几道常见SQL面试题

SQL数据库开发 7月13日

微信扫一扫
关注该公众号

点击关注上方“SQL数据库开发”，
设为“置顶或星标”，第一时间送达干货

SQL专栏

SQL基础知识第二版
SQL高级知识第二版

作者：你在我家门口

juejin.im/post/5cabf5025188251afe0a76ad

前言

今天楼主给大家列一下关于MySQL数据库几个常见面试题，如果大家对其中的问题感兴趣，可以自行扩展研究。

1. UNION ALL 与 UNION 的区别

- UNION和UNION ALL关键字都是将两个结果集合并为一个。
- UNION在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录再返回结果。
- 而UNION ALL只是简单的将两个结果合并后就返回。
- 由于UNION需要排序去重，所以 UNION ALL 的效率比 UNION 好很多。

2. TRUNCATE 与 DELETE 区别

- TRUNCATE 是DDL语句，而 DELETE 是DML语句。
- TRUNCATE 是先把整张表drop调，然后重建该表。而 DELETE 是一行一行的删除，所以 TRUNCATE 的速度肯定比 DELETE 速度快。
- TRUNCATE 不可以回滚，DELETE 可以。
- TRUNCATE 执行结果只是返回0 rows affected，可以解释为没有返回结果。
- TRUNCATE 会重置水平线（自增长列起始位），DELETE 不会。
- TRUNCATE 只能清理整张表，DELETE 可以按照条件删除。

一般情景下，TRUNCATE性能比DELETE好一点。

3. TIMESTAMP 与 DATETIME 的区别

相同点

TIMESTAMP 列的显示格式与 DATETIME 列相同。显示列宽固定在19字符，并且格式为YYYY-MM-DD HH:MM:SS。

不同点

TIMESTAMP

- 4个字节存储，时间范围：1970-01-01 08:00:01~2038-01-19 11:14:07。
- 值以UTC格式保存，涉及时区转化，存储时对当前的时区进行转换，检索时再转换回当前的时区。

DATETIME

- 8个字节存储，时间范围：1000-10-01 00:00:00~9999-12-31 23:59:59。
- 实际格式存储，与时区无关。

4. 什么是联合索引

两个或更多个列上的索引被称作联合索引，联合索引又叫复合索引。

5. 为什么要使用联合索引

- **减少开销：** 建一个联合索引(col1,col2,col3)，实际相当于建了(col1),(col1,col2),(col1,col2,col3)三个索引。减少磁盘空间的开销。
- **覆盖索引：** 对联合索引(col1,col2,col3)，如果有如下的sql：select col1,col2,col3 from test where col1=1 and col2=2。那么MySQL可以直接通过遍历索引取得数据，而无需回表，这减少了很多的随机io操作。覆盖索引是主要的提升性能的优化手段之一。
- **效率高：** 索引列越多，通过索引筛选出的数据越少。有1000W条数据的表，有如下sql `select from table where col1=1 and col2=2 and col3=3`，假设假设每个条件可以筛选出10%的数据，如果只有单值索引，那么通过该索引能筛选出 `1000W * 10%=100w` 条数据，然后再回表从100w条数据中找到符合 `col2=2 and col3= 3` 的数据，然后再排序，再分页；如果是联合索引，通过索引筛选出 `1000w * 10% * 10% * 10%=1w`，效率得到明显提升。

6. MySQL 联合索引最左匹配原则

- 在 MySQL 建立联合索引时会遵循最左前缀匹配的原则，即最左优先，在检索数据时从联合索引的最左边开始匹配。
- MySQL 会一直向右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)的索引则都可以用到，a,b,d的顺序可以任意调整。
- = 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的查询优化器会帮你优化成索引可以识别的形式。

7. 什么是聚集和非聚集索引

- 聚集索引就是以主键创建的索引。
- 非聚集索引就是以非主键创建的索引。

8. 什么是覆盖索引

覆盖索引（covering index）指一个查询语句的执行只用从索引页中就能够取得（如果不是聚集索引，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），避免了查到索引后，再做回表操作，减少I/O提高效率。

可以结合第10个问题更容易理解。

9. 什么是前缀索引

前缀索引就是对文本的前几个字符（具体是几个字符在创建索引时指定）创建索引，这样创建起来的索引更小。但是MySQL不能在ORDER BY或GROUP BY中使用前缀索引，也不能把它们用作覆盖索引。

创建前缀索引的语法：

```
ALTER TABLE table_name ADD
KEY(column_name(prefix_length))
```

10. InnoDB 与 MyISAM 索引存储结构的区别

- MyISAM索引文件和数据文件是分离的，索引文件仅保存数据记录的地址。
- 而在InnoDB中，表数据文件本身就是按B+Tree组织的一个索引结构，这棵树的叶节点data域保存了完整的数据记录。这个索引的key是数据表的主键，因此InnoDB表数据文件本身就是主索引，所以必须有主键，如果没有显示定义，自动为生成一个隐含字段作为主键，这个字段长度为6个字节，类型为长整型。
- InnoDB的辅助索引（Secondary Index，也就是非主键索引）存储的只是主键列和索引列，如果主键定义的比较大，其他索引也将很大。
- MyISAM引擎使用B+Tree作为索引结构，索引文件叶节点的data域存放的是数据记录的地址，指向数据文件中对应的值，每个节点只有该索引列的值。
- MyISAM主索引和辅助索引（Secondary key）在结构上没有任何区别，只是主索引要求key是唯一的，辅助索引可以重复，（由于MyISAM辅助索引在叶子节点上存储的是数据记录的地址，和主键索引一样，所以不需要再遍历一次主键索引）。

