Тема лекції 9:

Індекси як об’єкт в реляційних базах даних

- Архітектура таблиць і структур даних індексу
- Структура купи
- Структури кластерного індексу
- Структури некластерних індексів
- Індекси у віртуальних таблицях
- Фрагментація індексу

Поняття сторінки даних

- З точки зору фізичного зберігання даних у СУБД серверного типу елементарні області зберігання даних називаються сторінками.
- Це означає, що дані на фізичному диску зберігаються в сторінках.
- **Сторінка** – це блок фіксованої довжини неперервних віртуальних адрес пам’яті, який бере участь в операціях читання і запису як єдине ціле.
- В SQL Server розмір сторінки даних рівний 8 Кбайт (8 192 байт).
- В базах даних SQL Server об’єкти, зокрема, таблиці та індекси, зберігаються у вигляді **колекцій сторінок**.

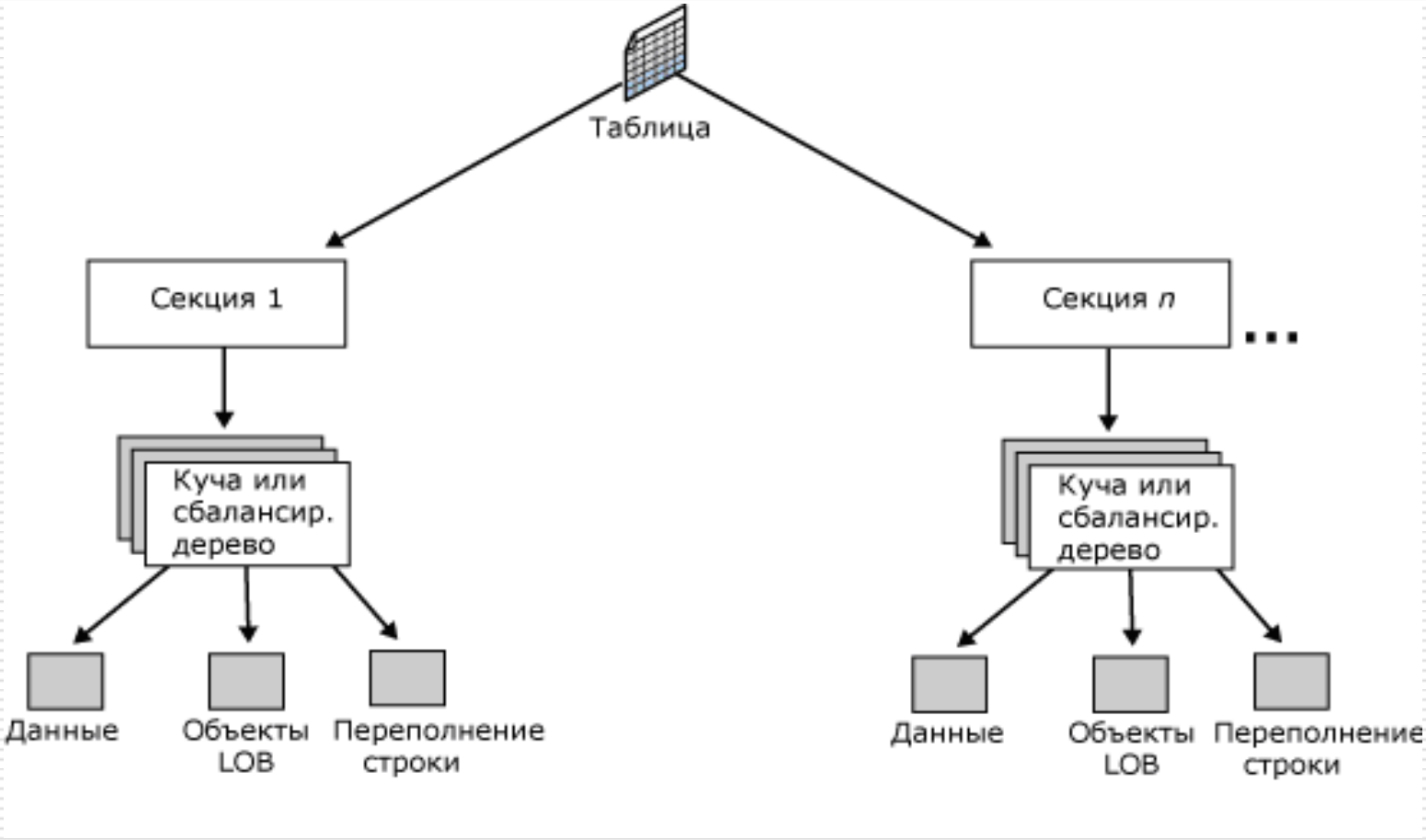
Поняття екстенту

- Сторінки об’єднуються в екстенти.
- Екстент – це одиниця пам’яті, яку сервер виділяє як єдине ціле при розміщенні даних на диску по мірі необхідності.
- Починаючи з SQL Server 2000, екстент складається з 8 сторінок.

