Архітектура СУБД

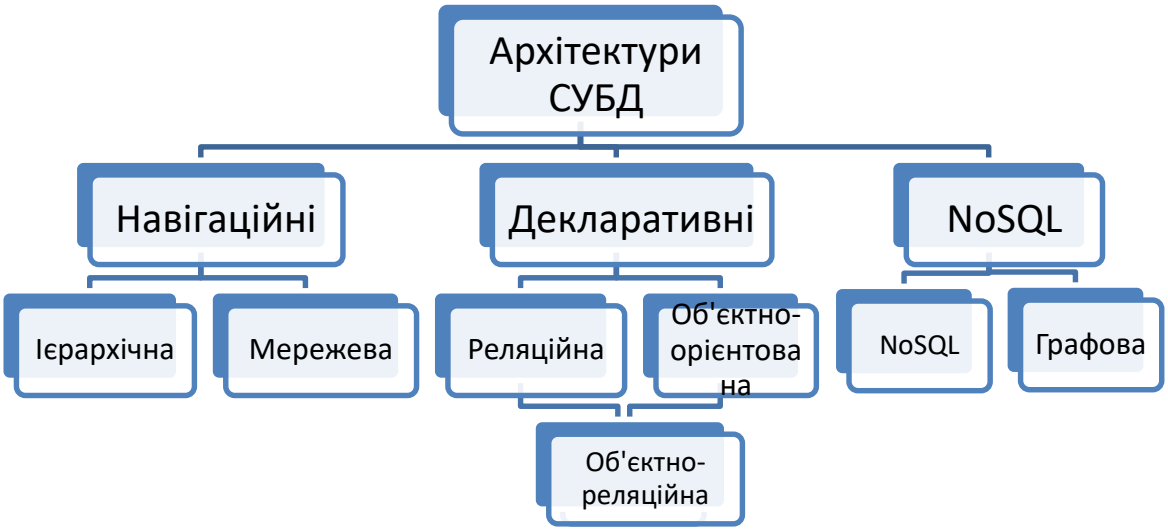


Рисунок 3.1 – Спроба класифікації архітектур СУБД

1. СУБД навігаційного типу – ієрархічні та мережеві.

1.1. Навігаційні бази даних

Навігаційні бази даних - такі бази даних, в яких записи або об'єкти знаходяться в основному за послідовними посиланнями з інших об'єктів. Навігаційні інтерфейси, як правило, процедурні, хоча деякі сучасні системи, такі як XPath (мова доступу до ієрархій типу XML) можна вважати одночасно навігаційними та декларативними.

Навігаційний доступ традиційно асоціюється з мережевою та ієрархічною архітектурами бази даних. Навігаційні методи використовують **"покажчики"** і **"шляхи"** для переміщення між записами даних («вузлами»). На відміну від реляційної моделі, в якій необхідно вказувати системі, які дані потрібні, в навігаційних СУБД потрібно вказувати як дістатись до даних.

Приклад: щоб отримати напрямок до будинку, в навігаційному підході це буде нагадувати щось на зразок "проїдьте на шосе 25,8 км, поверніть на вулиці Соборній наліво біля червоного будинку та зупиніться біля 3-го будинку по дорозі", в той час як при декларативному підході це буде нагадувати "Візит в зелений будинок (та) в наступних координатах ....".

Ієрархічні моделі також розглядаються як навігаційні, тому що в них дані посилаються вгору на батьків, вниз на листові елементи, є "шляхи", такі, як шляхи файлів/папок в ієрархічних файлових системах. Загалом, навігаційні системи будуть використовувати комбінації шляхів і команд, такі як "наступний", "попередній", "перший", "останній", "вгору", "вниз", "власник", і т.д. "Шляхи" часто утворюються шляхом об'єднання імен вузлів або адреси вузла. Приклад:

Node6.Node4.Node5.Node1

Node6/Node4/Node5/Node1

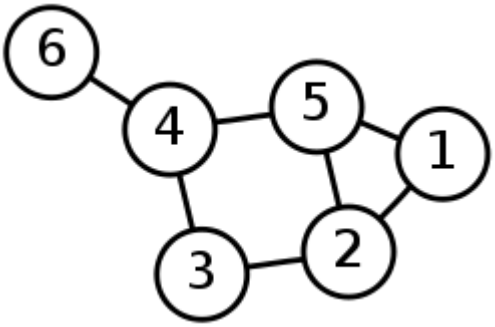


Рисунок 3.2 – Приклад вузлів бази даних в графі на 6 вершин і 7 ребер. (Числа використовуються тільки для ілюстрації. На практиці частіше використовуються більш осмислені імена. Інші потенційні атрибути не показані.)

Якщо не існує зв'язку між деякими вузлами, то зазвичай спрацьовує стандартна помилка з повідомленням, таких як "неіснуючий шлях". Шлях "Node6.Node2.Node1" буде неіснуючим в наведеному прикладі, тому що немає прямого зв'язку між вузлами 6 і 2.
Критики розглядають навігаційні методи як "неструктуровані столові спагеті", і порівнюють їх з "goto" зі структурного програмування. З цієї точки зору реляційні методи забезпечують більшу дисципліну і послідовність в організації даних і їх використанні, тому що їх коріння в теорії множин і численні предикатів.
Сучасні приклади навігаційного структурування:

– Document Object Model (DOM), стандарт побудови документів для World Wide Web, що використовується, наприклад, у веб-браузерах і тісно пов'язаний з JavaScript. "Двигун" DOM являє собою легку базу даних навігаційного типу.

