**Бази даних**

**База даних —** це певний набір даних, які пов'язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом.

**СУБД —** це програмні засоби, які виступають посередником між БД та її користувачами. Завдяки сукупності мовних та програмних засобів, СУБД сприяють створенню, ведення та спільного використання БД різними користувачами.

Сучасна програма СУБД складаються з ядра, процесору мови БД, підсистеми підтримки часу виконання та сервісних програм, які надають додаткові можливості для обслуговування інформаційних систем.

**Розподілені бази даних (РБД)** — це сукупність логічно пов'язаних між собою БД, які є розподіленими у комп'ютерній мережі.

Переваги РБД:

* ця модель відображає інформацію в найбільш простій для користувача формі;
* заснована на розвиненому математичному апараті, який дозволяє досить лаконічно описати основні операції з даними;
* дозволяє створювати мови маніпулювання даними не процедурного типу;
* маніпулювання даними на рівні вихідний БД і можливість зміни.

Недоліки РБД:

* найповільніший доступ до даних;
* трудомісткість розробки.

**Реляційна модель** використовує представлення даних як сукупність двовимірних таблиць особливого виду, відомого в математичній сфері як відношення.

Таким чином, перша модель являє собою набір таблиць, зв'язаних між собою. Кожний рядок і стовпчик в таблиці відповідає дійсності реального світу (наприклад, це інформація про людину).

**Реляційна модель** У реляційній моделі дані й взаємозв'язки між ними подаються за допомогою прямокутних таблиць. Рядки в реляційній базі даних називають записами, а стовпці — полями. Модель реляційної бази даних була вперше розроблена доктором Е. Ф. Коддом на початку 70-х років XX ст. як більш зручний засіб збереження, вибірки й маніпулювання даними, ніж ієрархічні й мережні бази даних. Модель двовимірної таблиці дозволяє звертатися до даних як по рядках, так і по стовпцях, що є значною перевагою[3].

Назва «реляційна» (relational) пов'язана з тим, що кожен запис у таблиці даних містить інформацію, яка стосується (related) якогось конкретного об'єкта. Крім того, зв'язані між бою (тобто такі, що знаходяться в певних відношеннях — relations) дані навіть різних типів в моделі можуть розглядатися як одне ціле.

Таблиця має такі властивості:

* кожний елемент таблиці являє собою один елемент даних;
* повторювані групи відсутні;
* усі стовпці в таблиці однорідні; це означає, що елементи стовпця мають однакову природу;
* стовпцям присвоєні унікальні імена;
* у таблиці немає двох однакових рядків

Порядок розміщення рядків і стовпців у таблиці довільний; таблиця такого типу називається відношенням. У сучасній практиці для рядка використовується термін «запис», а для стовпця термін «поле».

Основною відмінністю пошуку даних в ієрархічних, мережних і реляційних базах даних є те, що ієрархічні і мережні моделі даних здійснюють зв'язок і пошук між різними об'єктами за структурою, а реляційні — за значенням ключових атрибутів (наприклад, можна знайти всі записи, значення яких у полі «номер будинку» дорівнює 3, але не можна знайти 3-й рядок).

Оскільки реляційна структура концептуально проста, вона дозволяє реалізовувати невеликі і прості (і тому легкі для створення) бази даних, навіть персональні, сама можливість реалізації яких ніколи навіть і не розглядалася в системах з ієрархічною чи мережною моделлю.

Недоліком реляційної моделі даних є надмірність по полях (для створення зв'язків між різними об'єктами бази даних).

Майже всі наявні на сьогоднішній день комерційні бази даних і програмні продукти для їх створення використовують реляційну модель даних.

**Об'єктна модель** орієнтується на об'єктному програмуванні. Тобто, у такій моделі дані являють собою набір об'єктів и властивостей, зв'язаних між собою якими-небудь подібностями. Робота з об'єктами здійснюється за допомогою схованих у них методів. Прикладами об'єктних СУБД є Cache, Gemstone, ONTOS і т.д.

**Ієрархічна модель** Ієрархічна структура (модель) будується у вигляді ієрархічної деревоподібної структури, у якій для кожного головного об'єкта існує кілька підлеглих, а для кожного підлеглого об'єкта може бути тільки один головний. На найвищому рівні ієрархії перебуває кореневий об'єкт. Прикладом ієрархічної структури даних може бути організація каталогів на диску, різного роду класифікації, структура державної влади тощо.

Концептуальна схема ієрархічної моделі являє собою сукупність типів записів, пов'язаних типами зв'язків в одним чи кількома деревами. Усі типи зв'язків цієї моделі належать до виду «один до декількох» і зображуються у вигляді стрілок.

Таким чином, взаємозв'язки між об'єктами нагадують взаємозв'язки в генеалогічному дереві, за єдиним винятком: для кожного породженого (підлеглого) типу об'єкта може бути тільки один вхідний (головний) тип об'єкта. Тобто ієрархічна модель даних допускає тільки два типи зв'язків між об'єктами: «один до одного» і «один до декількох». Ієрархічні бази даних є навігаційними, тобто доступ можливий тільки за допомогою заздалегідь визначених зв'язків.

При моделюванні подій, як правило, необхідні зв'язки типу «багато до декількох». Як одне з можливих рішень зняття цього обмеження можна запропонувати дублювання об'єктів. Однак дублювання об'єктів створює можливості неузгодженості даних.

