模拟面试Day18

> Tencent 面试题第二篇，Written By CTENET

1、MySQL常见的存储引擎InnoDB、MyISAM的适用场景分别是?

区别

1）事务：MyISAM不支持，InnoDB支持

2）锁级别： MyISAM 表级锁，InnoDB 行级锁及外键约束

3）MyISAM存储表的总行数；InnoDB不存储总行数；

4）MyISAM采用非聚集索引，B+树叶子存储指向数据文件的指针。InnoDB主键索引采用聚集索引，B+树叶子存储数据

适用场景

MyISAM适合： 插入不频繁，查询非常频繁，如果执行大量的SELECT，MyISAM是更好的选择， 没有事务。

InnoDB适合： 可靠性要求比较高，或者要求事务； 表更新和查询都相当的频繁， 大量的INSERT或UPDATE

2、什么是聚集索引?

聚集索引就是按照拼音查询，非聚集索引就是按照偏旁等来进行查询。

其实，我们的汉语字典的正文本身就是一个聚集索引。比如，我们要查"安"字，就会很自然地翻开字典的前几页，因为"安"的拼音是"an"，而按照拼音排序 汉字的字典是以英文字母"a"开头并以"z"结尾的，那么"安"字就自然地排在字典的前部。

如果您翻完了所有以"a"开头的部分仍然找不到这个字，那么就 说明您的字典中没有这个字；

同样的，如果查"张"字，那您也会将您的字典翻到最后部分，因为"张"的拼音是"zhang"。

也就是说，字典的正文部分本身 就是一个目录，您不需要再去查其他目录来找到您需要找的内容。

我们把这种**正文内容本身就是一种按照一定规则排列的目录称为"聚集索引"**

3、读可重复读和读提交有什么区别?

对于可重复读来说，就是在 a 进入这个事务以后，那个他的这个数据在他的视图来说就是已经是固定了，如果说在 a 这个事务提交之前 B 的这个事务修改了那个数据，在 A 是看不到的

对于读提交的情况来说，就是说还是 a b 两个事务，b 事务修改一个数据然后并且提交以后，但 a a 还没有提交，然后 a 这个时候去读那个数据，就会读到 b 已经修改的数据。

> 读提交，指一个事务提交之后，它做的变更才能被其他事务看到。
>
> 可重复读，指一个事务执行过程中看到的数据，一直跟这个事务启动时看到的数据是一致的，MySQL InnoDB 引擎的默认隔离级别。
>
> 对于「读提交」和「可重复读」隔离级别的事务来说，它们是通过 MVCC 来实现的，它们的区别在于创建 Read View 的时机不同，大家可以把 Read View 理解成一个数据快照，就像相机拍照那样，定格某一时刻的风景。
>
> 「读提交」隔离级别是在「每个语句执行前」都会重新生成一个 Read View，而「可重复读」隔离级别是「启动事务时」生成一个 Read View，然后整个事务期间都在用这个 Read View。

4、事务四大特性（ACID）

第一种回答

原子性：一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被恢复（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

一致性：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。

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，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。事务隔离分为不同级别，包括读未提交（Read uncommitted）、读提交（read committed）、可重复读（repeatable read）和串行化（Serializable）。

持久性：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

第二种回答

原子性（Atomicity）

- 原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚，因此事务的操作如果成功就必须要完全应用到数据库，如果操作失败则不能对数据库有任何影响。

一致性（Consistency）

- 事务开始前和结束后，数据库的完整性约束没有被破坏。比如A向B转账，不可能A扣了钱，B却没收到。

隔离性（Isolation）

- 隔离性是当多个用户并发访问数据库时，比如操作同一张表时，数据库为每一个用户开启的事务，不能被其他事务的操作所干扰，多个并发事务之间要相互隔离。

同一时间，只允许一个事务请求同一数据，不同的事务之间彼此没有任何干扰。比如A正在从一张银行卡中取钱，在A取钱的过程结束前，B不能向这张卡转账。
关于事务的隔离性数据库提供了多种隔离级别，稍后会介绍到。　　持久性

(Durability)

- 持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的，即便是在数据库系统遇到故障的情况下也不会丢失提交事务的操作。

