# Mysql面试题

## 一、为什么用自增列作为主键

1．如果我们定义了主键(PRIMARY KEY)，那么InnoDB会选择主键作为聚集索引。

如果没有显式定义主键，则InnoDB会选择第一个不包含有NULL值的唯一索引作为主键索引。

如果也没有这样的唯一索引，则InnoDB会选择内置6字节长的ROWID作为隐含的聚集索引(ROWID随着行记录的写入而主键递增，这个ROWID不像ORACLE的ROWID那样可引用，是隐含的)。

2．数据记录本身被存于主索引（一颗B+Tree）的叶子节点上，这就要求同一个叶子节点内（大小为一个内存页或磁盘页）的各条数据记录按主键顺序存放。

因此每当有一条新的记录插入时，MySQL会根据其主键将其插入适当的节点和位置，如果页面达到装载因子（InnoDB默认为15/16），则开辟一个新的页（节点）。

3．如果表使用自增主键，那么每次插入新的记录，记录就会顺序添加到当前索引节点的后续位置，当一页写满，就会自动开辟一个新的页。

4．如果使用非自增主键（如果身份证号或学号等），由于每次插入主键的值近似于随机，因此每次新纪录都要被插到现有索引页得中间某个位置。此时MySQL不得不为了将新记录插到合适位置而移动数据，甚至目标页面可能已经被回写到磁盘上而从缓存中清掉，此时又要从磁盘上读回来，这增加了很多开销，同时频繁的移动、分页操作造成了大量的碎片，得到了不够紧凑的索引结构，后续不得不通过OPTIMIZE TABLE来重建表并优化填充页面。

## 二、为什么使用数据索引能提高效率

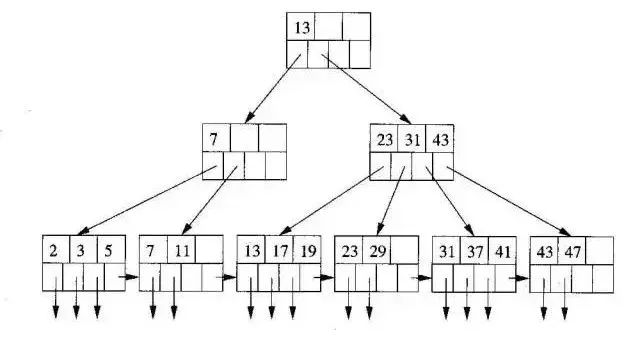
数据索引的存储是有序的

在有序的情况下，通过索引查询一个数据是无需遍历索引记录的

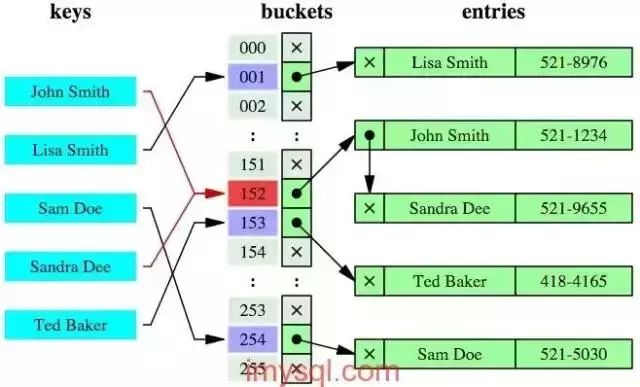
极端情况下，数据索引的查询效率为二分法查询效率，趋近于log2(N)。

## 三、B+树索引和哈希索引的区别

**B+树**是一个平衡的多叉树，从根节点到每个叶子节点的高度差值不超过1，而且同层级的节点间有指针相互链接，是有序的，如下图：



**哈希索引**就是采用一定的哈希算法，把键值换算成新的哈希值，检索时不需要类似B+树那样从根节点到叶子节点逐级查找，只需一次哈希算法即可，是无序的，如下图所示：



## 四、哈希索引的优势：

等值查询，哈希索引具有绝对优势（前提是：没有大量重复键值，如果大量重复键值时，哈希索引的效率很低，因为存在所谓的哈希碰撞问题。）

## 五、哈希索引不适用的场景：

不支持范围查询

不支持索引完成排序

不支持联合索引的最左前缀匹配规则

通常，B+树索引结构适用于绝大多数场景，像下面这种场景用哈希索引才更有优势：

在HEAP表中，如果存储的数据重复度很低（也就是说基数很大），对该列数据以等值查询为主，没有范围查询、没有排序的时候，特别适合采用哈希索引，例如这种SQL：

# 仅等值查询

select id, name from table where name='李明';

而常用的 InnoDB 引擎中默认使用的是B+树索引，它会实时监控表上索引的使用情况。

如果认为建立哈希索引可以提高查询效率，则自动在内存中的“自适应哈希索引缓冲区”建立哈希索引（在InnoDB中默认开启自适应哈希索引）。

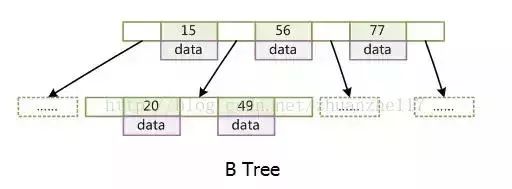
通过观察搜索模式，MySQL会利用index key的前缀建立哈希索引，如果一个表几乎大部分都在缓冲池中，那么建立一个哈希索引能够加快等值查询。

注意：在某些工作负载下，通过哈希索引查找带来的性能提升远大于额外的监控索引搜索情况和保持这个哈希表结构所带来的开销。

但某些时候，在负载高的情况下，自适应哈希索引中添加的read/write锁也会带来竞争，比如高并发的join操作。like操作和%的通配符操作也不适用于自适应哈希索引，可能要关闭自适应哈希索引。

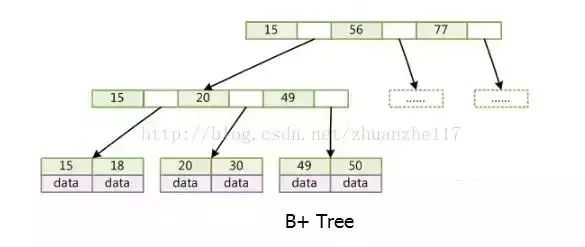
## B树和B+树的区别

1. B树，每个节点都存储key和data，所有节点组成这棵树，并且叶子节点指针为nul，叶子结点不包含任何关键字信息。



1. B+树，所有的叶子结点中包含了全部关键字的信息，及指向含有这些关键字记录的指针，且叶子结点本身依关键字的大小自小而大的顺序链接。

所有的非终端结点可以看成是索引部分，结点中仅含有其子树根结点中最大（或最小）关键字。 (而B 树的非终节点也包含需要查找的有效信息)



