# Бази даних

## Бази даних

1. База даних це сукупність взаємозалежних *даних* на машинних *носіях,* організована певним способом; *інформаційна модель предметної області.*

Відповідно до моделей опису даних БД визначають як *ієрархічні, мережні, реляційні.* У зв'язку з наявністю повного математичного опису реляційної моделі, універсальністю моделі й простотою реалізації найбільший розвиток одержали *реляційні бази даних.*

Створення БД складається з наступних етапів:

- розробка концептуальної моделі даних;

- розробка логічної моделі БД;

- конкретне конструювання.

Інструмент для всіх можливих дій із БД — *система керування базою даних (СУБД).* Коли говорять про БД у загальному випадку, мають на увазі БД разом із системою керування, якщо спеціально не застережене противне.

Розрізняють БД: однокористувацькі й багатокористувацькі, *розподілені* й централізовані.

2. База даних це набір *файлів, що* відносяться до одного інформаційного *об'єкта.*

**Основи баз даних**

У наші дні люди часто говорять про бази даних. Комп'ютери становлять невід'ємну частину сучасного суспільства, тому нерідко можна почути фрази начебто "Я пошукаю твій запис у базі даних". І мова йде не про великі ящики, де зберігаються купи папок, а про комп'ютерні системи, призначених для прискореного пошуку інформації.

Комп'ютери так міцно ввійшли в наше життя, тому що їх можна запрограмувати на виконання стомлюючих, повторюваних операцій і рішення завдань, які нам самим було б не під силу вирішити без їхньої обчислювальної швидкості і ємності інформаційних носіїв. Розміщення інформації на папері і розробка схеми зберігання паперів у папках і картотеках - досить чітко відпрацьований процес, але багато хто зітхнув з полегшенням, коли завдання звелося до переміщення електронних документів у папки на жорсткому диску.

Однієї з функцій баз даних є впорядкування й індексація інформації. Як й у бібліотечній картотеці, не потрібно переглядати половину архіву, щоб знайти потрібний запис. Усе виконується набагато швидше.

Не всі бази даних створюються на основі тих самих принципів, але традиційно в них застосовується ідея організації даних у вигляді записів. Кожен запис має фіксований набір полів. Записи містяться в таблиці, а сукупність таблиць формує базу даних.

Для роботи з базою даних необхідна СКБД (система керування базами даних), тобто програма, що бере на себе всі турботи, пов'язані з доступом до даних. Вона містить команди, що дозволяють створювати таблиці, вставляти в них записи, шукати й навіть видаляти записи.

СКБД управляє однією або декількома базами даних. База даних являє собою сукупність інформації, організованої у вигляді множин. Кожна множина містить записи уніфікованого виду. Самі записи складаються з полів. Звичайно множини називають таблицями, а записи - рядками таблиць. Така логічна модель даних. На жорсткому диску вся база даних може перебувати в одному файлі.

Рядка таблиць можуть бути зв'язані один з одним одним із трьох способів. Найпростіше відношення - "один до одного". У цьому випадку рядок першої таблиці відповідає одному єдиному рядку другої таблиці. На діаграмах таке відношення виражається записом 1:1.

Відношення "один до багатьох" означає ситуацію, коли рядок однієї таблиці відповідає декільком рядкам іншої таблиці. Це найпоширеніший тип відносин. На діаграмах він виражається записом 1:N.

Нарешті, при відношенні "багато хто до багатьох" рядки першої таблиці можуть бути пов'язані з довільним числом рядків у другій таблиці. Таке відношення записується як N:M.

**Системи керування базами даних**

Програміст, що працює з базою даних, не піклується про те, як ці дані зберігаються, і додатки, взаємодіючі із СКБД, не знають про спосіб запису даних на диск. "Зовні" видимий лише логічний образ даних, і це дозволяє міняти код СКБД, не торкаючись коду самих додатків.

Подібна обробка даних здійснюється за допомогою мови четвертого покоління (4GL), що підтримує запити, які записуються й виконуються негайно. Дані швидко втрачають свою актуальність, тому швидкість доступу до них важлива. Крім того, програміст повинен мати можливість формулювати нові запити. Вони називаються нерегламентованими (ad hoc), оскільки не зберігаються в самій базі даних і служать вузькоспеціалізованим цілям.

