



持续更新中

回复“1024”获取Java架构师资源



微信搜一搜

Java研发军团

Java研发军团《Java面试手册》V1.0
公众号后台回复“面试手册”

Mysql面试题

1、数据库存储引擎

数据库存储引擎是数据库底层软件组织，数据库管理系统（DBMS）使用数据引擎进行创建、查询、更新和删除数据。不同的存储引擎提供不同的存储机制、索引技巧、锁定水平等功能，使用不同的存储引擎，还可以获得特定的功能。现在许多不同的数据库管理系统都支持多种不同的数据引擎。存储引擎主要有：1. MyIsam, 2. InnoDB, 3. Memory, 4. Archive, 5. Federated。

2、InnoDB (B+树)

InnoDB 底层存储结构为B+树，B树的每个节点对应innodb的一个page，page大小是固定的，一般设为16k。其中非叶子节点只有键值，叶子节点包含完整数据



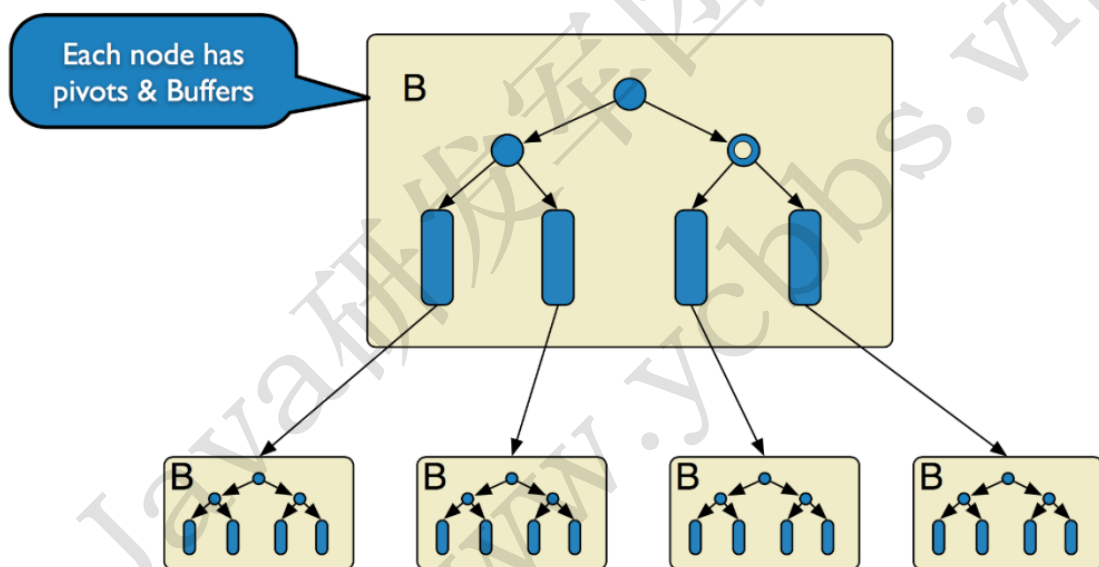
适用场景：

- 1) 经常更新的表，适合处理多重并发的更新请求。
- 2) 支持事务。
- 3) 可以从灾难中恢复（通过 bin-log 日志等）。
- 4) 外键约束。只有他支持外键。
- 5) 支持自动增加列属性 auto_increment。

2、TokuDB (Fractal Tree-节点带数据)

TokuDB 底层存储结构为 Fractal Tree, Fractal Tree 的结构与 B+树有些类似, 在 Fractal Tree 中, 每一个 child 指针除了需要指向一个 child 节点外, 还会带有一个 Message Buffer, 这个 Message Buffer 是一个 FIFO 的队列, 用来缓存更新操作。

例如, 一次插入操作只需要落在某节点的 Message Buffer 就可以马上返回了, 并不需要搜索到叶子节点。这些缓存的更新会在查询时或后台异步合并应用到对应的节点中。



TokuDB 在线添加索引, 不影响读写操作, 非常快的写入性能, Fractal-tree 在事务实现上有优势。他主要适用于访问频率不高的数据或历史数据归档

3、MyIASM

MyIASM是 MySQL默认的引擎, 但是它没有提供对数据库事务的支持, 也不支持行级锁和外键, 因此当 INSERT(插入)或 UPDATE(更新)数据时即写操作需要锁定整个表, 效率便会低一些。