## 七、为什么说B+比B树更适合实际应用中操作系统的文件索引和数据库索引？

1．B+的磁盘读写代价更低。

B+的内部结点并没有指向关键字具体信息的指针，因此其内部结点相对B树更小。

如果把所有同一内部结点的关键字存放在同一盘块中，那么盘块所能容纳的关键字数量也越多。一次性读入内存中的需要查找的关键字也就越多。相对来说IO读写次数也就降低了。

2．B+-tree的查询效率更加稳定。

由于非终结点并不是最终指向文件内容的结点，而只是叶子结点中关键字的索引。所以任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字查询的路径长度相同，导致每一个数据的查询效率相当。

## 八、MySQL联合索引

1．联合索引是两个或更多个列上的索引。

对于联合索引:Mysql从左到右的使用索引中的字段，一个查询可以只使用索引中的一部份，但只能是最左侧部分。

例如索引是key index (a,b,c). 可以支持a 、 a,b 、 a,b,c 3种组合进行查找，但不支持 b,c进行查找 .当最左侧字段是常量引用时，索引就十分有效。

2．利用索引中的附加列，您可以缩小搜索的范围，但使用一个具有两列的索引不同于使用两个单独的索引。

复合索引的结构与电话簿类似，人名由姓和名构成，电话簿首先按姓氏对进行排序，然后按名字对有相同姓氏的人进行排序。

如果您知道姓，电话簿将非常有用；如果您知道姓和名，电话簿则更为有用，但如果您只知道名不知道姓，电话簿将没有用处。

## 九、什么情况下应不建或少建索引

1．表记录太少；

2．经常插入、删除、修改的表；

3．数据重复且分布平均的表字段，假如一个表有10万行记录，有一个字段A只有T和F两种值，且每个值的分布概率大约为50%，那么对这种表A字段建索引一般不会提高数据库的查询速度；

4．经常和主字段一块查询但主字段索引值比较多的表字段。

## 十、什么是表分区？

表分区，是指根据一定规则，将数据库中的一张表分解成多个更小的，容易管理的部分。从逻辑上看，只有一张表，但是底层却是由多个物理分区组成。

## 十一、表分区与分表的区别

**分表**：指的是通过一定规则，将一张表分解成多张不同的表。比如将用户订单记录根据时间成多个表。

**分表与分区的区别在于**：分区从逻辑上来讲只有一张表，而分表则是将一张表分解成多张表。

## 十二、表分区有什么好处？

1、**存储更多数据**。分区表的数据可以分布在不同的物理设备上，从而高效地利用多个硬件设备。和单个磁盘或者文件系统相比，可以存储更多数据

2、**优化查询**。在where语句中包含分区条件时，可以只扫描一个或多个分区表来提高查询效率；涉及sum和count语句时，也可以在多个分区上并行处理，最后汇总结果。

3、**分区表更容易维护**。例如：想批量删除大量数据可以清除整个分区。

4、**避免某些特殊的瓶颈**，例如InnoDB的单个索引的互斥访问，ext3问价你系统的inode锁竞争等。

## 十三、分区表的限制因素

1. 一个表最多只能有1024个分区
2. MySQL5.1中，分区表达式必须是整数，或者返回整数的表达式。在MySQL5.5中提供了非整数表达式分区的支持。
3. 如果分区字段中有主键或者唯一索引的列，那么多有主键列和唯一索引列都必须包含进来。即：分区字段要么不包含主键或者索引列，要么包含全部主键和索引列。
4. 分区表中无法使用外键约束
5. MySQL的分区适用于一个表的所有数据和索引，不能只对表数据分区而不对索引分区，也不能只对索引分区而不对表分区，也不能只对表的一部分数据分区。

## 十四、如何判断当前MySQL是否支持分区？

命令：show variables like '%partition%' 运行结果:

have\_partintioning 的值为YES，表示支持分区。

## 十五、MySQL支持的分区类型有哪些？

* **RANGE分区**： 这种模式允许将数据划分不同范围。例如可以将一个表通过年份划分成若干个分区
* **LIST分区**： 这种模式允许系统通过预定义的列表的值来对数据进行分割。按照List中的值分区，与RANGE的区别是，range分区的区间范围值是连续的。
* **HASH分区** ：这中模式允许通过对表的一个或多个列的Hash Key进行计算，最后通过这个Hash码不同数值对应的数据区域进行分区。例如可以建立一个对表主键进行分区的表。
* **KEY分区**：上面Hash模式的一种延伸，这里的Hash Key是MySQL系统产生的。

## 十六、怎么解决事务问题？MySQL的事务隔离级别了解吗？

MySQL的四种隔离级别如下：

**未提交读(READ UNCOMMITTED)**

这就是上面所说的例外情况了，这个隔离级别下,其他事务可以看到本事务没有提交的部分修改。因此会造成脏读的问题(读取到了其他事务未提交的部分，而之后该事务进行了回滚)。

这个级别的性能没有足够大的优势，但是又有很多的问题，因此很少使用。

**已提交读(READ COMMITTED)**

其他事务只能读取到本事务已经提交的部分。这个隔离级别有不可重复读的问题，在同一个事务内的两次读取，拿到的结果竟然不一样，因为另外一个事务对数据进行了修改。

**REPEATABLE READ(可重复读)**

可重复读隔离级别解决了上面不可重复读的问题(看名字也知道)，但是仍然有一个新问题,就是幻读,当你读取id> 10 的数据行时，对涉及到的所有行加上了读锁,此时例外一个事务新插入了一条id=11的数据，因为是新插入的，所以不会触发上面的锁的排斥，那么进行本事务进行下一次的查询时会发现有一条id=11的数据，而上次的查询操作并没有获取到，再进行插入就会有主键冲突的问题。

**SERIALIZABLE(可串行化)**

这是最高的隔离级别，可以解决上面提到的所有问题，因为他强制将所以的操作串行执行，这会导致并发性能极速下降，因此也不是很常用。

## 十七、关于MVCC

MySQL InnoDB存储引擎，实现的是基于多版本的并发控制协议——MVCC (Multi-Version Concurrency Control)

**注**：与MVCC相对的，是基于锁的并发控制，Lock-Based Concurrency Control

**MVCC最大的好处**：读不加锁，读写不冲突。在读多写少的OLTP应用中，读写不冲突是非常重要的，极大的增加了系统的并发性能，现阶段几乎所有的RDBMS，都支持了MVCC。

* LBCC：Lock-Based Concurrency Control，基于锁的并发控制
* MVCC：Multi-Version Concurrency Control

