# MySQL面试常见问题

## 第一部分 通用模块

此部分对 MySQL 整体概念、执行流程、数据库引擎、查询缓存、表空间、回表查询、数据类型间的区别、内存表、临时表、删除表的 n 种方式、枚举、视图、数据恢复等相关知识点对应的面试题进行解答。

## 第二部分 索引模块

索引的好坏直接影响数据库的性能，所以索引的面试题也是面试中必问的问题，此部分为索引对应的面试题合集。

### 2.1 模糊查询与索引

示例： select \* from temp where col like ‘%a%’;

索引通常被用来快速发现带有特殊列的值的行，如果没有索引，MySQL必须进行全表扫描发现相关的行。对于超级大的表，这非常耗时，如果被查询的表的列上有索引，MySQL能够非常快速的检测要查询的行所在行的数据文件的位置（seek）而不用查看所有的数据，这非常快速。

加索引-查询的速度是不加索引的几百倍 （百万级数据或者千万级数据）

索引的类型：

* Primary key
* Unique 唯一索引
* INDEX
* FULLTEXT (全文索引)

使用索引的场景：

**（a），快速发现满足where字句（clause）的行**

**（b），如果要在多个索引之间进行选择，MySQL通常会使用发现最小数量的行的索引。**

**（c），为了发现min,max 值在特殊的索引列上**

**（d），sort 或者 group 一个表**

**索引在数据量较少的表上的查询显示不是很重要。**

**优化方式：**

1. **主键优化（Primary Key Optimization）**

主键代表着在至关重要的查询中使用的一个或一组列，如果你的表非常大，并且没有设置主键，你可以设置一个自增列作为主键。

1. **外键优化（Foreign Key Optimization）**

如果一个表有非常多的列，并且你的查询有非常多的不同的列的联合，把最低频率使用的数据列分割成单独的只包含少数列的表是非常有效的方式。每个单独的表有一个主键去扫描数据。

1. **列索引**

在一个单独的列上创建索引是最通常的类型。B-Tree 数据结构让索引快速的发现特殊的值，一些值的集合，或者是一些值的范围。 例如这些操作 = ,> ,<= between, in 在where 字句中。每个表的最大的索引的数量和最大的索引的长度是有存储引擎决定的。

* Index Prefixed （索引前缀）
* FULLTEXT Indexs （全文索引）
* Spatial Indexes （空间索引）
* Indexes In the Memory Storage Engine （在内存存储引擎中的索引）

1. **Multuple-Column Indexs （组合索引）**

MySQL 能够创建Composite Index ,其包含多个列。MySQL 能够使用多列索引在查询中尝试所有的列。

B-Tree 索引的特征

B-Tree 索引能够被使用在 = ， > , >= , < , <= 或者 BETWEEN 操作的列比较表达式中，同时该索引也可以被用于 Like 比较表达式中（如果该表达式不以通配符%开头）

例如：

该语句会使用索引

Select \* from tal\_name where key\_col like ‘’partick%”

而该语句不会使用索引

Select \* from tal\_name where key\_col like “%partick%”;

Select \* from tal\_name where key\_col like “partick%”;

**建立索引的原则：**

1. 最左前缀匹配原则，非常重要的原则，mysql 会一直向右边匹配直到遇到范围查询（> , < , between, like 就停止匹配，比如 a = 1 and b = 2 and c > 3 and d = 4） 如果建立 （a,b,c,d） 顺序的索引，d是用不到索引的，如果建立（a, b,d,c）的索引都可以用到，a,b,d的顺序可以任意调整。
2. = 和 in 可以乱序，比如 a = 1 and b = 2 and c = 3 建立 （a,b,c）索引可以任意顺序，mysql 的查询优化器会帮助你优化成索引可以识别的形式。
3. 尽量选择区分度高的列作为索引，区分度的公式是count(distinct col) / count(\*) ,表示字段中值不重复的比例，唯一键的区分度是1，而 一些状态，性别，字段在大数据面前区分度就是0，一般join 的字段我们要求区分度 是 0.1以上。
4. 添加索引列不能参与计算。
5. 尽量的扩展索引而不是新建索引。

