Mysql索引主要使用的两种数据结构

哈希索引

对于哈希索引来说，底层的数据结构就是哈希表，因此在绝大多数需求为单条记录查询的时候，可以选择哈希索引，查询性能最快；其余大部分场景，建议选择BTree索引。

BTree索引

覆盖索引介绍

什么是覆盖索引

如果一个索引包含（或者说覆盖）所有需要查询的字段的值，我们就称之为“覆盖索引”。我们知道InnoDB存储引擎中，如果不是主键索引，叶子节点存储的是主键+列值。最终还是要“回表”，也就是要通过主键再查找一次。这样就会比较慢覆盖索引就是把要查询出的列和索引是对应的，不做回表操作！

覆盖索引使用实例

现在我创建了索引(username,age)，我们执行下面的 sql 语句

```
select username , age from user where username = 'Java' and age = 22
```

在查询数据的时候：要查询出的列在叶子节点都存在！所以，就不用回表。

选择索引和编写利用这些索引的查询的3个原则

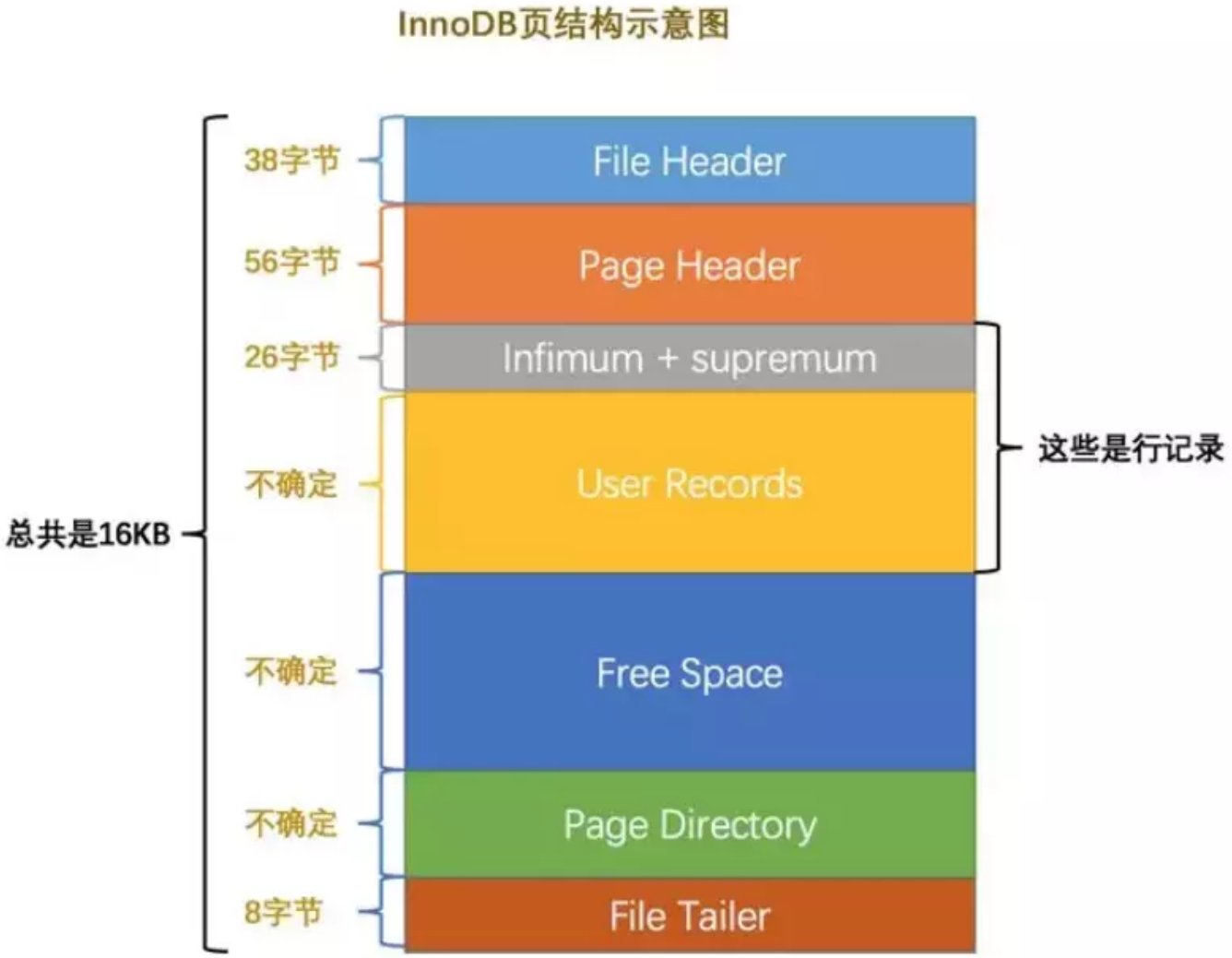
1. 单行访问是很慢的。特别是在机械硬盘存储中(SSD的随机I/O要快很多，不过这一点仍然成立）。如果服务器从存储中读取一个数据块只是为了获取其中一行，那么就浪费了很多工作。最好读取的块中能包含尽可能多所需要的行。使用索引可以创建位置引，用以提升效率。
2. 按顺序访问范围数据是很快的，这有两个原因。第一，顺序 I/O 不需要多次磁盘寻道，所以比随机I/O要快很多（特别是对机械硬盘）。第二，如果服务器能够按需要顺序读取数据，那么就不再需要额外的排序操作，并且GROUPBY查询也无须再做排序和将行按组进行聚合计算了。
3. 索引覆盖查询是很快的。如果一个索引包含了查询需要的所有列，那么存储引擎就 不需要再回表查找行。这避免了大量的单行访问，而上面的第1点已经写明单行访 问是很慢的。