Організація таблиць

- Таблиця мітиться в одній або декількох секціях,
- Кожна секція містить рядки даних або в кучі, або в структурі кластерного індексу.
- Сторінки кучі або кластерного індексу об’єднуються в одну або декілька **одиниць розподілу** в залежності від типів стовпців у рядках даних.

Організація таблиць



Поняття секції (Partition)

- **Секція** (partition) – це базова одиниця організації даних.
- За замовчуванням таблиця або індекс має єдину секцію, яка містить усі сторінки таблиці або індексу, і секція розміщена в одній файловій групі. (Така організація даних була у версіях до SQL Server 2000 включно)
- Якщо таблиця або індекс використовують декілька секцій (з SQL Server 2005), то дані розділяються горизонтально так, що групи рядків співставляються з окремими секціями, базуючись на вказаному стовпці.

Поняття секції (Partition)

- Секції можуть зберігатись в одній або декількох файлових групах в базі даних (з SQL Server 2005).
- Однак таблиця або індекс розглядаються як єдина логічна сутність при виконанні запитів або операцій модифікації даних.
- Для перегляду секцій, в яких розміщуються таблиці або індекси, можна використати віртуальну таблицю (view) каталога ***sys.partitions***.

Методи організації сторінок даних всередині секції

- ***Кластерні таблиці*** – це таблиці, які мають кластерний індекс. В таких таблицях рядки даних зберігаються у порядку ключа кластерного індексу.
- ***Купи*** – це таблиці, які не мають кластерного індексу. В купах рядки даних зберігаються без визначеного порядку, і будь-який порядок в послідовності сторінок даних відсутній.
- Якщо купа або кластерна таблиця містить декілька секцій, то кожна секція має структуру купи або збалансованого дерева відповідно, і яка містить групу рядків для вказаної секції. Наприклад, якщо кластерна таблиця містить чотири секції, то є чотири збалансованих дерева, по одному на кожну секцію.

Поняття одиниці розподілу

- ***Одиниця розподілу*** – це колекція сторінок в купі або збалансованому дереві, яка використовується для керування даними відповідно до типів сторінок.
- У наступній таблиці перелічуються одиниці розподілу, які використовуються для керування даними в таблицях та індексах.

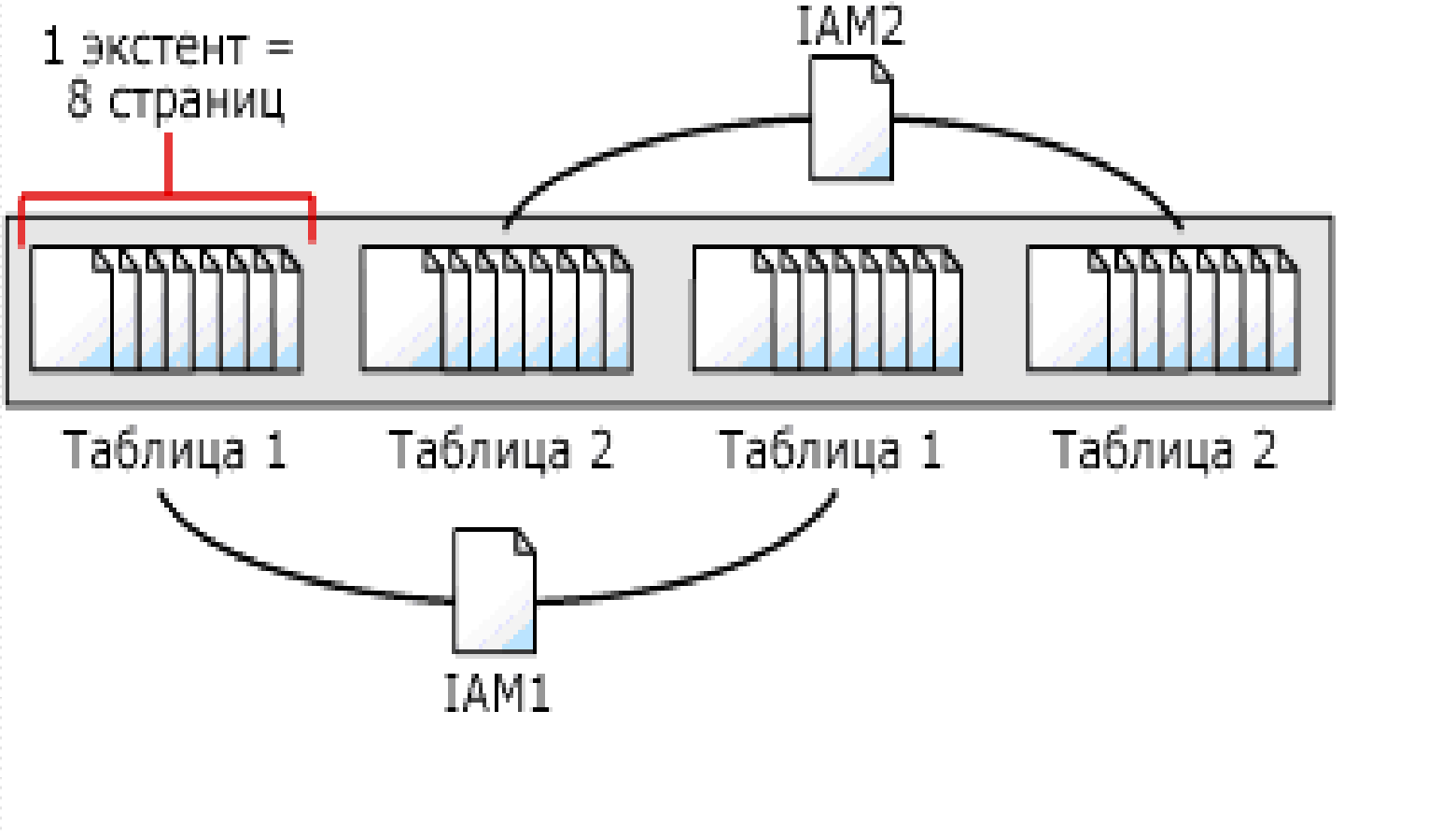
Одиниці розподілу в організації таблиць та індексів