基于多版本的并发控制协议。纯粹基于锁的并发机制并发量低，MVCC是在基于锁的并发控制上的改进，主要是在读操作上提高了并发量。

## 十八、在MVCC并发控制中，读操作可以分成两类：

* **快照读 (snapshot read)**：读取的是记录的可见版本 (有可能是历史版本)，不用加锁（共享读锁s锁也不加，所以不会阻塞其他事务的写）
* **当前读 (current read)**：读取的是记录的最新版本，并且，当前读返回的记录，都会加上锁，保证其他事务不会再并发修改这条记录

## 十九、行级锁定的优点：

* 当在许多线程中访问不同的行时只存在少量锁定冲突。
* 回滚时只有少量的更改
* 可以长时间锁定单一的行。

## 二十、行级锁定的缺点：

* 比页级或表级锁定占用更多的内存。
* 当在表的大部分中使用时，比页级或表级锁定速度慢，因为你必须获取更多的锁。
* 如果你在大部分数据上经常进行GROUP BY操作或者必须经常扫描整个表，比其它锁定明显慢很多。
* 用高级别锁定，通过支持不同的类型锁定，你也可以很容易地调节应用程序，因为其锁成本小于行级锁定。

## 二十一、MySQL优化

* 开启查询缓存，优化查询；
* explain你的select查询，这可以帮你分析你的查询语句或是表结构的性能瓶颈。EXPLAIN 的查询结果还会告诉你你的索引主键被如何利用的，你的数据表是如何被搜索和排序的；
* 当只要一行数据时使用limit 1，MySQL数据库引擎会在找到一条数据后停止搜索，而不是继续往后查少下一条符合记录的数据；
* 为搜索字段建索引；
* 使用 ENUM 而不是 VARCHAR。如果你有一个字段，比如“性别”，“国家”，“民族”，“状态”或“部门”，你知道这些字段的取值是有限而且固定的，那么，你应该使用 ENUM 而不是VARCHAR；
* Prepared StatementsPrepared Statements很像存储过程，是一种运行在后台的SQL语句集合，我们可以从使用 prepared statements 获得很多好处，无论是性能问题还是安全问题；
* Prepared Statements 可以检查一些你绑定好的变量，这样可以保护你的程序不会受到“SQL注入式”攻击；
* 垂直分表；
* 选择正确的存储引擎。

## 二十二、key和index的区别

* key 是数据库的物理结构，它包含两层意义和作用，一是约束（偏重于约束和规范数据库的结构完整性），二是索引（辅助查询用的）。包括primary key, unique key, foreign key等；
* index是数据库的物理结构，它只是辅助查询的，它创建时会在另外的表空间（mysql中的innodb表空间）以一个类似目录的结构存储。索引要分类的话，分为前缀索引、全文本索引等；

## 二十三、Mysql中MyISAM和InnoDB的区别有哪些？

**区别：**

1. InnoDB支持事务，MyISAM不支持

对于InnoDB每一条SQL语言都默认封装成事务，自动提交，这样会影响速度，所以最好把多条SQL语言放在begin和commit之间，组成一个事务；

1. InnoDB支持外键，而MyISAM不支持。对一个包含外键的InnoDB表转为MYISAM会失败；
2. InnoDB是聚集索引，数据文件是和索引绑在一起的，必须要有主键，通过主键索引效率很高。

但是辅助索引需要两次查询，先查询到主键，然后再通过主键查询到数据。因此主键不应该过大，因为主键太大，其他索引也都会很大。

而MyISAM是非聚集索引，数据文件是分离的，索引保存的是数据文件的指针。主键索引和辅助索引是独立的。

1. InnoDB不保存表的具体行数，执行select count(\*) from table时需要全表扫描。而MyISAM用一个变量保存了整个表的行数，执行上述语句时只需要读出该变量即可，速度很快；
2. Innodb不支持全文索引，而MyISAM支持全文索引，查询效率上MyISAM要高；

**如何选择：**

1. 是否要支持事务，如果要请选择innodb，如果不需要可以考虑MyISAM；
2. 如果表中绝大多数都只是读查询，可以考虑MyISAM，如果既有读写也挺频繁，请使用InnoDB；
3. 系统奔溃后，MyISAM恢复起来更困难，能否接受；
4. MySQL5.5版本开始Innodb已经成为Mysql的默认引擎(之前是MyISAM)，说明其优势是有目共睹的，如果你不知道用什么，那就用InnoDB，至少不会差。

## 二十四、数据库表创建注意事项

### 1．字段名及字段配制合理性

* 剔除关系不密切的字段；
* 字段命名要有规则及相对应的含义（不要一部分英文，一部分拼音，还有类似a.b.c这样不明含义的字段）；
* 字段命名尽量不要使用缩写（大多数缩写都不能明确字段含义）；
* 字段不要大小写混用（想要具有可读性，多个英文单词可使用下划线形式连接）；
* 字段名不要使用保留字或者关键字；
* 保持字段名和类型的一致性；
* 慎重选择数字类型；
* 给文本字段留足余量；

### 2．系统特殊字段处理及建成后建议

* 添加删除标记（例如操作人、删除时间）；
* 建立版本机制；

### 3．表结构合理性配置

* 多型字段的处理，就是表中是否存在字段能够分解成更小独立的几部分（例如：人可以分为男人和女人）；
* 多值字段的处理，可以将表分为三张表，这样使得检索和排序更加有调理，且保证数据的完整性！

### 4．其它建议

* 对于大数据字段，独立表进行存储，以便影响性能（例如：简介字段）；
* 使用varchar类型代替char，因为varchar会动态分配长度，char指定长度是固定的；
* 给表创建主键，对于没有主键的表，在查询和索引定义上有一定的影响；
* 避免表字段运行为null，建议设置默认值（例如：int类型设置默认值为0）在索引查询上，效率立显；
* 建立索引，最好建立在唯一和非空的字段上，建立太多的索引对后期插入、更新都存在一定的影响（考虑实际情况来创建）；

## 二十五、事务

关系性数据库需要遵循ACID规则，具体内容如下：

* **原子性**：事务是最小的执行单位，不允许分割。事务的原子性确保动作要么全部完成，要么完全不起作用；
* **一致性**：执行事务前后，数据保持一致；
* **隔离性**：并发访问数据库时，一个用户的事物不被其他事物所干扰，各并发事务之间数据库是独立的；
* **持久性**: 一个事务被提交之后。它对数据库中数据的改变是持久的，即使数据库 发生故障也不应该对其有任何影响。