## 第三部分 事务模块

事务决定了程序的稳定性，在 MySQL 中的地位也是首屈一指，也是面试中必问的面试题，此部分为事务对应的面试题合集。

数据库事务是访问并可能操作各种数据项的一个数据库操作序列，这些操作要么全部执行，要么全部不执行，是一个不可分割的工作单位。事务由事务开始和事务结束之间大的执行的全部数据库操作组成。

数据库事务的四大特性：

* 原子性

事务包含的数据库操作要么全部完成，要么不执行。如果在事务中的一系列操作中发生错误，数据库会发生回滚。

* 一致性

事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。

* 隔离性

数据库允许多个并发事务同时对其数据进行读写和修改的能力（事务执行的中间结果对其他事务透明），隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致行。事务隔离的级别为 。

* 1. 读未提交 （Read uncommitted）
  2. 读提交 （read committed）
  3. 可重复读 （repeatable read）
  4. 串行化 （Serializable）
* 持久性

对于任意已经提交的事务，系统必须保证该事务对数据库的改变不被丢失，即使数据库出现故障。

事务的语法：

Begin transaction

End transaction

Commit 用于提交事务

Rollback 用于在事务包含的一系列操作中发生错误的时候，将已经更改的数据回滚。

Savepoint 保存保存点

使用事务的优点：

1. 把逻辑相关的操作分成了一个组
2. 在数据永久改变前，可以预览数据变化
3. 能够保证数据的读一致性

## 第四部分 锁

锁包括：全局锁、表锁、行锁、死锁、乐观锁、悲观锁等，不同的数据库引擎支持的锁支持粒度也是不同的，此部分的面试题，让你彻底搞定锁相关的面试题。

## 第五部分 日志

日志看似不起眼，却是 MySQL 主备同步和容灾恢复以及问题排除的关键，当然也是面试中必问的问题，这部分会对不同的数据库引擎中的重点日志，进行详细的介绍。

## 第六部分 MySQL 操作命令和内置函数

MySQL 的操作命令，对于程序员或者 DBA 来说也是必须具备的一项技能，比如，用户和权限的创建、数据库相关信息的查询等，都离不开对 MySQL 命令行的掌握。对内置函数的掌握程度，代表了你对 MySQL 的掌握程度，善用 MySQL 提供的内置函数，会让你有事半功倍的效果，内置函数也是笔试中必考的面试题。

## 第七部分 性能优化和分布式

性能优化和分布式是面试中决定你高度的关键指标，其中性能优化包括了慢查询的分析和处理，对分布式的掌握体现了你的技术深度。

## 第八部分 开放性问题

很多大公司最后也会问一下没有标准答案的开放性问题，以考察面试者的技术能力边界和对待问题的分析思路，这部分助你更平稳的获得 offer。

## 第九部分 常用操作命令

### 9.1 explain 命令

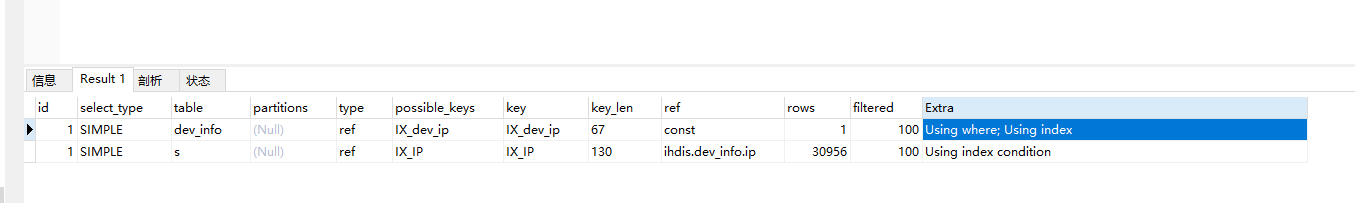
**使用方式** explain select \* from tal\_name