Тип одиниці розподілу	Дані, для керування, якими цей тип використовується
IN_ROW_DATA	Рядки даних або індексу, які містять усі дані, крім даних великого об'єкта (LOB). Сторінки мають тип `Data` або `Index`.
LOB_DATA	Дані великого об'єкта, які зберігаються в одному або декількох з наступних типів даних: `text, ntext, image, xml, varchar(max), nvarchar(max), varbinary(max)` або користувацькі типи CLR (CLR UDT). Сторінки мають тип `Text/Image`.
ROW_OVERFLOW_DATA	Дані змінної довжини, які зберігаються в стовпцях типів `varchar, nvarchar, varbinary` або `sql_variant`, і які перевищують обмеження розміру рядка в 8 060 байт. Сторінки мають тип `Text/Image`.

Організація дискового простору, зайнятого об’єктами

- Кожна одиниця розподілу містить як мінімум одну сторінку карти розподілу індекса (Index Allocation Map, ***IAM-сторінка***) для кожного з файлів, в яких містяться всі екстенти
- Для файлу може існувати декілька IAM-сторінок, якщо розмір екстентів файлу, призначеного одиниці розподілу, перевищує об’єм, який може бути записаний в одній IAM-сторінці.
- IAM-сторінки для кожної одиниці розподілу виділяються за необхідністю і розміщуються у файлі в довільному порядку.
- Усі IAM-сторінки, які відносяться до однієї одиниці розподілу, об’єднуються в ланцюг

Організація дискового простору, зайнятого об’єктами



Структура купи

- Купа – це таблиця без кластерного індексу.
- За замовчуванням в купі є одна секція.
- Якщо купа має декілька секцій, то кожна з них має структуру купи, яка містить дані для відповідної секції. Наприклад, якщо в купі є чотири секції, то існують чотири структури купи, по одній на кожну секцію.
- В залежності від типів даних в купі, кожна структура купи має одну або декілька одиниць розподілу для збереження і керування даними певної секції.
- У кожної купи існує хоча б одна одиниця розподілу IN_ROW_DATA на кожну секцію.