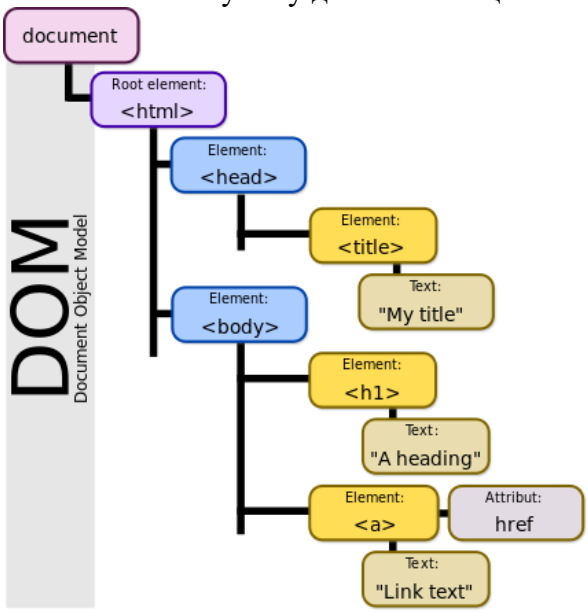


Рисунок 3.3 – Приклад ієрархії DOM в документі HTML

– World Wide Web в цілому та Wikipedia потенційно можуть вважатися формами навігаційних баз даних, хоча вони і зроблені на зрозумілих для людини текстах, а не на інших даних (у великих масштабах, Web являє собою мережеву модель, а в меншому масштабі, такому, як домен та розмітка URL, вона використовує ієрархії, зокрема DOM).
– останнім часом широкого розвитку набули ряд нових навігаційних баз даних - графові бази даних. Ця категорія баз даних часто розглядається як один з чотирьох типів баз даних NoSQL.

У порівнянні з реляційними базами даних, навігаційні бази даних більш точно відповідають структурі об’єктно-орієнтованих застосунків. Це є **головною перевагою**, яка викликає високий інтерес та використання таких СУБД в реальних застосунках. Вони зазвичай не вимагають досить витратних операцій з'єднання (JOIN). Реляційні бази даних, як правило, швидші при виконанні тієї ж операції на великій кількості елементів даних.
Приклади навігаційних СУБД: AllegroGraph, Bigdata, CloudGraph, Cytoscape, DEX, Filament, GiraffeDB, GraphBase, Horton, HyperGraphDB, InfiniteGraph, Neo4j, InfoGrid, OpenLink Virtuoso, OrientDB, OQGRAPH , R2DF, sones GraphDB, VertexDB

Ієрархічні бази даних

Ієрархічна структура представляє сукупність елементів, пов'язаних між собою за певними правилами. Об'єкти, пов'язані ієрархічними відносинами, утворюють орієнтований граф. До основних понять ієрархічної структури відносяться: рівень, елемент (вузол, або сегмент), зв'язок. **Вузол** - це сукупність атрибутів даних, що описують деякий об'єкт. На схемі ієрархічного дерева вузли представляються вершинами графа. Кожен вузол на нижчому рівні пов'язаний лише з одним вузлом, що знаходиться на вищому рівні. Ієрархічне дерево має тільки одну вершину, не підпорядковану ніякий інший вершині і знаходиться на самому верхньому (першому) рівні. Залежні вузли знаходяться на другому, третьому і т.д. рівнях. Кількість дерев у базі даних визначається числом кореневих записів. До кожного некореневого запису бази даних існує тільки один (ієрархічний) шлях від кореневого запису.

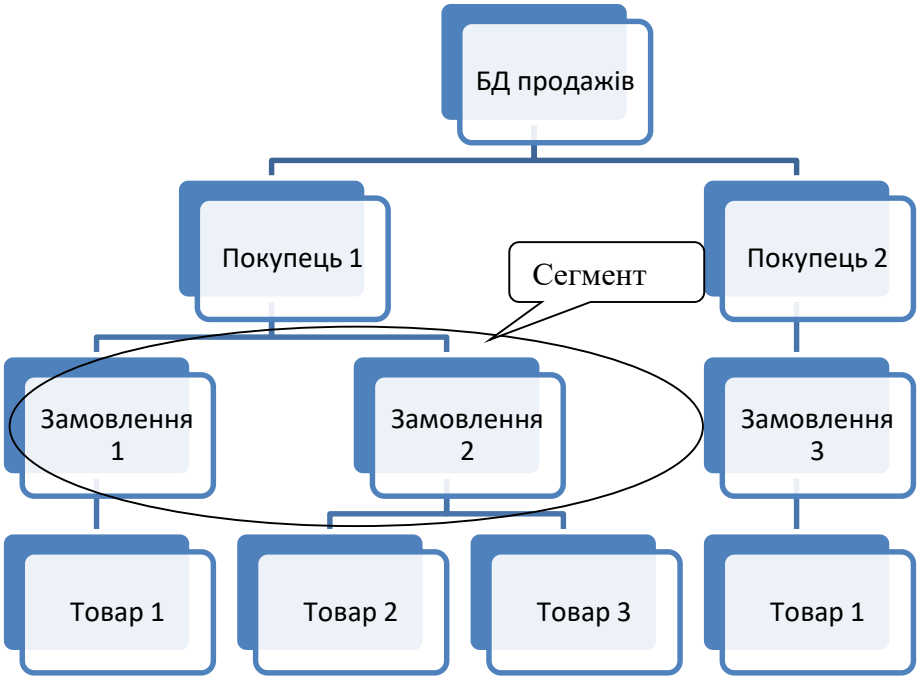


Рисунок 3.4 – Приклад ієрархічної структури СУБД

Ієрархічною базою даних є Реєстр Windows, в якому локально зберігається вся інформація, необхідна для нормального функціонування комп'ютерної системи (дані про конфігурацію комп'ютера і встановлені драйвери, відомості про встановлені програми, налаштування графічного інтерфейсу та ін.). Одне налаштування визначається один раз і, таким чином, лише в одній ієрархії, хоча групи SOFTWARE, SYSTEM і т.п. повторюються в різних гілках.

