

27 | 从数据页的角度理解B+树查询

2019-08-12 陈旻



我们之前已经了解了B+树和Hash索引的原理，这些索引结构给我们提供了高效的索引方式，不过这些索引信息以及数据记录都是保存在文件上的，确切说是存储在页结构中。

对数据库的存储结构以及页结构的底层进行了解，可以加深我们对索引运行机制的认识，从而你对索引的存储、查询原理，以及对SQL查询效率有更深入的理解。

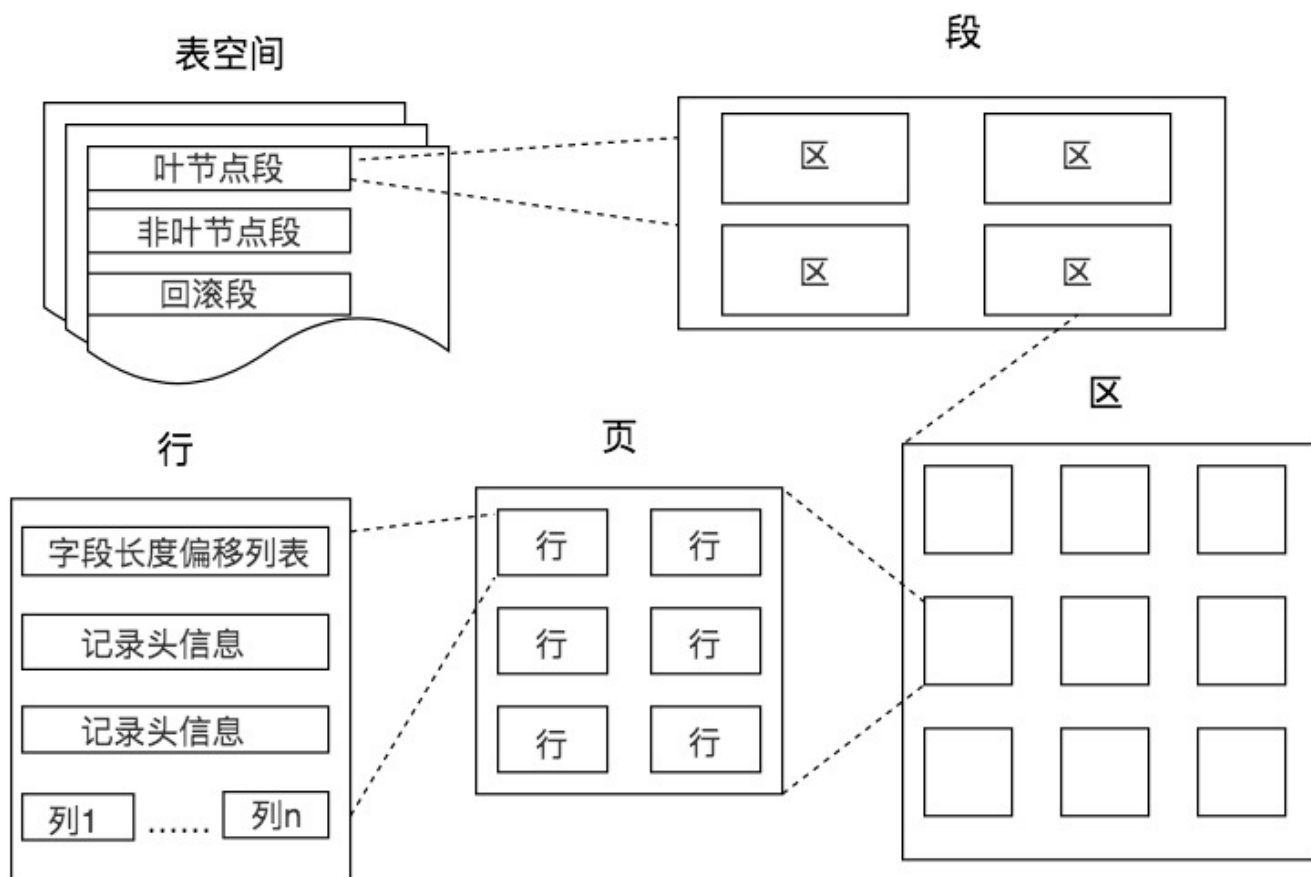
今天的课程主要包括下面几个部分：

1. 数据库中的存储结构是怎样的？页、区、段和表空间分别指的是什么？
2. 为什么页（Page）是数据库存储空间的基本单位？
3. 从数据页的角度来看，B+树是如何进行查询的？