ISAM 执行读取操作的速度很快, 而且不占用大量的内存和存储资源。在设计之初就预想数据组织成有固定长度的记录, 按顺序存储的。---ISAM 是一种静态索引结构。

缺点是它不 支持事务处理。

4、Memory

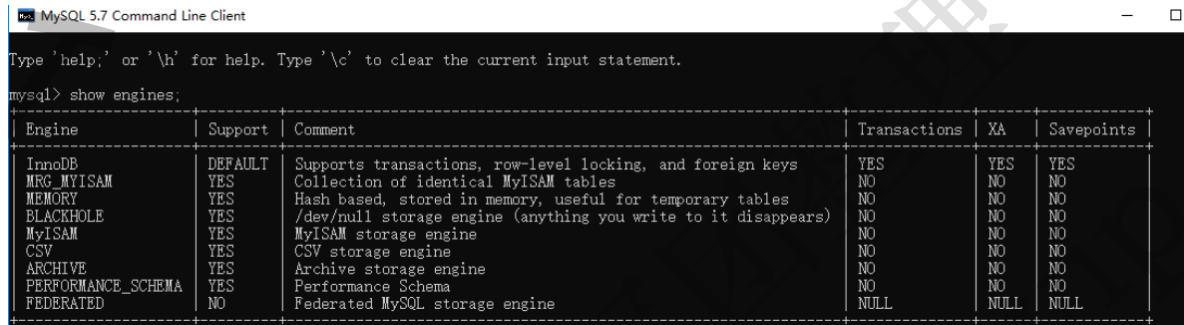
Memory (也叫 HEAP) 堆内存: 使用存在内存中的内容来创建表。每个 MEMORY 表只实际对应一个磁盘文件。MEMORY 类型的表访问非常得快, 因为它的数据是放在内存中的, 并且默认使用HASH 索引。但是一旦服务关闭, 表中的数据就会丢失掉。Memory 同时支持散列索引和 B 树索引, B树索引可以使用部分查询和通配查询, 也可以使用 <, > 和 = 等操作符方便数据挖掘, 散列索引相等的比较快但

是对于范围的比较慢很多

5、数据库引擎有哪些

如何查看mysql提供的所有存储引擎

```
mysql> show engines;
```



Engine	Support	Comment	Transactions	XA	Savepoints
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
MyISAM	YES	MyISAM storage engine	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL

mysql常用引擎包括：MYISAM、Innodb、Memory、MERGE

1. MYISAM：全表锁，拥有较高的执行速度，不支持事务，不支持外键，并发性能差，占用空间相对较小，对事务完整性没有要求，以select、insert为主的应用基本上可以使用这引擎
2. Innodb:行级锁，提供了具有提交、回滚和崩溃回复能力的事务安全，支持自动增长列，支持外键约束，并发能力强，占用空间是MYISAM的2.5倍，处理效率相对会差一些
3. Memory:全表锁，存储在内容中，速度快，但会占用和数据量成正比的内存空间且数据在mysql重启时会丢失，默认使用HASH索引，检索效率非常高，但不适用于精确查找，主要用于那些内容变化不频繁的代码表
4. MERGE：是一组MYISAM表的组合

6、InnoDB与MyISAM的区别

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；
2. 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；
3. 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。
4. 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；
5. Innodb不支持全文索引，而MyISAM支持全文索引，查询效率上MyISAM要高

7、索引

索引 (Index) 是帮助 MySQL 高效获取数据的数据结构。常见的查询算法,顺序查找,二分查找,二叉排序树查找,哈希散列法,分块查找,平衡多路搜索树 B 树 (B-tree)，索引是对数据库表中一个或多个列的值进行排序的结构，建立索引有助于快速获取信息。

你也可以这样理解：索引就是加快检索表中数据的方法。数据库的索引类似于书籍的索引。在书籍中，索引允许用户不必翻阅完整本书就能迅速地找到所需要的信息。在数据库中，索引也允许数据库程序迅速地找到表中的数据，而不必扫描整个数据库

mysql 有4种不同的索引：

主键索引 (PRIMARY)