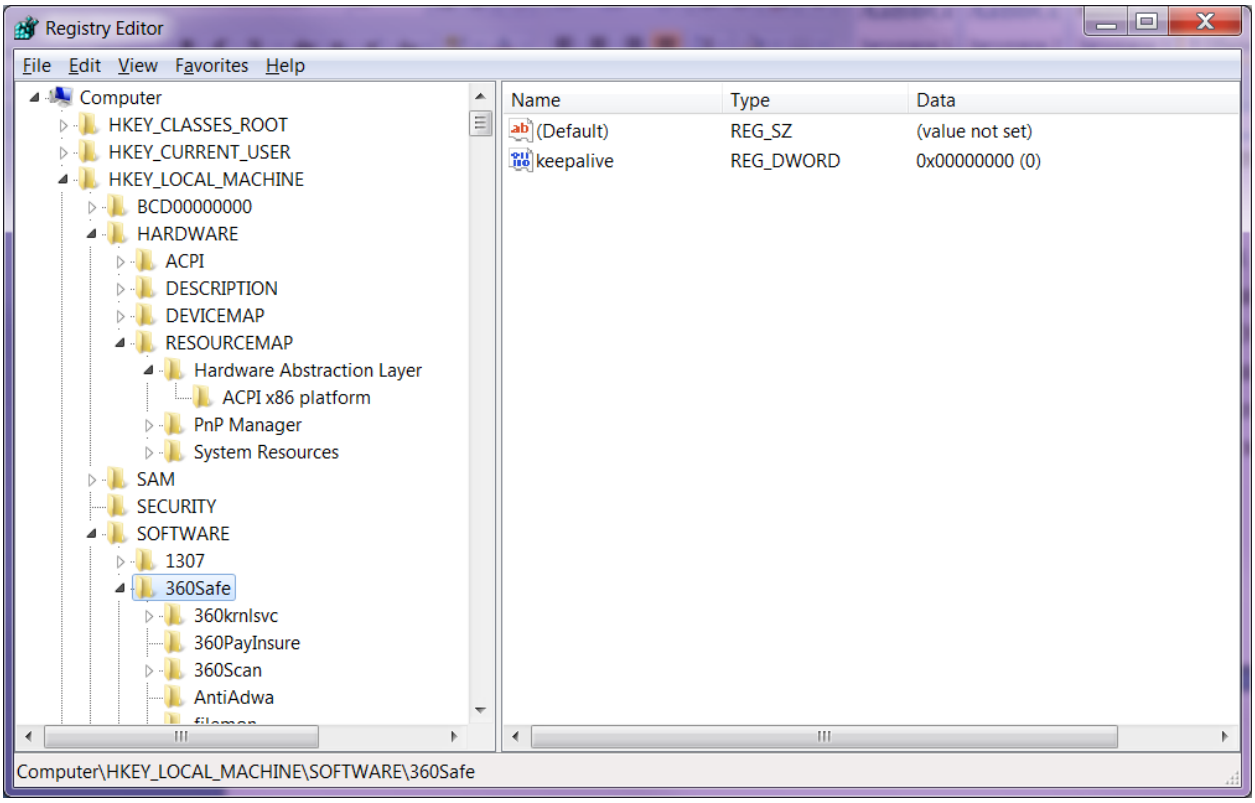


Рисунок 3.5 – Приклад реєстру Windows як ієрархічної СУБД

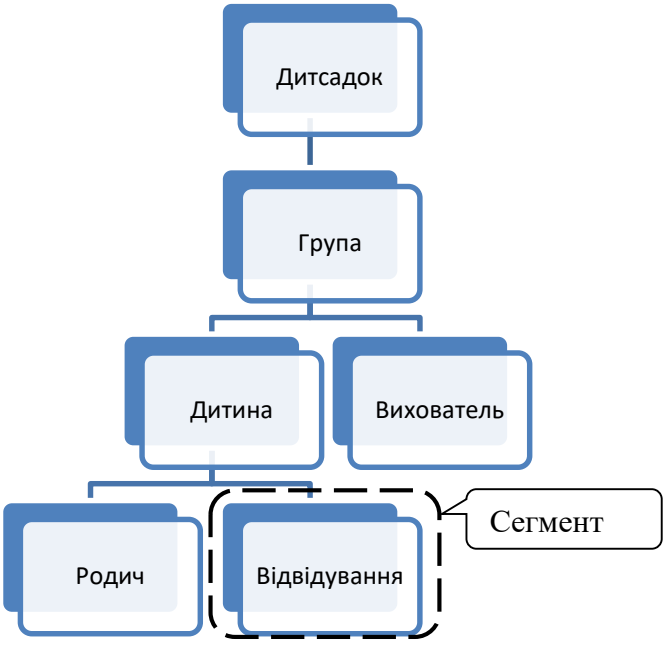


Рисунок 3.6 – Ієрархічна структура БД дитячого садку

Як бачимо, ні в базі продажів, ні в базі дитячого садку ієрархія не вирішує проблем, пов'язаних зі зв'язком «багато до багатьох». В останньому випадку множина сутностей Родич має такий зв'язок з мн.сутн. Дитина, і в разі, якщо у двох дітей один родич, його доводиться дублювати.

До основних недоліків ієрархічних моделей слід віднести: неефективність реалізації відносин типу N:N, повільний доступ до сегментів даних нижніх рівнів ієрархії, чітка

орієнтація на певні типи запитів та ін. Тому раніше створені ієрархічні СУБД піддаються істотним модифікаціям у напрямку, в першу чергу, мережевих архітектур.

Управляюча частина ієрархічної моделі

У рамках ієрархічної моделі, як і в інших, також виділяють мовні засоби опису даних (МОД (мова опису даних) – аналог Data Definition Language (DDL) для SQL) і засоби маніпулювання даними (ММД (мова маніпулювання даними) – аналог Data Manipulation Language (DML) для SQL). Кожна фізична база описується набором операторів її створення, що обумовлюють як її логічну структуру (множини сутностей та зв'язки між ними), так і фізичну структуру зберігання даних (включаючи типи атрибутів та назви).

