请写出下面MySQL数据类型表达的意义（int(0),char(16),varchar(16),datetime,text）

Int(0)整形，宽度是0位

char(16)定长字符串

Varchar(16)可变长度字符串，长度是16位

datetime保存时间日期的

text 存储字符串类型，存储大文本数据的

MySQL基础操作

MySQL存储引擎

MySQL锁机制

MySQL事务处理，存储过程，触发器

整数类型

TINYINT、SMALLINT、MEDIUMINT、INT、BIGINT

属性：UNSIGNED

长度：可以为整数类型指定宽度，例如：INT(11)、对大多数应用是没有意义的，它不会限制值的合法范围，只会影响显示字符的个数

请写出下面MySQL数据类型表达的意义(int(0),char(16),varchar(16),datetime,text)

## 实数类型

FLOAT、BOUBLE、DECIMAL

DECIMAL可存储比BIGINT还大的整数；可以用于存储精确的小数

FLOAT和DOUBLE类型支持使用标准的浮点进行近似计算

### 字符串类型

VARCHAR、CHAR、TEXT、BLOB

VARCHAR类型用于存储可变长度字符串，它比定长类型更节省空间

VARCHAR使用1或2额外字节记录字符串的长度，列长度小于255字节，使用1个字节表示，否则用2个

**字符串类型**

VARCHAR长度，如果存储内容超出指定长度，会被截断

CHAR是定长的，根据定义的字符串长度分配足够的空间

CHAR会根据需要采用空格进行填充以方便比较

CHAR 适合存储很短的字符串，或者所有值都接近同一个长度

CHAR长度，超出设定的长度，会被截断

对于经常变更的数据，CHAR比VARCHAR更好，CHAR不容易产生碎片

对于非常短的列，CHAR比VARCHAR在存储空间上更有效率

只分配真正需要的空间，更长的列会消耗更多内存

尽量避免使用BLOB/TEXT类型，查询会使用临时表，导致严重的性能开销。

**枚举**

有时可以使用枚举代替常用的字符串类型把不重复的集合存储成一个预定义的集合

非常紧凑，把列表值压缩到一个或两个字节内部存储的是整数

尽量避免使用数字作为ENUM枚举的常量，易混乱

排序是按照内部存储的整数进行排序

枚举表会使表大小大大减小

**日期和时间类型**

尽量使用TIMESTAMP、比DATETIME空间效率高

用整数保存时间戳的格式通常不方便处理

如果需要存储微妙，可以使用bigint存储

**列属性**

Auto\_increment、default、not null、zerofill

常见操作

MySQL的链接和关闭：mysql -u -p -h -P

-u指定用户名

-p指定用户密码

-h指定主机

-P端口

其他：

\G:把打印的结果垂直显示

\c:取消当前mysql的命令

\q:退出mysql

\s:显示mysql服务器状态

\h:帮助信息

\d:改变执行服务

**InnoDB表引擎**

默认事务类型引擎，最重要最广泛的存储引擎，性能非常优秀数据存储在共享表空间，可以通过配置分开

对主键查询的性能高于其他类型的存储引擎

内部做了很多优化，从磁盘读取数据时自动在内存构建hash索引，插入数据时自动构建插入缓冲区。

**MyISAM表引擎**

5.1版本前，MyISAM是默认的存储引擎

拥有全文索引，压缩，空间函数

不支持事务和行级锁，不支持崩溃后的

安全恢复

表存储在两个文件，MYD和MYI

设计简单，某些场景下性能很好

**其他表引擎**

Archive、Blackhole、CSV、Memory

优先选择InnoDB

**基础概念**

表锁是日常开发当中常见的问题，因此也是面试当中最常见的考察点，当多个查询同一时刻进行数据修改时，就会产生并发控制的问题。

共享锁和排他锁，其实就是读锁和写锁

**读锁：**

共享的、不堵塞，多个用户可以同时读一个资源，互不干扰

**写锁：**

排他的、一个写锁会阻塞他的写锁和读锁，这样可以只允许一个人进行写入，防止其他用户读取正在写入的资源。

**锁粒度：**

表锁、系统性能开销最小，会锁定整张表、MyISAM使用表锁

行锁，最大程度的支持并发处理，但是也带来了最大的锁开销，InnoDB实现行级锁。

**事务处理：**

MySQL提供事务处理的表引擎、InnoDB

