链滴

MySQL 高频面试题

作者：dql

原文链接：https://ld246.com/article/1681863698561

来源网站：链滴

1.请你谈谈 MySQL 事务隔离级别，MySQL 的默认隔离级别是什么？为了达到事务四大特性，数据库定义了 4 种不同的事务隔离级别：

READ-UNCOMMITTED（读取未提交）：最低的隔离级别，允许脏读，也就是可能读取 到其他会话未提交事务修改的数据，可能会导致脏读、幻读或不可重复读。

READ-COMMITTED（读取已提交）： 只能读取到已经提交的数据。Oracle 等多数数 据库默认都是级别 （不重复读），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。

MySQL 默认采用的 REPEATABLE_READ 隔离级别。

==2、可重复读解决了哪些问题？==

可重复读的核心就是一致性读(consistent read);保证多次读取同一个数据时，其值都和事 务开始时的**内容是一致**，禁止读取到别的事务未提交的数据，会造成幻读。

而事务更新数据的时候，只能用当前读。如果当前的记录的行锁被其他事务占用的话，就 需要进入锁待。

查询只承认在==事务启动前==就已经**提交**完成的**数据**。

可重复读解决的是重复读的问题，可重复读在**快照读**的情况下是*不会有幻读*，但**当**前读的候会**有幻读**。

==3、对 SQL 慢查询会考虑哪些优化？==

 分析语句，是否加载了不必要的字段/数据。

 分析 SQL 执行计划（expl索引信息。

 如果 SQL 很复杂，优化 SQL 结构。

 按照可能的优化点执行表结构变更、增加索引、SQL 改写等操作。

 查看优化后的执行时间和执行计划。

 如果表数据量太大，考虑分表。

 利用缓存，减少查询次数

4、==MySQL 为什么 InnoDB 是默认引擎？==

聚集索引是指数据库表行中数据的物理顺序与键值的逻辑（索引）顺序相同。**一个**表只能有*一 个聚簇索引**，因为一个表的**物理顺序只有一种**情况，所以，对应的聚簇索引只能有一个。聚簇索引的叶子节点就是数据节点，既存储索引值，又在叶子节点存储行数据。 Innodb 创建表后生成的件有： frm:创建表的语句 idb:表里面的**数据+索引文件**

5、==MySQL 索引底层结构为什么使用 B+树==?

 哈希虽然能够提供 O(1) 的单数据行操作性能，但是对于**范围查询**和**排序**却无法很好地支持，最终导致**全表扫描**；**B 树**能够在**非叶节子点中存储数据**，但是这也导致在**查询连续** 据时可能会带来**更多的随机 I/O**，而 **B+树**的所有叶节点可以通过**指针相互连接**，能够*减 少顺序遍历**时产生的额外**随机 I/O**； 第一，**B 树一个节点**里存的是**数据**，而 **B+**存储的是**索引（地址**），所以 **B 树**里**一个节 点**存不了很多**个数据**，但是 **B+树**个节点**能存**很多索引**，**B+树**叶子节点存**所有的数据**。

 第二，B+树的叶子节点是数据阶段用了**一个链表串联**起来，便于范围查找。

6、B+ 树的叶子节点链表是单向还是双向?