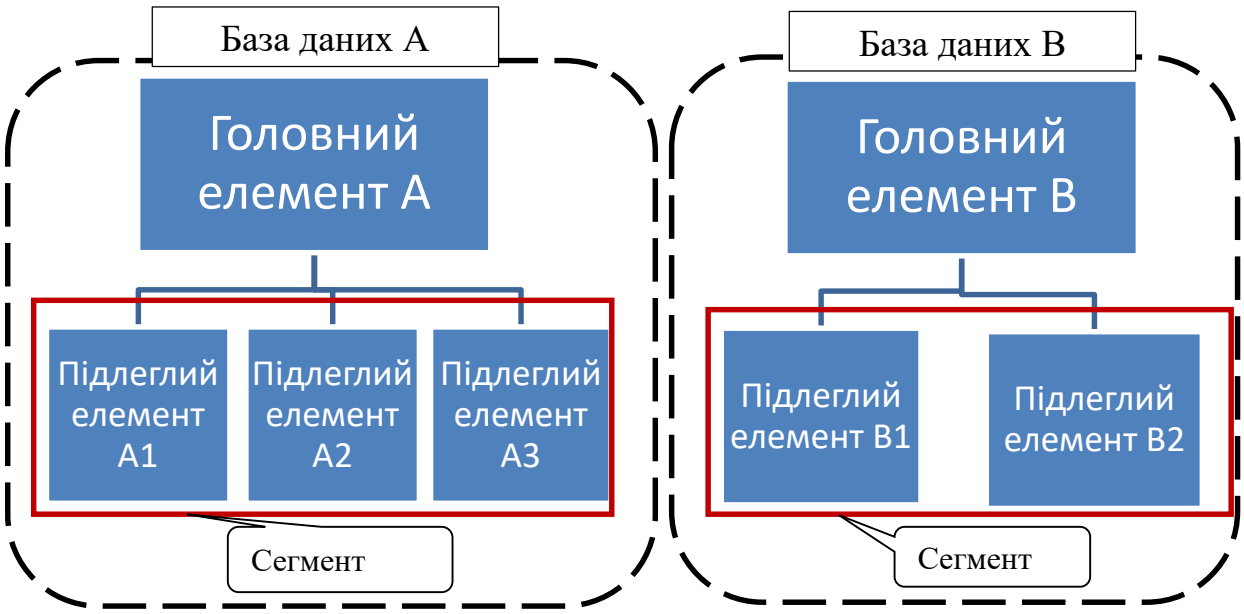


Рисунок 3.7 – Сегменти ієрархічної СУБД

Крім визначення імені БД і способу доступу описи мають містити визначення типів сегментів, що складають БД, у відповідності з ієрархією, починаючи з кореневого сегменту. Кожна фізична БД містить тільки один кореневий сегмент (і одне дерево ієрархії), але в системі може бути кілька фізичних БД.
Серед операторів маніпулювання даними можна виділити оператори пошуку даних, оператори пошуку даних з можливістю модифікації, оператори модифікації даних. Набір операцій маніпулювання даними невеликий, але цілком достатній.

Приклади типових операторів пошуку даних

- знайти вказане дерево БД;
- перейти від одного дерева до іншого;
- знайти екземпляр сегмента, що задовольняє умові пошуку;
- перейти від одного сегмента до іншого всередині дерева;
- перейти від одного сегмента до іншого в порядку обходу ієрархії.

Приклади типових операторів пошуку даних з можливістю модифікації:

- знайти і утримати для подальшої модифікації єдиний екземпляр сегмента, що задовольняє умові пошуку;
- знайти і утримати для подальшої модифікації наступний екземпляр сегмента з тими ж умовами пошуку.

Приклади типових операторів модифікації ієрархічно організованих даних, *які виконуються після виконання одного з операторів другої групи* (пошуку даних з можливістю модифікації):

- вставити новий екземпляр сегмента у вказану позицію;
- оновити поточний екземпляр сегмента;
- видалити поточний екземпляр сегмента.

В ієрархічній моделі автоматично підтримується цілісність посилань між предками і нащадками. Основне правило: ніякої нащадок не може існувати без свого батька (безпосереднього предка).

Відомі ієрархічні СУБД

- Типовим представником (найбільш відомим і поширеним) є Information Management System (IMS) фірми IBM з 1968 г.
- Time-Shared Date Management System (TDMS) компанії System Development Corporation;
- Mark IV Multi-Access Retrieval System компании Control Data Corporation;
- System 2000 розробки SAS Institute;
- Сервери каталогів, такі, як LDAP та Active Directory (допускають чітке подання у вигляді дерева);
- За принципом ієрархічної БД побудовані ієрархічні файлові системи та Реєстр Windows.
- InterSystems Caché

Мережеві бази даних

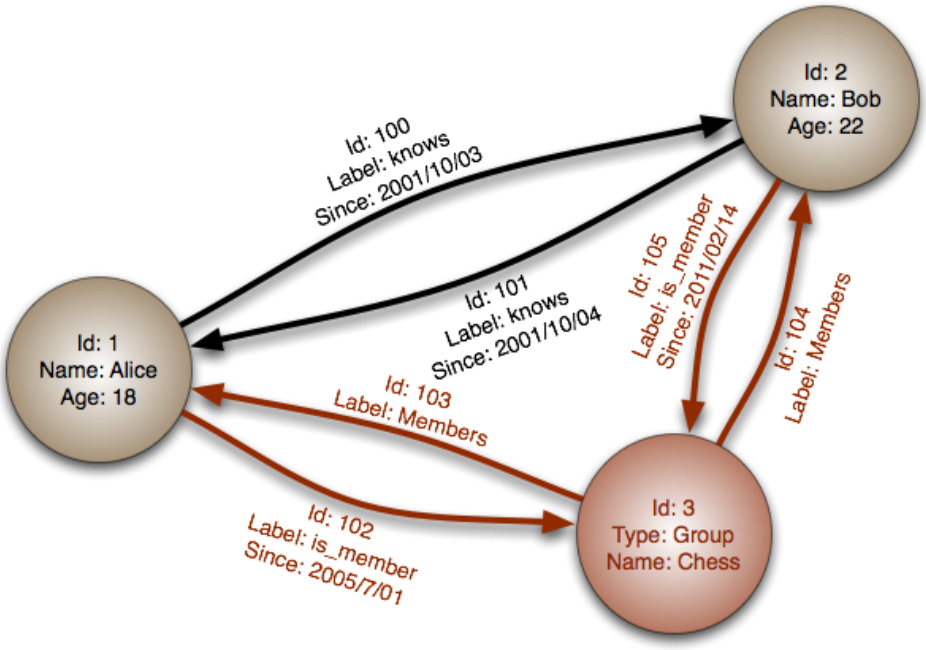


Рисунок 3.8 – Приклад вузлів мережевої бази даних. Label визначає тип зв'язку, а Id зв'язку – екземпляр.

Мережева модель даних - логічна модель даних, яка є розширенням ієрархічного підходу та використовує строгу математичну теорію, що описує структурний аспект, аспект цілісності і аспект обробки даних в мережевих базах даних.

Різниця між ієрархічною моделлю даних і мережевою полягає в тому, що в ієрархічних структурах запис-нащадок повинен мати в точності одного предка, а в мережевій структурі даних у нащадка може бути будь-яке число предків.

Мережева БД складається з набору екземплярів певного типу записів і набору екземплярів певного типу зв'язків між цими записами.
Тип зв'язку визначається для двох типів запису: предка і нащадка. Примірник типу зв'язку складається з одного примірника типу запису предка і впорядкованого набору екземплярів типу запису нащадка. Для даного типу зв'язку L з типом запису предка P і типом запису нащадка C повинні виконуватися наступні дві умови:

– кожен екземпляр типу запису P є предком тільки в одному екземплярі типу зв'язку L;
– кожен екземпляр типу запису C є нащадком не більше ніж в одному екземплярі типу зв'язку L.