数据库中的存储结构是怎样的

记录是按照行来存储的，但是数据库的读取并不以行为单位，否则一次读取（也就是一次I/O操作）只能处理一行数据，效率会非常低。因此在数据库中，不论读一行，还是读多行，都是将这些行所在的页进行加载。也就是说，数据库管理存储空间的基本单位是页（Page）。

一个页中可以存储多个行记录（Row），同时在数据库中，还存在着区（Extent）、段（Segment）和表空间（Tablespace）。行、页、区、段、表空间的关系如下图所示：



从图中你能看到一个表空间包括了一个或多个段，一个段包括了一个或多个区，一个区包括了多个页，而一个页中可以有多行记录，这些概念我简单给你讲解下。

区（**Extent**）是比页大一级的存储结构，在InnoDB存储引擎中，一个区会分配**64**个连续的页。因为InnoDB中的页大小默认是**16KB**，所以一个区的大小是**64*16KB=1MB**。

段（**Segment**）由一个或多个区组成，区在文件系统是一个连续分配的空间（在InnoDB中是连续的**64**个页），不过在段中不要求区与区之间是相邻的。段是数据库中的分配单位，不同类型的数据库对象以不同的段形式存在。当我们创建数据表、索引的时候，就会相应创建对应的段，比如创建一张表时会创建一个表段，创建一个索引时会创建一个索引段。

表空间（**Tablespace**）是一个逻辑容器，表空间存储的对象是段，在一个表空间中可以有一个或多个段，但是一个段只能属于一个表空间。数据库由一个或多个表空间组成，表空间从管理上可以划分为系统表空间、用户表空间、撤销表空间、临时表空间等。

在InnoDB中存在两种表空间的类型：共享表空间和独立表空间。如果是共享表空间就意味着多张表共用一个表空间。如果是独立表空间，就意味着每张表有一个独立的表空间，也就是数据和索引信息都会保存在自己的表空间中。独立的表空间可以在不同的数据库之间进行迁移。

你可以通过下面的命令来查看InnoDB的表空间类型：

```
mysql > show variables like 'innodb_file_per_table';
```

```
mysql> show variables like 'innodb_file_per_table'
```

Variable_name	Value
innodb_file_per_table	ON

你能看到innodb_file_per_table=ON，这就意味着每张表都会单独保存为一个.ibd文件。

数据页内的结构是怎样的

页（Page）如果按类型划分的话，常见的有数据页（保存B+树节点）、系统页、Undo页和事务数据页等。数据页是我们最常使用的页。

表页的大小限定了表行的最大长度，不同DBMS的表页大小不同。比如在MySQL的InnoDB存储引擎中，默认页的大小是16KB，我们可以通过下面的命令来进行查看：

```
mysql> show variables like '%innodb_page_size%';
```

```
mysql> show variables like 'innodb_page_size';
```

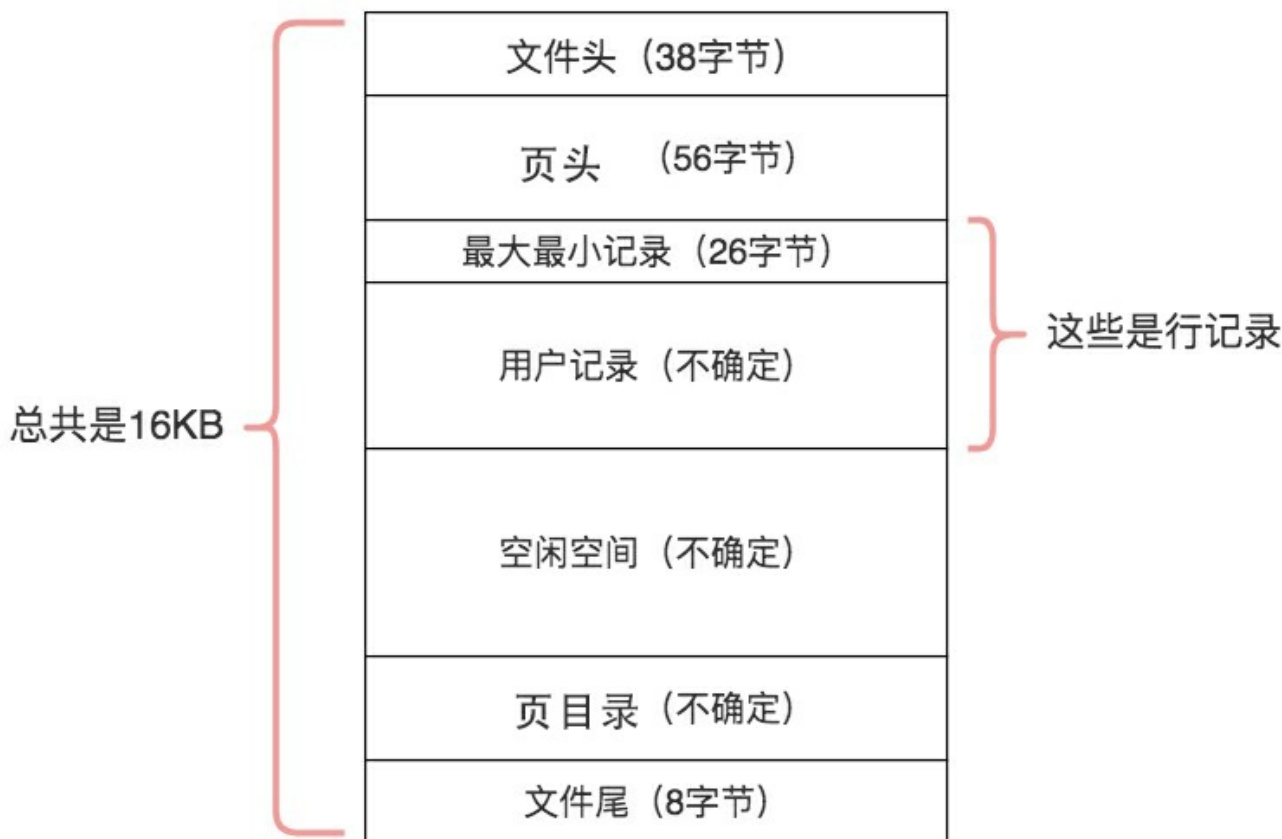
Variable_name	Value
innodb_page_size	16384