为了达到上述事务特性，数据库定义了几种不同的事务隔离级别：

* **READ\_UNCOMMITTED（未授权读取）**: 最低的隔离级别，允许读取尚未提交的数据变更，可能会导致脏读、幻读或不可重复读；
* **READ\_COMMITTED（授权读取）**: 允许读取并发事务已经提交的数据，可以阻止脏读，但是幻读或不可重复读仍有可能发生；
* **REPEATABLE\_READ（可重复读）**: 对同一字段的多次读取结果都是一致的，除非数据是被本身事务自己所修改，可以阻止脏读和不可重复读，但幻读仍有可能发生。
* **SERIALIZABLE（串行）**: 最高的隔离级别，完全服从ACID的隔离级别。所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不可重复读以及幻读。但是这将严重影响程序的性能。通常情况下也不会用到该级别。

这里需要注意的是：Mysql 默认采用的 REPEATABLE\_READ隔离级别 Oracle 默认采用的 READ\_COMMITTED隔离级别。事务隔离机制的实现基于锁机制和并发调度。其中并发调度使用的是MVVC（多版本并发控制），通过保存修改的旧版本信息来支持并发一致性读和回滚等特性。

## 二十六、锁

MyISAM和InnoDB存储引擎使用的锁：

MyISAM采用表级锁(table-level locking)。

InnoDB支持行级锁(row-level locking)和表级锁，默认为行级锁。

**表级锁和行级锁对比：**

* 表级锁：Mysql中锁定 粒度最大的一种锁，对当前操作的整张表加锁，实现简单，资源消耗也比较少，加锁快，不会出现死锁。其锁定粒度最大，触发锁冲突的概率最高，并发度最低，MyISAM和 InnoDB引擎都支持表级锁。
* 行级锁：Mysql中锁定 粒度最小的一种锁，只针对当前操作的行进行加锁。行级锁能大大减少数据库操作的冲突。其加锁粒度最小，并发度高，但加锁的开销也最大，加锁慢，会出现死锁。

**InnoDB存储引擎的锁的算法有三种：**

* Record lock：单个行记录上的锁
* Gap lock：间隙锁，锁定一个范围，不包括记录本身
* Next-key lock：record+gap 锁定一个范围，包含记录本身

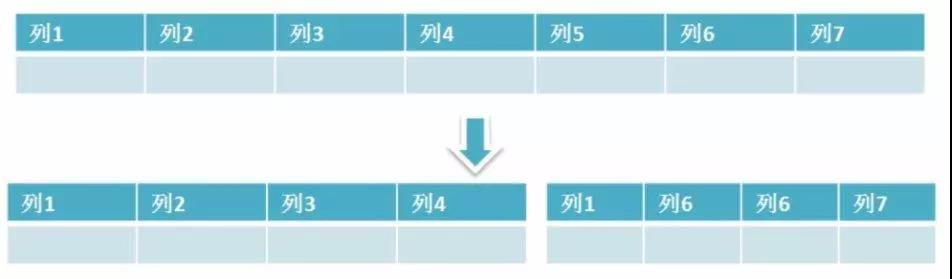
## 二十七、大表优化

当MySQL单表记录数过大时，数据库的CRUD性能会明显下降，一些常见的优化措施如下：

* **限定数据的范围**： 务必禁止不带任何限制数据范围条件的查询语句。比如：我们当用户在查询订单历史的时候，我们可以控制在一个月的范围内。；
* **读/写分离**： 经典的数据库拆分方案，主库负责写，从库负责读；
* **缓存**： 使用MySQL的缓存，另外对重量级、更新少的数据可以考虑使用应用级别的缓存；
* **垂直分区**：

根据数据库里面数据表的相关性进行拆分。 例如，用户表中既有用户的登录信息又有用户的基本信息，可以将用户表拆分成两个单独的表，甚至放到单独的库做分库。

简单来说垂直拆分是指数据表列的拆分，把一张列比较多的表拆分为多张表。 如下图所示，这样来说大家应该就更容易理解了。



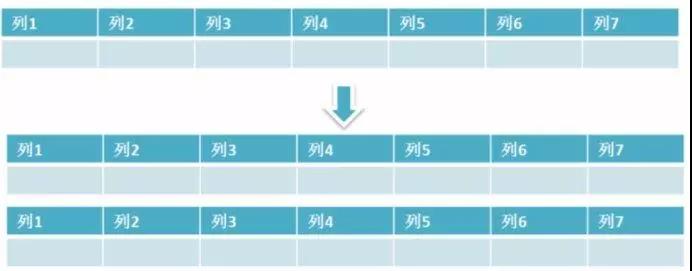
垂直拆分的优点： 可以使得行数据变小，在查询时减少读取的Block数，减少I/O次数。此外，垂直分区可以简化表的结构，易于维护。

垂直拆分的缺点： 主键会出现冗余，需要管理冗余列，并会引起Join操作，可以通过在应用层进行Join来解决。此外，垂直分区会让事务变得更加复杂；

* **水平分区**：

保持数据表结构不变，通过某种策略存储数据分片。这样每一片数据分散到不同的表或者库中，达到了分布式的目的。 水平拆分可以支撑非常大的数据量。

水平拆分是指数据表行的拆分，表的行数超过200万行时，就会变慢，这时可以把一张的表的数据拆成多张表来存放。举个例子：我们可以将用户信息表拆分成多个用户信息表，这样就可以避免单一表数据量过大对性能造成影响。



水品拆分可以支持非常大的数据量。需要注意的一点是:分表仅仅是解决了单一表数据过大的问题，但由于表的数据还是在同一台机器上，其实对于提升MySQL 并发能力没有什么意义，所以 水品拆分最好分库 。

水平拆分能够 支持非常大的数据量存储，应用端改造也少，但 分片事务难以解决 ，跨界点Join 性能较差，逻辑复杂。《Java工程师修炼之道》的作者推荐 尽量不要对数据进行分片，因为拆分会带来逻辑、部署、运维的各种复杂度 ，一般的数据表在优化得当的情况下支撑千万以下的数据量是没有太大问题的。如果实在要分片，尽量选择客户端分片架构，这样可以减少一次和中间件的网络 I/O。

**下面补充一下数据库分片的两种常见方案：**

* 客户端代理： 分片逻辑在应用端，封装在jar包中，通过修改或者封装JDBC层来实现。 当当网的 Sharding-JDBC 、阿里的TDDL是两种比较常用的实现。
* 中间件代理： 在应用和数据中间加了一个代理层。分片逻辑统一维护在中间件服务中。 我们现在谈的 Mycat、360的Atlas、网易的DDB等等都是这种架构的实现。

