Кафедра управління інформаційною безпекою

Спеціальність «Кібербезпека», 5 курс навчання

ЛЕКЦІЯ. ФІЗИЧНА ОРГАНІЗАЦІЯ БАЗ ДАНИХ

План:

1. Організація зберігання інформації.
2. Індексація.
3. Хешування.
4. В-дерева.
5. Інвертовані файли.

Література

1. Гайна Г.А. Основи проектування баз даних: Навчальний посібник. К.: КНУБА, 2005. – 204 с.

2. Гайна Г.А. Організація баз даних і знань. Мови баз даних: Конспект лекцій.−К.:КНУБА, 2002. – 64 с.

3. *Гайна Г.А., Попович Н.Л.* Організація баз даних і знань. Організація реляційних баз даних: Конспект лекцій.−К.:КНУБА, 2000. – 76 с.

1. Організація зберігання інформації.

Фізична організація даних відповідає за їх зберігання, управління, форми представлення і структури даних.

Фізичне проектування являє собою процес визначення характеристик сховища даних і доступу до них в БД. Властивості сховища даних залежать від пристроїв зберігання, засобів доступу до даних, що підтримуються системою і від СУБД. На етапі фізичного проектування визначається місцезнаходження даних на пристроях зберігання, загальна продуктивність системи.

В реляційних БД складні фізичні процеси організації даних приховані від користувача, але вони мають великий вплив на продуктивність роботи з БД.

Процес пошуку і представлення даних користувачу виконується в декілька етапів (рис. 1).

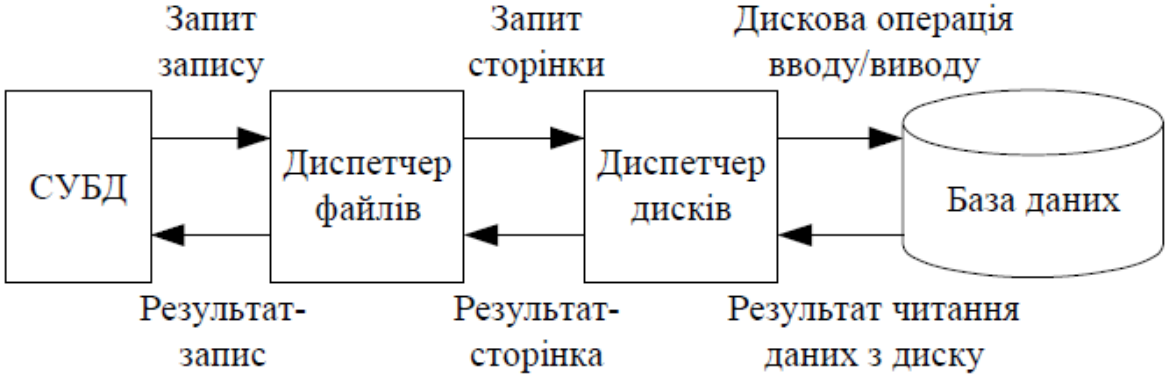

Рис. 1. Організація доступу до даних

Спочатку визначається необхідний запис, для знаходження якого викликається диспетчер файлів. Диспетчер файлів визначає сторінку, на якій знаходиться запис, і для її отримання викликає диспетчер дисків. Диспетчер дисків визначає фізичне положення необхідної сторінки на диску і передає її диспетчеру файлів, а той – СУБД. Головними одиницями операцій обміну системи СУБД − БД є сторінки даних. З точки зору СУБД БД виглядає як набір записів, з точки зору диспетчера файлів БД виглядає як набір сторінок. Кожна сторінка пам'яті має унікальний ідентифікатор. Кожен запис зберігається повністю на одній сторінці.

Для організації зберігання даних застосовується технологія *кластерізації* – фізично близьке розташування записів у просторі пам'яті середовища зберігання БД.

Пам'ять сторінок – організація простору зовнішньої або віртуальної пам'яті в БД, яка передбачає поділ простору пам'яті на сторінки фіксованого розміру. Для розміщення сторінок, що викликаються, в оперативній пам'яті використовуються спеціальні області – *буфери*. Оновлений в буферах зміст сторінок повертається у зовнішню пам'ять. Розмір сторінок, кількість буферів, алгоритми вибору буферів для розміщення сторінок суттєво впливають на ефективність роботи системи.

