# 一．基础知识

1. 数据库三范式

第一范式：每一列的属性都是原子不可再分的。对于关系型数据库，这是基本要求，只要建的表是合法的，就一定满足。

第二范式：第二范式建立在第一范式的基础上。第二范式要求属性完全依赖于主键，即每一行的数据都可以通过主键唯一区分。因此主键都是唯一非空的。

第三范式：属性不依赖于其他非主属性，而是直接依赖于主键。简而言之就是消除传递依赖。对于一对多，多对多的关系时，我们就需要额外建表，然后通过外键来建立联系。

1. 分别说一下范式和反范式的优缺点

范式就是满足三大范式，列属性原子不可再分，属性依赖于主键，并被主键唯一区分，消除传递依赖。反范式就是不满足数据库的三大范式，比如不满足第二范式，不创建主键，在spring security中存储的token表就没有主键。还是就是不满足第三范式，将数据都放在一张表中。

范式的优点就是数据简洁，没有重复数据。

1. 更新的效率更高，相较于反范式可能要一次更新几条，范式只需要更新一次，
2. 另外由于重复数据少，因此加载到内存也更快，并且使用distinct和group by也更快，甚至不需要distinct和group by。

缺点就是分为了多张表，通常需要关联，可能还会导致索引无效，从而效率更低。

反范式的优点就是简单，避免了表之间的关联，当数据比内存大的时候，就避免了随机IO，可能比范式种多个关联更快。缺点就是效率较范式要低，因为更新可能需要同时更新多条数据，并且冗余数据过多，导致表的数据增多，查询可能也会变慢。

现实中我们通常是范式和反范式混用

1. 约束的类型

主键约束，非空约束，唯一约束，外键约束，默认约束。检查约束（mysql不支持,可以设置但是没效果）

# 二．引擎

1. MyISAM和InnoDb数据库的区别
   1. 存储结构：MyISAM每张表分为了3个文件，frm（表的结构）,myd（表的数据）,myi（索引），这是由于它把索引和数据是分开存储的。而InnoDb所有表都存储在了一个文件中。
   2. 事物：MyISAM不支持，InnoDb支持
   3. 外键：MyISAM不支持，InnoDb支持
   4. 锁：MyISAM是表级锁，InnoDb是行级锁。
   5. 索引：MyISAM特有全文索引，InnoDb特有哈希索引。
   6. select count(\*):MyISAM更快，因为它内部维护了一个计数器。
   7. 查询：如果查询非主属性且没有索引覆盖的话，MyISAM更快，因为MyISAM查到对应的结点后，直接通过指针就能去文件中取，而InnoDb还需要进行一次回表操作，取主键索引再查一次。
   8. 增删改：InnoDb更快，因为他的锁的粒度更低。
2. MyISAM和InnoDb索引的区别
   1. MyISAM的索引是非聚簇索引，而InnoDb是聚簇索引；
   2. InnoDb索引的叶子结点存储的是主键和其他索引列，因此如果查询时做到覆盖索引，查询效率会十分高，不然的话需要再进行一次寻址。MyISAM的叶子结点存储的是行数据的地址，需要再进行一次寻址操作c才能取到数据。

# 三．索引

1. 什么是索引

索引其实就是一种数据结构，用来方便数据的查找。常见的有B+树和哈希索引，其中InnoDB默认的是B+树，也是我们平时使用最多的。另外MyISAM还支持R树索引和全文索引。

1. 索引的优缺点

索引的好处：

1. 大大加快了查询的效率，通过B树等数据结构，减少了查询的次数。
2. 避免服务器的排序和临时表。因为B树的叶子结点本身有序，可以直接返回。
3. 将随机I/O变为顺序I/O

索引的缺点：

1. 创建额外的空间来存储索引。
2. 在增删改的时候还需要额外的时间来维护索引。

因此不是所有的情况都需要用上索引，如果说使用索引带来效率的提升大于维护索引的成本，那是可以的。在数据量比较少的时候，全表扫描可能比索引更快。在数据量比较多的时候，索引才能发挥它的优势。

1. 索引的类型

唯一索引，主键索引，普通索引，前缀索引，哈希索引(innodb)，全文索引(myisam)，

1. B+ 树和 B 树的区别，为什么是B+树

B+树和B树都是一种平衡多叉树，他们的最大区别就是B树的非叶子结点存储了数据，而B+树的非叶子结点是不存储数据的，数据全部存储在叶子结点，并且叶子结点是对他的父结点的一个拷贝，并且多了一个指针指向下一个叶子结点。

因为B+树的能够很好的解决范围查询，并且由于B+树的非叶子结点没有存储数据，一次也能够加载更多的数据到内存中去，从而查询更快。不过他的缺点就是每次查询都必须要到叶子结点才能拿到数据。而B树可以在非叶子结点就能够拿到数据并返回。

1. B+树的叶子结点存储的是什么数据

如果是主键的话，存储了整行的数据，如果不是主键的话，存储的是主键值以及其他有索引列的数据。

1. 聚簇索引和非聚簇索引

聚簇索引：叶子结点放的是行数据

非聚簇索引：叶子结点放的是主键或者行数据地址，需要再次查询或者寻址才能拿到数据，因此聚簇索引效率要比非聚簇索引高很多，所以要利用好聚簇索引

1. 覆盖索引

覆盖索引就是所查询的列都建立了索引，由于索引的叶子结点存储的是主键值和索引值，因此覆盖索引后能够减少一次回表操作，直接在索引上就能够拿到数据。因此平时写的时候尽量只写必要的查询列，不写select \* ,增加覆盖索引的概率。