1 row in set, 1 warning (0.00 sec)

在SQL Server的页大小为8KB，而在Oracle中我们用术语“块”（Block）来代表“页”，Oracle支持的块大小为2KB，4KB，8KB，16KB，32KB和64KB。

数据库I/O操作的最小单位是页，与数据库相关的内容都会存储在页结构里。数据页包括七个部分，分别是文件头（File Header）、页头（Page Header）、最大最小记录（Infimum+supremum）、用户记录（User Records）、空闲空间（Free Space）、页目录（Page Directory）和文件尾（File Tailer）。

页结构的示意图如下所示：



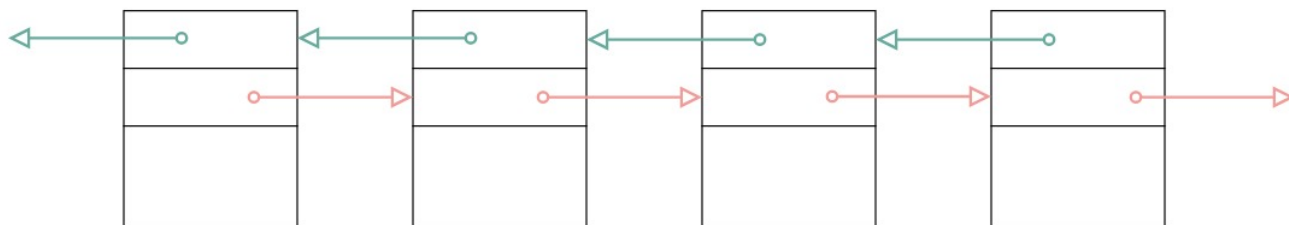
这7个部分到底有什么作用呢？我简单梳理下：

名称	占用大小	说明
File Header	38字节	文件头，描述页的信息
Page Header	56字节	页头，页的状态信息
Infimum + Supremum	26字节	最小和最大记录，这是两个虚拟的行记录
User Records	不确定	用户记录，存储行记录内容
Free Space	不确定	空闲空间，页中还没有被使用的空间
Page Directory	不确定	页目录，存储用户记录的相对位置
File Trailer	8字节	文件尾，校验页是否完整

实际上，我们可以把这7个数据页分成3个部分。

首先是文件通用部分，也就是文件头和文件尾。它们类似集装箱，将页的内容进行封装，通过文件头和文件尾校验的方式来确保页的传输是完整的。

在文件头中有两个字段，分别是FIL_PAGE_PREV和FIL_PAGE_NEXT，它们的作用相当于指针，分别指向上一个数据页和下一个数据页。连接起来的页相当于一个双向的链表，如下图所示：