服务器层不管理事务，由下层的引擎实现，所以同一个事务中，使用多种存储引擎不靠谱

在非事务的表上执行事务操作MySQL不会发出提醒，也不会报错

**存储过程：**

为以后的使用而保存的一条或多条MySQL语句的集合存储过程就是有业务逻辑和流程的集合

可以在存储过程中创建表，更新数据，删除等等

**使用场景：**

通过把处理封装在容易使用的单元中，简化复杂的操作保证数据的一致性简化对变动的管理

**触发器：**

提供给程序员和数据分析员来保证数据完整性的一种方法，它是与表事件相关的特殊存储过程。

**使用场景：**

可通过数据库中的相关表实现级联更改

实时监控某张表中的某个字段的更改而需要做出相应的处理

某些业务编号的生成等

滥用会造成数据库及应用程序的维护困难

牢记以上基础知识点，理解数据类型CHAR和VARCHAR的差异，表存储引擎InnDB和MyISAM的区别

# MySQL创建高性能索引

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。

MySQL索引的基础和类型

MySQL索引的创建原则

MySQL索引的注意事项

**索引的基础**

索引类似于书籍的目录，要想找到一本书的某个特定主题，需要先查找书的目录，定位对应的页码**。**

存储引擎使用类似的方式进行数据查询，先去索引当中找到对应的值，然后根据匹配的索引找到对应的数据行

**索引对性能的影响**

大大减少服务器需要扫描的数据量

帮助服务器避免排序和临时表

将随机I/O变顺序I/O

大大提高查询速度，降低写的速度，占用磁盘

索引的使用场景

对于非常小的表，大部分情况下表扫描效率更高中到大型表，索引非常有效

特大型的表，建立和使用索引的代价将随之增长，可以使用分区技术解决

**索引的类型**

索引有很多种类型，都是实现在存储引擎层的

普通索引：最基本的索引，没有任何约束限制

唯一索引：与普通索引类似，但是具有唯一性约束

主键索引：特殊的唯一索引，不允许有空值

一个表只能有一个主键索引，可以有多个唯一索引主键索引一定是唯一索引，

唯一索引不是主键索引主键可以与外键构成参照完整性约束，防止数据不一致

组合索引：将多个列组合在一起创建索引，可以覆盖多个列

外键索引：只有InnoDB类型的表才可以使用外键索引，保证数据的一致性、完整性和实现级联操作

全文索引：MySQL自带的全文索引只能用于MyISAM,并且只能对英文进行全文检索。

**MySQL索引的创建原则**

1. 最适合索引的列是出现在WHERE子句中的列，或连接子句中的列而不是出现在SELECT关键字后的列。
2. 索引列的基数越大，索引的效果越好
3. 对字符串进行索引，应该制定一个前缀长度，可以节省最大的索引空间
4. 根据情况创建复合索引、复合索引可以提高查询效率
5. 避免创建过多索引，索引会额外占用磁盘空间，降低写操作效率
6. 主键尽可能选择较短的数据类型，可以有效减少索引的磁盘占用提高查询效率

**注意**

1. 复合索引遵循前缀原则

复合索引应用场景：如a表示文章的章节b表示小节c表示文章的段，在这样的场景下创建复合索引可以快速定位到第1章第2节里的第3段WHERE a = 1 and b = 2 and c = 3

Key(a,b,c)#是创建了一个索引同时作用在了abc这样的组合，它的效果是缩短查询的范围

WHERE a = 1 and b = 2 and c = 3

WHERE a = 1 and b = 2

WHERE a = 1

WHERE b = 2 and c = 3 这种情况下不会使用到索引，因为它跳过a了

WHERE a = 1 and c = 3这种情况下也不会用到索引，因为跳过了b

如果说我经常性的查询条件where a and b 或者这时 where a这样的查询这种情况下创建什么样的索引最为合适

答案：

Key(a,b)

1. like查询，%不能在前，可以使用全文索引

where name like "%wang%" 这样即使name创建索引也会失效,like 前面有了%号了

如果我们就是需要这样的，语句就是要%号就是要索引也起作用，我们可以使用全文检索引擎 Sphinx，Lucene这样的概念

1. column is null 可以使用索引（如果想使用一个列判断是否使用到null也是可以使用到索引的）
2. 如果MySQL估计使用索引比全表扫描更慢，会放弃使用索引