双向链表

==7、如何查询慢 SQL 产生的原因==

 分析 SQL 执行计划（explain extended），思考可能的优化点，是否命中索引等。

 没有索引或者没有用到索引(这是查询慢最常见的问题，是程序设计的缺陷)。

 内存不足。

 网络速度慢。

 是否查询出的数据量过大（可以采用多次查询，其他的方法降低数据量）。

 是否返回了不必要的行和列。

 锁或者死锁。

 I/O 吞吐量小，形成了瓶颈效应。

 sp_lock,sp_who,活动的用户查看,原因是读写竞争资源。

==8、索引失效的情况有哪些？==

 like **以%开头索引无效**，当 **like 以&
尾**，索引**有效**。

 or 语句前后没有同时使用索引，当且仅当** or 语句**查询条件的**前后列均为索引**时，**索 生效**。

 组合索引，使用的不是第一列索引时候，索引失效，即最**左匹配**规则。

 数据类型出现隐式转换，如 varchar 不加单引号的时候可能会自动转换为 int 类型，这 个时候引失效。

 在索引列上使用 IS NULL 或者 IS NOT NULL 时候，索引失效，因为索引是不索引空值 得。

 **在索引字段上使用，NOT、 <>
！ = **、时候是不会使用索引的，对于这样的处理只会进 行**全表扫描**。

 对索引字段进行计算操作，**函数操作**时不会使用索引。

 当全表扫描速度比索引速度快的时候不会使用索引。

9、==**MySQL 事务的特性有什么，说一下分别是什么意思？**==

 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。

 一致性或可串性。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。

 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务。

 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，事务处理结果也会得到保存。

==10、介绍下 MySQL 聚簇索引与非聚簇索引的区别（InnoDB 与 Myisam 引 擎）？==

聚簇索引：数据和索引放在一块

非聚簇索引： 数据和索引分开存储，索引结构的叶子节点指向数据的对应行

1

2

聚集索引是指数据库表行中数据的物理顺序与键值的逻辑（索引）顺序相同。一个表只能有一 个聚簇

引，因为一个表的物理顺序只有一种情况，所以，对应的**聚簇索引只能有一个**。 聚簇索引的**叶节点就是数据节点**，既**存储索引值**，又在叶子节点存储行数据。 Innodb 创建表后生成的文件： frm:创建表的语句 idb:表里面的数据+索引文件 **非聚集索引（**MyISAM 引擎的底层实现）的辑顺序与磁盘上行的物理存储顺序不同。

非聚簇索引的叶子节点仍然是**索引节点**，只不过有**指向对应数据块的指针**。索引命中后需要**回表查 询**。 Myisam 创建表后生成的文件有： **frm**:创建表的语句 **MYD**:表里面的数文件（myisam data） **MYI**:表里面的索引文件（myisam index）** innodb** 的**次索引指向主键的引用** (聚簇索引) **myisam** 的**次索引和主索引都指向物理行** (非聚簇索引)

==11、然后给一个联合索引(a,b)和一个语句,select * from table where b = ‘xxx’ ， 判断是否能中索引？为什么？==

不能命中。 对于查询 SELECT * FROM TABLE WHERE a=xxx and b=xxx，显然是可以使用（a，b）这个联合索引的。 对于单个的 a 列查询 SELECT * FROM TABLEWHERE a=xxx，也可以使用这个（，b） 索引。 但对于 b 列的查询 SELECT *FROM TABLE WHERE b=xxx，则不可以使用这棵 B+树索引。在 innoDb 数据引擎中，可以发现叶子节点上的 b 值为 1、2、1、4、1、2，显然不是排序 的因此对于 b 列的查询使用不到（a，b）的索引

==12、MySQL 索引分类?==

单列索引

 普通索引：MySQL 中基本索引类型，没有什么限制，允许在定义索引的列中插入重复值 和空，纯粹为了**查询数据更快一点**。

 唯一索引：索引列中的**值必须是唯一**的，但是允许为**空值**，

**** 主键索引**：是一种**特殊的唯一索引**，**不**允许有**空值**。

组合索引： 多个字段组合上创建的索引，只有在查询条件中使用了这些字段的左边字段时，索引会被使 用，使用组合索引时遵循**最左前缀集合**。

全文索引： 只有在 **MyISAM 引擎上才能使用**，只能CHAR,VARCHAR,TEXT 类型字段上使用文 索引，介绍了要求，说说什么是全文索引，就是在一堆文字中，通过其中的某个关键字等，就 能到该字段所属的记录行，比有"你是个靓仔，靓女 ..." 通过靓仔，可能就可以找到该条记录

空间索引： 空间索引是对空间数据类型的字段建立的索引，**MySQL 中的空间数据类型有四种*， GEOMETRY、POINT、LINESTRING、POLYGON。在创建空间索引时，使用 SPATIAL 关 键字要求，引擎为 MyISAM，创建空间索引的列，必须将其声明为 NOT NULL。

1==3、谈谈你对 SQL 注入式攻击的理解？==

所谓 SQL 注入式攻击，就是攻击者**把 SQL 命令插入到 Web 表单的输入域或页面请求的 查询字符**，**欺骗服务器执行恶意的 SQL 命令**。 如何防范 SQL 注入式攻击？ 在利用表单输入的内容构造 QL 命令之前，**把所有输入内容过滤一番**就可以了。过滤输入内 容可以按多种方式进行。

 对于动态构造 SQL 查询的场合 a. 替换单引号，即把所有单独出现的单引号改成两个单引号，防止击者修改 SQL 命令的含 义。 b. 删除用户输入内容中的所有连字符 c. 对于用来执行查询的数据库帐，限制其权限。用不同的用户帐户执行查询、插入、更新、 删除操作。

 用存储过程来执行所有的查询。

 限制表单或查询字符串输入的长度。

 检查用户输入的合法性。

 将用户登录名称、密码等数据加密保存。

 检查提取数据的查询所返回的记录数量。

14、==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（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，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**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 行这些写作时，还要操作索引文件。

==**15、幻读是什么，用什么隔离级别可以防止幻读?** ==

幻读是**一个事务**在**前后两次查询**同一个范围的时候、**后一次查询**看到了**前一次查**询**看到的行**。 在可重复读隔离级别下，普通的查询是**快照读**，是不会看到别的事务插入的数据的因此，幻读在**“当前读**”下才会出现。 SERIALIZABLE(可串行化)可以防止幻读：最高的隔离级，**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。

==16、limit 1000000 加载很慢的话，你是怎么解决的呢？==

方案一：如果 id 是连续的，可以这样，返回上次查询的最大记录(偏移量)，再往下 limit select id，name from employee where id>1000000 limit 10.