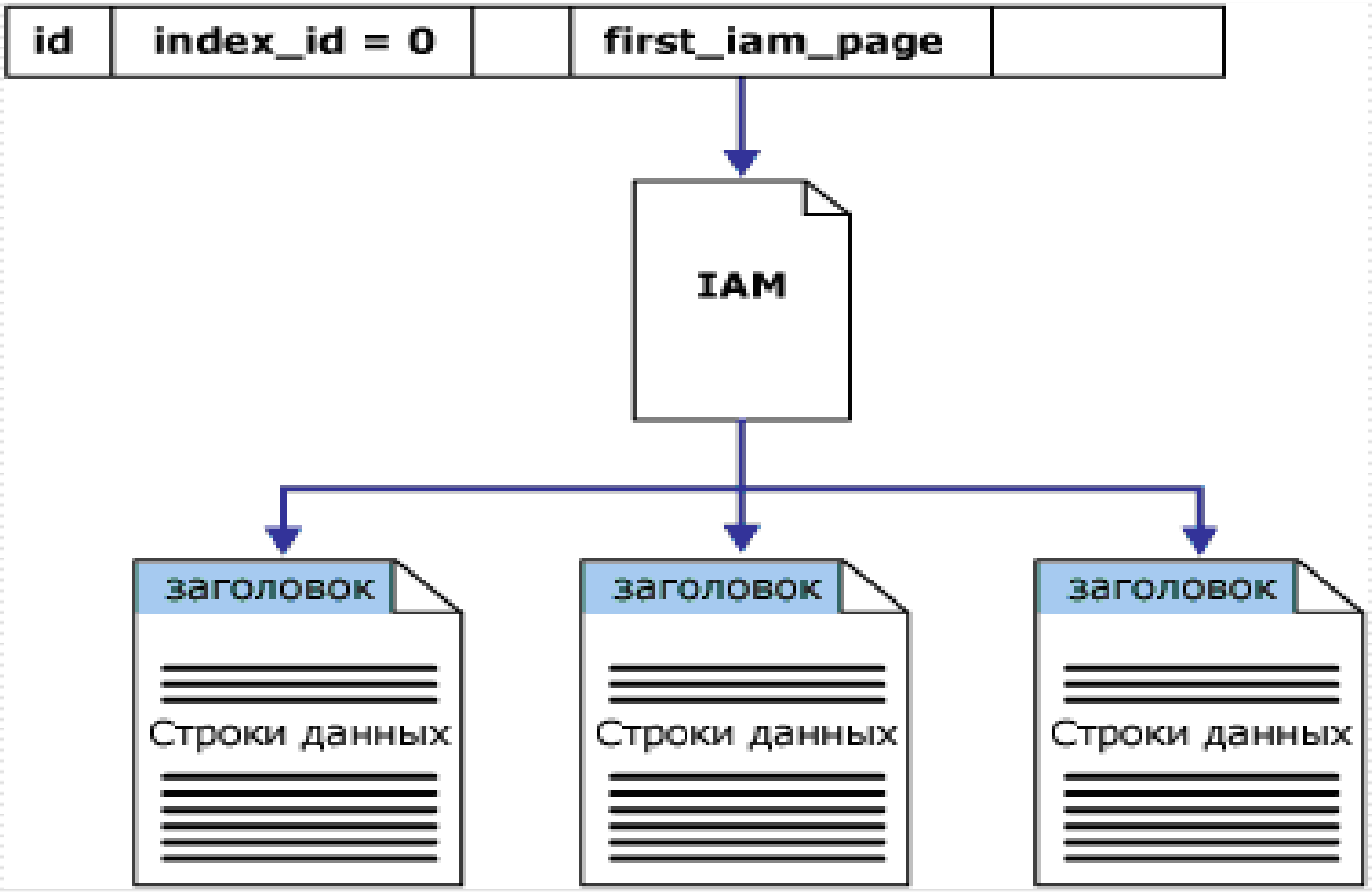
Робота з купою

- SQL Server використовує IAM-сторінки для переміщення по купі.
- Сторінки даних і рядки в цих сторінках невпорядковані і незв’язані.
- Єдиним логічним з’єднанням сторінок даних є дані, які записані в IAM-сторінки.
- Перегляд таблиць або послідовне читання в купі може виконуватись шляхом перегляду IAM-сторінок для знаходження екстентів, які містять сторінки купи.

Використання IAM-сторінки в односекційній купі

- Для кожної купи існує один рядок у віртуальній таблиці (view) **sys.partitions** з index_id=0 у кожній секції.
- Стовпец first_iam_page в системній віртуальній таблиці **sys.system_internals_allocation_units** вказує на першу IAM-сторінку в ланцюжку IAM-сторінок, який керує виділенням простору купи у певній секції.

Використання IAM-сторінки в односекційній купі



Індекси як засіб швидкого пошуку даних

- Індекс є фізичним об’єктом бази даних, який має структуру збалансованого B-дерева.
- Використання індексів у вигляді збалансованих дерев дозволяє досягти високої швидкодії при реалізації пошуку даних за ключем.
- Індекс застосовується оптимізатором СУБД для пришвидшення доступу до даних у порівнянні з лінійним скануванням таблиці.
- Індекси можуть використовуватись для надання обмежень унікальності потенційного ключа базової таблиці.