Компоненти системи БД, які призначені для зберігання даних, мають різну ємкість і різну швидкість доступу до даних. Організація ієрархії пам'яті комп'ютера наведена на рис. 2.

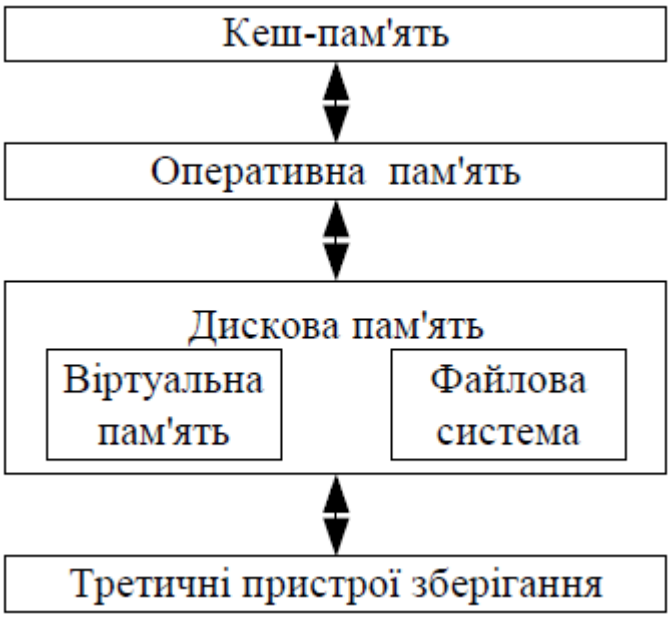

Рис. 2. Ієрархія пристроїв пам'яті бази даних

Кеш-пам'ять – швидка буферна пам'ять відносно невеликого розміру, яка зберігає останні дані до яких виконувався доступ. Вона реалізується апаратними або програмно-апаратними засобами.

Віртуальна пам'ять – це пам'ять, яка моделюється за допомогою апаратних засобів і програмного забезпечення. Вона дозволяє користувачу оперувати з об'ємами даних, які значно перевершують простір пам'яті, що виділяється в оперативній пам'яті, таким чином нібито вони знаходились в оперативній пам'яті.

Пам'ять третична – запам'ятовуючий пристрій великого об'єму і з низькою вартістю. Найчастіше це стойки з компакт-дисками або касети магнітної стрічки.

Організація зберігання даних має таку ієрархію:

− атрибути відображаються у байтову послідовність постійної або змінної довжини, яка називається *полями*;

− поля об'єднуються в набори даних постійної або змінної довжини, які називаються *записом*;

− записи зберігаються у фізичних *блоках* (*сторінках*);

− набори записів, які відповідають відношенням, зберігаються у вигляді *файлів*.

Розрізняють такі засоби фізичної організації файлів даних: послідовний, індексно-послідовний, прямий.

Послідовний файл – файл, до записів якого можливий послідовний доступ у порядку їх фізичного розміщення в пам'яті. Послідовна організація ефективна, коли застосування при кожному зверненні до БД обробляє значну кількість записів.

Індексно-послідовний файл − файл, до записів якого можливий прямий доступ за допомогою створеного для нього індексу по заданому ключу, а також послідовний доступ згідно з їх впорядкуванням по цьому ключу індексування. Індексно-послідовна організація ефективна, коли достатньо багато застосувань потребують послідовної обробки і достатньо багато застосувань потребують прямого доступу.

Файл прямого доступу − файл, в якому забезпечується прямий доступ до його записів за їх безпосередньою або посередньою адресою у середовищі зберігання або за заданими ключем за допомогою будь-якого методу відображення ключа в адресу. Пряма організація необхідна, коли для більшості застосувань потрібен прямий доступ до записів.

2. Індексація.

Індекс − структура даних, яка допомагає СУБД швидше знайти окремі записи в файлі і скоротити час виконання запитів користувачів. Основне призначення індексів полягає в забезпеченні ефективного прямого доступу до записів таблиці по ключу. Файл, що містить індексні записи називається *індексним файлом*.

Залежно від організації індексної і основної області розрізняють два типи файлів: зі щільним індексом і з розрідженим індексом. *Щільний індекс* вміщує індексні записи для всіх значень ключа пошуку в даному файлі. *Розріджений індекс* вміщує індексні записи тільки для деяких значень ключа пошуку в даному файлі.

У файлах зі щільним індексом основна область вміщує послідовність записів однакової довжини, які *розташовані у довільному порядку*, а структура індексного запису має такий вигляд: (значення ключа, номер запису).