方案二：在业务允许的情况下限制页数： 建议跟业务讨论，有没有必要查这么后的分页啦。因为绝大数用户都不会往后翻太多页。

方案三：order by + 索引（id 为索引） select id，name from employee order by id limit 100000，10

方案四：利用延迟关联或者子查询优化超多分页场景。（先快速定位需要获取的 id 段，然后 再关联）SELECT a.* FROM employee a, (select id from employee where 条件 LIMIT 1000000,10) b where a.id=b.id

17、==**什么是散列表? select * 和 select 1?** ==

哈希表（Hash table，也叫散列表），是根据关键码值(Key value)而直接进行访问的数据结 构。也是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散函数，存放记录的数组叫做散列表。 有时候为了提高效率，只是为了测试下某个表中是否存在记录，用 1 来代替。

1==8、介绍下 MySQL 的主从复制原理？产生主从延迟的原因==?

 主从复制原理： 主库将变更写入 binlog 日志，然后从库连接到主库之后，从库有一个 IO 线程，将库的 binlog 日志拷贝到自己本地，写入一个 relay 中继日志中。 接着从库 中有一个 SQL 线程会从继日志读取 binlog，然后执行 binlog 日志中的内容，也就是 在自己本地再次执行一遍 SQL。

 主从延迟：

a. 主库的从库太多

b. 从库硬件配置比主库差

c. 慢 SQL 语句过多

d. 主从库之间的网络延迟

e. 主库读写压力大

==19、MySQL 中有哪几种锁？==

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。

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。

页面锁：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，发度一般

==20.数据库三范式==

三范式总结

第一范式:需要满足**列字段的原子性**(不能再分)

第二范式:在满足第一范式的基础上**列字段**需要跟**主键有直接关联关系**(列字段依赖于主键,可通过主键所代表的表对象,定义关联字段)

第三范式:在满足第一,第二范式的基础上,**列字段不能冗余(**可以通过表关联展示的字段,应放到关联中)

==**21.mysql中int(10)和char(10)以及varchar(10)的区别**==

● int(10)的10表示显示的**数据的长度**，不是存储数据的大小；chart(10)和varchar(10)的10表示储**数据的大小**，即表示存储多少个字符。

int(10) 10位的数据长度 9999999999，占32个字节，int型4位

char(10) 10位固定字符串，不足补空格 最多10个字符

varchar(10) 10位可变字符串，不足补空格 最多10个字符

● char(10)表示存储定长的10个字符，不足10个就用空格补齐，占用更多的存储空间

● varchar(10)表示存储10个变长的字符，存储多少个就是多少个，空格也按一个字符存储，这一点和char(10)的空格不同的，char(10)的空格表示占位不算一个字符

==22.谈谈sql优化的经验==

查询语句，**where左侧**的**条件查询**字段**不要使用函数**或者**表达式**；

使用Explain语句分析sql可能优化的地方，进行优化

select查询语句只需要使用一条记录时，**limit 1**;

不要直接**select***，而应该使用具体需要查询的字段；

避免在**where子句中对字段进行NULL**;

为**搜索字段创建索引**；

使用 LIKE%abc%不会走索引，而使用 LIKE abc%会走索引;

字段设计尽可能使用**NOT NULL**;