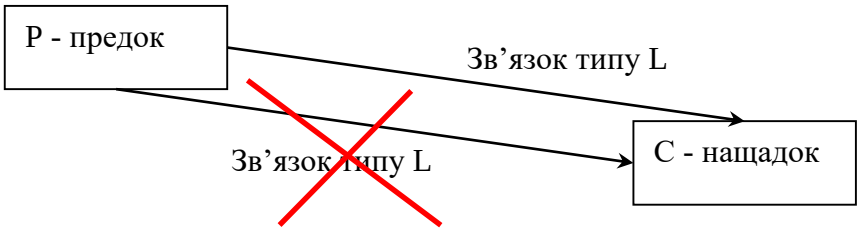


Рисунок 3.9 – Обмеження на зв’язки мережевої бази даних

Тобто, постулюється відсутність паралельних зв’язків одного типу між тими ж предком і нащадком.
Згідно рис.3.7, в наведеній моделі даних задіяні такі типи зв’язків:

- Knows;
- Members;
- Is_member.

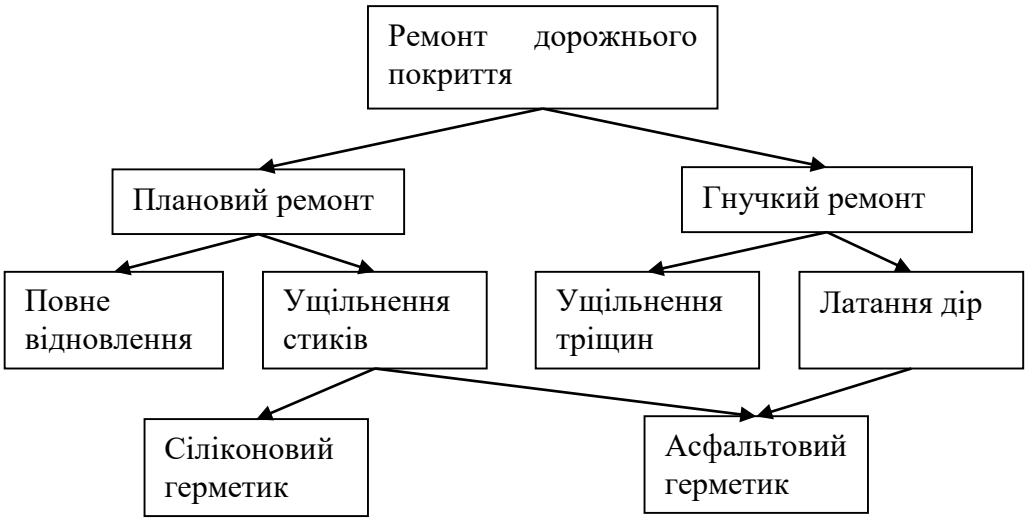


Рисунок 3.10 – Приклад мережевої моделі даних (https://en.wikipedia.org/wiki/Network_model)

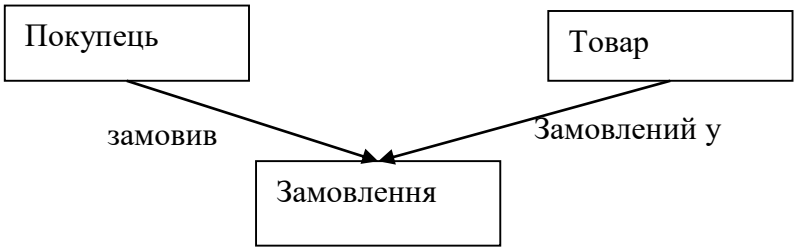


Рисунок 3.11 – Приклад мережевої моделі даних

Незважаючи на те, що ця модель вирішує деякі проблеми, пов'язані з ієрархічною моделлю, виконання простих запитів залишається досить складним процесом.

Також, оскільки логіка процедури вибірки даних залежить від фізичної організації цих даних, то застосування не є повністю незалежним від моделі даних. Тобто якщо необхідно змінити структуру даних, то потрібно змінити і застосування.

Керуюча частина мережевої моделі

Приклад набору операцій маніпулювання даними:

- знайти певний запис в наборі однотипних записів;
- перейти від предка до першого нащадка за деяким зв'язком;
- перейти до наступного нащадка у деякому зв'язку;
- перейти від нащадка до предка за деяким зв'язком;
- створити новий запис;
- знищити запис;
- модифікувати запис;
- включити в зв'язок;
- виключити з зв'язку;
- переставити в інший зв'язок і т. д.

Приклади мережевих СКБД

- IDS (Integrated Data Store) компанії General Electric — найперша мережева СУБД, розроблена Чарльзом Бахманом у 1960 р.
- IDS/2 або IDS/II) компанії Honeywell, що придбала IDS у General Electric, пізніше — компанії Bull
- СООБЗ Cerebrum (Мережева об'єктно-орієнтована система управління базою знань)
- Caché (позіціонується як мультімодельна; в 2014 році сайт DB-Engines поставив Caché на перше місце в рейтингу популярності об’єктно-орієнтованих СУБД).

Посилання:
http://en.wikipedia.org/wiki/Navigational_database
http://en.wikipedia.org/wiki/NoSQL
http://en.wikipedia.org/wiki/Graph_database

СУБД реляційного типу

Реляційна база даних

Реляційна база даних — база даних, заснована на реляційній моделі даних. Слово «реляційний» походить від англ. *relation*. Для роботи з реляційними БД застосовують реляційні СКБД. Реляційна база даних — це БД, яку користувач бачить як набір нормалізованих відношень різного ступеню. Відношення може бути представлене як таблиця, яка фіксує або певну множину елементів, або наявність зв'язку між елементами різних множин.