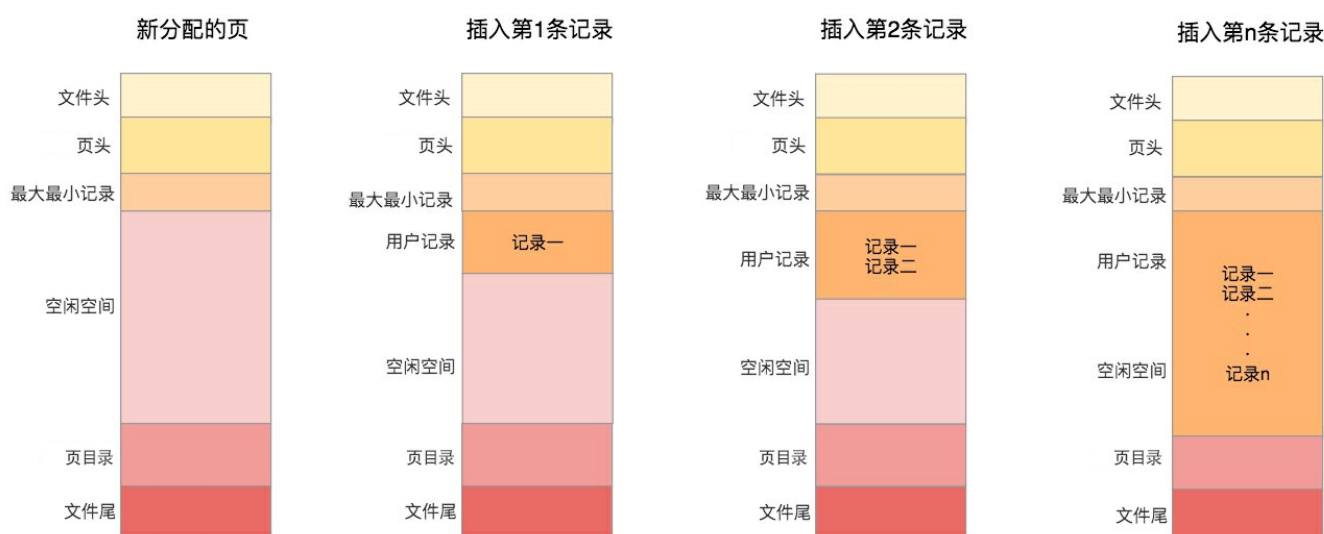


数据页之间组成的双向链表

需要说明的是采用链表的结构让数据页之间不需要是物理上的连续，而是逻辑上的连续。

我们之前讲到过Hash算法，这里文件尾的校验方式就是采用Hash算法进行校验。举个例子，当我们进行页传输的时候，如果突然断电了，造成了该页传输的不完整，这时通过文件尾的校验和（checksum值）与文件头的校验和做比对，如果两个值不相等则证明页的传输有问题，需要重新进行传输，否则认为页的传输已经完成。

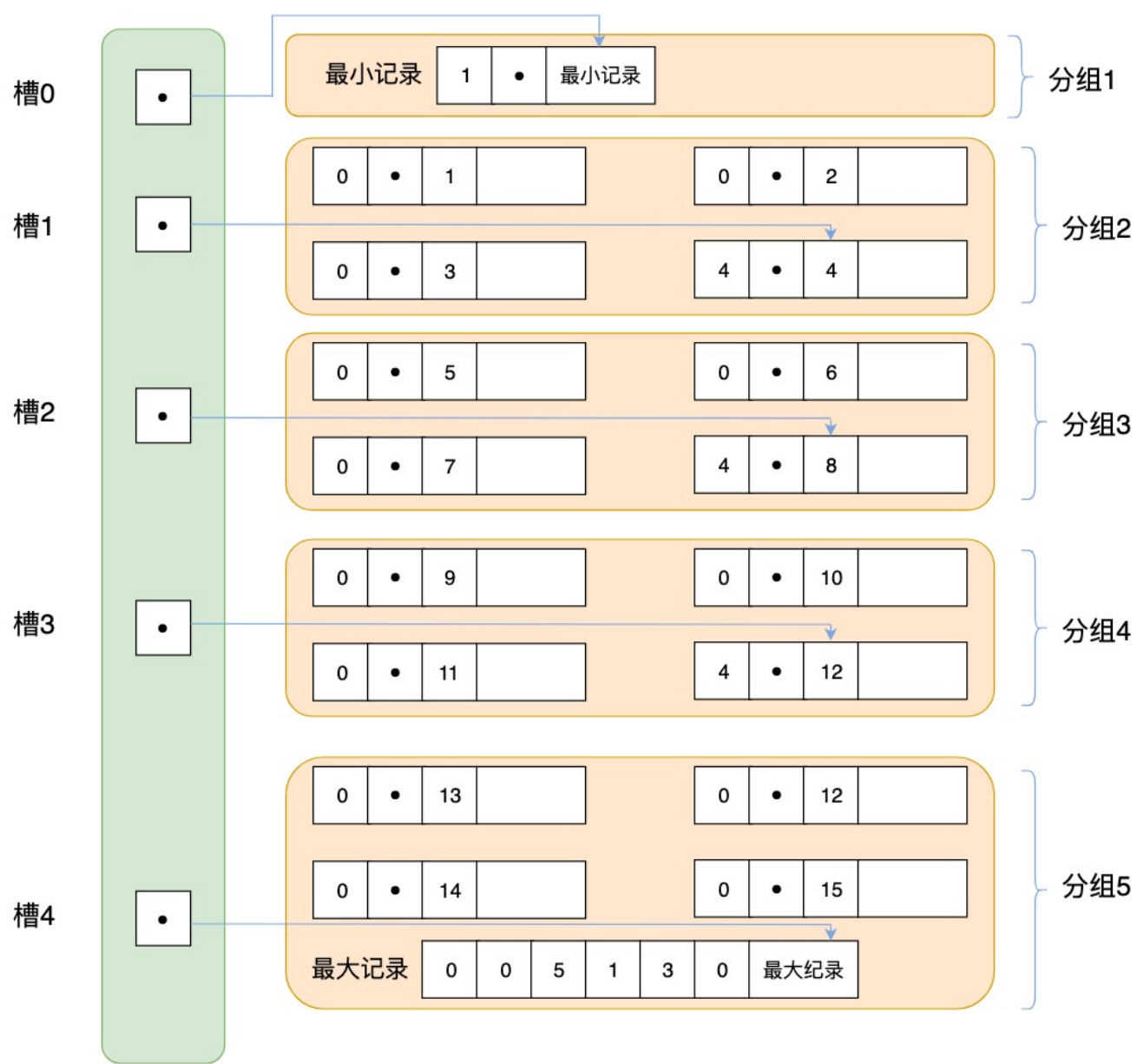
第二个部分是记录部分，页的主要作用是存储记录，所以“最小和最大记录”和“用户记录”部分占了页结构的主要空间。另外空闲空间是个灵活的部分，当有新的记录插入时，会从空闲空间中进行分配用于存储新记录，如下图所示：