唯一索引 (UNIQUE)

普通索引 (INDEX)

全文索引 (FULLTEXT)

索引并非是越多越好，创建索引也需要耗费资源，一是增加了数据库的存储空间，二是在插入和删除时要花费较多的时间维护索引

索引加快数据库的检索速度

索引降低了插入、删除、修改等维护任务的速度

唯一索引可以确保每一行数据的唯一性

通过使用索引，可以在查询的过程中使用优化隐藏器，提高系统的性能

索引需要占物理和数据空间

8、常见索引原则有

1. 选择唯一性索引，唯一性索引的值是唯一的，可以更快速的通过该索引来确定某条记录。
2. 为经常需要排序、分组和联合操作的字段建立索引。
3. 为常用作查询条件的字段建立索引。
4. 限制索引的数目：
越多的索引，会使更新表变得很浪费时间。尽量使用数据量少的索引
5. 如果索引的值很长，那么查询的速度会受到影响。尽量使用前缀来索引
6. 如果索引字段的值很长，最好使用值的前缀来索引。
7. 删除不再使用或者很少使用的索引
8. 最左前缀匹配原则，非常重要的原则。
9. 尽量选择区分度高的列作为索引区分度的公式是表示字段不重复的比例
10. 索引列不能参与计算，保持列“干净”：带函数的查询不参与索引。
11. 尽量的扩展索引，不要新建索引

9、数据库的三范式是什么

第一范式：列不可再分

第二范式：行可以唯一区分，主键约束

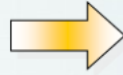
第三范式：表的非主属性不能依赖与其他表的非主属性 外键约束

且三大范式是一级一级依赖的，第二范式建立在第一范式上，第三范式建立第一第二范式上。

10、第一范式(1st NF - 列都是不可再分)

第一范式的目标是确保每列的原子性:如果每列都是不可再分的最小数据单元（也称为最小的原子单元），则满足第一范式（1NF）

BuyerID	Address
1	中国北京市
2	美国纽约市
3	英国利物浦
4	日本东京市
...	...



BuyerID	Country	City
1	中国	北京
1	中国	北京
4	日本	东京
2	美国	纽约
...

11、第二范式(2nd NF - 每个表只描述一件事情)

首先满足第一范式，并且表中非主键列不存在对主键的部分依赖。第二范式要求每个表只描述一件事情。

Orders		Orders	
字段	例子	字段	例子
订单编号	001	订单编号	001
产品编号	A001	订购日期	2000-2-3
订购日期	2000-2-3		
价格	\$29.00		
...	...		

Products	
字段	例子
产品编号	A001
价格	\$29.00

12、第三范式(3rd NF - 不存在对非主键列的传递依赖)

第三范式定义是，满足第二范式，并且表中的列不存在对非主键列的传递依赖。除了主键订单编号外，顾客姓名依赖于非主键顾客编号。

Orders		Orders	
字段	例子	字段	例子
订单编号	001	订单编号	001
订购日期	2000-2-3	订购日期	2000-2-3
顾客编号	AB001	顾客编号	AB001
顾客姓名	Tony		
...	...		

13、数据库是事务

事务(Transaction)是作为单个逻辑工作单元执行的一系列操作，这些操作作为一个整体一起向系统提交，要么都执行、要么都不执行。事务是一个不可分割的工作逻辑单元，事务必须具备以下四个属性，简称 ACID 属性：

原子性 (Atomicity)

1. 事务是一个完整的操作。事务的各步操作是不可分的（原子的）；要么都执行，要么都不执行。

一致性 (Consistency)

2. 当事务完成时，数据必须处于一致状态。

隔离性 (Isolation)

3. 对数据进行修改的所有并发事务是彼此隔离的，这表明事务必须是独立的，它不应以任何方式依赖于或影响其他事务。

永久性 (Durability)

4. 事务完成后，它对数据库的修改被永久保持，事务日志能够保持事务的永久性

14、SQL优化

- 1、查询语句中不要使用select *
- 2、尽量减少子查询，使用关联查询 (left join,right join,inner join) 替代
- 3、减少使用IN或者NOT IN ,使用exists , not exists或者关联查询语句替代
- 4、or 的查询尽量用 union或者union all 代替(在确认没有重复数据或者不用剔除重复数据时，union all会更好)
- 5、应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。
- 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，如： select id from t where num is null 可以在num上设置默认值0，确保表中num列没有null值，然后这样查询： select id from t where num=0