Мова четвертого покоління дозволяє створювати схеми - точні визначення даних і відносин між ними. Схема зберігається як частина бази даних і може бути змінена без шкоди для даних.

Схема призначена для контролю цілісності даних. Якщо, приміром, оголошено, що поле містить ціле значення, то СКБД відмовиться записувати в нього числа із плаваючою комою або рядки. Відносини між записами теж чітко контролюються, і неузгоджені дані не допускаються. Операції можна групувати в транзакції, виконувані за принципом "все або нічого".

СКБД забезпечує безпеку даних. Користувачам надаються певні права доступу до інформації. Деяким користувачам дозволено лише переглядати дані, тоді як інші користувачі можуть міняти вміст таблиць.

СКБД підтримує паралельний доступ до бази даних. Додатки можуть звертатися до бази даних одночасно, що підвищує загальну продуктивність системи. Крім того, окремі операції можуть "распаралелюватися" для ще більшого поліпшення продуктивності.

Нарешті, СКБД допомагає відновлювати інформацію у випадку непередбаченого збою, непомітно для користувачів створюючи резервні копії даних. Всі зміни, внесені в базу даних, реєструються, тому багато операцій можна скасовувати й виконувати повторно.

**Системи керування файлами**

Найпростіша база даних організована у вигляді набору звичайних файлів. Ця модель нагадує картотечну організацію документів, при якій папки зберігаються в ящиках, а в кожній папці підшите деяке число сторінок.

Системи керування файлами не можна класифікувати як СКБД, тому що звичайно вони є частиною операційної систем і нічого не знають про внутрішній зміст файлів. Це знання закладене в прикладних програмах, що працюють із файлами. Як приклад можна привести таблицю користувачів UNIX, що зберігається у файлі /etc/passwd. Програми, що звертаються до цього файлу, знають, що в його першому полі перебуває ім'я користувача, що кінчається двокрапкою. Якщо додатку потрібно відредагувати цю інформацію, він повинний безпосередньо відкрити файл і подбати про правильне форматування полів.

Така модель бази даних дуже незручна, оскільки вона вимагає використовувати мову третього покоління (3GL). У результаті час програмування запитів збільшується, а програміст повинен мати більше високу кваліфікацію, тому що йому потрібно продумати не тільки логічну, але й фізичну структуру зберігання даних. Це приводить до того, що між додатком і файлом утворюється тісний зв'язок. Вся інформація про поля таблиць закодована в додатку. Інший додаток, що звертається до того ж файлу, змушений дублювати існуючий код.

Зі збільшенням числа додатків росте складність керування базою даних. Зміни схеми даних приводять до необхідності зміни кожного програмного компонента, для якого це актуально. Формування нових запитів займає стільки часу, що найчастіше губить усякий зміст.

Системи керування файлами не можуть перешкодити дублюванню інформації. Гірше того, не існує механізмів, що запобігають непогодженості даних. Уявіть собі файл, що містить відомості про всіх хто служить у компанії. У кожному рядку є поле, де записане ім'я начальника. Під керівництвом одного начальника працює багато службовців, тому його ім'я буде неминуче повторюватися. Якщо десь це ім'я буде записано неправильно, формально вийде, що в службовця інший начальник. При заміні начальника його ім'я прийде "виловлювати" по всій базі даних.

Безпека звичайних файлів контролюється операційною системою. Окремий файл може бути заблокований для перегляду або модифікації з боку того або іншого користувача, але це виконується тільки на рівні операційної системи. У конкретний момент часу лише один додаток може здійснювати запис у файл, що знижує загальну продуктивність

**Ієрархічні бази даних**

Ієрархічні бази даних підтримують деревоподібну організацію інформації. Зв'язки між записами виражаються у вигляді відносин предок/нащадок, а в кожного запису є рівно один батьківський запис. Це допомагає підтримувати цілісність посилань. Коли запис видаляється з дерева, всі його нащадки також повинні бути вилучені.

Ієрархічні бази даних мають централізовану структуру, тобто безпеку даних легко контролювати. На жаль, певні знання про фізичний порядок зберігання записів все-таки необхідні, тому що відносини предок/нащадок реалізуються у вигляді фізичних покажчиків з одного запису на інші.