В індексних файлах зі щільним індексом для кожного запису в основній області існує один запис з індексної області. *Всі записи* в індексній області *впорядковані* за значенням ключа. Файли зі щільним індексом називаються *індексно-прямими файлами*.

У файлах з розрідженим індексом основна область вміщує послідовність записів однакової довжини, *які розташовані у впорядкованому порядку*, а структура індексного запису має такий вигляд: (значення ключа першого запису блока, номер блока з цим записом).

Розріджений індекс будується для впорядкованих файлів. Файли з розрідженим індексом називаються *індексно-послідовними файлами*.

Індекси доцільно створювати по стовпцю або групі стовпців у таких випадках:

− часто виконується пошук у БД за атрибутами, які перелічені в умові WHERE оператора SELECT;

− часто виконується об'єднання таблиць за певними атрибутами;

− часто виконується сортування таблиць (використання ORDER BY в операторі SELECT).

Індекси не доцільно створювати по стовпцю або групі стовпців у таких випадках:

− рідко використовуються для пошуку;

− містять значення, які часто змінюються, що призводить до частого оновлення індексу і, відповідно, уповільнює роботу;

− містять незначну кількість значень.

Як правило в БД визначають два види індексів:

− індекси, які використовуються запитами для доступу до даних;

− індекси, які забезпечують посилкову цілісність між таблицями.

Більша частина БД підтримує B-дерева, крім того деякі з них працюють із хеш-таблицями. Деякі системи представляють багатомірні структури даних,

такі, як варіанти R-дерев і квадрадерев (*kd-дерева*). Окремі системи підтримують також бітові вектори і багатотабличні індекси з'єднання. Системи, що розташовуються в оперативній пам'яті, крім того підтримують структури даних, які мають невеликі коефіцієнти розгалуження, такі як Т-дерева, бінарні дерева.

3. Хешування.

Хешування − технологія прямого доступу до записів БД за відомими значеннями їхніх ключів. Час доступу суттєво не залежить від кількості записів, що зберігаються. Суть методу хешування полягає в тому, що за значеннями ключа *k*, або його певних характеристик, обчислюється деяка хеш-фукція *h*(*k*) і отримане значення береться в якості адреси початку пошуку. Недоліком більшості хеш-функцій є те, що вони не гарантують отримання унікальної адреси, оскільки кількість можливих значень поля хешування може бути значно більшою за кількість адрес, які доступні для запису. Ситуація, коли різним значенням ключів відповідає одне значення хеш-функції називається *колізією*. Значення ключів, які мають одну й ту ж хеш-функцію називаються *синонімами*. Для вирішення колізій застосовують спеціальні методи: послідовного перебору, введення області переповнення, багатократне хешування. Хеш-структура показана на рис. 3.

Рис. 3. Організація хеш-структури

Хеш-структури можуть забезпечити відповідь на запитання до БД за одне звернення до диску при умові відсутності синонімів. Хеш-структури мають низьку ефективність при запитах на визначення діапазонів, знаходження екстремумів і деяких інших.

4. B-дерева.

У багатьох СУБД для збереження індексів використовується структура даних, яка називається деревом. Серед дерев найбільш поширені бінарні дерева, B-дерева та їх різновиди (B^+-дерева, B^*-дерева і т.ін.). B-дерево є збалансованим, впорядкованим за значенням ключа деревом. Найбільш розповсюдженими серед B-дерев є B^+-дерева.

Приклад. На рис. 4 наведено фрагмент B^+-дерева і його зв'язок з БД.

5. Інвертовані файли.

Для забезпечення прискореного доступу за вторинними ключами використовуються структури, які називаються інвертованими списками. Інвертований список являє собою дворівневу індексну структуру. На першому рівні знаходиться файл або частина файлу, в який впорядковано розташовані значення вторинних ключів. Кожен запис із вторинним ключем має покажчик на номер першого блоку в ланцюгу блоків, які містять номери записів з даним значенням вторинного ключа. На другому рівні знаходиться ланцюг блоків, які містять номери записів, що вміщують одне і те ж саме значення вторинного ключа. При цьому блоки другого рівня впорядковані за значенням вторинного ключа. На третьому рівні знаходиться безпосередньо файл даних. Пошук даних виконується у такій послідовності:

− на першому рівні знаходиться значення вторинного ключа;

− за знайденим покажчиком читається блок другого рівня, який містить номери всіх записів із заданим значенням вторинного ключа;

− у робочу область завантажується зміст всіх записів із заданим значенням вторинного ключа.