15、简单说一说drop、delete与truncate的区别

SQL中的drop、delete、truncate都表示删除，但是三者有一些差别

delete和truncate只删除表的数据不删除表的结构

速度,一般来说: drop> truncate >delete

delete语句是dml,这个操作会放到rollback segment中,事务提交之后才生效;

如果有相应的trigger,执行的时候将被触发. truncate,drop是ddl, 操作立即生效,原数据不放到rollbacksegment中,不能回滚. 操作不触发trigger

16、什么是视图

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

17、什么是内联接、左外联接、右外联接？

内联接 (Inner Join)：匹配2张表中相关联的记录。

左外联接 (Left Outer Join)：除了匹配2张表中相关联的记录外，还会匹配左表中剩余的记录，右表中未匹配到的字段用NULL表示。

右外联接 (Right Outer Join)：除了匹配2张表中相关联的记录外，还会匹配右表中剩余的记录，左表中未匹配到的字段用NULL表示。在判定左表和右表时，要根据表名出现在Outer Join的左右位置关系

18、并发事务带来哪些问题？

在典型的应用程序中，多个事务并发运行，经常会操作相同的数据来完成各自的任务（多个用户对同一数据进行操作）。并发虽然是必须的，但可能会导致以下的问题。

脏读 (Dirty read) : 当一个事务正在访问数据并且对数据进行了修改，而这种修改还没有提交到数据库中，这时另外一个事务也访问了这个数据，然后使用了这个数据。因为这个数据是还没有提交的数据，那么另外一个事务读到的这个数据是“脏数据”，依据“脏数据”所做的操作可能是不正确的。

丢失修改 (Lost to modify) : 指在一个事务读取一个数据时，另外一个事务也访问了该数据，那么在第一个事务中修改了这个数据后，第二个事务也修改了这个数据。这样第一个事务内的修改结果就被丢失，因此称为丢失修。例如：事务1读取某表中的数据A=20，事务2也读取A=20，事务1修改A=A-1，事务2也修改A=A-1，最终结果A=19，事务1的修改被丢失。

不可重复读 (Unrepeatableread) : 指在一个事务内多次读同一数据。在这个事务还没有结束时，另一个事务也访问该数据。那么，在第一个事务中的两次读数据之间，由于第二个事务的修改导致第一个事务两次读取的数据可能不太一样。这就发生了在一个事务内两次读到的数据是不一样的情况，因此称为不可重复读。

幻读 (Phantom read) : 幻读与不可重复读类似。它发生在一个事务 (T1) 读取了几行数据，接着另一个并发事务 (T2) 插入了一些数据时。在随后的查询中，第一个事务 (T1) 就会发现多了一些原本不存在的记录，就好像发生了幻觉一样，所以称为幻读。

不可重复读和幻读区别：

不可重复读的重点是修改比如多次读取一条记录发现其中某些列的值被修改，幻读的重点在于新增或者删除比如多次读取一条记录发现记录增多或减少了

19、事务隔离级别有哪些?MySQL的默认隔离级别是?

SQL 标准定义了四个隔离级别：

READ-UNCOMMITTED(读取未提交) : 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(读取已提交) : 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(可串行化) : 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

隔离级别	脏读	不可重复读	幻影读
READ-UNCOMMITTED	√	√	√
READ-COMMITTED	×	√	√
REPEATABLE-READ	×	×	√
SERIALIZABLE	×	×	×

MySQL InnoDB 存储引擎的默认支持的隔离级别是 REPEATABLE-READ (可重复)。我们可以通过 SELECT @@tx_isolation; 命令来查看


```
mysql> SELECT @@tx_isolation;
+-----+
| @@tx_isolation |
+-----+
| REPEATABLE-READ |
+-----+
```