比如说我们一张表里面有100条数据写条件是这样写 where id > 1 and id <100 即使你对id创建了索引那在搜索的时候应该搜索是不是应该搜索2到99的数据啊，那在搜索时是怎么样搜索的啊！首先找2这条数据先去索引里面看一下它在什么位置然后找到索引在去定位我们的数据行然后在去看3在去看索引然后在去找数据行，在去看4在去看索引在去找数据行我们是不是一直到99都要这样去看啊！所以在这里面大家注意本身上来说我们这样来查询的话是不是首先我们会多了什么步骤，是不是查询索引的步骤所以mysql认为你本身就100条数据。你要查询其中的98条数据而这98条数据都需要去看索引是不是就很慢了我尽量干什么，我是不是直接把这一百条数据我都让你扫一遍不就得了然后给你返回其中的98条不就ok了码所以这样效率会更高他会自动去放弃不用我们手段去放弃这是我们优化器做的事

1. 如果or前面的条件中的列有索引，后面的没有，索引都不会被用到
2. 列类型是字符串，查询时一定要给值加引号，否则索引失败

比如有一个这样的列 name varchar(16)里面存了一个 "100"的字符串

where name = 100; 这样能搜到但是索引用不到索引，即使你对name创建了索引他的索引会失效为什么？因为人家本身是字符串类型，你给了人家一个整形。所以我们要给100家引号 where name = "100"

**解题方法：**

在一些MySQL索引基础考题中，我们可以轻松的通过索引基础和类型来解决此类问题，对于一些索引创建注意事项方面的考点，我们可以通过索引创建原则和注意事项来解决。

**创建MySQL复合索引应该注意那些事项？**

**我们应该遵循前缀原则**

# MySQL的SQL语句编写考点

**有A(id,sex,par,c1,c2),B(id,age,c1,c2)两张表，其中A.id与B.id关联，现在要求写出一条SQL语句，将B中age>50的记录的c1,c2更新到A表中统一记录中的c1,c2字段中**

**A(id,sex,par,c1,c2) A表的字段**

**B(id,age,c1,c2) B表的字段**

**第一种方法**

**update A,B set A.c1 = B.c1,A.c2 = B.c2**

**where A.id=B.id and B.age > 50**

**第二种方法**

**update A inner join B on A.id = B.id**

**set A.c1 = B.c1,A.c2 = B.c2**

**where B.age > 50**

**MySQL的关联查询语句**

**关联更新**

**UPDATE A,B SET A.c1 = B.c1,A.c2=B.c2 WHERE A.id=B.id**

**UPDATE A INNER JOIN B ON A.id=B.id SET A.c1=B.c1,A.c2=B.c2 WHERE**

**六种关联查询**

交叉链接(CROSS JOIN),内连接(INNER JOIN),外连接(LEFTJOIN/RIGHT JOIN),联合查询(UNION与UNION ALL),全连接(FULL JOIN)

**交叉连接**

SELECT \* FROM A,B(,C)或者

SELECT \* FROM A CROSS JOIN B(CROSS JOIN C)

没有任何关联条件，结果是笛卡尔积，结果集会很大，没有意义，很少使用

**内链接**

SELECT \* FROM A,B WHERE A.id=B.id或者

SELECT \* FROM A INNER JOIN B ON A.id=B.id

多表中同时符合某种条件的数据记录的集合

**内链接分为三类**

等值连接：ON A.id=B.id

不等值连接：ON A.id > B.id

自连接：SELECT \* FROM A T1 INNER JOIN A T2 ON T1.id=T2.pid

INNER JOIN 可以缩写为 JOIN

**外连接**

左外连接：LEFT OUTER JOIN ,以左表为主，先查询出左表，按照ON后的关联条件匹配右表，没有匹配到的用NULL填充，可以简写成LEFT JOIN

右外连接：RIGHT OUTER JOIN,以右表为主，先查询出右表，按照ON后的关联条件匹配左表，没有匹配到的用NULL填充，可以简写成RIGHT JOIN

**联合查询**

SELECT \* FROM A UNION SELECT \* FROM B UNION....