第三部分是索引部分，这部分重点指的是页目录，它起到了记录的索引作用，因为在页中，记录是以单向链表的形式进行存储的。单向链表的特点就是插入、删除非常方便，但是检索效率不高，最差的情况下需要遍历链表上的所有节点才能完成检索，因此在页目录中提供了二分查找的方式，用来提高记录的检索效率。这个过程就好比是给记录创建了一个目录：

1. 将所有的记录分成几个组，这些记录包括最小记录和最大记录，但不包括标记为“已删除”的记录。
2. 第1组，也就是最小记录所在的分组只有1个记录；最后一组，就是最大记录所在的分组，会有1-8条记录；其余的组记录数量在4-8条之间。这样做的好处是，除了第1组（最小记录所在组）以外，其余组的记录数会尽量平分。
3. 在每个组中最后一条记录的头信息中会存储该组一共有多少条记录，作为n_owned字段。

4. 页目录用来存储每组最后一条记录的地址偏移量，这些地址偏移量会按照先后顺序存储起来，每组的地址偏移量也被称之为槽（slot），每个槽相当于指针指向了不同组的最后一个记录。如下图所示：



页目录存储的是槽，槽相当于分组记录的索引。我们通过槽查找记录，实际上就是在做二分查找。这里我以上面的图示进行举例，5个槽的编号分别为0，1，2，3，4，我想查找主键为9的用户记录，我们初始化查找的槽的下限编号，设置为low=0，然后设置查找的槽的上限编号high=4，然后采用二分查找法进行查找。

首先找到槽的中间位置 $p=(low+high)/2=(0+4)/2=2$ ，这时我们取编号为2的槽对应的分组记录中最大的记录，取出关键字为8。因为9大于8，所以应该会在槽编号为(p,high]的范围进行查找