## 二十八、Mysql中有哪些不同的表格？

共有5种类型的表格：

MyISAM

Heap

Merge

INNODB

ISAM

## 二十九、CHAR和VARCHAR的区别？

1.CHAR和VARCHAR类型在存储和检索方面有所不同

2.CHAR列长度固定为创建表时声明的长度，长度值范围是1到255

当CHAR值被存储时，它们被用空格填充到特定长度，检索CHAR值时需删除尾随空格。

## 三十、主键和候选键有什么区别？

表格的每一行都由主键唯一标识,一个表只有一个主键。

主键也是候选键。按照惯例，候选键可以被指定为主键，并且可以用于任何外键引用。

## 三十一、myisamchk是用来做什么的？

它用来压缩MyISAM表，这减少了磁盘或内存使用。

## 三十二、MyISAM Static和MyISAM Dynamic有什么区别？

在MyISAM Static上的所有字段有固定宽度。动态MyISAM表将具有像TEXT，BLOB等字段，以适应不同长度的数据类型。

MyISAM Static在受损情况下更容易恢复。

## 三十三、如果一个表有一列定义为TIMESTAMP，将发生什么？

每当行被更改时，时间戳字段将获取当前时间戳。

列设置为AUTO INCREMENT时，如果在表中达到最大值，会发生什么情况？

它会停止递增，任何进一步的插入都将产生错误，因为密钥已被使用。

## 三十四、怎样才能找出最后一次插入时分配了哪个自动增量？

LAST\_INSERT\_ID将返回由Auto\_increment分配的最后一个值，并且不需要指定表名称。

## 三十五、LIKE声明中的％和\_是什么意思？

％对应于0个或更多字符，\_只是LIKE语句中的一个字符。

## 三十六、列对比运算符是什么？

在SELECT语句的列比较中使用=，<>，<=，<，> =，>，<<，>>，<=>，AND，OR或LIKE运算符。

## 三十七、BLOB和TEXT有什么区别？

BLOB是一个二进制对象，可以容纳可变数量的数据。TEXT是一个不区分大小写的BLOB。

BLOB和TEXT类型之间的唯一区别在于对BLOB值进行排序和比较时区分大小写，对TEXT值不区分大小写。

## 三十八、mysql\_fetch\_array和mysql\_fetch\_object的区别是什么？

以下是mysql\_fetch\_array和mysql\_fetch\_object的区别：

mysql\_fetch\_array（） – 将结果行作为关联数组或来自数据库的常规数组返回。

mysql\_fetch\_object – 从数据库返回结果行作为对象。

## 三十九、MyISAM表格将在哪里存储，并且还提供其存储格式？

每个MyISAM表格以三种格式存储在磁盘上：

·“.frm”文件存储表定义

·数据文件具有“.MYD”（MYData）扩展名

索引文件具有“.MYI”（MYIndex）扩展名

## 四十、Mysql如何优化DISTINCT？

DISTINCT在所有列上转换为GROUP BY，并与ORDER BY子句结合使用。

1

SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;

## 四十一、如何显示前50行？

在Mysql中，使用以下代码查询显示前50行：

SELECT\*FROM

LIMIT 0,50;

## 四十二、可以使用多少列创建索引？

任何标准表最多可以创建16个索引列。

NOW（）和CURRENT\_DATE（）有什么区别？

NOW（）命令用于显示当前年份，月份，日期，小时，分钟和秒。

CURRENT\_DATE（）仅显示当前年份，月份和日期。

## 四十三、什么是非标准字符串类型？

TINYTEXT

TEXT

MEDIUMTEXT

LONGTEXT

## 四十四、什么是通用SQL函数？

CONCAT(A, B) – 连接两个字符串值以创建单个字符串输出。通常用于将两个或多个字段合并为一个字段。

FORMAT(X, D)- 格式化数字X到D有效数字。

CURRDATE(), CURRTIME()- 返回当前日期或时间。

NOW（） – 将当前日期和时间作为一个值返回。

MONTH（），DAY（），YEAR（），WEEK（），WEEKDAY（） – 从日期值中提取给定数据。

HOUR（），MINUTE（），SECOND（） – 从时间值中提取给定数据。

DATEDIFF（A，B） – 确定两个日期之间的差异，通常用于计算年龄

SUBTIMES（A，B） – 确定两次之间的差异。

FROMDAYS（INT） – 将整数天数转换为日期值。

## 四十五、MYSQL支持事务吗？

在缺省模式下，MYSQL是autocommit模式的，所有的数据库更新操作都会即时提交，所以在缺省情况下，mysql是不支持事务的。

但是如果你的MYSQL表类型是使用InnoDB Tables 或 BDB tables的话，你的MYSQL就可以使用事务处理,使用SET

AUTOCOMMIT=0就可以使MYSQL允许在非autocommit模式，在非autocommit模式下，你必须使用COMMIT来提交你的更改，或者用ROLLBACK来回滚你的更改。

## 四十六、mysql里记录货币用什么字段类型好

NUMERIC和DECIMAL类型被Mysql实现为同样的类型，这在SQL92标准允许。他们被用于保存值，该值的准确精度是极其重要的值，例如与金钱有关的数据。当声明一个类是这些类型之一时，精度和规模的能被(并且通常是)指定。

例如：

salary DECIMAL(9,2)

在这个例子中，9(precision)代表将被用于存储值的总的小数位数，而2(scale)代表将被用于存储小数点后的位数。

因此，在这种情况下，能被存储在salary列中的值的范围是从-9999999.99到9999999.99。

## 四十七、mysql有关权限的表都有哪几个？

Mysql服务器通过权限表来控制用户对数据库的访问，权限表存放在mysql数据库里，由mysql\_install\_db脚本初始化。这些权限表分别user，db，table\_priv，columns\_priv和host。

## 四十八、列的字符串类型可以是什么？

字符串类型是：

SET

BLOB

ENUM

CHAR

TEXT

## 四十九、MySQL数据库作发布系统的存储，一天五万条以上的增量，预计运维三年,怎么优化？

a. 设计良好的数据库结构，允许部分数据冗余，尽量避免join查询，提高效率。

b. 选择合适的表字段数据类型和存储引擎，适当的添加索引。

c. mysql库主从读写分离。

d. 找规律分表，减少单表中的数据量提高查询速度。

e. 添加缓存机制，比如memcached，apc等。

f. 不经常改动的页面，生成静态页面。

g. 书写高效率的SQL。比如 SELECT \* FROM TABEL 改为 SELECT field\_1, field\_2, field\_3 FROM TABLE。