就是把多个结果集集中在一起，UNION前的结果为基准，需要注意的是联合查询的列数要相等，相同的记录行会合并

如果使用UNION ALL ,不会合并重复的记录行

**全连接**

MySQL不支持全连接

可以使用LEFT JOIN 和 UNION 和RIGHT JOIN 联合使用

SELECT \*FROM A LEFT JOIN B ON A.id UNION

SELECT \*FROM A RIGHT JOIN B ON A.id=B.id

**嵌套查询**

用一条SQL语句的结果作为另外一条SQL语句的条件

SELECT \*FROM A WHERE id IN (SELECT id FROM B)

一般不建议使用，因为效率不好把控

### 解题方法

根据考题要搞清楚表的结够和多表之间的关系，根据想要的结果思考使用哪种关联方式，通常把要查询的列先写出来，然后分析这些列都属于哪些表，才考虑使用关联查询

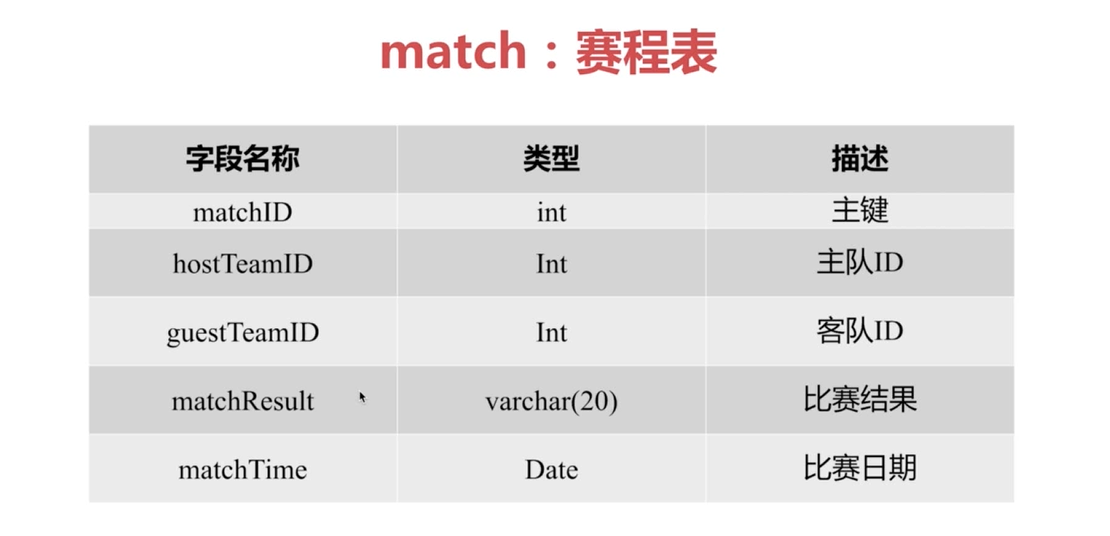
**真题：**

为了记录足球比赛的结果，设计表如下：

第一张表



第二张表



其中，match赛程表中的hostTeamID与guestTeamID都和team表中的teamID关联，查询2006-6-1到2006-7-1之间举行的所有比赛，并且用以下形式列出：拜仁2：0不来梅2006-6-21

方法一、

Select hostTeamID,matchResult,matchTime guestTeamID from match

where matchTime between "2006-6-1" and "2006-7-1";

方法二、

select t1.teamName,m.matchResult,t2.teamName,m.matchTime from

match as m left join team as t1 on m hostTeameID = t1.teamID ,

left join team t2 on m.guestTeameID=t2.gueatTeamID

where m.matchTime between "2006-6-1" and "2006-7-1"

请简述项目中优化SQL语句执行效率的方法，从那些方面，SQL语句性能如何分析？

查找分析查询速度慢的原因

优化查询过程中的数据访问

优化长难的查询语句

优化特定类型的查询语句

分析SQL查询慢的方法

记录慢查询日志

分析查询日志，不要直接打开慢查询日志进行分析，这样比较浪费时间和精力，可以使用pt-query-digest工具进行分析

分析我们的这个查询SQL语句比较慢的原因是什么我们可以使用show profile

show profile：

set profiling=1;开启，服务器上执行的所有语句会检测消耗的时间，存到临时表中。

show profiles

show profile for query 临时表ID

mysql> set profiling=1; #打开临时表

Query OK, 0 rows affected (0.00 sec)

mysql> show profiles; #查看临时表

Empty set (0.00 sec)

mysql> use test; #使用数据库

Database changed

mysql> select \* from a;