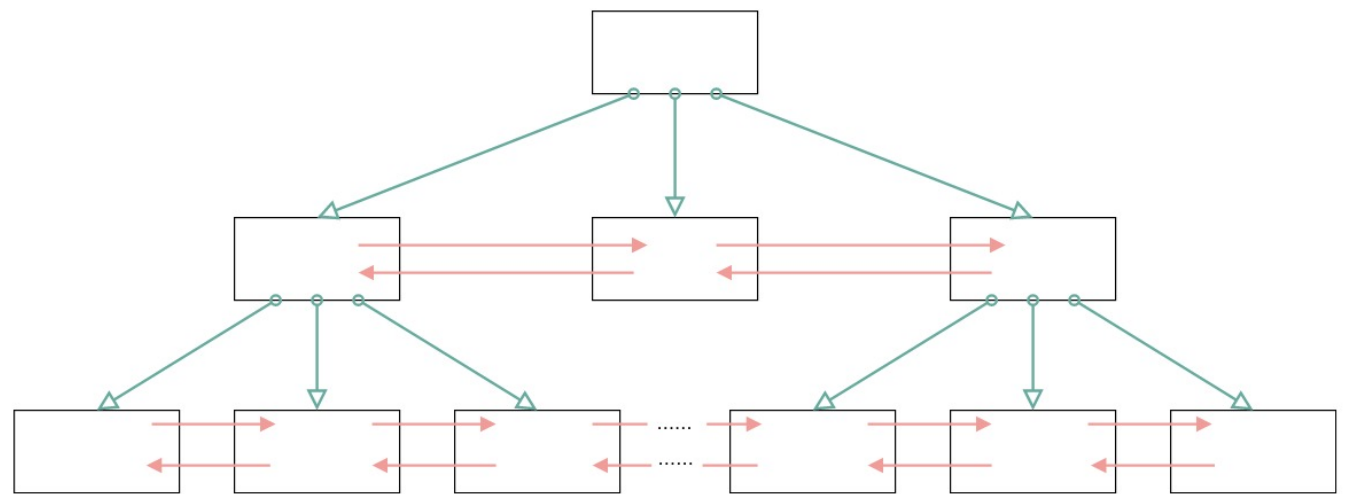
接着重新计算中间位置 $p'=(p+high)/2=(2+4)/2=3$ ，我们查找编号为3的槽对应的分组记录中最大的记录，取出关键字为12。因为9小于12，所以应该在槽3中进行查找。

遍历槽3中的所有记录，找到关键字为9的记录，取出该条记录的信息即为我们想要查找的内容。

从数据页的角度看B+树是如何进行查询的

MySQL的InnoDB存储引擎采用B+树作为索引，而索引又可以分成聚集索引和非聚集索引（二级索引），这些索引都相当于一棵B+树，如图所示。一棵B+树按照节点类型可以分成两部分：

- 1. 叶子节点，B+树最底层的节点，节点的高度为0，存储行记录。
- 2. 非叶子节点，节点的高度大于0，存储索引键和页面指针，并不存储行记录本身。



我们刚才学习了页结构的内容，你可以用页结构对比，看下B+树的结构。

在一棵B+树中，每个节点都是一个页，每次新建节点的时候，就会申请一个页空间。同一层上的节点之间，通过页的结构构成一个双向的链表（页文件头中的两个指针字段）。非叶子节点，包括了多个索引行，每个索引行里存储索引键和指向下一层页面的页面指针。最后是叶子节点，它存储了关键字和行记录，在节点内部（也就是页结构的内部）记录之间是一个单向的链表，但是对记录进行查找，则可以通过页目录采用二分查找的方式来进行。

当我们从页结构来理解B+树的结构的时候，可以帮我们理解一些通过索引进行检索的原理：

1.B+树是如何进行记录检索的？

如果通过B+树的索引查询行记录，首先是从B+树的根开始，逐层检索，直到找到叶子节点，也就是找到对应的数据页为止，将数据页加载到内存中，页目录中的槽（slot）采用二分查找的方式先找到一个粗略的记录分组，然后再在分组中通过链表遍历的方式查找记录。