Основні типи індексів в SQL Server

- кластерний
- некластерний

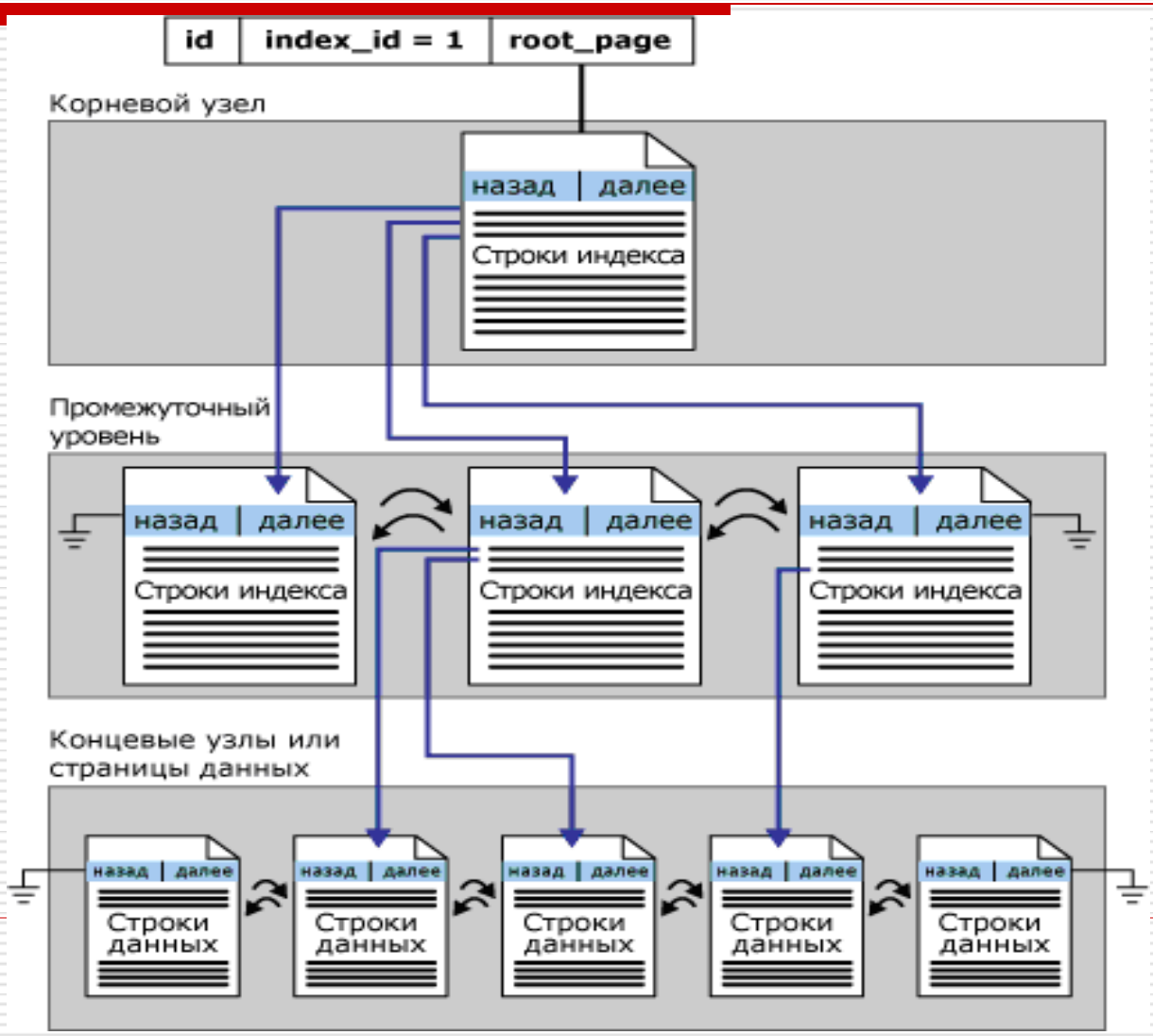
Структури кластерного індексу

- Кластерний індекс реалізується у вигляді збалансованого дерева, яке підтримує швидку вибірку рядків за їх ключовими значеннями в кластерному індексі.
- Сторінки на кожному рівні індексу, включаючи сторінки даних на кінцевому рівні, зв’язані у ***двонапрямлений список***.
- Переміщення з одного рівня на інший виконується за допомогою ключових значень.

Структури кластерного індексу

- Кожна сторінка в збалансованому дереві індексу називається вузлом індексу.
- Верхній вузол дерева називається кореневим.
- Вузли нижнього рівня індексу називаються кінцевими.
- Усі рівні між кореневими і кінцевими вузлами називаються проміжними.
- В кластерному індексі кінцеві вузли містять сторінки даних базової таблиці.
- На сторінках кореневого і проміжного вузлів знаходяться рядки індексу.
- Кожен рядок індексу містить ключове значення і вказівник або на сторінку проміжного рівня дерева, або на рядок даних кінцевого рівня індексу.
- На кожному рівні сторінки зв’язані у двонапрямлений список.