## 五十、锁的优化策略

1. 读写分离

2. 分段加锁

3. 减少锁持有的时间

4. 多个线程尽量以相同的顺序去获取资源

不能将锁的粒度过于细化，不然可能会出现线程的加锁和释放次数过多，反而效率不如一次加一把大锁。

## 五十一、索引的底层实现原理和优化

B+树，经过优化的B+树

主要是在所有的叶子结点中增加了指向下一个叶子节点的指针，因此InnoDB建议为大部分表使用默认自增的主键作为主索引。

## 五十二、什么情况下设置了索引但无法使用

1. 以“%”开头的LIKE语句，模糊匹配

2. OR语句前后没有同时使用索引

3. 数据类型出现隐式转化（如varchar不加单引号的话可能会自动转换为int型）

4. 使用不等于查询；

5. 列参与了数学运算或者函数；

6. 在字符串like时左边是通配符.类似于'%aaa'；

7. 当mysql分析全表扫描比使用索引快的时候不使用索引；

8. 当使用联合索引，前面一个条件为范围查询，后面的即使符合最左前缀原则，也无法使用索引；

## 五十三、实践中如何优化MySQL

最好是按照以下顺序优化：

1. SQL语句及索引的优化

2. 数据库表结构的优化

3. 系统配置的优化

4. 硬件的优化

详细可以查看<https://mikechen.cc/3305.html>

## 五十四、优化数据库的方法

1. 选取最适用的字段属性，尽可能减少定义字段宽度，尽量把字段设置NOTNULL，例如’省份’、’性别’最好适用ENUM

2. 使用连接(JOIN)来代替子查询

3. 适用联合(UNION)来代替手动创建的临时表

4. 事务处理

5. 锁定表、优化事务处理

6. 适用外键，优化锁定表

7. 建立索引

8. 优化查询语句

## 五十五、简单描述mysql中，索引，主键，唯一索引，联合索引的区别，对数据库的性能有什么影响（从读写两方面）

索引是一种特殊的文件(InnoDB数据表上的索引是表空间的一个组成部分)，它们包含着对数据表里所有记录的引用指针。

普通索引(由关键字KEY或INDEX定义的索引)的唯一任务是加快对数据的访问速度。

普通索引允许被索引的数据列包含重复的值。如果能确定某个数据列将只包含彼此各不相同的值，在为这个数据列创建索引的时候就应该用关键字UNIQUE把它定义为一个唯一索引。也就是说，唯一索引可以保证数据记录的唯一性。

主键，是一种特殊的唯一索引，在一张表中只能定义一个主键索引，主键用于唯一标识一条记录，使用关键字 PRIMARY KEY 来创建。

索引可以覆盖多个数据列，如像INDEX(columnA, columnB)索引，这就是联合索引。

索引可以极大的提高数据的查询速度，但是会降低插入、删除、更新表的速度，因为在执行这些写操作时，还要操作索引文件。

## 五十六、SQL注入漏洞产生的原因？如何防止？

SQL注入产生的原因：程序开发过程中不注意规范书写sql语句和对特殊字符进行过滤，导致客户端可以通过全局变量POST和GET提交一些sql语句正常执行。

防止SQL注入的方式：

开启配置文件中的magic\_quotes\_gpc 和 magic\_quotes\_runtime设置

执行sql语句时使用addslashes进行sql语句转换

Sql语句书写尽量不要省略双引号和单引号。

过滤掉sql语句中的一些关键词：update、insert、delete、select、 \* 。

提高数据库表和字段的命名技巧，对一些重要的字段根据程序的特点命名，取不易被猜到的。

**为表中得字段选择合适得数据类型**

字段类型优先级: 整形>date,time>enum,char>varchar>blob,text

优先考虑数字类型，其次是日期或者二进制类型，最后是字符串类型，同级别得数据类型，应该优先选择占用空间小的数据类型

**存储时期**

Datatime:以 YYYY-MM-DD HH:MM:SS 格式存储时期时间，精确到秒，占用8个字节得存储空间，datatime类型与时区无关

Timestamp:以时间戳格式存储，占用4个字节，范围小1970-1-1到2038-1-19，显示依赖于所指定得时区，默认在第一个列行的数据修改时可以自动得修改timestamp列得值

Date:（生日）占用得字节数比使用字符串.http://datatime.int储存要少，使用date只需要3个字节，存储日期月份，还可以利用日期时间函数进行日期间得计算

Time:存储时间部分得数据

注意:不要使用字符串类型来存储日期时间数据（通常比字符串占用得储存空间小，在进行查找过滤可以利用日期得函数）

使用int存储日期时间不如使用timestamp类型

## 五十七、索引的目的是什么？

快速访问数据表中的特定信息，提高检索速度

创建唯一性索引，保证数据库表中每一行数据的唯一性。

加速表和表之间的连接

使用分组和排序子句进行数据检索时，可以显著减少查询中分组和排序的时间

## 五十八、索引对数据库系统的负面影响是什么？

负面影响：

创建索引和维护索引需要耗费时间，这个时间随着数据量的增加而增加；索引需要占用物理空间，不光是表需要占用数据空间，每个索引也需要占用物理空间；当对表进行增、删、改、的时候索引也要动态维护，这样就降低了数据的维护速度。

## 五十九、为数据表建立索引的原则有哪些？

在最频繁使用的、用以缩小查询范围的字段上建立索引。

在频繁使用的、需要排序的字段上建立索引

## 六十、什么情况下不宜建立索引？

对于查询中很少涉及的列或者重复值比较多的列，不宜建立索引。

对于一些特殊的数据类型，不宜建立索引，比如文本字段（text）等

## 六十一、解释MySQL外连接、内连接与自连接的区别

先说什么是交叉连接: 交叉连接又叫笛卡尔积，它是指不使用任何条件，直接将一个表的所有记录和另一个表中的所有记录一一匹配。

内连接 则是只有条件的交叉连接，根据某个条件筛选出符合条件的记录，不符合条件的记录不会出现在结果集中，即内连接只连接匹配的行。

外连接 其结果集中不仅包含符合连接条件的行，而且还会包括左表、右表或两个表中

的所有数据行，这三种情况依次称之为左外连接，右外连接，和全外连接。

左外连接，也称左连接，左表为主表，左表中的所有记录都会出现在结果集中，对于那些在右表中并没有匹配的记录，仍然要显示，右边对应的那些字段值以NULL来填充。右外连接，也称右连接，右表为主表，右表中的所有记录都会出现在结果集中。左连接和右连接可以互换，MySQL目前还不支持全外连接。

## 六十二、Myql中的事务回滚机制概述

事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位，事务回滚是指将该事务已经完成的对数据库的更新操作撤销。

要同时修改数据库中两个不同表时，如果它们不是一个事务的话，当第一个表修改完，可能第二个表修改过程中出现了异常而没能修改，此时就只有第二个表依旧是未修改之前的状态，而第一个表已经被修改完毕。而当你把它们设定为一个事务的时候，当第一个表修改完，第二表修改出现异常而没能修改，第一个表和第二个表都要回到未修改的状态，这就是所谓的事务回滚