5、说一下你理解的分库分表

根据业务字段的hash值来确定分片的，比如user_id不同的用户信息就会存储到不同分片当中，他是多个分片同时提供服务。

> 当数据量过大造成事务执行缓慢时，就要考虑分表，因为减少每次查询数据总量是解决数据查询缓慢的主要原因。你可能会问：“查询可以通过主从分离或缓存来解决，为什么还要分表？”但这里的查询是指事务中的查询和更新操作。
>
> 为了应对高并发，一个数据库实例撑不住，即单库的性能无法满足高并发的要求，就把并发请求分散到多个实例中去，这种就是分库。
>
> 总的来说，分库分表使用的场景不一样： 分表是因为数据量比较大，导致事务执行缓慢；分库是因为单库的性能无法满足要求。

6、讲一讲MySQL中char和varchar的区别?

char是固定长度的字符串类型，varchar是可变长度的字符串类型。

拿char(128)和varchar(128)举例来说。char(128)是无论字符串大小，都会在磁盘上分配128个字符的内存空间。而varchar(128)会根据字符本身的长短来分配内存空间。

补充

在MySQL中，CHAR和VARCHAR都是用于存储字符类型数据的数据类型，它们的区别在于存储方式和使用场景。

CHAR类型用于存储固定长度的字符串，其长度在定义表时就已经固定，且最大长度为255个字符。当存储的字符串长度小于定义的长度时，MySQL会在其后面补充空格使其长度达到定义的长度。由于存储的长度是固定的，因此CHAR类型的读取速度比VARCHAR类型更快。

VARCHAR类型则用于存储可变长度的字符串，其长度可以在存储数据时动态地改变，但最大长度也为255个字符。当存储的字符串长度小于定义的长度时，MySQL不会在其后面补充空格。由于存储的长度是可变的，因此VARCHAR类型的存储空间相对更小，但读取速度比CHAR类型稍微慢一些。

那与varchar相比，char字段是不是一无是处呢?

大部分情况，是的，最好使用varchar。不过考虑一个极端的场景：某个字段的最大长度是100字节，但是会频繁修改。如果使用char(100)，则插入记录后就分配了100个字节，后续修改不会造成页分裂、页空隙等问题，而varchar(100)由于没有提前分配存储空间，后续修改时可能出现页分裂，进而导致性能下降。

7、讲一讲你理解的外键约束

举例来说，某一个字段是表b的主键，但是它也是表a中的字段，表a中该字段的使用范围取决于表b。外键约束主要是用来维护两个表的一致性。

补充

外键约束的作用是维护表与表之间的关系，确保数据的完整性和一致性。让我们举一个简单的例子：

假设你有两个表，一个是学生表，另一个是课程表，这两个表之间有一个关系，即一个学生可以选修多门课程，而一门课程也可以被多个学生选修。在这种情况下，我们可以在学生表中定义一个指向课程表的外键，如下所示：

```
CREATE TABLE students (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  course_id INT,
  FOREIGN KEY (course_id) REFERENCES courses(id)
);
```

这里，students表中的course_id字段是一个外键，它指向courses表中的id字段。这个外键约束确保了每个学生所选的课程在courses表中都存在，从而维护了数据的完整性和一致性。

如果没有定义外键约束，那么就有可能出现学生选了不存在的课程或者删除了一个课程而忘记从学生表中删除选修该课程的学生的情况，这会破坏数据的完整性和一致性。因此，使用外键约束可以帮助我们避免这些问题。

8、TCP和UDP的区别?

1.TCP是面向连接的协议，建立和释放连接需要进行三次握手和四次挥手。UDP是面向无连接的协议，无需进行三次握手和四次挥手。说明udp比TCP实时性更强。

2.TCP 是流式传输，没有边界，但保证顺序和可靠。UDP 是一个包一个包的发送，是有边界的，但可能会丢包和乱序。

3.TCP连接的可靠性强，UDP的可靠性不强。

4.TCP只能一对一，UDP支持一对多和多对多。

5.TCP的头部开销比UDP大。TCP 首部长度较长，会有一定的开销，首部在没有使用「选项」字段时是 20 个字节，如果使用了「选项」字段则会变长的。UDP 首部只有 8 个字节，并且是固定不变的，开销较小。

TCP 提供了一系列的可靠传输机制来保证它这个传输是可靠的，相较而言的话，那它的传输速度就是慢的.