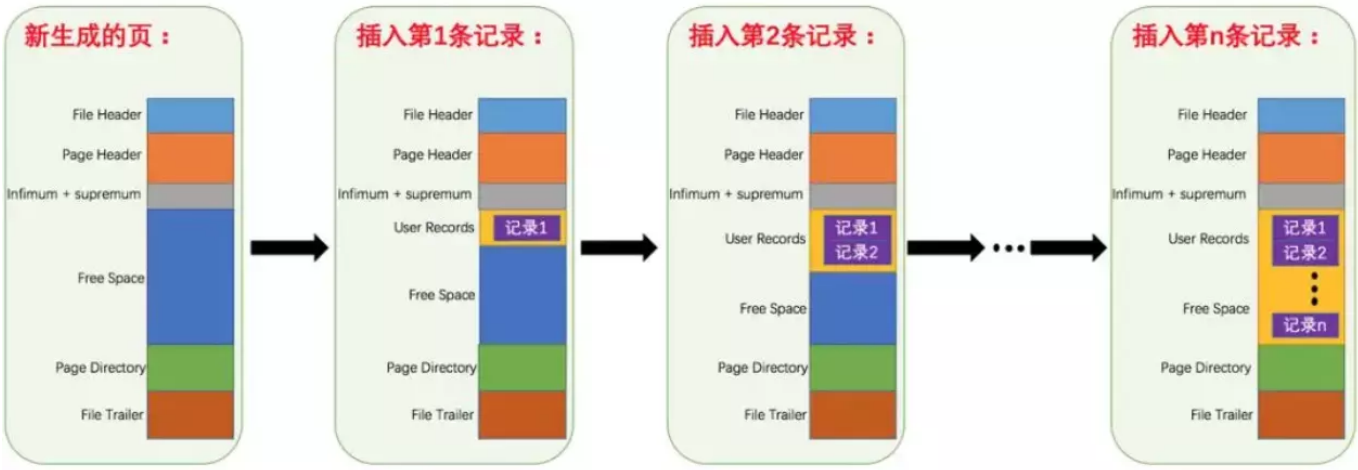
为什么索引能提高查询速度

> 以下内容整理自： 地址： https://juejin.im/post/5b55b842f265da0f9e589e79 作者 ： Java3y

先从 MySQL 的基本存储结构说起

MySQL的基本存储结构是页(记录都存在页里边)：





- **各个数据页可以组成一个双向链表**
- **每个数据页中的记录又可以组成一个单向链表**
  - 每个数据页都会为存储在它里边儿的记录生成一个页目录，在通过主键查找某条记录的时候可以在页目录中使用二分法快速定位到对应的槽，然后再遍历该槽对应分组中的记录即可快速找到指定的记录
  - 以其他列(非主键)作为搜索条件：只能从最小记录开始依次遍历单链表中的每条记录。