## 六十三、SQL语言包括哪几部分？每部分都有哪些操作关键字？

SQL语言包括数据定义(DDL)、数据操纵(DML),数据控制(DCL)和数据查询（DQL）四个部分。

数据定义：Create Table,Alter Table,Drop Table, Craete/Drop Index等

数据操纵：Select ,insert,update,delete,

数据控制：grant,revoke

数据查询：select

## 六十四、完整性约束包括哪些？

数据完整性(Data Integrity)是指数据的精确(Accuracy)和可靠性(Reliability)。

分为以下四类：

1) 实体完整性：规定表的每一行在表中是惟一的实体。

2) 域完整性：是指表中的列必须满足某种特定的数据类型约束，其中约束又包括取值范围、精度等规定。

3) 参照完整性：是指两个表的主关键字和外关键字的数据应一致，保证了表之间的数据的一致性，防止了数据丢失或无意义的数据在数据库中扩散。

4) 用户定义的完整性：不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性即是针对某个特定关系数据库的约束条件，它反映某一具体应用必须满足的语义要求。

与表有关的约束：包括列约束(NOT NULL（非空约束）)和表约束(PRIMARY KEY、foreign key、check、UNIQUE) 。

## 六十五、什么是锁？

数据库是一个多用户使用的共享资源。当多个用户并发地存取数据时，在数据库中就会产生多个事务同时存取同一数据的情况。若对并发操作不加控制就可能会读取和存储不正确的数据，破坏数据库的一致性。

加锁是实现数据库并发控制的一个非常重要的技术。当事务在对某个数据对象进行操作前，先向系统发出请求，对其加锁。加锁后事务就对该数据对象有了一定的控制，在该事务释放锁之前，其他的事务不能对此数据对象进行更新操作。

**基本锁类型：锁包括行级锁和表级锁**

## 六十六、什么叫视图？游标是什么？

**视图**：是一种虚拟的表，具有和物理表相同的功能。可以对视图进行增，改，查，操作，视图通常是有一个表或者多个表的行或列的子集。对视图的修改不影响基本表。它使得我们获取数据更容易，相比多表查询。

**游标**：是对查询出来的结果集作为一个单元来有效的处理。游标可以定在该单元中的特定行，从结果集的当前行检索一行或多行。可以对结果集当前行做修改。一般不使用游标，但是需要逐条处理数据的时候，游标显得十分重要。

## 六十七、什么是存储过程？用什么来调用？

存储过程是一个预编译的SQL语句，优点是允许模块化的设计，就是说只需创建一次，以后在该程序中就可以调用多次。如果某次操作需要执行多次SQL，使用存储过程比单纯SQL语句执行要快。可以用一个命令对象来调用存储过程。

## 六十八、如何通俗地理解三个范式？

第一范式：1NF是对属性的原子性约束，要求属性具有原子性，不可再分解；

第二范式：2NF是对记录的惟一性约束，要求记录有惟一标识，即实体的惟一性；

第三范式：3NF是对字段冗余性的约束，即任何字段不能由其他字段派生出来，它要求字段没有冗余。。

**范式化设计优缺点:**

**优点:**

可以尽量得减少数据冗余，使得更新快，体积小

**缺点:**

对于查询需要多个表进行关联，减少写得效率增加读得效率，更难进行索引优化

**反范式化:**

**优点:**

可以减少表得关联，可以更好得进行索引优化

**缺点:**

数据冗余以及数据异常，数据得修改需要更多的成本

## 六十九、什么是基本表？什么是视图？

答：基本表是本身独立存在的表，在 SQL 中一个关系就对应一个表。 视图是从一个或几个基本表导出的表。视图本身不独立存储在数据库中，是一个虚表

## 七十、试述视图的优点？

答：(1) 视图能够简化用户的操作 (2) 视图使用户能以多种角度看待同一数据； (3) 视图为数据库提供了一定程度的逻辑独立性； (4) 视图能够对机密数据提供安全保护。

## 七十一、NULL是什么意思

答：NULL这个值表示UNKNOWN(未知):它不表示“”(空字符串)。对NULL这个值的任何比较都会生产一个NULL值。您不能把任何值与一个 NULL值进行比较，并在逻辑上希望获得一个答案。

使用IS NULL来进行NULL判断

## 七十二、主键、外键和索引的区别？

主键、外键和索引的区别

**定义：**

主键–唯一标识一条记录，不能有重复的，不允许为空

外键–表的外键是另一表的主键, 外键可以有重复的, 可以是空值

索引–该字段没有重复值，但可以有一个空值

**作用：**

主键–用来保证数据完整性

外键–用来和其他表建立联系用的

索引–是提高查询排序的速度

**个数：**

主键–主键只能有一个

外键–一个表可以有多个外键

索引–一个表可以有多个唯一索引

## 七十三、你可以用什么来确保表格里的字段只接受特定范围里的值?

答：Check限制，它在数据库表格里被定义，用来限制输入该列的值。

触发器也可以被用来限制数据库表格里的字段能够接受的值，但是这种办法要求触发器在表格里被定义，这可能会在某些情况下影响到性能。

## 七十四、说说对SQL语句优化有哪些方法？

（1）Where子句中：where表之间的连接必须写在其他Where条件之前，那些可以过滤掉最大数量记录的条件必须写在Where子句的末尾.HAVING最后。

（2）用EXISTS替代IN、用NOT EXISTS替代NOT IN。

（3） 避免在索引列上使用计算

（4）避免在索引列上使用IS NULL和IS NOT NULL

（5）对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

（6）应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描

（7）应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描

## 七十五、索引是个什么样的数据结构呢?

索引的数据结构和具体存储引擎的实现有关, 在MySQL中使用较多的索引有Hash索引,B+树索引等,而我们经常使用的InnoDB存储引擎的默认索引实现为:B+树索引。

## 七十六、在建立索引的时候,都有哪些需要考虑的因素呢?

建立索引的时候一般要考虑到字段的使用频率,经常作为条件进行查询的字段比较适合.如果需要建立联合索引的话,还需要考虑联合索引中的顺序.此外也要考虑其他方面,比如防止过多的所有对表造成太大的压力.这些都和实际的表结构以及查询方式有关。

## 七十七、联合索引是什么?为什么需要注意联合索引中的顺序?

MySQL可以使用多个字段同时建立一个索引,叫做联合索引.在联合索引中，如果想要命中索引，需要按照建立索引时的字段顺序挨个使用,否则无法命中索引。

具体原因为：

MySQL使用索引时需要索引有序，假设现在建立了"name,age,school"的联合索引，那么索引的排序为：先按照name排序，如果name相同,则按照age排序，如果age的值也相等，则按照school进行排序。

当进行查询时，此时索引仅仅按照name严格有序，因此必须首先使用name字段进行等值查询，之后对于匹配到的列而言，其按照age字段严格有序，此时可以使用age字段用做索引查找，以此类推。因此在建立联合索引的时候应该注意索引列的顺序，一般情况下，将查询需求频繁或者字段选择性高的列放在前面。此外可以根据特例的查询或者表结构进行单独的调整。