+----+------+------+-----+

| id | c1 | c2 | sex |

+----+------+------+-----+

| 1 | b1 | b2 | 1 |

| 2 | b11 | b22 | 1 |

| 3 | b111 | b222 | 1 |

+----+------+------+-----+

3 rows in set (0.00 sec)

mysql> show profiles; #查看临时表，此时可以看到每一条sql语句查询所消耗的时间

+------------+------------+------------------+

| Query\_ID | Duration | Query | #Query 是SQL语句；

+------------+------------+------------------+ #Duration是查询时消耗的时间

| 1 | 0.00011750 | ELECT DATABASE() |

| 2 | 0.00006575 | select \* from a |

+------------+------------+------------------+

2 rows in set (0.00 sec)

mysql> show profile for query 2; #查看select \* from a 语句的查询历程

+---------------------------------+------------+

| Status | Duration |

+---------------------------------+------------+

| starting | 0.000023 |

| Waiting for query cache lock | 0.000004 |

| checking query cache for query | 0.000008 |

| checking privileges on cached | 0.000003 |

| checking permissions | 0.000016 |#花费时间比较长的是这步

| sending cached result to clien | 0.000007 |

| logging slow query | 0.000002 |

| cleaning up | 0.000002 |

+---------------------------------+------------+

8 rows in set (0.00 sec)

mysql>

###### 使用 show status

show status 会返回一些计数器，show global status 查询服务器级别的所有计数

有时根据这些计数，可以猜测出那些操作代价较高或者消耗时间多

mysql> show status; #重点，由于教程讲的比较简略这里不做过多记录，过后需要上网了解

###### 使用 show processlist

观察是否有大量线程处于不正常的状态或者特征

mysql> show processlist;#我们可以看到这里只有一个线程

+-------+------+-----------+------+---------+------+-------+-----------------------+

| Id | User | Host | db | Command | Time | State | Info |

+-------+------+-----------+------+---------+------+-------+-----------------------+

| 430 | root | localhost | test | Query | 0 | NULL | show processlist |

+-------+------+-----------+------+---------+------+-------+-----------------------+

1 row in set (0.00 sec)

mysql>

###### 使用explain:

分析单条的SQL语句

可以看到索引的使用情况，还有扫描行数，还有索引的查询类型。

mysql> explain select \*from a\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: a

type: ALL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: 3

Extra:

1 row in set (0.00 sec)

mysql> desc select \* from a\G; #我们也可以使用 desc 也是 一样的结果

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

id: 1

select\_type: SIMPLE

table: a

type: ALL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: 3

Extra:

1 row in set (0.00 sec)

mysql>

# 优化查询过程中的数据访问

访问数据太多导致查询性能下降

确定应用程序是否在检索大量超过需要的数据，可能是太多行或列

确认MySQL服务器是否在分析大量不必要的数据行

**避免使用如下SQL语句**

查询不需要的记录，使用limit解决

多表关联返回全部列，指定A.id,A.name,B.age

总是取出全部列，SELECT \* 会让优化器无法完成索引覆盖扫描的优化

重复查询相同的数据，可以缓冲数据，下次直接读取缓存

**是否在扫描额外的记录**

使用explain 来进行分析，如果发现查询需要扫描大量的数据但只返回少数的行，可以通过如下技巧去优化：

使用索引覆盖扫描，把所有用的列都放到索引中，这样存储引擎不需要回表获取对应行就可以返回结果

改变数据库和表结构，修改数据表范式

重写SQL语句，让优化器可以以更忧的方式执行查询

**一个复杂查询还是多个简单查询**

MySQL内部每秒能扫描内存中上百万行数据，相比之下，响应数据给客户端就要慢得多

使用尽可能少的查询是好的，但是有时将一个大的查询分解为多个小的查询是很有必要的

**切分查询**

将一个大的查询分为多个小的相同的查询

一次性删除1000万的数据要比一次删除1万，暂停一会的方案更加损耗服务器开销

**分解关联查询**

可以将一条关联语句分解成多条SQL来执行

让缓存的效率更高

执行单个查询可以减少锁的竞争

在应用层做关联可以更容易对数据库进行拆分

查询效率会有大幅提升

**优化count()查询**

count(\*)中\*会忽略所有的列，直接统计所有列数，因此不要使用count(列名)

MyISAM中，没有任何WHERE条件的count(\*)非常块

当有WHERE条件，MyISAM的count统计不一定比其他引擎快

可以使用explain查询近似值，用近似值替代count(\*)