2.普通索引和唯一索引在查询效率上有什么不同？

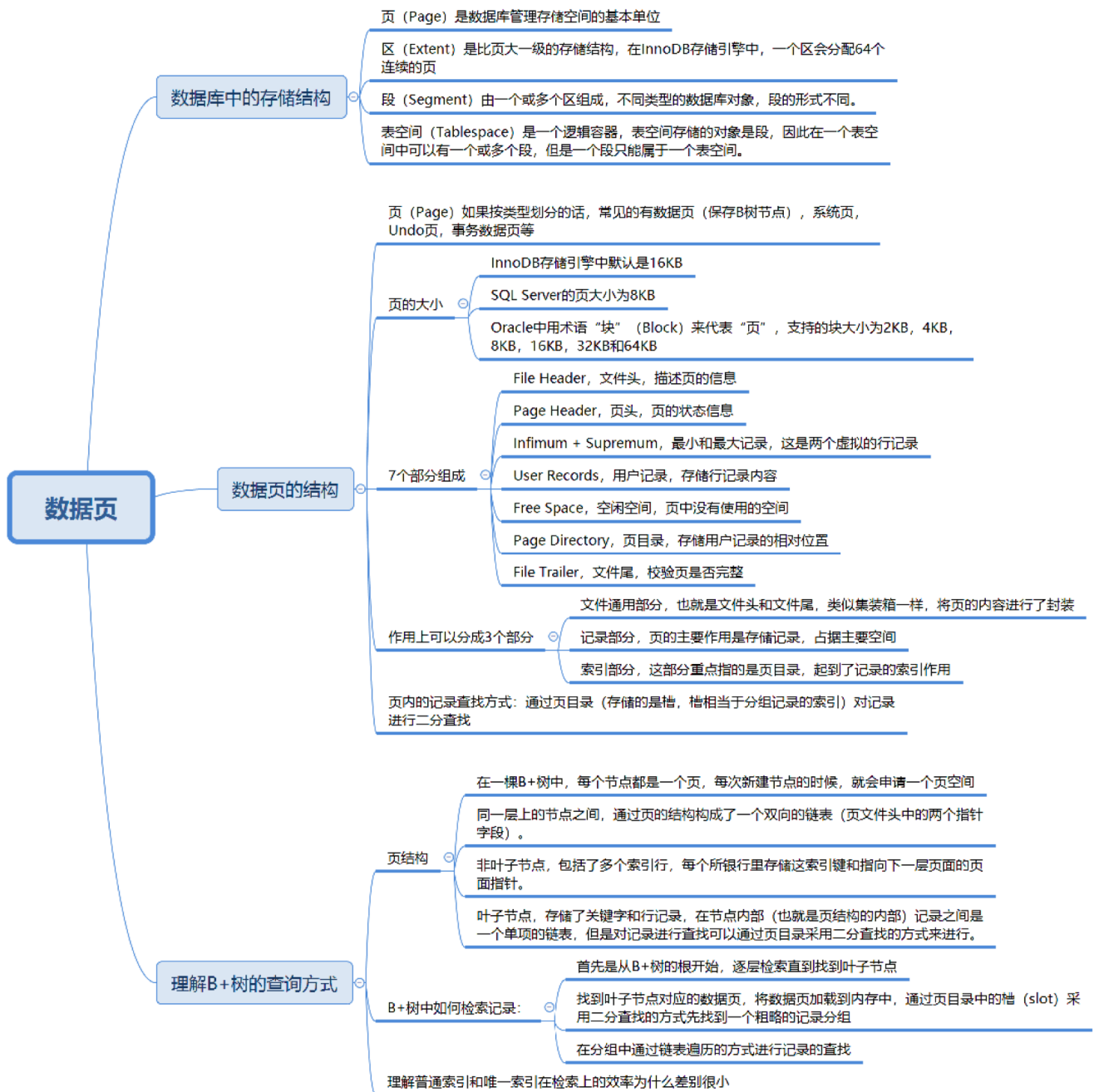
我们创建索引的时候可以是普通索引，也可以是唯一索引，那么这两个索引在查询效率上有什么不同呢？

唯一索引就是在普通索引上增加了约束性，也就是关键字唯一，找到了关键字就停止检索。而普通索引，可能会存在用户记录中的关键字相同的情况，根据页结构的原理，当我们读取一条记录的时候，不是单独将这条记录从磁盘中读出去，而是将这个记录所在的页加载到内存中进行读取。InnoDB存储引擎的页大小为**16KB**，在一个页中可能存储着上千个记录，因此在普通索引的字段上进行查找也就是在内存中多几次“判断下一条记录”的操作，对于**CPU**来说，这些操作所消耗的时间是可以忽略不计的。所以对一个索引字段进行检索，采用普通索引还是唯一索引在检索效率上基本上没有差别。

总结

今天我们学习了数据库中的基本存储单位，也就是页（**Page**），磁盘**I/O**都是基于页来进行读取的，在页之上还有区、段和表空间，它们都是更大的存储单位。我们在分配空间的时候会按照页为单位来进行分配，同一棵树上同一层的页与页之间采用双向链表，而在页里面，记录之间采用的单向链表的方式。

链表这种数据结构的特点是增加、删除比较方便，所以在对记录进行删除的时候，有时候并不是真的删除了记录，而只是逻辑上的删除，也就是在标记为上标记为“已删除”。但链表还有个问题就是查找效率低，因此在页结构中还专门设计了页目录这个模块，专门给记录做一个目录，通过二分查找法的方式进行检索提升效率。



今天的内容到这里就结束了, 最后我给你留两道思考题吧。按照聚集索引存储的行记录在物理上连续的, 还是逻辑上连续的? 另外, 通过B+树进行记录的检索流程是怎样的?