## 七十八、同时有多个事务在进行会怎么样呢?

多事务的并发进行一般会造成以下几个问题：

**脏读：**A事务读取到了B事务未提交的内容,而B事务后面进行了回滚。

**不可重复读：**当设置A事务只能读取B事务已经提交的部分，会造成在A事务内的两次查询,结果竟然不一样，因为在此期间B事务进行了提交操作。

**幻读：**A事务读取了一个范围的内容，而同时B事务在此期间插入了一条数据.造成"幻觉"。

## 七十九、Innodb使用的是哪种隔离级别呢?

InnoDB默认使用的是可重复读隔离级别。

## 八十、MySQL都有哪些锁呢？像上面那样子进行锁定岂不是有点阻碍并发效率了？

从锁的类别上来讲，有共享锁和排他锁。

共享锁：又叫做读锁。当用户要进行数据的读取时，对数据加上共享锁。共享锁可以同时加上多个。

排他锁：又叫做写锁。当用户要进行数据的写入时，对数据加上排他锁。排他锁只可以加一个，他和其他的排他锁,共享锁都相斥。

用上面的例子来说就是用户的行为有两种，一种是来看房，多个用户一起看房是可以接受的。一种是真正的入住一晚，在这期间，无论是想入住的还是想看房的都不可以。

锁的粒度取决于具体的存储引擎，InnoDB实现了行级锁，页级锁，表级锁。

他们的加锁开销从大大小，并发能力也是从大到小。

## 八十一、为什么要尽量设定一个主键？

主键是数据库确保数据行在整张表唯一性的保障，即使业务上本张表没有主键，也建议添加一个自增长的ID列作为主键。设定了主键之后，在后续的删改查的时候可能更加快速以及确保操作数据范围安全。

## 八十二、主键使用自增ID还是UUID？

推荐使用自增ID，不要使用UUID。

因为在InnoDB存储引擎中，主键索引是作为聚簇索引存在的，也就是说，主键索引的B+树叶子节点上存储了主键索引以及全部的数据(按照顺序)，如果主键索引是自增ID，那么只需要不断向后排列即可，如果是UUID，由于到来的ID与原来的大小不确定，会造成非常多的数据插入，数据移动，然后导致产生很多的内存碎片，进而造成插入性能的下降。

总之，在数据量大一些的情况下，用自增主键性能会好一些。

## 八十三、MySQL支持哪些存储引擎?

MySQL支持多种存储引擎，比如InnoDB，MyISAM，Memory，Archive等等.在大多数的情况下，直接选择使用InnoDB引擎都是最合适的，InnoDB也是MySQL的默认存储引擎。

## 八十四、MySQL的binlog有有几种录入格式?分别有什么区别?

有三种格式，statement，row和mixed。

statement模式下，记录单元为语句。即每一个sql造成的影响会记录。由于sql的执行是有上下文的，因此在保存的时候需要保存相关的信息，同时还有一些使用了函数之类的语句无法被记录复制。

row级别下，记录单元为每一行的改动，基本是可以全部记下来但是由于很多操作，会导致大量行的改动(比如alter table)，因此这种模式的文件保存的信息太多，日志量太大。

mixed。 一种折中的方案，普通操作使用statement记录，当无法使用statement的时候使用row。

此外，新版的MySQL中对row级别也做了一些优化，当表结构发生变化的时候，会记录语句而不是逐行记录。

## 八十五、超大分页怎么处理?

超大的分页一般从两个方向上来解决。

数据库层面，这也是我们主要集中关注的(虽然收效没那么大),类似于select \* from table where age > 20 limit 1000000,10这种查询其实也是有可以优化的余地的。 这条语句需要load1000000数据然后基本上全部丢弃，只取10条当然比较慢。 当时我们可以修改为select \* from table where id in (select id from table where age > 20 limit 1000000,10)。这样虽然也load了一百万的数据，但是由于索引覆盖，要查询的所有字段都在索引中，所以速度会很快。 同时如果ID连续的好，我们还可以select \* from table where id > 1000000 limit 10，效率也是不错的，优化的可能性有许多种，但是核心思想都一样，就是减少load的数据。

从需求的角度减少这种请求。主要是不做类似的需求(直接跳转到几百万页之后的具体某一页。只允许逐页查看或者按照给定的路线走，这样可预测，可缓存)以及防止ID泄漏且连续被人恶意攻击。

解决超大分页，其实主要是靠缓存，可预测性的提前查到内容,缓存至redis等k-V数据库中，直接返回即可。

## 八十六、关心过业务系统里面的sql耗时吗?统计过慢查询吗?对慢查询都怎么优化过?

在业务系统中，除了使用主键进行的查询，其他的我都会在测试库上测试其耗时，慢查询的统计主要由运维在做，会定期将业务中的慢查询反馈给我们。

慢查询的优化首先要搞明白慢的原因是什么？是查询条件没有命中索引？是load了不需要的数据列?还是数据量太大？

所以优化也是针对这三个方向来的：

* 首先分析语句，看看是否load了额外的数据，可能是查询了多余的行并且抛弃掉了，可能是加载了许多结果中并不需要的列，对语句进行分析以及重写。
* 分析语句的执行计划,然后获得其使用索引的情况，之后修改语句或者修改索引，使得语句可以尽可能的命中索引。
* 如果对语句的优化已经无法进行，可以考虑表中的数据量是否太大，如果是的话可以进行横向或者纵向的分表。

## 八十七、上面提到横向分表和纵向分表,可以分别举一个适合他们的例子吗?

* 横向分表是按行分表。假设我们有一张用户表，主键是自增ID且同时是用户的ID。数据量较大，有1亿多条，那么此时放在一张表里的查询效果就不太理想。我们可以根据主键ID进行分表，无论是按尾号分，或者按ID的区间分都是可以的。 假设按照尾号0-99分为100个表，那么每张表中的数据就仅有100w。这时的查询效率无疑是可以满足要求的。
* 纵向分表是按列分表。假设我们现在有一张文章表。包含字段id-摘要-内容。而系统中的展示形式是刷新出一个列表，列表中仅包含标题和摘要，当用户点击某篇文章进入详情时才需要正文内容。此时，如果数据量大，将内容这个很大且不经常使用的列放在一起会拖慢原表的查询速度。我们可以将上面的表分为两张。id-摘要，id-内容。当用户点击详情，那主键再来取一次内容即可。而增加的存储量只是很小的主键字段。代价很小。

当然，分表其实和业务的关联度很高，在分表之前一定要做好调研以及benchmark。不要按照自己的猜想盲目操作。

## 八十八、其他

<http://www.360doc.com/content/21/0116/15/70209069_957290635.shtml>