1. InnoDB索引页
2. 概述

页是InnoDB管理存储空间的基本单位，InnoDB为了不同的目的设计了不同类型的页，下面介绍存放表中记录的页，称为索引页。

1. 索引页结构

索引页可以划分为多个部分，有的部分占用的字节数是确定的，有的部分占用的字节数是不确定的。

1. File Header

文件头部， 38字节， 页的一些通用信息。

1. Page Header

页面头部，56字节，索引页专有的信息。

1. Infimum + Supremum

页面中的最小记录和最大记录，26字节。

1. User Records

用户记录，不确定字节数，用户存储的记录内容。

1. Free Space

空闲空间，不确定字节数，页中尚未使用的空间。

1. Page Directory

页目录，不确定字节数，页中某些记录的相对位置。

1. File Trailer

文件尾部，8个字节，校验页是否完整。

1. 记录的存储
2. 用户记录

自己存储的记录会按照指定的行格式存储到User Records部分，在一开始生成页的时候，并没有User Records部分，每当插入一条记录时，都会从Free Space部分申请一个记录大小的空间划分到User Records部分。

1. 记录头信息

之前我们介绍了记录中额外的头信息，但是没有详细地介绍，下面具体地讲一讲。

1. deleted\_flag

这个属性用来标记当前记录是否被删除，占用1bit，值为1时表示记录被删除了。其实记录依然存在于磁盘上，只是逻辑上被删除了，所有被删除掉的记录都会组成一个垃圾链表，记录在这个链表中占用的空间称为可重用空间，之后若有新的记录插入到表中，可以覆盖掉这些记录空间。

1. heap\_no

索引页中的记录的在堆中的相对位置，在页前面的记录的heap\_no比较小，在页后面的记录的heap\_no比较大。

1. 伪记录

InnoDB自动给每个页中添加了两条记录，称为伪记录，一条代表页面中的最小记录（Infimum记录），一条代表页面中的最大记录（Supremum记录）。记录也是可以比较大小的，通过比较记录的主键，InnoDB指定任何用户记录都比Infimum记录大，比Supremum记录小。这两个记录不放在User Records部分，而是单独放在Infimum + Supremum部分中。

1. record\_type

当前记录的类型，共有4种类型的记录，0表示普通记录，1表示B+树非叶节点的目录项纪录，2表示Infimum记录，3表示Supremum记录。

1. next\_record

表示从当前记录的真实数据到下一条记录的真实数据的距离，如果是正数，说明下一条记录在后面，否则在前面。这表明这个结构是链表，下一条记录指的是按照主键值由小到大排序的下一条记录。因此Infimum记录的下一条记录就是本页中主键值最小的用户记录，本业中主键值最大的用户记录的下一个记录就是Supremum记录。

1. 页目录

既然我们已经知道了记录在页中是按照主键值由小到大的顺序串联成一个单向链表，那么如何快速地在页中查找记录呢？

类似目录的想法，提出了页目录。

将所有正常的记录（包括Infimum和Supremum记录，不包括已经删除的记录）划分为几个组，每个组的最后一条记录（组内最大）的头信息中的n\_owned属性表示该组内共有几条记录，将该记录在页面中的地址偏移量（真实数据与第0个字节的距离）提取出来，按照顺序存储到靠近页尾部的地方，这就是页目录，页目录中的这些地址偏移量称为槽，每个槽占用两个字节。

由于记录是按照主键值从大到小排序的，因此可以使用二分法查找，通过在页目录中进行二分法查询，按照主键值查询，找到该记录所在的分组，然后该组的头记录开始通过next\_record遍历查找即可，每个分组记录条数最多8条，因此代价很小。

那么划分分组的依据是什么呢？

InnoDB规定了对于Infimum记录所在的分组只能有一条记录，Supremum记录所在的分组拥有的记录数在1-8之间，剩下的分组在4-8之间。初始情况下，一个数据页只有Infimum记录和Supremum记录，它们分属于两个组，也分属于两个槽，之后每插入一条记录，都会从页目录中找到对应记录的主键值比待插入记录的主键值要大且差值最小的槽，然后把该槽对应的记录的n\_owned值+1，表示本组中又添加了一条记录，直到到达上限8条。之后会将组中的记录拆分成两个组，新增一个槽。

1. 页面头部

索引页中记录的状态信息，占用固定的56个字节。

1. 文件头部

通用于各种类型的页，描述了一些通用的信息，如页面的编号，上一个页和下一个页，还有当前页面的校验和，用于校验页的完整性，占用固定的38个字节。

1. 文件尾部

InnoDB会将页在磁盘和内存中不断刷新，如果修改后的页数据在刷新过程中意外中断了，如何判断页面数据是否刷新成功？

通过文件头部和文件尾部的校验和，文件尾部由8个字节组成，前4个字节代表页的校验和， 对应于文件头部的校验和，每当一个页面在内存中发生修改时，在刷新之前就要把页面的校验和计算出来，因为文件头部在页面的前面，因此文件头部的校验和会首先刷新到磁盘上，当页面完全写完后，校验和也会写到页的尾部，如果页面刷新成功，页首和页尾的校验和应该是一致的，如果意外终止，则头尾校验和不一致，说明刷新发生了错误。