这里需要注意的是：与 SQL 标准不同的地方在于 InnoDB 存储引擎在 REPEATABLE-READ（可重读）事务隔离级别下使用的是 Next-Key Lock 锁算法，因此可以避免幻读的产生，这与其他数据库系统(如 SQL Server)是不同的。所以说 InnoDB 存储引擎的默认支持的隔离级别是 REPEATABLE-READ（可重读）已经可以完全保证事务的隔离性要求，即达到了 SQL 标准的 SERIALIZABLE(可串行化) 隔离级别。因为隔离级别越低，事务请求的锁越少，所以大部分数据库系统的隔离级别都是 READCOMMITTED(读取提交内容)，但是你要知道的是 InnoDB 存储引擎默认使用 REPEATABLE-READ（可重读）并不会有任何性能损失

InnoDB 存储引擎在 分布式事务 的情况下一般会用到 SERIALIZABLE(可串行化) 隔离级别。

20、大表如何优化？

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；

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：可以使得列数据变小，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；

21、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。



水平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，所以水平拆分最好分库。

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，跨节点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。

1. 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。

2. 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。

详细内容可以参考：MySQL大表优化方案: <https://segmentfault.com/a/1190000006158186>

22、分库分表之后,id 主键如何处理

因为要是分成多个表之后，每个表都是从1开始累加，这样是不对的，我们需要一个全局唯一的id来支持。

生成全局id有下面这几种方式：

UUID：不适合作为主键，因为太长了，并且无序不可读，查询效率低。比较适合用于生成唯一的名字的标示比如文件的名字。

数据库自增id：两台数据库分别设置不同步长，生成不重复ID的策略来实现高可用。这种方式生成的id有序，但是需要独立部署数据库实例，成本高，还会有性能瓶颈。

利用redis生成id：性能比较好，灵活方便，不依赖于数据库。但是，引入了新的组件造成系统更加复杂，可用性降低，编码更加复杂，增加了系统成本。

Twitter的snowflake算法： Github 地址：<https://github.com/twitter-archive/snowflake>。

美国的Leaf分布式ID生成系统：Leaf 是美团开源的分布式ID生成器，能保证全局唯一性、趋势递增、单调递增、信息安全，里面也提到了几种分布式方案的对比，但也需要依赖关系数据库、Zookeeper等中间件。感觉还不错。美团技术团队的一篇文章：<https://tech.meituan.com/2017/04/21/mt-leaf.html>

23、存储过程(特定功能的 SQL 语句集)

一组为了完成特定功能的 SQL 语句集，存储在数据库中，经过第一次编译后再次调用不需要再次编译，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。

24、存储过程优化思路

1. 尽量利用一些 sql 语句来替代一些小循环，例如聚合函数，求平均函数等。
2. 中间结果存放于临时表，加索引。
3. 少使用游标。sql 是个集合语言，对于集合运算具有较高性能。而 cursors 是过程运算。比如对一个 100 万行的数据进行查询。游标需要读表 100 万次，而不使用游标则只需要少量几次读取。
4. 事务越短越好。sqlserver 支持并发操作。如果事务过多过长，或者隔离级别过高，都会造成并发操作的阻塞，死锁。导致查询极慢，cpu 占用率极高。
5. 使用 try-catch 处理错误异常。
6. 查找语句尽量不要放在循环内

25、触发器(一段能自动执行的程序)

触发器是一段能自动执行的程序，是一种特殊的存储过程，触发器和普通的存储过程的区别是：触发器是当对某一个表进行操作时触发。诸如：update、insert、delete 这些操作的时候，系统会自动调用执行该表上对应的触发器。SQL Server 2005 中触发器可以分为两类：DML 触发器和DDL 触发器，其中 DDL 触发器它们会影响多种数据定义语言语句而激发，这些语句有 create、alter、drop 语句。

26、数据库并发策略

并发控制一般采用三种方法，分别是乐观锁和悲观锁以及时间戳。

27、MySQL 中有哪几种锁？

- 1、表级锁：开销小，加锁快；不会出现死锁；锁定粒度大，发生锁冲突的概率最高，并发度最低。
- 2、行级锁：开销大，加锁慢；会出现死锁；锁定粒度最小，发生锁冲突的概率最低，并发度也最高。
- 3、页级锁：开销和加锁时间界于表锁和行锁之间；会出现死锁；锁定粒度界于表锁和行锁之间，并发度一般