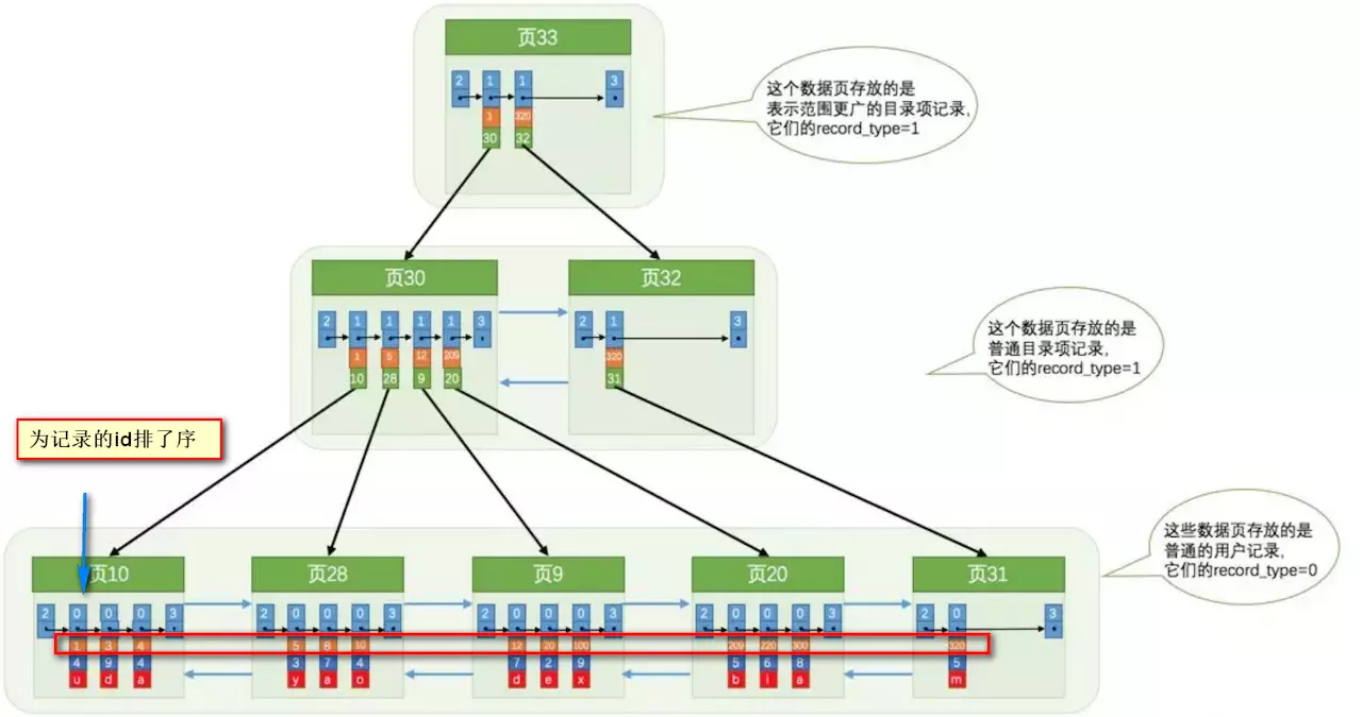
所以说，如果我们写select * from user where indexname = 'xxx'这样没有进行任何优化的sql语句，默认会这样做：

