times New Roman, 14 пт, звичайний, Регістр – як в реченні	Вирівнювання: по ширині Відступи: 0,63 см Інтервал міжрядковий 1,5 Інтервали: перед – 0 пт, після – 0 пт Додаткові параметри Поз.табуляції: 0.63 см Вирівняти по поз.таб. Нумерований: Рівень: 1 Почати з: 1 Вирівнювання: зліва Вирівняти по: 6,75 см

й ПЗ	Times New Roman, 14 пт, звичайний, Регістр – як в реченні	Вирівнювання: по ширині Відступи: 0,63 см Інтервал міжрядковий 1,5 Інтервали: перед – 0 пт, після – 0 пт Додаткові параметри Поз.табуляції: 1,5 см Вирівняти по лівому Маркований: Рівень: 1 Вирівняти по: 1,63 см Відступ: 2,27 см
	Times New Roman, 14 пт, звичайний, Регістр – як в реченні	Вирівнювання: по ширині Відступи: без абзацу Інтервал міжрядковий 1,5 Інтервали: перед – 3 пт, після – 3 пт
	Times New Roman, 14 пт, звичайний, Регістр – як в реченні	Вирівнювання: по центру Відступи: без абзацу Інтервал міжрядковий 1,5 Інтервали: перед – 3 пт, після – 3 пт