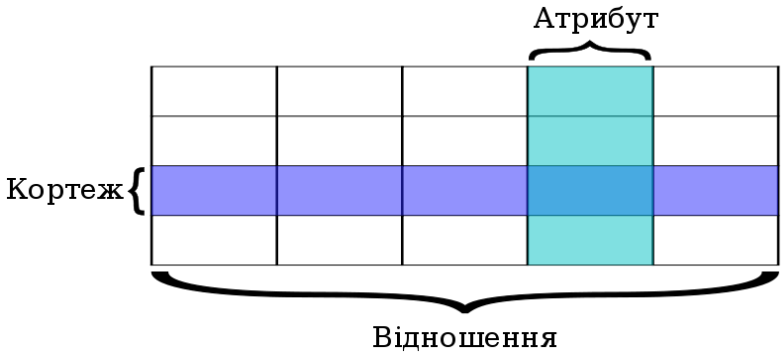


Рисунок 3.12 – Візуальне зображення відношення в реляційній СУБД

Реляці́йна моде́ль да́них — розроблена **Едгаром Коддом** в 1970 логічна модель даних, що описує:

- структури даних у вигляді наборів відношень, що, можливо, змінюються в часі;
- теоретико-множинні операції над даними: об'єднання, перетин, різниця і декартів добуток;
- спеціальні реляційні операції: селекція, проекція, з'єднання і розподіл;
- а також спеціальні правила, що забезпечують цілісність даних.

Обробка даних в реляційній моделі ґрунтується на принципах реляційної алгебри.

Реляційна система керування базами даних (РСКБД; інакше Система керування реляційними базами даних, СКРБД) — СКБД, що керує реляційними базами даних.

Ця модель характеризується простотою структури даних, зручним для користувача табличним представленням і можливістю використання формального апарату алгебри відношень і реляційного обчислення для обробки даних.

Реляційна модель орієнтована на організацію у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має такі властивості:

- кожний елемент (комірка) таблиці - один елемент даних;
- всі комірки в стовпці таблиці однорідні, тобто мають однаковий тип;
- кожний стовпець має унікальне для таблиці ім'я;
- однакові рядки в таблиці відсутні;
- порядок наступності рядків і стовпців може бути довільним.

Базовими поняттями реляційних СКБД є:

- атрибут
- відношення
- кортеж

Відношення

Відношення — фундаментальне поняття реляційної моделі даних. З цієї причини модель і називається *реляційною*.

Відношення має просту графічну інтерпретацію, воно може буде представлене у вигляді таблиці, стовпці (поля, атрибути) якої відповідають доменам (множинам припустимих значень), а рядки (записи, кортежі) - наборам з *n* значень, що взяті з доменів атрибутів. Кількість рядків m називають *кардинальністю* відношення, або *потужністю* відношення.

Під *атрибутом* розуміємо властивість (входження домена у відношення). Рядки відношення називаються *кортежами*.

Рисунок 3.13 – Візуальне зображення відношення в реляційній СУБД

Об’єктно-орієнтовані БД

Визначення об'єктно-орієнтованих БД

Об'єктно-орієнтована база даних (ООБД) - база даних, в якій дані оформлені у вигляді моделей об'єктів, що включають прикладні програми, які управляються зовнішніми подіями. ООБД відрізняються від реляційних БД, які є таблично-орієнтованими. *Об'єктно-орієнтовані системи управління базами даних* (ООСУБД) дозволяють працювати з об'єктами баз даних так само, як з об'єктами в пам’яті програми в об'єктно-орієнтованих мовах програмування і мають таку саму модель даних.
Деякі об'єктно-орієнтовані бази даних розроблені для щільної взаємодії з такими об'єктно-орієнтованими мовами програмування як Python, Java, C#, Visual Basic.NET, C++, OBJECTIVE-C і Smalltalk;
Приклади об'єктно-орієнтованих СУБД: Cache, Wakanda (2012), Gemstone (1982-2010, 2013), Realm (2014).

Об'єктно-орієнтована модель базується на концепціях:

- об'єкту і ідентифікатора об'єкту;
- атрибутів і методів;

– класів;
– ієрархії і успадкування класів.

Будь-яка сутність реального світу моделюється у вигляді об'єкту. Будь-який об'єкт при своєму створенні отримує унікальний ідентифікатор, що генерується системою, який пов'язаний з об'єктом у весь час його існування і не змінюється при зміні стану об'єкту.
Кожен об'єкт має стан і поведінку.

– Стан об'єкту - набір значень його атрибутів.
– Поведінка об'єкту - набір методів (програмний код), що оперують над станом об'єкту.
– Значення атрибуту об'єкту - це теж деякий об'єкт або множина об'єктів.
– Стан і поведінка об'єкту інкапсульовані в об'єкті.

Взаємодія між об'єктами проводиться на основі передачі повідомлень і виконанні відповідних методів. Методи визначаються не лише у визначенні класу в прикладній програмі, вони містяться у визначенні класу в БД.

Множина об'єктів з одним і тим же набором атрибутів і методів утворює **клас об'єктів**. Допускається наявність примітивних зумовлених класів, об'екти-екземляри яких не мають атрибутів: цілі числа, рядки і так далі. Клас – множина допустимих значень атрибуту (напр., ціле число), називається доменом цього атрибуту.
Допускається породження нового класу на основі вже існуючого класу - **успадкування**. В цьому випадку новий клас, званий **підкласом** існуючого класу (суперкласу) успадковує всі атрибути і методи суперкласу. У підкласі, крім того, можуть бути визначені додаткові атрибути і методи. Розрізняються випадки простого і множинного успадкування. У першому випадку підклас може визначатися тільки на основі одного суперкласу, в другому випадку суперкласів може бути декілька. Якщо в мові або системі підтримується одиничне успадкування класів, набір класів утворює деревовидну ієрархію. При підтримці множинного успадкування класи зв'язані в орієнтований граф з коренем, званий **решіткою класів**. Об'єкт підкласу вважається таким, що належить кожному суперкласу цього класу.

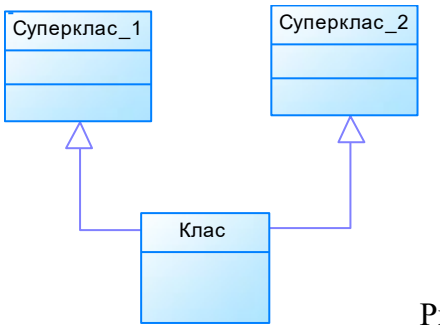


Рисунок 3.14 – Елементи решітки класів

Допускається **перевизначення атрибутів і методів суперкласу в підкласі** (перевантаження методів). Ця можливість збільшує гнучкість, але породжує додаткову проблему: при компіляції об'єктно-орієнтованої програми можуть бути невідомі структура і програмний код методів об'єкту, хоча його клас (у загальному випадку - суперклас) відомий. Для вирішення цієї проблеми застосовується так званий метод пізнього зв‘язування, що означає, по суті справи, інтерпретаційний режим виконання програми з розпізнаванням деталей реалізації об'єкту під час посилання повідомлення до нього.