Перевага ієрархічної бази даних полягає в тому, що її навігаційна природа забезпечує швидкий доступ при проходженні вздовж заздалегідь визначених зв'язків. Однак негнучкість моделі даних і, зокрема, неможливість наявності в об'єкта декількох батьків, а також відсутність прямого доступу до даних роблять її непридатною в умовах частого виконання запитів, не запланованих заздалегідь. Ще одним недоліком ієрархічної моделі даних є те, що інформаційний пошук з нижніх рівнів ієрархії не можна спрямувати по вище розміщених вузлах.

**Мережна модель** У мережній моделі один і той же об'єкт може одночасно виступати як у ролі головного, так і підлеглого елемента. Це означає, що кожний об'єкт може брати участь у довільній кількості зв'язків. Зв'язок у цьому випадку може встановлюватися явно, коли значення деяких полів є посилання на дані, що містяться в іншому файлі. Прикладом мережної структури БД може бути структура автобусних маршрутів (із будь-якого населеного пункту існують маршрути в інші).

Подібно до ієрархічної, мережну модель також можна подати у вигляді орієнтованого графа. Але в цьому випадку граф може містити цикли, тобто вершина може мати кілька батьківських вершин.

Така структура набагато гнучкіша і виразніша від попередньої і придатна для моделювання більш ширшого класу завдань.

Ієрархічні і мережні бази даних часто називають базами даних з навігацією. Ця назва відбиває технологію доступу до даних, використовувану при написанні програм обробки мовою маніпулювання даними. При цьому доступ до даних по шляхах, не передбачених при створенні бази даних, може потребувати нерозумно тривалого часу.

Підвищуючи ефективність доступу до даних і скорочуючи таким чином час відповіді на запит, принцип навігації разом з цим підвищує і ступінь залежності програм і даних. Програми обробки даних виявляються жорстко прив'язаними до поточного стану структури бази даних і повинні бути переписані при її змінах. Операції модифікації і видалення даних вимагають перевстановлення покажчиків, а маніпулювання даними залишається записоорієнтованим. Крім того, принцип навігації не дозволяє істотно підвищувати рівень мови маніпулювання даними, щоб зробити його доступним користувачу-непрограмісту чи навіть програмісту-непрофесіоналу. Для пошуку запису-мети в ієрархічній або мережній структурі програміст повинен спочатку визначити шлях доступу, а потім — крок за кроком переглянути всі записи, що трапляються на цьому шляху.

Наскільки складними є схеми представлення ієрархічних і мережних баз даних, настільки і трудомістким є проектування конкретних прикладних систем на їхній основі. Як показує досвід, тривалі терміни розроблення прикладних систем нерідко призводять до того, що вони постійно перебувають на стадії розроблення і доопрацювання. Складність практичної реалізації баз даних на основі ієрархічної і мережної моделей визначила створення реляційної моделі даних[2].

**Цілісність бази даних** [database integrity] — стан бази даних, коли всі значення даних правильні в тому сенсі, що відображають стан реального світу (в межах заданих обмежень по точності та часовій узгодженості) і підпорядковуються правилам взаємної не суперечливості. Підтримка цілісності бази даних включає перевірку цілісності і відновлення з будь-якого неправильного стану, яке може бути виявлено; це входить у функції адміністратора бази даних.

**Правила цілісності бази даних** - це правила, які дають змогу уникнути введення некоректних даних у БД, а також забезпечити можливість зв'язування декількох таблиць. Ці правила можуть бути описані при створенні чи модифікації таблиці.

Типи правил цілісності

CHECK - Контроль допустимих значень атрибутів.

NOT NULL/NULL - Заборона/ дозвіл на використання не заданих або не визначених значень.

UNIQUE - Контроль унікальності значень атрибутів.

PRIMARY KEY - Первинний ключ.

FOREIGN KEY - Зовнішній ключ.

PRIMARY KEY

PRIMARY KEY (PK) використовується для ідентифікації рядків таблиці, в нього є певні особливості:

В одній таблиці БД може бути тільки один PK.

Рядки, які мають це правило, не можуть мати не визначені або не задані дані.

Будучи призначеним для декількох стовпців(складений ключ), це обмеження ставить унікальність комбінацій відповідних значень, хоча окреме значення в кожному стовпці складеного ключа не обов'язково має бути унікальним.

FOREIGN KEY

FOREIGN KEY (FG) використовується для того, щоб зв'язать дві таблиці, в нього є такі особливості :

Поле у головній таблиці, на яке буде адресуваться FG повинне бути оголошене PK або UNIQUE.

Поле у головній таблиці і поле з якої відбувається адресація повинні мати один й теж самий тип даних

Також за допомогою FG можна реалізувати каскадне видалення і оновлення даних.

CHECK

Обмеження CHECK дозволяє встановити умову, якій має задовольняти значення, що вводиться в таблицю, перш ніж воно буде прийнято. Обмеження CHECK складається із ключового слова CHECK та предикатом, який використовує вказане поле. Будь-яка спроба модифікувати або вставити значення поля, яке могло б зробити цей предикат невірним буде відхилена. Таким чином, можна запобігти введення небажаних даних. Обмеження CHECK може використовуватися у вигляді деякої маски введення, що забезпечить контроль заданого формату даних, що вводяться.

UNIQUE

Обмеження UNIQUE (унікальний), як і обмеження PRIMARY KEY, обмежує множину значень для вказаних стовпців унікальними значеннями. Як і обмеження PRIMARY KEY, обмеження UNIQUE може бути обмеженням таблиці, і тоді воно визначає унікальність комбінацій значень відповідних стовпців.

NOT NULL/NULL

Обмеження NULL / NOT NULL дозволяють або забороняють введення в поле NULL-значень. Очевидно, що обмеження NOT NULL має бути вказано для первинних ключів, оскільки в іншому випадку під загрозою опиниться цілісність даних.