Це означає, що пошук запису здійснюється методом прямого обходу дерева. Записи, розташовані в одній половині дерева, шукаються швидше, ніж в іншій. Звідси виникає необхідність правильно впорядковувати записи, щоб час їхнього пошуку був мінімальним. Це важко, тому що не всі відносини, що існують у реальному світі, можна виразити в ієрархічній базі даних. Відносини "один до багатьох" є природними, але практично неможливо описати відносини "багато хто до багатьох" або ситуації, коли запис має кілька предків. Доти поки в додатках будуть кодуватися відомості про фізичну структуру даних, будь-які зміни цієї структури будуть грозити перекомпіляцією.

**Мережні бази даних**

Мережна модель розширює ієрархічну модель, дозволяючи групувати зв'язки між записами в множини. З логічної точки зору зв'язок - це не сам запис. Зв'язки лише виражають відносини між записами. Як й в ієрархічній моделі, зв'язки ведуть від батьківського запису до дочірнього, але цього разу підтримується множинне спадкування.

Відповідно специфікації CODASYL, мережна модель підтримує DDL (Data Definition Language - мову визначення даних) і DML (Data Manipulation Language - мова обробки даних). Це спеціальні мови, призначені для визначення структури бази даних і складання запитів. Незважаючи на їхню наявність, програміст як і раніше повинен знати структуру бази даних.

У мережній моделі допускаються відносини "багато хто до багатьох", а записи не залежать друг від друга. При видаленні запису віддаляються й всі її зв'язки, але не самі зв'язані записи.

У мережній моделі потрібно, щоб зв'язки встановлювалися між існуючими записами щоб уникнути дублювання й перекручування цілісності. Дані можна ізолювати у відповідних таблицях і зв'язати із записами в інших таблицях.

Програмістові не потрібно піклуватися про те, як організується фізичне зберігання даних на диску. Це послабляє залежність додатків і даних. Але в мережній моделі потрібно, щоб програміст пам'ятав структуру даних при формуванні запитів.

Оптимальну структуру бази даних складно сформувати, а готову структуру важко міняти. Якщо вид таблиці перетерплює зміни, всі відносини з іншими таблицями повинні бути встановлені заново, щоб не порушилася цілісність даних. Складність подібного завдання приводить до того, що програмісти найчастіше скасовують деякі обмеження цілісності заради спрощення додатків

**Реляційні бази даних**

У порівнянні з розглянутими вище моделями реляційна модель жадає від СКБД набагато більш високого рівня складності. У ній робиться спроба позбавити програміста від виконання рутинних операцій по керуванню даними, настільки характерних для ієрархічної й мережної моделей.

У реляційній моделі база даних являє собою централізоване сховище таблиць, що забезпечує безпечний одночасний доступ до інформації з боку багатьох користувачів. У рядках таблиць частина полів містить дані, стосовні безпосередньо до запису, а частина - посилання на записі інших таблиць. Таким чином, зв'язки між записами є невід'ємною властивістю реляційної моделі.

Кожен запис таблиці має однакову структуру. Наприклад, у таблиці, що містить опис автомобілів, у всіх записів буде той самий набір полів: виробник, модель, рік випуску, пробіг і т.д. Такі таблиці легко зображувати в графічному виді.

У реляційній моделі досягається інформаційна й структурна незалежність. Записи не зв'язані між собою настільки, щоб зміна однієї з них торкнулося інших, а зміна структури бази даних не обов'язково приводить до перекомпіляції працюючих з нею додатків.

У реляційних СКБД застосовується мова SQL, що дозволяє формулювати довільні, нерегламентовані запити. Це мова четвертого покоління, тому будь-який користувач може швидко навчитися становити запити. До того ж, існує безліч додатків, що дозволяють будувати логічні схеми запитів у графічному виді. Все це відбувається за рахунок жорсткості вимог до продуктивності комп'ютерів. На щастя, сучасні обчислювальні потужності більш ніж адекватні.

Реляційні бази даних страждають від розходжень у реалізації мови SQL, хоча це й не проблема реляційної моделі. Кожна реляційна СКБД реалізує якусь підмножину стандарту SQL плюс набір унікальних команд, що ускладнює завдання програмістам, які намагаються перейти від однієї СКБД до іншої. Доводиться робити нелегкий вибір між максимальною переносимістю й максимальною продуктивністю. У першому випадку потрібно дотримуватися мінімального загального набору команд, підтримуваних у кожній СКБД. У другому випадку програміст просто зосереджується на роботі в даній конкретній СКБД, використовуючи переваги її унікальних команд і функцій.