28、MySQL 中有哪些不同的表格？

共有 5 种类型的表格：

- 1、MyISAM
- 2、Heap
- 3、Merge
- 4、INNODB
- 5、ISAM

29、简述在 MySQL 数据库中 MyISAM 和 InnoDB 的区别

MyISAM：

不支持事务，但是每次查询都是原子的；
支持表级锁，即每次操作是对整个表加锁；
存储表的总行数；

一个 MYISAM 表有三个文件：索引文件、表结构文件、数据文件；
采用非聚集索引，索引文件的数据域存储指向数据文件的指针。辅索引与主索引基本一致，但是辅索引不用保证唯一性。

InnoDB：

支持 ACID 的事务，支持事务的四种隔离级别；
支持行级锁及外键约束：因此可以支持写并发；

不存储总行数：

一个 InnoDB 引擎存储在一个文件空间（共享表空间，表大小不受操作系统控制，一个表可能分布在多个文件里），也有可能为多个（设置为独立表空间，表大小受操作系统文件大小限制，一般为 2G），受操作系统文件大小的限制；

主键索引采用聚集索引（索引的数据域存储数据文件本身），辅索引的数据域存储主键的值；因此从辅索引查找数据，需要先通过辅索引找到主键值，再访问辅索引；最好使用自增主键，防止插入数据时，为维持 B+树结构，文件的大调整。

30、MySQL 中 InnoDB 支持的四种事务隔离级别名称，以及逐级之间的区别？

SQL 标准定义四个隔离级别为：

- 1、read uncommitted：读到未提交数据
- 2、read committed：脏读，不可重复读
- 3、repeatable read：可重读
- 4、serializable：串行事物

31、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。

32、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。

33、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。

34、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。

35、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。

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

36、你怎么看到为表格定义的所有索引？

索引是通过以下方式为表格定义的：

```
SHOW INDEX FROM <tablename>;
```

37、LIKE 声明中的%和_是什么意思？

%对应于 0 个或多个字符，_只是 LIKE 语句中的一个字符

如何在 Unix 和 MySQL 时间戳之间进行转换？

UNIX_TIMESTAMP 是从 MySQL 时间戳转换为 Unix 时间戳的命令

FROM_UNIXTIME 是从 Unix 时间戳转换为 MySQL 时间戳的命令

38、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。

39、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。

40、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。

41、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

42、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。

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

43、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;
```

44、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。

45、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。

46、什么是非标准字符串类型？

- 1、TINYTEXT
- 2、TEXT
- 3、MEDIUMTEXT
- 4、LONGTEXT

47、什么是通用 SQL 函数？

- 1、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。
- 2、FORMAT(X, D)- 格式化数字 X 到 D 有效数字。
- 3、CURRDATE(), CURRTIME()- 返回当前日期或时间。
- 4、NOW () - 将当前日期和时间作为一个值返回。
- 5、MONTH () , DAY () , YEAR () , WEEK () , WEEKDAY () - 从日期值中提取给定数据。
- 6、HOUR () , MINUTE () , SECOND () - 从时间值中提取给定数据。
- 7、DATEDIFF (A , B) - 确定两个日期之间的差异，通常用于计算年龄
- 8、SUBTIMES (A , B) - 确定两次之间的差异。
- 9、FROMDAYS (INT) - 将整数天数转换为日期值

48、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。

49、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：

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。

```
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。

50、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。

51、列的字符串类型可以是什么？

字符串类型是：

- 1、SET
- 2、BLOB
- 3、ENUM
- 4、CHAR
- 5、TEXT

52、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？

- 1、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。
- 2、选择合适的表字段数据类型和存储引擎，适当的添加索引。
- 3、MySQL 库主从读写分离。
- 4、找规律分表，减少单表中的数据量提高查询速度。
- 5、添加缓存机制，比如 memcached，apc 等。
- 6、不经常改动的页面，生成静态页面。
- 7、书写高效率的 SQL。比如 SELECT * FROM TABEL 改为 SELECT field_1,field_2, field_3 FROM TABLE.