欢迎你在评论区写下你的思考, 我会和你一起交流, 也欢迎把这篇文章分享给你的朋友或者同事, 一起来交流。

SQL 必知必会

从入门到数据实战

陈旸

清华大学计算机博士



新版升级：点击「👤请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言



Monday

👍 6

关于数据页有几个疑问：

- 1、分组1为什么只有一条记录？
- 2、分组之间采用的是二分查找，表示分组1，分组2，...，分组n之间是通过某种顺序进行排列的。假设除了最后一个分组n外，所有的分组都存储满了8条记录，这时新增一条记录x按有序性应该放置在分组2中，是不是分组2，3，分组n-1都需要进行记录的改动？
- 3、最大最小记录是专门给数据页内二分查找赋值low与high设计的？

2019-08-12



我行我素

👍 6

- 1.逻辑上连续的，在文中有说采用链表的结构让数据页之间不需要是物理上的连续，而是逻辑上的连续；
- 2.二分查找法，在总结图中有详细的说明，首先是从B+树的根开始，逐层检索直到叶子节点，找到叶子节点对应的数据页，将数据页加载到内存中，通过页目录中的槽采用二分查找的方式先找到一个粗略的记录分组，在分组中通过链表遍历的方式进行记录的查找

2019-08-12



Monday

👍 4

Mysql如text、MediumText等类型最大存储长度为64k、16M，已经超过了页16k，请问包含这几种类型的记录都怎么存储的？

2019-08-12



老毕

👍 2

逻辑上连续，物理上未必连续。数据库的最小分配单位是页，连续分配一定数量的物理页构成段。每页存储只够存储有限的记录，记录不断增加，一定会超出页，继而超出段，无法保证物理连续。进一步说，数据库的磁盘分配操作基于操作系统，单个“页”是否一定在物理介质上连续，也不一定。

B+树只有叶节点含有数据指针，总是从根节点开始检索，直到叶节点。如果是范围检索，只需要一次根到叶的检索，然后就能借助叶节点的双向链表结构特性，顺序移动直到找出所有满足范围的数据。

2019-08-13



往事随风，顺其自然

👍 1

逻辑上连续的，记录之间采用单链表方式实现

2019-08-12



小智e

👍 0

请问一下老师，这些内容的源头是在哪里呢？想去源头看看

2019-08-23



习惯沉淀

👍 0

老师，“在B+树中，每一个节点都是一个页”这句，有点不太理解，非叶子节点不是不存放记录吗，非叶子节点就是页，只不过页里面不存放数据是这意思吗？

2019-08-20



asdf100

👍 0

对页里查找记录时，可先根据二分查找法，先找到记录所在槽，时间复杂度为 $O(\log_2 n)$ 。槽内有多条记录，再遍历所有记录，直到找到指定记录为止，时间复杂度为 $O(n)$ 。

2019-08-13



asdf100

👍 0

遍历槽 3 中的所有记录，找到关键字为 9 的记录，取出该条记录的信息即为我们想要查找的内容。由于单链表，所以查询复杂度为 $O(n)$

2019-08-13



阿恺

👍 0

关于普通索引和唯一索引的效率问题还想请教一下：普通索引不是主键，不是物理索引，叶子节点记录的是行的位置，实际可能分散在多个页中，那就存在多次IO读取，效率会降低，为什么说差不多？

2019-08-13



wusiration

👍 0

问题1：聚集索引是在物理上连续，因为聚集索引是指表中数据行按索引的排序方式进行存储，表设置了聚集索引，数据行会按照索引列的值在磁盘上物理存储和排序；（摘抄自23索引的概

览)

问题2: 先是从**B+**树的根开始, 逐层检索找到相应叶子节点对应的数据页, 将数据页加载到内存中, 通过页目录中的槽采用二分查找的方式先找到一个粗略的记录分组, 在分组中通过链表遍历的方式进行记录的查找

2019-08-12



星星

0

老师, 怎么理解**B+**树里面的关键字? 是不是可以理解为就是索引的编号呢。。

2019-08-12



往事随风, 顺其自然

0

查找页目录的时候, 需要遍历所有的数据页目录? 还是说创建索引后这些页目录已经排好序了

2019-08-12



Ant

0

普通索引和唯一索引在查找中查询效率差别不大, 除非是两条记录刚好分到了两个不同的页中, 但是插入时这个就有较大区别了, 这个会在下节课中讲吗?

2019-08-12



许童童

0

按照聚集索引存储的行记录在物理上连续的, 还是逻辑上连续的?

逻辑上连续, 因为链表没有强制要求使用连续的物理空间, 当然如果物理连续, 配合局部性原理, 性能会更好。

另外, 通过 **B+** 树进行记录的检索流程是怎样的?

先从根节点开始通过二分查找, 确定到要找的页节点或叶子节点, 如果是页节点就继续, 直到找到叶子节点, 到叶子节点, 再通过页目录, 也是二分查找到对应的组, 再通过单链表遍历找到相应的数据。

2019-08-12



遁地之鹅

0

getpagesize

2019-08-12