增加汇总表

使用缓存

**优化关联查询**

确定ON或者USING子句的列上有索引

确保GROUP BY和ORDER BY 中只有一个表中的列，这样MySQL才有可能使用索引

**优化子查询**

尽量少用子查询，尽可能使用关联查询类替代。

**优化GROUP BY和DISTINCT**

这两种查询均可使用索引优化，是最有效的优化方法

关联查询中，使用标识列进行分组的效率会更高

如果不需要ORDER BY ,进行GROUP BY 时使用ORDER BY

NULL,MySQL不会在进行文件排序

WITH ROLLUP超级聚合，可以挪到应用程序处理

**优化LIMIT分页**

LIMIT偏移量大的时候查询效率较低

可以记录上次查询的最大ID，下次查询时直接根据ID来查询

**优化UNION查询**

UNION ALL 的效率高于UNION

解题方法:

对于此类考题，先说明如何定位低效SQL语句，然后根据SQL语句可能低效的原因做排查，先从索引着手，如果索引没有问题，考虑以上几个方面，数据访问的问题，长难查询句的问题还是一些特定类型优化的问题，逐一回答。

# MYSQL高可扩展和高可用考点

简述MySQL分表操作和分区的工作原理，分别说说分区和分表的使用场景和各自的优缺点。

**分区表的原理**

**分库表的原理**

**MySQL的复制原理及负载均衡**

**分区表的原理**

**工作原理：**

对用户而言，分区表是一个独立的逻辑表，但是底层MySQL将其分成了多个物理子表，这对用户来说是透明的，每一个分区表都会使用一个独立的表文件。

创建表时使用partition by子句定义每个分区存放的数据，执行查询时，优化器会根据分区定义过滤那些没有我们需要的数据分区，这样查询只需要查询所需数据在的分区即可

分区的主要目的是将数据按照一个较粗的粒度分在不同的表中，这样可以将相关的数据存放在一起，而且如果想一次性删除整个分区的数据也很方便

**使用场景**

1. 表非常大，无法全部存在内存，或者只在表的最后有热点数据，其他都是历史数据
2. 分区表的数据更易维护，可以对立的分区进行独立的操作
3. 分区表的数据可以分布在不同的机器上，从而高校使用资源
4. 可以使用分区表来避免某些特殊的瓶颈
5. 可以备份和恢复独立的分区

**限制**

1. 一个表最多只能有1024个分区
2. 5.1版本中，分区表表达式必须是整个数，5.5可以使用列分区
3. 分区字段中如果有主键和唯一索引列，那么主键列和唯一列都必须包含进来
4. 分区表中无法使用外键约束
5. 需要对现有的表结构进行修改
6. 所有分区都必须使用相同的存储引擎
7. 分区函数中可以使用的函数和表达式会有一些限制
8. 某些存储引擎不支持分区
9. 对于MyISAM的分区表，不能使用load index into cache
10. 对于MyISAM表，使用分区表时需要打开更多的文件描述符