53、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。

54、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。

55、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）

56、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 阿里 P8 架构师谈：MySQL 慢查询优化、索引优化、以及表等优化总结

57、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

58、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

59、数据库中的事务是什么？

事务 (transaction) 是作为一个单元的一组有序的数据库操作。如果组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有操作完成，事务则提交，其修改将作用于所有其他数据库进程。如果一个操作失败，则事务将回滚，该事务所有操作的影响都将取消。

事务特性：

- 1、原子性：即不可分割性，事务要么全部被执行，要么就全部不被执行。

- 2、一致性或可串行性。事务的执行使得数据库从一种正确状态转换成另一种正确状态
- 3、隔离性。在事务正确提交之前，不允许把该事务对数据的任何改变提供给任何其他事物
- 4、持久性。事务正确提交后，其结果将永久保存在数据库中，即使在事务提交后有了其他故障，事务的处理结果也会得到保存。

或者这样理解：

事务就是被绑定在一起作为一个逻辑工作单元的 SQL 语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。为了确保要么执行，要么不执行，就可以使用事务。要将有组语句作为事务考虑，就需要通过 ACID 测试，即原子性，一致性，隔离性和持久性。

60、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。

61、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

62、存储时期

Datetime: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，datetime 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.datetime.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

63、对于关系型数据库而言，索引是相当重要的概念，请回答有关索引的几个问题

1、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

2、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。

3、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

4、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

64、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——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。

65、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

66、SQL 语言包括哪几部分？每部分都有哪些操作关键

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

67、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库数据扩散。
- 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)。

68、什么是锁？

答：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

69、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。

70、什么是存储过程？用什么来调用？

答：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。

71、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

72、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。

73、试述视图的优点？

- (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

74、NULL 是什么意思

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

75、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

76、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？

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。

77、说说对 SQL 语句优化有哪些方法？（选择几条）

- 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

78、什么是乐观锁

乐观锁认为一个用户读数据的时候，别人不会去写自己所读的数据；悲观锁就刚好相反，觉得自己读数据库的时候，别人可能刚好在写自己刚读的数据，其实就是持一种比较保守的态度；时间戳就是不加锁，通过时间戳来控制并发出现的问题。

79、什么是悲观锁

悲观锁就是在读取数据的时候，为了不让别人修改自己读取的数据，就会先对自己读取的数据加锁，只有自己把数据读完了，才允许别人修改那部分数据，或者反过来说，就是自己修改某条数据的时候，不允许别人读取该数据，只有等自己的整个事务提交了，才释放自己加上的锁，才允许其他用户访问那部分数据。

80、什么是时间戳

时间戳就是在数据库表中单独加一列时间戳，比如“TimeStamp”，每次读出来的时候，把该字段也读出来，当写回去的时候，把该字段加1，提交之前，跟数据库的该字段比较一次，如果比数据库的值大的话，就允许保存，否则不允许保存，这种处理方法虽然不使用数据库系统提供的锁机制，但是这种方法可以大大提高数据库处理的并发量，以上悲观锁所说的加“锁”，其实分为几种锁，分别是：排它锁（写锁）和共享锁（读锁）。

81、什么是行级锁

行级锁是一种排他锁，防止其他事务修改此行；在使用以下语句时，Oracle 会自动应用行级锁：

1. INSERT、UPDATE、DELETE、SELECT ... FOR UPDATE [OF columns] [WAIT n | NOWAIT];
2. SELECT ... FOR UPDATE 语句允许用户一次锁定多条记录进行更新
3. 使用 COMMIT 或 ROLLBACK 语句释放锁。

82、什么是表级锁

表示对当前操作的整张表加锁，它实现简单，资源消耗较少，被大部分 MySQL 引擎支持。最常使用的 MYISAM 与 INNODB 都支持表级锁定。表级锁定分为表共享读锁（共享锁）与表独占写锁（排他锁）。

83、什么是页级锁

页级锁是 MySQL 中锁定粒度介于行级锁和表级锁中间的一种锁。表级锁速度快，但冲突多，行级冲突少，但速度慢。所以取了折衷的页级，一次锁定相邻的一组记录。BDB 支持页级锁



持续更新中

回复“1024”获取Java架构师资源



微信搜一搜

Java研发军团

Java研发军团《Java面试手册》V1.0
公众号后台回复“面试手册”