**Об’єктно-орієнтовані бази даних**

Об’єктно-орієнтована база даних (ООБД) дозволяє програмістам, які працюють із мовами третього покоління, інтерпретувати всі свої інформаційні сутності як об'єкти, що зберігаються в оперативній пам'яті. Додатковий інтерфейсний рівень абстракції забезпечує перехоплення запитів, що звертаються до тих частин бази даних, які перебувають у постійному сховищі на диску. Зміни, внесені в об'єкти, оптимальним образом переносяться з пам'яті на диск.

Перевагою об’єктно-орієнтованих баз даних є спрощений код. Додатки одержують можливість інтерпретувати дані в контексті тієї мови програмування, на якому вони написані. Реляційна база даних повертає значення всіх полів у текстовому виді, а потім вони приводяться до локальних типів даних. В об’єктно-орієнтованих базах даних цей етап ліквідований. Методи маніпулювання даними завжди залишаються однаковими незалежно від того, перебувають дані на диску або в пам'яті.

Дані в об’єктно-орієнтованих базах даних здатні прийняти вид будь-якої структури, яку можна виразити використовуваною мовою програмування. Відносини між сутностями також можуть бути довільно складними. Об’єктно-орієнтована база даних управляє кеш-буфером об'єктів, переміщаючи об'єкти між буфером і дисковим сховищем у міру необхідності.

За допомогою об’єктно-орієнтованих баз даних вирішуються дві проблеми. По-перше, складні інформаційні структури виражаються в них краще, ніж у реляційних базах даних, а по-друге, усувається необхідність транслювати дані з того формату, що підтримується в СКБД. Наприклад, у реляційній СКБД розмірність цілих чисел може становити 11 цифр, а у використовуваній мові програмування - 16. Програмістові прийде враховувати цю ситуацію.

Об’єктно-орієнтовані СКБД виконують багато додаткових функцій. Це окупається сповна, якщо відносини між даними дуже складні. У такому випадку продуктивність об’єктно-орієнтованих баз даних виявляється вище, ніж у реляційних СКБД. Якщо ж дані менш складні, додаткові функції виявляються надлишковими. В об'єктній моделі даних підтримуються нерегламентовані запити, але мовою їхнього складання не обов'язково є SQL. Логічне подання даних може не відповідати реляційній моделі, тому застосування мови SQL стане безглуздим. Найчастіше зручніше обробляти об'єкти в пам'яті, виконуючи відповідні види пошуку.

Великим недоліком об’єктно-орієнтованих баз даних є їхні тісні зв'язки із застосовуваною мовою програмування. До даних, що зберігаються в реляційній СКБД, можуть звертатися будь-які додатки, тоді як, приміром,

Java-об'єкт, поміщений в об’єктно-орієнтовану базу даних, буде становити інтерес лише для додатків, написаних на Java

**Об’єктно-реляційні бази даних**

Об’єктно-реляційні СКБД поєднують у собі риси реляційної й об'єктної моделей. Їхнє виникнення породжене тим, що реляційні бази даних добре працюють із убудованими типами даних і набагато гірше - з користувальницькими, нестандартними. Коли з'являється новий важливий тип даних, доводиться або включати його підтримку в СКБД, або змушувати програміста самостійно управляти даними в додатку.

Не всяку інформацію має сенс інтерпретувати у вигляді ланцюжків символів або цифр. Уявимо собі музичну базу даних. Пісню, закодовану у вигляді аудіофайлу, можна помістити в текстове поле великого розміру, але як у такому випадку буде здійснюватися текстовий пошук?

Перебудова СКБД із метою включення в неї підтримки нового типу даних - не кращий вихід з положення. Замість цього об’єктно-реляційна СКБД дозволяє завантажувати код, призначений для обробки "нетипових" даних. Таким чином, база даних зберігає свою табличну структуру, але спосіб обробки деяких полів таблиць визначається ззовні, тобто програмістом

**Реляційна модель**

*Реляційна модель це модель даних, що* описує *структуру даних,* припустимі *операції* над даними й спеціальні правила, що забезпечують *цілісність даних.*

Розроблена Едгаром Коддом (Edgar Codd) у фірмі *IBM* в 1970 р. Дані представляються у вигляді двомірних *таблиць,* над якими допускаються традиційні теоретико-множинні операції (об'єднання, перетинання, різниця й декартовий добуток) і спеціальні реляційні операції (селекція, проекція, з'єднання й розподіл).