Структура кластерного індексу для однієї секції



Структура кластерного індексу для однієї секції

- Для кожного кластерного індексу таблиця ***sys.partitions*** містить один рядок зі значенням index_id рівним 1 для кожної секції.
- За замовчуванням кластерний індекс займає одну секцію.
- Якщо кластерний індекс займає декілька секцій, то кожна секція містить збалансоване дерево, в якому розміщені дані цієї секції.

Структура кластерного індексу для однієї секції

- ***sys.system_internals_allocation_units*** містить вказівники на кореневі вузли кластерного індексу для кожної секції.
- SQL Server рухається вниз по індексу, щоб знайти рядок, який відповідає відповідному ключеві кластерного індексу.
- Щоб знайти діапазон ключів, SQL Server спочатку знаходить початкове значення ключа в діапазоні, а потім сканує сторінки даних, використовуючи вказівники на наступну і попередню сторінку.
- Щоб знайти першу сторінку в ланцюгу сторінок даних, SQL Server рухається по крайніх лівих вказівниках від кореня індексу.

Спосіб зберігання кластерних таблиць

- ***Кластерна таблиця*** – це таблиця, з кластерним індексом.
- В кластерному індексі кінцевий рівень не містить ключів індексу і вказівників, а містить самі дані.
- Це означає, що дані вже не зберігаються в структурі купи. Тепер вони зберігаються на кінцевому рівні індексу і відсортовані за ключем індексу.
- Для кожної таблиці можна визначити лише ***один кластерний індекс***.

Робота з кластерними таблицями

- При внесенні нового рядка у кластерну таблицю, якщо у потрібної сторінки немає вільного місця, відбувається процес розщеплення сторінок
- При розщепленні сторінок половина рядків або індексних входів переміщується з повністю заповненої сторінки у нову виділену сторінку
- Таким чином, замість однієї повністю заповненої сторінки з'являються дві, заповнені наполовину.
- Так з'являється місце для нового рядка або індексного входу зі збереженням фізичного порядку сортування.
- Доступ до певного рядка кластерної таблиці виконується за алгоритмомпошуку в збалансованому дереві за значеннямключа, який є унікальний для кожного рядка таблиці.

Переваги кластеризованих таблиць

- системі SQL Server для доступу до даних не потрібно йти за вказівником; дані зберігаються безпосередньо в індексі;
- дані сортуються за ключем індексу, що є головною перевагою: системі не потрібно буде виконувати операцію сортування під час доступу до даних, вони вже посортовані.

Інструкція створення кластерного індексу

CREATE [UNIQUE]
CLUSTERED INDEX <назва індексу>
ON <об'єкт> (<стовпець_1> [ASC | DESC]
[,...< стовпець_n>])

- SQL Server створює унікальний індекс, якщо для таблиці визначене обмеження первинного ключа або унікальності.
- Якщо визначено первинний ключ, то SQL Server створює кластерний індекс за замовчуванням, якщо такого індексу ще немає в таблиці.

Структури некластерних індексів

- Некластерні індекси мають таку ж структуру збалансованого дерева, що й кластерні індекси; але з такою різницею:
  - рядки даних в базовій таблиці не сортуються і зберігаються в тому порядку, який оснований на їх некластерних ключах;
  - кінцевий рівень некластерного індексу складається зі сторінок індексу замість сторінок даних.
- Некластерні індекси можуть визначатися на таблиці або віртуальній таблиці (view) з кластерним індексом, або на купі.
- Кожен рядок некластерного індексу містить некластерне ключове значення і вказівник на рядок.