Структура об'єктної моделі описуються за допомогою трьох ключових понять:

- **Інкапсуляція** - кожен об'єкт володіє деяким внутрішнім станом (зберігає усередині себе запис даних), а також набором методів - процедур, за

допомогою яких (і тільки таким чином) можна дістати доступ до даних, що визначають внутрішній стан об'єкту, або змінити їх.

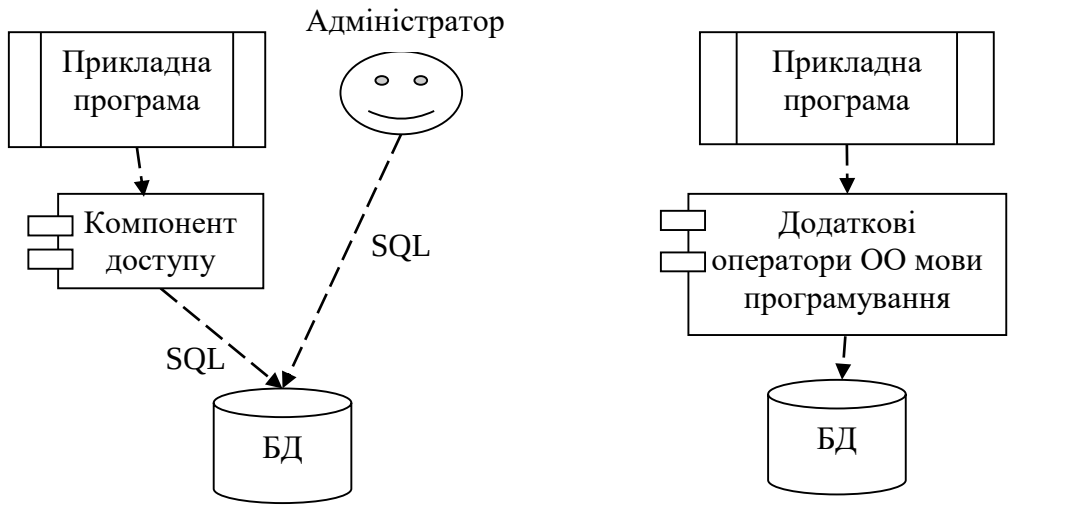
- **Успадкування** - можливість створювати з класів нові класи, які успадковують структуру і методи своїх предків, додаючи до них риси, що відображають їх власну індивідуальність. Успадкування може бути просте (один предок) і множинне (декілька предків).

- **Поліморфізм** - різні об'єкти можуть мати однаковий інтерфейс методів, але по-різному реагувати на однакові зовнішні події залежно від того, як реалізовані їх методи**.**

Цілісність даних:
Для підтримки цілісності об'єктно-орієнтований підхід пропонує використовувати наступні засоби:

- автоматична підтримка відношень успадкування;
- можливість оголосити деякі поля даних і методи об'єкту як "приховані", не видимі для інших об'єктів; такі поля і методи використовуються тільки методами самого об'єкту;
- створення процедур контролю цілісності усередині об'єкту.

Засоби маніпулювання даними:
В об'єктно-орієнтованому програмуванні відсутні загальні засоби маніпулювання даними, такі як реляційна алгебра або реляційне числення. Робота з даними ведеться за допомогою однієї з об'єктно-орієнтованих мов програмування загального призначення, звичайно це Smalltalk, C# або Java.



Звернення до бази в реляційній СУБД

Звернення до бази в об’єктно-орієнтованій СУБД

Рисунок 3.15 – Звернення до БД в РСУБД і ООСУБД

Переваги моделі ООБД

1) Визначення призначених для користувача абстракцій

Об'єктно-орієнтовані бази даних надають можливість визначати нові абстракції і управляти реалізацією таких абстракцій. Ці нові абстракції можуть відповідати структурам даних, потрібним в складних завданнях, новим абстрактним типам даних. Зокрема,

можливо створення нового класу з атрибутами і методами; успадкування атрибутів і методів від суперкласів; створення екземплярів класу, кожен з яких володіє унікальним об'єктним ідентифікатором; витягання цих екземплярів поодинці або групами; а також завантаження і виконання методів.

2) Полегшене проєктування деяких зв'язків

У об'єктно-орієнтованих базах даних підтримується засіб інверсних зв'язків для виразу взаємних посилань між двома об'єктами (бінарний зв'язок). Така система забезпечує посилальну цілісність шляхом встановлення відповідного зворотного посилання відразу ж після створення прямого посилання. (Objectstore)

3) Відсутність потреби у визначенні первинних ключів

У моделі ООБД є поняття ідентифікаторів об'єктів, що автоматично генеруються системою і гарантовано унікальних для кожного об'єкту. В моделі ООБД усувається потреба у визначенні ключів. Ідентифікатор об'єкту не може бути модифікований застосунком. Ідентичність об’єкту існує незалежно від його атрибутів.

4) Менша потреба в з'єднаннях

В об'єктно-орієнтованих базах даних неявні з'єднання, що породжуються ієрархічною вкладеністю об'єктів (успадкування), відрізняються від явних об'єднань (асоціації). Останні нагадують реляційні з'єднання: два об'єкти порівнюються явним чином по значеннях атрибутів або об'єктних ідентифікаторів. Більш того, всі явні з'єднання (засновані на порівнянні значень або ідентифікаторів) не можуть бути виражені на реляційній мові запитів, оскільки в РБД будь-який атрибут може містити тільки атомарні значення і РБД не може обробити зворотнє посилання на багато об’єктів.