1. 对于联合（多列）索引，哪些情况下能用到索引，哪些情况用不到索引

联合索引通常在使用and条件的时候使用，在5.0版本之前，即使建立了索引，但是无法使用，会进行全表扫面，5.0开始进行了优化，能够使用索引，但是需要合并索引，就是将分别用单个索引查询出来的结果进行合并，这样会消耗大量的资源，并且查询优化器也不会把这些成本计入。从而导致还不如进行全表的顺序扫描，因此多条件的时候最好建立联合索引。

对于联合索引的建立，我们通常把使用频率最高的放在前面，但是有时候，也会把区分度最高的放在前面，这个是情况而定。

对于联合索引，有一个最左前缀匹配原则，即在检索的时候从最左边开始匹配，比如建立了(key1,key2,key3)的联合索引，相当于建立了(key1)的索引，(key1,key2)的索引还有(key1,key2,key3)的索引。像(key2,key3)和(key1,key3)这种情况就用不到联合索引了。

另外，如果在联合索引中使用范围查询就会使范围查询后面的列失效。如果使用表达式，函数的话，索引也会是失效的。

因此联合索引的顺序选择十分重要，通常把使用频率最高的索引放在最前面。

1. 什么时候用不到索引
2. 虽然条件中有索引，但是加入了表达式就不行，比如where price + 1 = 5;
3. 使用了通配符开头的like操作，直接like可以
4. 字符串不加单引号会失效
5. or条件会使索引失效
6. 查询优化器经过计算后，可能使用索引的成本更高，从而不使用索引。
7. Mysql 索引重建

重建索引在常规的数据库维护操作中经常使用。在数据库运行了较长时间后，索引都有损坏的可能，这时就需要重建。对数据重建索引可以起到提高检索效率。REPAIR TABLE `table\_name` QUICK;

另外，要经常清理索引的碎片。optimze table或者重建表()

# 四．事务

1. 介绍一下数据库的事务

事物有四大特性：原子性，隔离性，一致性和持久性。

原子性是：事物要么全部失败，要么全部成功。保证原子性是通过redo/undo log保证的。

隔离性：其他事务不能访问当前事务。

一致性：事务都是从一个一致性状态转换到另外一个一致性状态。

持久性：事务操作成功了之后都是永久生效的。

上面4个特性加起来就是事务了。

1. Mysql 有哪些隔离级别

读未提交：可以读取到还没有提交到的事务，如果事务回滚了，会造成脏读

读已提交：只能看到已经提交的事务所作的改变，可以解决脏读，但是还有可重复读，幻读的问题。

可重复读：保证多个事务在并发读取的时候，可以看到相同的行，即解决了不可重复读，但是还是有幻读。主要是通过行级锁来实现的。

可串行化：这是MySql的最高隔离级别，不允许并发操作，事务只能串行化执行，这样就不会有并发带来的问题，从而幻读也解决了。

InnoDb的默认隔离级别是可重复读。

1. Binlog 和 Redolog ，undolog?

redolog记录着更新成功后的值。如果事务成功提交后，还没来得及写入data file就挂了的话，可以通过redolog进行恢复。从而保证持久性

undolog记录着更新之前的值。如果事务失败了，可以通过undolog进行回滚。从而保证了一致性

binlog是二进制日志，记录着数据库表结构的表更还有表数据的修改的二进制文件。当我们需要复制数据库的时候就用到了binlog

1. Mysql 什么情况会造成脏读、不可重复度、幻读？如何解决

脏读：由于读取到了进行了回滚的事务，从而读取到了不正确的数据。将隔离级别设置为读未提交及以上，

不可重复度：在一个事务中，前后两次读取同一行的数据不一致。就是由于在两次读取中间进行了一次更新的操作，从而导致前后两次读取的结果不一致。将隔离级别设置为可重复度及以上。

幻读：在一个事务中，前后两次读取的数据的行数不一致。就是由于在两次读取中间进行了一次插入操作，从而前后两次读取的结果不一致。将隔离级别设置为可串行话可以解决。

1. Mysql 事务是如何实现的

通过MVCC（多版本并发控制）实现的

1. 谈一谈 MVCC 多版本并发控制

MVCC主要是为了提高数据库并发的性能，做到即使有读写冲突时，也能不加锁，做到并发读。

# 锁

1. Innodb 的默认加锁方式是什么，是怎么实现的
2. 如何高效处理大库DDL
3. 共享锁和排他锁的使用场景，
4. 关系型数据库和非关系数据库的优缺点
5. Mysql 什么情况会造成慢查，如何查看慢查询
6. 如何处理慢查询，你一般是怎么处理慢查询的
7. Mysql 中 varchar 和 char 的区别
8. 数据库外键的优缺点
9. 有没有使用过数据库的视图
10. Mysql 中插入数据使用自增 id 好还是使用 uuid，为什么？
11. Mysql 有哪些数据类型，使用的时候有没有什么注意点
12. Mysql 集群有哪几种方式，分别适用于什么场景
13. Mysql 主从模式如何保证主从强一致性
14. Mysql 集群如何保证主从可用性
15. Mysql 读写分离有哪些解决办法