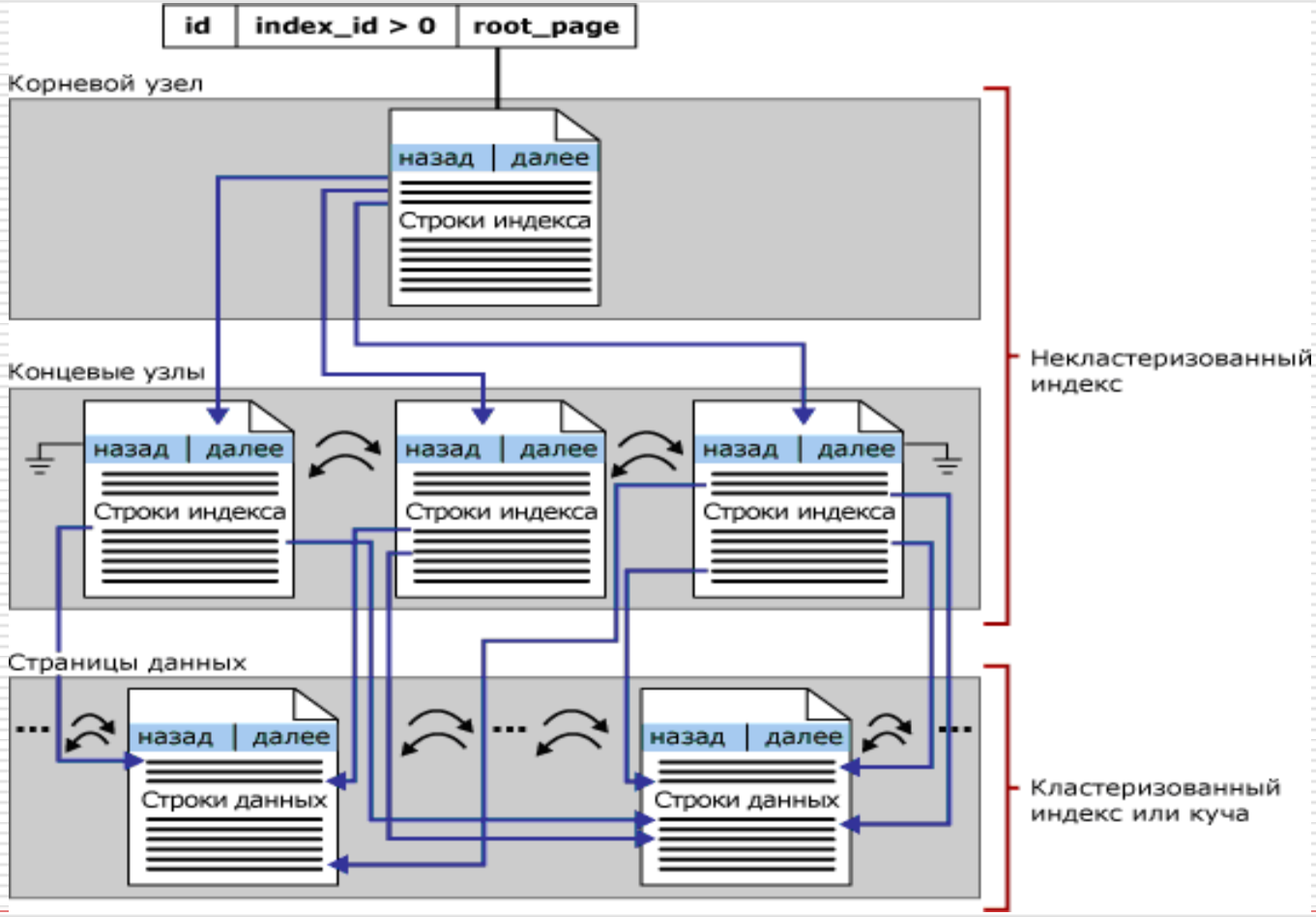
Вказівник в рядках некластерного індексу

- Якщо таблиця є ***купою***, то вказівник є вказівником на рядок.
  - Вказівник будується на основі ідентифікатора файлу (ID), номера сторінки і номеру рядка на сторінці.
  - Увесь вказівник називається ідентифікатором рядка (RID).

Вказівник в рядках некластерного індексу

- Якщо для таблиці (або віртуальної таблиці (view)) створено індекс, то вказівник – це ключ кластерного індексу для рядка.
  - якщо кластерний індекс не є унікальним індексом, то SQL Server робить його унікальним, щоб використовувати в некластерних індексах.
  - SQL Server отримує рядок даних шляхом пошуку за кластерним індексом, використовуючи ключ некластерного індексу, який зберігається в кінцевому рядку некластерного індексу.

Структура некластерного індексу для однієї секції



Структура некластерного індексу

- Для некластерних індексів є один рядок в таблиці **sys.partitions** зі значенням стовпця index_id>0 для кожної секції.
- За замовчуванням некластерний індекс займає одну секцію.
- Якщо некластерний індекс займає декілька секцій, то кожна секція має структуру збалансованого дерева, в якому розміщені індексні рядки цієї секції.
- Колекції сторінок збалансованих дерев скріпляються вказівниками root_page в системній віртуальній таблиці **sys.system_internals_allocation_units**.