简单的说：

- **主索引的区别：** InnoDB的数据文件本身就是索引文件。而MyISAM的索引和数据是分开的。
- **辅助索引的区别：** InnoDB的辅助索引data域存储相应记录主键的值而不是地址。而MyISAM的辅助索引和主索引没有多大区别。

11. 为什么尽量选择单调递增数值类型的主键

InnoDB中数据记录本身被存于主索引（B+树）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结点和位置，如果页面达到装载因子（InnoDB默认为15/16），则开辟一个新的页。

如果使用自增主键，那么每次插入新的记录，记录就会顺序添加到当前索引结点的后续位置，当一页写满，就会自动开辟一个新的页，这样就会形成一个紧凑的索引结构，近似顺序填满。由于每次插入时也不需要移动已有数据，因此效率很高，也不会增加很多开销在维护索引上。

如果使用非自增主键，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入到现有索引页的中间某个位置，此时MySQL不得不为了将新记录查到合适位置而移动元素，甚至目标页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。

简单的说：

索引树只能定位到某一页，每一页内的插入还是需要通过比较、移动插入的。所以有序主键可以提升插入效率。

12. 建表时，int 后面的长度的意义

int占多少个字节，已经是固定的了，长度代表了显示的最大宽度。如果不够会用0在左边填充，但必须搭配zerofill使用。也就是说，int的长度并不影响数据的存储精度，长度只和显示有关。

13. SHOW INDEX 结果字段代表什么意思

Table：

- 表名。

Non_unique：

- 0：该索引不含重复值。
- 1：该索引可含有重复值。

Key_name：

- 索引名称，如果是注解索引，名称总是为PRIMARY。

Seq_in_index：

- 该列在索引中的序号，从 1 开始。例如：存在联合索引 idx_a_b_c (a,b,c)，则a的Seq_in_index=1，b=2，c=3。

Column_name：

- 列名。

Collation：

- 索引的排列顺序：A（ascending），D (descending)，NULL (not sorted)。

Cardinality：

- 一个衡量该索引的唯一程度的值，可以使用ANALYZE TABLE（INNODB） 或者 myisamchk -a（MyISAM）更新该值。
- 如果表记录太少，该字段的意义不大。一般情况下，该值越大，索引效率越高。

Sub_part：

- 对于前缀索引，用于索引的字符个数。如果整个字段都加上了索引，则显示为NULL。

Null：

- YES：该列允许NULL值。
- ''：该列不允许NULL值。

Index_type：

- 索引类型，包括(BTREE, FULLTEXT, HASH, RTREE)。

1. 如何解决like'%字符串%'时索引失效?

LIKE问题：like 以通配符开头 ('%abc...')，mysql索引失效会变成全表扫描的操作。

- 罪魁祸首是%，不是LIKE，LIKE 条件是 type = range 级别
- %xxx%：全表扫描
- %xxx：全表扫描
- xxx%：range

解决办法：

使用覆盖索引，可以由 ALL 变为INDEX，为啥呢？覆盖索引之后就能使用使用索引进行全表扫描。这里要注意一下，使用符合索引的时候，命中一个字段就可以，不用全部命中。

14. MySQL高效分页

存在SQL：SELECT * FROM ttl_product_info ORDER BY id LIMIT N,M。其中 LIMIT N,M 存在的问题最大：取出N+M行，丢弃前N行，返回 N ~ N+M 行的记录，如果N值非常大，效率极差（表记录1500w，N=10000000,M=30 需要9秒）。

解决办法：SQL：SELECT id FROM ttl_product_info WHERE id > N LIMIT M，id 列是索引列，id > N属于 range 级别，效率自然高，然后从位置开始取30条记录，效率极高（表记录1500w，N=10000000,M=30，需要0.9毫秒）。

当然想要实现上述效果的前提是：

- id是唯一索引，而且单调递增。
- N 的值是上一次查询的记录的最后一条id，（需要前端保存一下，不能直接用传统的方法获得）
- 不支持跨页查询，只能按照第1，2，3，4页这样查询逐页查询。

总结

为了保持文章结构的完整性，这里强行加上一段总结。。。

参考文章：

> https://segmentfault.com/a/1190000015416513
> https://blog.csdn.net/bigtree_3721/article/details/73151472

最后给大家分享我写的SQL两件套：《SQL基础知识第二版》和《SQL高级知识第二版》的PDF电子版。里面有各个语法的解释、大量的实例讲解和批注等等，非常通俗易懂，方便大家跟着一起来实操。

有需要的可以下载学习，只需要在下面的公众号「数据前线」(非本号)，后台回复关键字：SQL，就行

数据前线

后台回复关键字：**1024**，获取一份精心整理的技术干货

后台回复关键字：**进群**，带你进入高手如云的交流群。

推荐阅读

- 知乎：程序员上班时戴耳机是在听什么？
- 我用一根网线，控制了整栋楼的网络
- 看不见的二本学校学生
- 一个程序员被骗去养猪
- 再见了，学术硕士！

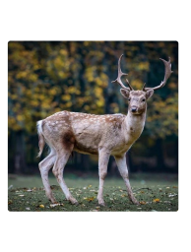
喜欢此内容的人还喜欢

16个 Redis 常见使用场景，面试有内容聊啦

顶级架构师

面试：如何从 100 亿 URL 中找出相同的 URL?

Java经验总结



面试官：如果要存ip地址，用什么数据类型比较好

IT哈哈