1. **定位到记录所在的页：需要遍历双向链表，找到所在的页**
2. **从所在的页内中查找相应的记录：由于不是根据主键查询，只能遍历所在页的单链表了**

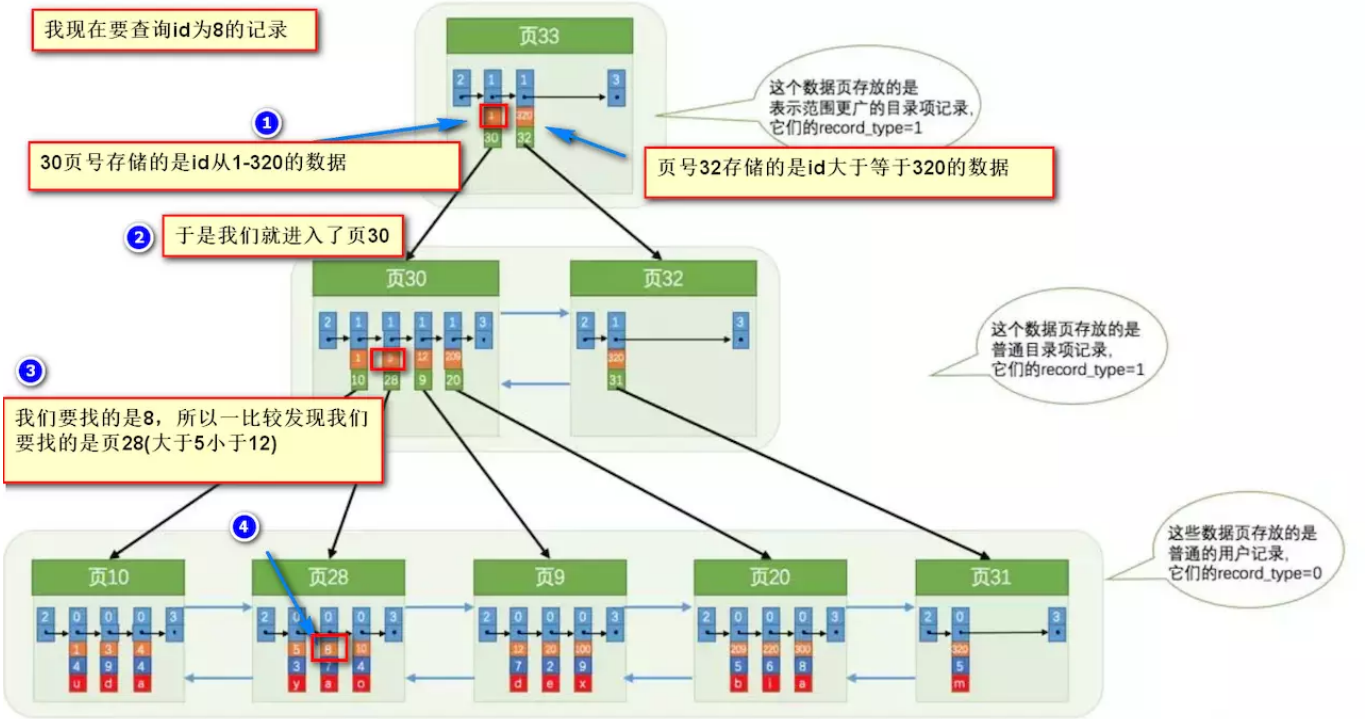
很明显，在数据量很大的情况下这样查找会很慢！这样的时间复杂度为O（n）。

使用索引之后

索引做了些什么可以让我们查询加快速度呢？其实就是将无序的数据变成有序(相对)：



要找到id为8的记录简要步骤：



很明显的是：没有用索引我们是需要遍历双向链表来定位对应的页，现在通过**“目录”**就可以很快地定位到对应的页上了！（二分查找，时间复杂度近似为O(logn)）

其实底层结构就是B+树，B+树作为树的一种实现，能够让我们很快地查找出对应的记录。

关于索引其他重要的内容补充

> 以下内容整理自：《Java工程师修炼之道》

最左前缀原则

MySQL中的索引可以以一定顺序引用多列，这种索引叫作联合索引。如User表的name和city加联合索引就是(name,city)，而最左前缀原则指的是，如果查询的时候查询条件精确匹配索引的左边连续一列或几列，则此列就可以被用到。如下：

```
select * from user where name=xx and city=xx ;  / / 可以命中索引
select * from user where name=xx ; // 可以命中索引
select * from user where city=xx ; // 无法命中索引
```

这里需要注意的是，查询的时候如果两个条件都用上了，但是顺序不同，如 `city= xx and name =xx`，那么现在的查询引擎会自动优化为匹配联合索引的顺序，这样是能够命中索引的。

由于最左前缀原则，在创建联合索引时，索引字段的顺序需要考虑字段值去重之后的个数，较多的放前面。ORDER BY子句也遵循此规则。

注意避免冗余索引

冗余索引指的是索引的功能相同，能够命中索引(a, b)就肯定能命中索引(a) ，那么索引(a)就是冗余索引。如（name,city ）和（name ）这两个索引就是冗余索引，能够命中前者的查询肯定是能够命中后者的 在大多数情况下，都应该尽量扩展已有的索引而不是创建新索引。

MySQL 5.7 版本后，可以通过查询 sys 库的 `schema_redundant_indexes` 表来查看冗余索引