我们很多时候编写一条SQL语句，往往想知道这条SQL语句执行是否高效。或者说，我们建立好的索引在这条语句中是否使用到了，就可以使用explain命令来分析一下。

**使用explain 命令我们可以知道一下信息：**

表的读取顺序，数据读取操作的类型，那些索引可以使用，那些索引实际使用了，表之间的引用，那张表有多少行被优化器查询等信息

**explaion** 命令输出的列



|  |  |  |
| --- | --- | --- |
| 列名称 | 值 | 释义 |
| Id | 一组数字 | 包含一组数字，表示查询中执行select字句或操作表的顺序。Id值越大，优先级越高，越先被执行。 |
| Select\_type | **Simple** | 简单查询，该查询不会包含union或子查询 |
| **Primary** | Primary key 如果查询包含union或子查询，则最外层的查询被标识为primary key |
| Union | 表示此查询是union中的第二个或者随后的查询 |
| Dependent | Union 满足union中的第二个或者随后的查询，其次取决于外面的查询 |
| Union resutl | Union的结果 |
| **Subquery** | 子查询中的第一个select 语句 |
| depentdent subquery | 子查询的第一个select语句，同时取决于外面的查询 |
| **derived** | 包含在from字句中子查询 |
| uncacheable subquery | 满足是子查询中第一个select 语句，同时意味着select 中的某些特性取决于结果被缓存于一个Item\_cache中。 |
| uncacheable union | 满足此查询时union中第二个或者随后的查询，同时意味着select中的某些特性阻止结果被缓存与一个item\_cache 中 |
| Table | 表名（有别名用别名） | 显示了对应行正在访问那个表 |
| Type | ALL | 全表扫描 （性能最差） |
| Index | 索引扫描 |
| Range | 范围扫描 |
| Ref | 索引访问，也称为索引查找，它返回所有匹配某个单个值的行。此类型通常出现在多表join查询 |
| Eq\_ref | 使用这种索引查找。最多返回一天符合条件的记录，使用唯一性索引或者主键查找时会出现该值，非常高效 |
| Const,system | 该表最多有一个匹配行，在查询开始读取，或者该表是系统表，只有一行匹配。 |
| Null | 在执行阶段不需要访问表 |
| Possible\_keys |  | 这里列显示查询可能使用那些索引来查找 |
| Key\_len |  | 这一列显示了在索引里使用的字节数，当key列值为null 时，则该列也是null |
| ref |  | 这一列显示了那些字段或者常量被用来和key配合从表中查询记录出来 |
| Rows |  | 这一列显示了估计要找到所需的行而读取的行数，这个值是个估计值，原则上越小越好。 |
| extra | Using index | 使用覆盖索引，表示查询索引就可查到所需数据，不用扫描数据文件，往往说明性能不错。 |
|  | Using where | 在存储引擎检索行后再进行过滤，使用where从句来限制哪些行将于下一张表匹配或者是返回给用户 |
|  | Using temporary | 在查询结果排序时会使用一个临时表，一般出现排序，分组，和多表join的情况，查询效率低下，建议优化。 |
|  | Using filesort | 对结果使用一个外部索引，而不是按索引次序从表里读取行，一般有出现值，都建议优化去掉。因为这样的查询CPU资源消耗大。 |