Використання моделі дозволило створити як самі *реляційні бази даних,* так і *системи керування реляційними базами даних.*

Від англ. relation — відношення.

**Реляційна база даних**

*Реляційна база даних це база даних,* побудована на основі *реляційної моделі,* тобто БД, що має табличний спосіб вистави даних, а на зовнішньому рівні, що задається набором однорідних *таблиць.* Кожний об'єкт записується рядком у таблиці. Рядок називається *записом.* Запис складається з *полів* різного типу.

Реляційна база даних створюється й потім управляється за допомогою спеціальних засобів — *реляційних систем керування базами даних (РСУБД).*

Історично РБД діляться на:

РБД (РСУБД), створені для дуже більших (більше 1 Гбайт) баз даних *архітектури « клієнт-сервер».* Перші розробки виконані для більших комп'ютерів *IBM,* у яких використовується *мова SQL;*

РБД (РСУБД), створені спеціально для ПК, *типу dbase,* у яких архітектура така, що база й *користувач* перебувають на одному *комп'ютері.*

У цей час намітилася тенденція їх зближення. Так, у СУБД другого типу вводиться мова SQL, що дозволяє взаємодія БД різного типу.

**Реляційна система керування базою даних**

*Реляційна система керування базою даних це система керування реляційною базою даних,* побудована на *реляційній моделі.*

На практиці існує розподіл реляційних систем керування базами даних на потужні системи *архітектури «клієнт-сервер»* для великої кількості *транзакцій* мережні *протоколи, що підтримують різні типу Oracle, Gupta*, Informix, і системи для *невеликого числа користувачів* персональних комп'ютерів - *це* Msaccess, серія dbasex (dbaseii, dbaseiii, dbaseiv), FOX, Clipper і ін.

Кожна реляційна система керування базою даних — це досить потужна *мова програмування* зі специфічним ухилом на обробку *таблиць.*

Останні версії цих систем мають не тільки гарні швидкісні якості, але й мають удалий користувацький *інтерфейс.* До складу реляційної системи керування базою даних звичайно входить *мова SQL.*

До складу багатьох реляційних систем керування базами даних для персональних комп’ютерів входять три *модулі:* командна мова, що інтерпретує й/або система, що компілює, і *користувацька оболонка.*

**Системи керування базами даних**

Система керування базою даних це сукупність програмних засобів, що забезпечує можливість створення *бази даних (БД), доступу* до *даних* і *керування* базою даних.

Керування — це створення, доповнення, модифікація й формування результуючих *документів,* підтримка бази в актуальному стані й збереження її цілісності, запобігання *несанкціонованого доступу* до неї.

До складу системи керування базою даних входять:

- *мова програмування;*

- *генератори програм* — полегшують складання програм створення БД і їх обробку;

- *компілятори* — генерують програми керування в машинних кодах для прискорення роботи програм і їх незалежності від *середовища* СУБД; - генератори звітів — дозволяють *користувачеві* оперативно створювати вихідні документи, робити *вибірку, сортування* й розрахунки;

- засобу документування — дозволяють одночасно зі створенням БД створювати опису її в текстовому й графічному видах, опису програм з *лістингами.*

Відповідно до типів БД існують ієрархічні, мережні й реляційні СУБД.

Практично всі системи керування базами даних підтримують роботу в *мережі* з *архітектурою «клієнт-сервер».*

**Мова структурованих запитів (SQL)**

Мова структурованих запитів це високорівнева *мова,* призначена для роботи з *базами даних.*

Дозволяє модифікувати *дані,* становити й виконувати запити, виводити результати у вигляді звітів. Розроблений фірмою *IBM на* початку 800х рр. У цей час є загальноприйнятим *стандартом* для *систем керування базами даних реляційного типу.*

Мова SQL звичайно використовується разом з універсальними *мовами програмування 3, Pascal* і ін. або мовами керування базами даних Foxpro, *dbase* IV і ін.

По англ. Structured Query Language (SQL).