**工作原理**

通过一些HASH算法或者工具实现将一张数据表垂直或者水平进行物理切分。

**使用场景**

1. 单表记录条数达到百万到千万级别时
2. 解决表锁的问题

**分表方式**

水平分割

表很大，分割后可以降低在查询时需要读的数据和索引的页数，同时也降低了索引的层数，提高查询速度。

**使用场景**

1. 表中的数据本身就有独立性，例如表中分表记录各个地区的数据或者不同时期的数据，特别是有些数据常用，有些不常用
2. 需要把数据存放在多个介质上

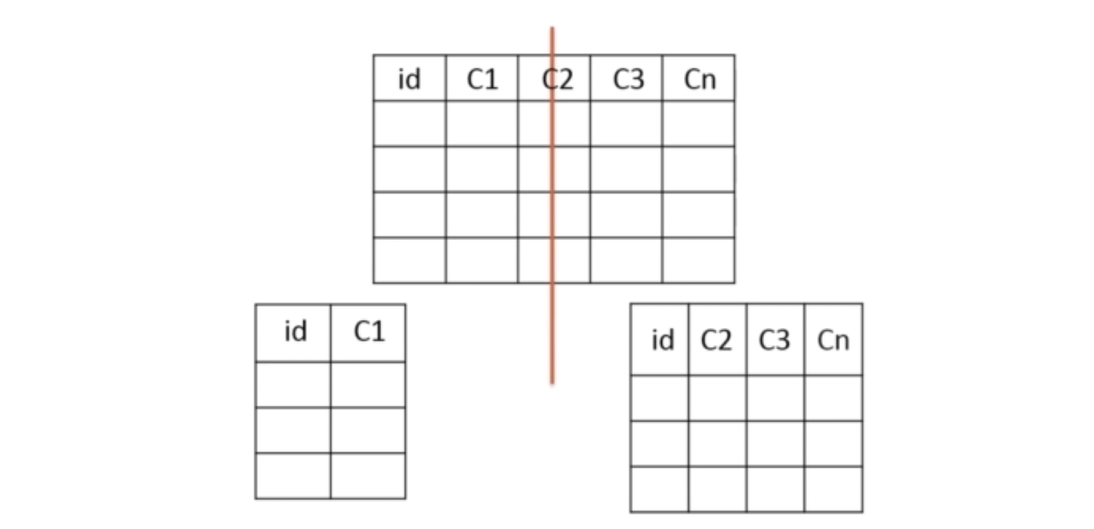
**水平分表缺点**

1. 给应用增加复杂度通常查询时需要多个表名，查询所有数据都需UNION操作
2. 在许多数据库应用中，这种复杂性会超过它带来的优点，查询时会增加读一个索引层的磁盘次数。

**分表方式**

**垂直分表**

把主键和一些列放在一个表，然后把主键和另外的列放在另一个表中

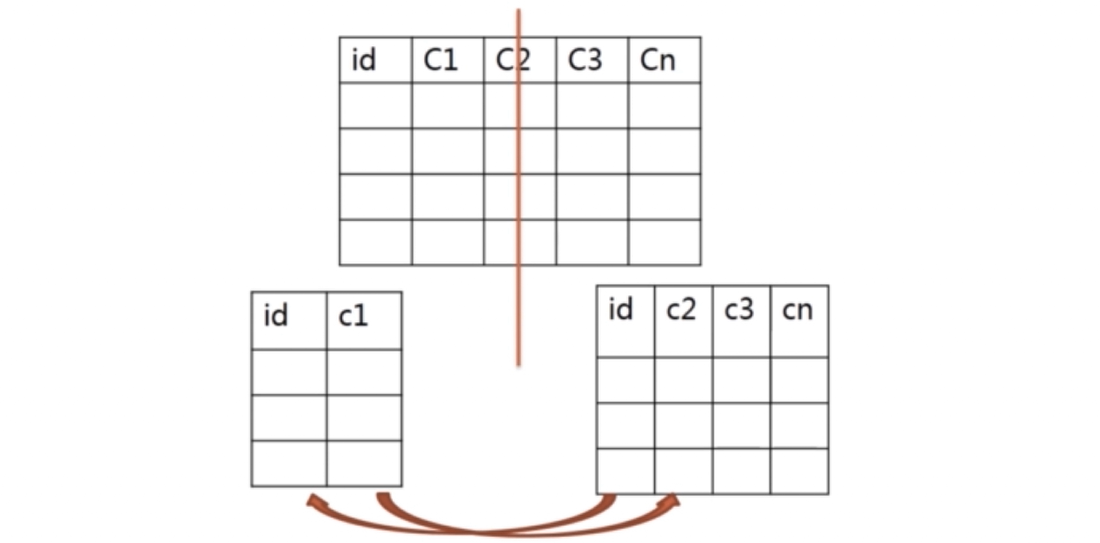


**使用场景**

1. 如果一个表中某些列常用，而另外一些列不常用
2. 可以使用数据行变小，一个数据页能存储更多数据，查询时减少I/O次数

**垂直分表缺点**

管理冗余列，查询所有数据需要JOIN操作



**分表缺点**

有些分表的策略基于应用层的逻辑算法，一旦逻辑算法改变，整个分表逻辑都回改变，扩展性较差。

对于应用层来说，逻辑算法无疑增加开发成本。

##### MySQL的复制原理及负载均衡

**MySQL主从复制工作原理**

在主库上把数据更改记录到二进制日志

从库将主库的日志复制到自己的中继日志

从库读取中继日志中的事件，将其重放到从库数据中

**MySQL主从复制解决的问题**

数据分布：随意停止或开始复制，并在不同地理位置分布数据备份

负载均衡：降低单个服务器的压力

高可用和故障切换：帮助应用程序避免单点失败

升级测试：可以使用更高版本的MySQL作为从库

**解题方法**

充分掌握分区和分表的工作原理和适用场景，在面试中，此类题通常比较灵活，会给一些现有公司遇到问题的场景，大家可以根据分区和分表以及MySQL复制，负载均衡的适用场景来根据情况进行回答。