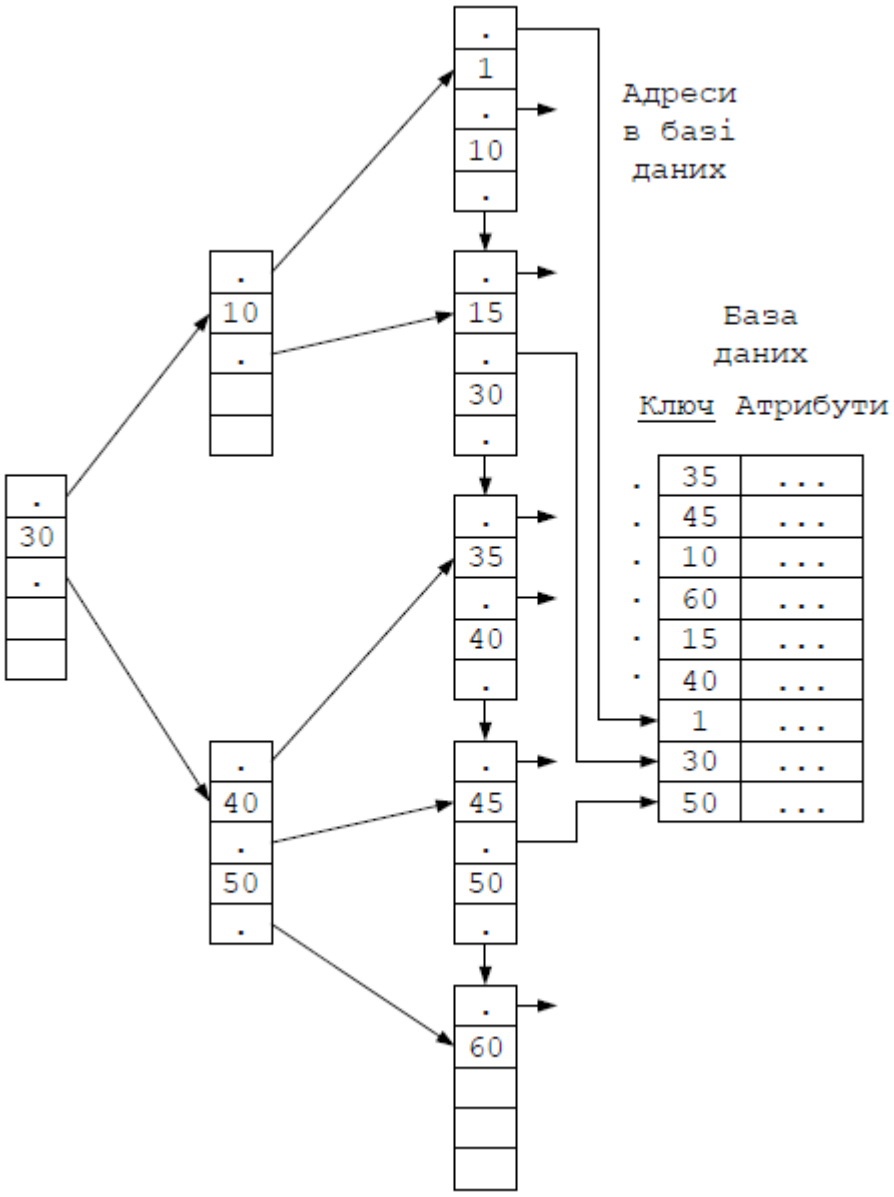

Рис. 4. Приклад побудови B^+-дерева

Контрольні запитання.

1. Порівняти послідовну, індексно-послідовну і пряму організацію файлів.

2. Чому розподіл записів по блоках впливає на швидкість роботи системи?

3. Пояснити різницю між організацією пошукових структур за первинним ключем і за вторинним ключем.

4. Дати характеристику сторінкової організації даних в СУБД.

5. Що собою являє технологія хешування і для чого вона використовується в базах даних?

6. У чому полягає основне призначення індексів? Які переваги і недоліки застосування індексів?

7. Які індекси називаються щільними і які розрідженими? Яка різниця в їх застосуваннях?

8. Що собою являє ієрархія пам'яті бази даних? Які види пам'яті існують?

Розробив: к.т.н., доц. Полотай О.І.