Структура некластерного індексу

- На противагу кластерним індексам некластерні індекси не містять усіх рядків даних на кінцевому рівні індексу.
- Замість цього на кінцевому рівні зберігаються усі ключові стовпці і вказівники на рядки таблиці.
- Використання і запис вказівників залежить від того, чи є базова таблиця купою, чи має кластерний індекс.
- Оскільки некластерні індекси не містять повністю рядків даних, то для кожної таблиці можна створити до 249 таких індексів

Інструкція створення некластерного індексу

CREATE [UNIQUE]
NONCLUSTERED INDEX <назва індексу>
ON <об'єкт> (<стовпець_1> [ASC | DESC]
[,...< стовпець_n>])

Індекси у віртуальних таблицях (view)

- Для зберігання віртуальної таблиці ***без індексу*** не вимагається до додаткового місця на диску. SQL Server об’єднує опис віртуальної таблиці з запитом, в якому вона використовується, оптимізує, генерує план виконання запиту і повертає дані.
- У випадку, коли віртуальна таблиця обробляє або об’єднує ***велику кількість рядків***, і до такої віртуальної таблиці часто звертаються запити, доцільно таку віртуальну таблицю індексувати.

Індекси у віртуальних таблицях (view)

- При індексації віртуальна таблиця обробляється, і результат зберігається у файлі даних, так як це робиться для кластерної таблиці.
- SQL Server автоматично обслуговує цей індекс у випадку зміни даних в базовій таблиці.
- Індекси у віртуальних таблицях можуть пришвидшити вибірку даних, але індексація забирає додаткові ресурси при модифікації даних в базовій таблиці.
- ***Рекомендується:*** індексувати віртуальні таблиці, якщо віртуальна таблиця обробляє велику кількість рядків, використовує агрегатні функції, причому дані в базовій таблиці змінюються не дуже часто.

Фрагментація індексу

- При операціях вставки даних в таблицю, ці дані записуються на вказаній сторінці серед сторінок кінцевого рівня кластерного індексу. Ключі некластерного індексу також повинні бути вставлені на потрібну сторінку серед сторінок рівня листових вершин некластерного індексу. Якщо на такій сторінці немає місця, то SQL Server повинен виділити нову сторінку і прив’язати її до відповідного індексу.
- Такі ситуації призводять до фрагментації індексу. У таких випадках логічний порядок сторінок даних не відповідає їх фізичному порядку.
- До фрагментації індексу може призвести й виконання операцій UPDATE і DELETE.

Усунення фрагментації

- З метою усунення фрагментації сторінок даних необхідно періодично реорганізовувати усі індекси всіх таблиць.
- Період виконання дефрагментації залежить від типу середовища.
  - фрагментація не дуже заважає в OLTP-середовищі.
  - для OLAP-середовища засоби протидії фрагментації повинні бути ефективними та застосовуватись частіше.

Зменшення фрагментації (параметр FILLFACTOR)

- Щоб зменшити фрагментацію, при ***створенні індексу*** можна використовувати параметр FILLFACTOR, який визначає, до якого процентного співвідношення при створенні індексу повинні бути заповнені сторінки кінцевого рівня.
  - При малому значенні параметра FILLFACTOR фрагментація є малоймовірною. У цьому випадку на рівні листових вершин можна розмістити рядки більшого об’єму без розбиття сторінки.
  - З другого боку, при малих значеннях параметра FILLFACTOR індекс буде більшим, тому з самого початку на кожній сторінці кінцевого рівня зберігається менше даних.

Дефрагментація індексу

- Якщо проіндексована таблиця не є лише для читання, то рано чи пізно станеться так, що її індекси будуть фрагментовані.
- Для підвищення швидкості доступу до даних такі індекси можна дефрагментувати за допомогою інструкції ALTER INDEX.

Параметри дефрагментації індексу

- REORGANIZE: Реорганізація індексу означає, що сторінки рівня листових вершин сортуються за допомогою алгоритму бульбашки. REORGANIZE сортує лише сторінки даних, а не записи на сторінках; це означає, що параметр FILLFACTOR при реорганізації використовувати неможна.

Параметри дефрагментації індексу

- REBUILD:Відбувається перебудова (rebuilding) всього індексу.
  - Це вимагає більше часу, ніж реорганізація індексу, але дає кращі результати.
  - Можна вказати параметр FILLFACTOR, щоб сторінки знову заповнювались до бажаного об'єму.
  - Якщо параметр FILLFACTOR не вказується, то сторінки рівня листових вершин заповнюються повністю.

Параметри дефрагментації індексу

- REBUILD:
  - При ***перебудові*** індексів також можна вказати параметр ONLINE.
  - Якщо цей параметр не вказано, то перебудова індексу виконується в автономному режимі, що означає блокування таблиці на протязі всього процесу перебудови. Перебудова в автономному режимі виконується швидше, ніж в робочому режимі, але через блокування даних вона не може використовуватись у той час, коли необхідно мати доступ до даних.

Дякую за увагу