例题：

设定网站的用户数量在千万级，但是活跃用户的数量只有1%，如何通过优化数据库提高活跃用户的访问速度？

答：

我们什么时候会用到我们的分区和分表，刚才已经说过了分区和分表的适用场景是什么呀，是不是都会有一个适用场景叫做什么呀，

我们是不是可以把常用的数据分出来对不对把不常用的数据我们也分到另外一个地方所以在这里面呢无论你使用分区还是使用分库分表

都可以来解答我们这道题。如果使用分区的话我们就这样来解答我们可以使用什么呀，MYSQL的分区因为mysql分区可以帮助我们什么呀

按照我们的一个规则按照什么样的规则我们可以把活跃用户分在一个区不活跃的用户分到另外一个区然后我们在进行查询的时候在进行

操作活跃用户的时候我们是不是只操作活跃用户的那个区啊本身这个区就数据是比较小的可以提高我们的访问数度

另外一种方式我们就可以通过分库分表，我们可以通过什么呀，水平切分的方式对不对然后呢把我们的活跃用户的数据呢切分成一个表

不活跃用户的数据放到另外一张表里面去我们在查询的时候呢只查询什么活跃用户的数据表是不是就可以了！所以在这当中呢我们可以

根据咱们的适用场景我们就能知道这个地方可以使用水平分表或者是咱们的分区就可以来解决所以本类考点我们是不是就知道该怎样的去解答了

# MYSQL的安全性

SQL查询的安全方案

MySQL的其他安全设置

1. 使用预处理语句防止SQL注入

PDO里的prepare

预处理语句:

Delete from user where id=?;

1. 写入数据库的数据要进行特殊字符的转义

比如里面带的引号，双引号这样的内容都需要把它转义掉，中是在我们的应用层上去做一些处理的。也是为了防止里面的SQL注入

3、查询错误信息不要返回给用户，将错误记录到日志

比如我们在查询的时候返回了一些错误信息，我们不要把这些错误反回到我们的应用当中有时候应用里面的信息会显示给我们的用户

用户会获取到我们的数据库里面的一些信息这样的话本身来说就是一种不安全的显示。所在这当中我们可以把错误给他屏蔽掉然后记录到我

们的错误日志里面去，我定期查询错误的时候我看日志就可以了。

**注意：**PHP端尽量使用PDO对数据库进行相关操作，PDO拥有对预处里语句很好的支持的方法，MySQLi也有，但是可扩展性不如PDO，效率略高于PDO，MySQL函数在新版本中已经趋向于淘汰，所以不建议使用，而且它没有很好的支持预处理的方法。这就是为什么我们经常会说pdo和mysqli要比我们的mysql函数库的安全性要高它的原因在这。

**哪对于mysql的其他的安全设置来说我们在这也给大家简单的去说一下**

1. 定期做数据备份
2. 不给查询用户root权限，合理分配权限
3. 关闭远程访问数据库权限

你不能说我们可以远程访问他，对吧我们通常情况下不允许远程访问这样的话有可能会被破解掉你的密码注意一下

如果有黑客惦记你的数据库的时候他会做一个暴力破解或者说通过一些其他的方式来破解对我们的数据库安全来说是非常不利的注意一下。

1. 修改root口令，不用默认口令，使用较复杂的口令
2. 删除多余的用户
3. 改变roo用户的名称
4. 限制一般用户浏览其他库
5. 限制用户对数据文件的访问权限

因为我们都知道怎么的mysql所有的数据是不是都存储在咱们的文件里面所以在这我们要对文件来做一些权限的限制比如说我们三个应用每个应用都有各自的一个用户

各自的用户不光要限制他的mysql的权限还要设置什么呀还要设置咱们的mysql data目录下的每一个库的权限每一个目录的权限

解题方法

通常情况下，SQL安全的考点都在防SQL注入的问题，因此只要遇到此类考点，优先考虑SQL注入的防护手段。

**为什么使用PDO和MySQLi连接数据库会比mysql函数库更加安全？**

因为PDO和mysqli本身呢支持我们的预处理的方式所以预处理方式呢本身就可以防我们sql注入，所以在这要比我们的mysql函数库要更加的安全，而mysql函数库本身是不支持预处理的。