### 9.2 profile 命令

Profile 代码示例

**set profiling = 1;**

**select \* from ihdis.cure\_record where shift\_id = 'd8a8870a7dbf4928b6c88fa7dbb41d8d';**

**set profiling = 0;**

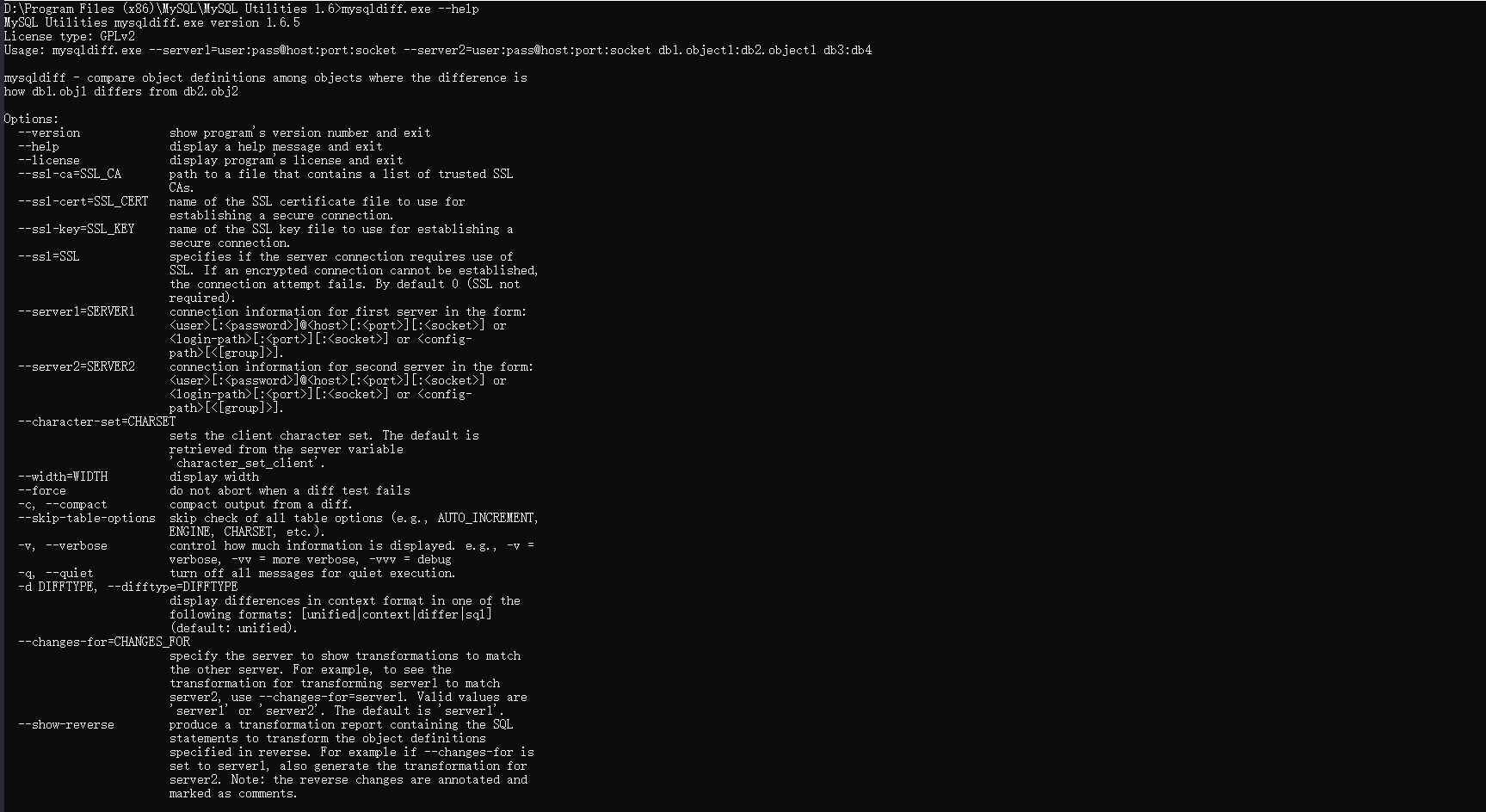
**show PROFILEs;**

**show PROFILE all for QUERY 95;**

## 第十部分 主从一致性校验

### 10.1 mysqldiff

使用mysqldiff 进行主从一致性校验。



Usage:

mysqldiff.exe --server1=user:pass@host:port:socket --server2=user:pass@host:port:socket db1.object1:db2.object1 db3:db4

使用方式