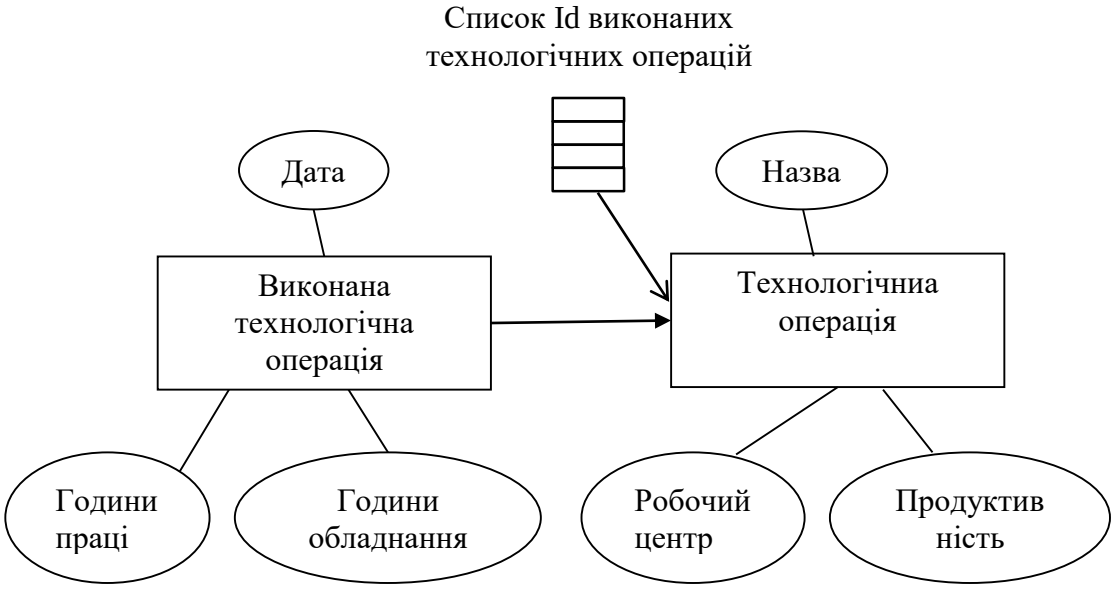


Рисунок 3.16 – Зв’язок в ER-діаграмі. В об’єктно-орієнтованій моделі на стороні «багато» (ТО) має бути записано багато ключів лівої множини сутностей, для кожної дати і ТО

5) Виграш в продуктивності за рахунок відповідності структури збереження даних потребам програміста, але **програш** по причині нерозвинутості оптимізації запитів.

6) Підтримка версій

У РБД не підтримуються робота з версіями (автоматичне збереження історії атрибутів і методів). Така підтримка є в деяких ООБД, хоча і з обмеженими можливостями.

Недоліки моделі ООБД

1) Відсутність інтероперабельності між РБД і ООБД , тобто неможливість формалізації і автоматизації конвертування РБД в ООБД або навпаки.

2) Недостатність засобів для оптимізації запитів
Оптимізацію запитів до ООБД ускладнює додаткова складність самої об'єктно-орієнтованої моделі даних.
3) Відсутність засобів забезпечення запитів
Мова запитів не сумісна з ANSI SQL. Серед цих засобів забезпечення запитів немає вкладених підзапитів, запитів до множин (union, intersection, difference), агрегатних функцій і GROUP BY, з'єднання декількох класів - можливостей, повністю підтримуваних в РБД.
Крім того, відсутній стандарт для об'єктних запитів, хоча робилися спроби розробити об'єктну мову SQL.
4) Відсутність підтримки збережених запитів – розрізів.
5) Проблеми з безпекою – відсутня підтримка авторизації, не автоматизована система блокування.

Недоліки пов'язані з відсутністю розвинених засобів маніпулювання даними. Це завдання вирішується двома способами - **розширення ОО-мов у бік управління даними (стандарт ODMG), або додавання об'єктних властивостей в реляційні СУБД (Sql-3,** а також так звані **об'єктно-реляційні СУБД).**

Об'єктно-реляційні СУБД

Інший спосіб об'єднання можливостей реляційного і об'єктно-орієнтованого підходу до управління даними запропонував відомий американський вчений Майкл Стоунбрейкер. Згідно його переконанням реляційну СУБД потрібно просто доповнити засобами доступу до складних даних. При цьому ядро СУБД не вимагає переробки, як у випадку з Sql3, і зберігає всі властиві реляційним системам можливості. Об'єктні розширення реалізуються у вигляді надбудов, які динамічно підключаються до ядра.

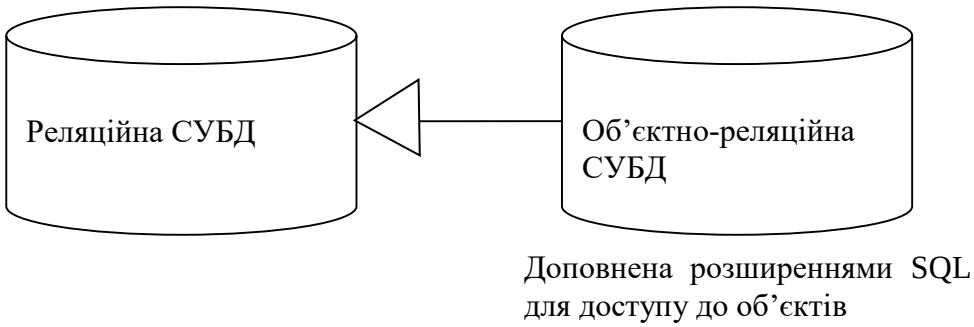


Рисунок 3.17 – Успадкування об’єктно-реляційної СУБД від реляційної СУБД

На основі цієї ідеї під керівництвом М.Стоунбрейкера в університеті Берклі (Каліфорнія, США) була розроблена СУБД Postgres.

Серед іншого, Postgres володіє властивостями, які дозволяють назвати її *темпоральною* СУБД. При будь-якому оновленні запису створюється його нова копія, а попередній варіант продовжує існувати вічно. Навіть після видалення запису всі накопичені варіанти зберігаються в базі даних. Можна витягувати з бази даних будь-який варіант запису, якщо вказати момент або інтервал часу, коли цей варіант був поточним. Досягнення цих властивостей дозволило також переглянути схеми журналізації і відкату транзакцій.
Багато ідей ранніх об'єктно-реляційних баз даних увійшли в стандарт SQL:1999 через структуровані типи. Фактично, будь-яка СУБД, яка дотримується об'єктно-орієнтованих аспектів SQL:1999, може бути названа об'єктно-реляційною. Наприклад, розробники IBM

DB2, БД Oracle та Microsoft SQL Server заявляють, що підтримують цю технологію і роблять це з різним ступенем успіху.