选择合适的字段类型，选择标准是尽可能小，尽可能定长，尽可能使用整数；

对于枚举类型的字段，尽量使用**enum**,不要使用**varchar**

==23.最左前缀原则与最左匹配原则==

最左优先，根据业务需求，创建多列索引时，where子句中使用最频繁的一列放在最左边

最左匹配原则：mysql会一直向右匹配，直到遇到查询范围（<,>,between,like）就停止匹配；

=和in可以乱序

==**24.百万级别或以上的数据如何删除**==

查询Mysql官方手册得知：**删除数据**的速度和**创建索引**的**数量成正比**

先删除索引；

删除无用的数据；

删除完成后，重新创建索引

2==5.CHAR与VARCHAR的区别==

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；

varchar可变长度

==26.MySql里记录货币用什么字段类型好==

decimal,numeric

==2==□==**8.Mysql有关权限的表有哪几个**==

user:记录允许连接到服务器的用户账户信息，里面的权限式是全局级的

db：记录各个账号在各个数据库上的操作权限

table_priv：记录数据表级的操作权限

columns_priv：记录数据列级的操作权限

host:配合db权限表对给定主机上数据库级操作权限作更细致的的控制，不受Grant和Revoke语影响。

==29.什么是索引==

一种特殊的文件，包含着对**数据表里**所有**记录**的**引用指针排好序的数据结构**

索引（Index）是帮助数据库高效获取数据的数据结构。索引是在基于数据库表创建的，它包含一个中某些列的值以及记录对应的地址，并且把这些值存储在一个数据结构中

==30.索引的优缺点==

优点：加快数据检索的速度；可以在查找的过程中，使用优化隐藏器，提高系统性能

缺点：

时间方面：**创建索引**和**维护索引**要**耗费时间**，对表中数据的**增删改**，同时也需要**维索引**；

空间方面：索引需要**占用物理空间**

==31.索引分类==

主键索引：数据列不能为null,不允许重复；表主键，一张表只能创建一个

唯一索引：数据列不允许重复，允许为null值；一张表允许创建多个

ALTER TABLE table_name ADD UNIQUE (column); 创建唯一索引

ALTER TABLE table_name ADD UNIQUE (column1,column2); 创建唯一组合索引

普通索引：基本的索引类型，没有唯一性限制，允许null值

ALTER TABLE table_name ADD INDEX index_name(column);创建普通索引

ALTER TABLE table_name ADD INDEX index_name(column1,column2,column3);创建组合索引

全文索引：是目前搜索引擎使用的一种关键技术；ALTER TABLE table_name ADD FULLTEXT (c

lumn);

类似于like+%模糊查询，但比它快N倍 select * from test where match(content) against（‘aaaa’
;

==32.唯一索引比普通索引快吗, 为什么==

唯一索引不一定比普通索引快, 还可能慢. 1. 查询时, 在未使用 limit 1 的情况下, 在匹配到一条数据后, 唯一索引即返回, 普通索引会继续匹配

下一条数据, 发现不匹配后返回. 如此看来唯一索引少了一次匹配, 但实际上这个消耗微乎其微. 2. 更新, 这个情况就比较复杂了. 普通索引将记录放到 change buffer 中语句就执行完毕了. 而对

唯一索引而言, 它必须要**校验唯一性**, 因此, 必须将**数据页读入内存**确定没有冲突, 然后才能继操作. 对于**写多读少**的情况, **普通索引**利用 **change buffer** 有效减少了对磁盘的访问次数, 此**普通索引**性能要高于**唯一索引**.

1. ==MySQL由哪些部分组成, 分别用来做什么==

Server

连接器: 管理连接, 权限验证.

分析器: 词法分析, 语法分析.

优化器: 执行计划生成, 索引的选择.

执行器: 操作存储引擎, 返回执行结果.

存储引擎: 存储数据, 提供读写接口.

2. ==MySQL查询缓存有什么弊端, 应该什么情况下使用, 8.0版本对查询缓存有什么变更.==

查询缓存可能会**失效非常频繁**, 对于一个表, 只要有更新, 该表的**全部查询缓存**都会被**清空**. 因此对于频繁更新的表来说, 查询缓存不一定能起到正面效果.

对于读远多于写的表可以考虑使用查询缓存.

8.0版本的查询缓存功能被删了 (￣.￣).

3. ==MyISAM和InnoDB**的区别有哪些==

InnoDB支持事务, MyISAM不支持.

InnoDB支持行级锁, MyISAM支持表级锁.

InnoDB支持多版本并发控制(MVVC), MyISAM不支持.

InnoDB支持外键, MyISAM不支持.

MyISAM支持全文索引, InnoDB部分版本不支持(但可以使用Sphinx插件)

4. **MySQL怎么恢复半个月前的数据**

通过整库备份+binlog进行恢复. 前提是要有定期整库备份且保存了binlog日志.

5. 做过哪些MySQL索引相关优化

尽量使用**主键查询**: **聚簇索引上存储了全部数据**, 相比普通索引查询, 减少了回表的消耗.

MySQL5.6之后引入了**索引下推优化**, 通过适当的使用联合索引, 减少回表判断的消耗.

若频繁查询某一列数据, 可以**考虑利用覆盖索**引避免回表.

联合索引将高频字段放在最左边.

6. ==一千万条数据的表, 如何分页查询==

数据量过大的情况下, limit offset 分页会由于扫描数据太多而越往后查询越慢. 可以配合当前页最后

一条ID进行查询, SELECT * FROM T WHERE id > #{ID} LIMIT #{LIMIT} . 当然, 这种情况下**ID必须有序的**, 这也是有序ID的好处之一.

7. ==订单表数据量越来越大导致查询缓慢, 如何处理==

分库分表. 由于历史订单使用率并不高, 高频的可能只是近期订单, 因此, 将订单表按照时间进行拆分, 据数据量的大小考虑按月分表或按年分表. 订单ID最好包含时间(如根据雪花算法生成), 此时既能根据单ID直接获取到订单记录, 也能按照时间进行查询.

、