UDP 没有做这个可靠的控制，它只是尽力而为，所以说它的传输速度是快的，而且占用的资源也会更小一些。具体使用的话要看不同的那个业务场景来进行相关的使用。

9、TCP是如何保证可靠传输的?

tcp的序列号可以避免乱序的问题，保证收到的tcp报文都是有序的。

在 TCP 中，当发送端的数据到达接收主机时，接收端主机会返回一个确认应答消息，表示已收到消息。

TCP 针对数据包丢失的情况，会用重传机制解决。

用快重传解决个别报文段的丢失问题。

使用滑动窗口实现流量控制。使用接收方确认报文中的窗口字段来控制发送方发送窗口大小，进而控制发送方的发送速率，使得接收方来得及接收。

使用基于窗口的拥塞控制，来尽量避免避免网络拥塞。

第一种回答

- **确认和重传**：接收方收到报文就会确认，发送方发送一段时间后没有收到确认就会重传。
- **数据校验**：TCP报文头有校验和，用于校验报文是否损坏。
- **数据合理分片和排序**：tcp会按最大传输单元(MTU)合理分片，接收方会缓存未按序到达的数据，重新排序后交给应用层。而UDP：IP数据报大于1500字节，大于MTU。这个时候发送方的IP层就需要分片，把数据报分成若干片，是的每一片都小于MTU。而接收方IP层则需要进行数据报的重组。由于UDP的特性，某一片数据丢失时，接收方便无法重组数据报，导致丢弃整个UDP数据报。
- **流量控制**：当接收方来不及处理发送方的数据，能通过滑动窗口，提示发送方降低发送的速率，防止包丢失。
- **拥塞控制**：当网络拥塞时，通过拥塞窗口，减少数据的发送，防止包丢失。

第二种回答

- 建立连接（标志位）：通信前确认通信实体存在。
- 序号机制（序号、确认号）：确保了数据是按序、完整到达。
- 数据校验（校验和）：CRC校验全部数据。
- 超时重传（定时器）：保证因链路故障未能到达数据能够被多次重发。
- 窗口机制（窗口）：提供流量控制，避免过量发送。
- 拥塞控制：同上。

第三种回答

首部校验

这个校验机制能够确保数据传输不会出错吗？ 答案是不能。

原因

TCP协议中规定，TCP的首部字段中有一个字段是校验和，发送方将伪首部、TCP首部、TCP数据使用累加和校验的方式计算出一个数字，然后存放在首部的校验和字段里，接收者收到TCP包后重复这个过程，然后将计算出的校验和和接收到的首部中的校验和比较，如果不一致则说明数据在传输过程中出错。

这就是TCP的数据校验机制。 但是这个机制能够保证检查出一切错误吗？**显然不能。**

因为这种校验方式是累加和，也就是将一系列的数字（TCP协议规定的是数据中的每16个比特位数据作为一个数字）求和后取末位。 但是小学生都知道A+B=B+A，假如在传输的过程中有前后两个16比特位的数据前后颠倒了（至于为什么这么巧合？我不知道，也许路由器有bug？也许是宇宙中的高能粒子击中了电缆？反正这个事情的概率不为零，就有可能会发生），那么校验和的计算结果和颠倒之前是一样的，那么接收端肯定无法检查出这是错误的数据。

解决方案

传输之前先使用MD5加密数据获得摘要，跟数据一起发送到服务端，服务端接收之后对数据也进行MD5加密，如果加密结果和摘要一致，则认为没有问题

补充

TCP保证可靠传输的机制包括以下几个方面：

1. 序号和确认号：TCP通过序号和确认号来保证数据的可靠传输。发送方将每个数据报文都标记一个唯一的序号，接收方收到数据后需要回复一个确认号，表示已经成功接收到了这个数据。
2. 超时重传：发送方在发送数据后会设置一个计时器，如果在规定的时间内没有收到确认，则会重新发送数据。这可以保证即使某个数

据包在传输过程中丢失，也能够被及时地重传。
3. 滑动窗口：TCP通过滑动窗口机制来控制发送方和接收方的数据发送和接收速率。发送方通过滑动窗口机制来控制发送数据的数量和速度，接收方则通过滑动窗口机制来控制接收数据的数量和速度。
4. 确认机制：TCP通过确认机制来确保数据的可靠传输。发送方将每个数据报文都标记一个唯一的序号，接收方收到数据后需要回复一个确认号，表示已经成功接收到了这个数据。
5. 拥塞控制：TCP通过拥塞控制机制来避免网络拥塞。发送方会根据网络状况和接收方的反馈来调整数据发送的速率，以避免网络拥塞。

10、流量控制是使用什么数据结构来实现的?

流量控制是使用滑动窗口来实现的。接收方确认报文中的窗口字段可以用来控制发送方窗口的大小。

如果窗户的值为0，则发送方停止发送数据，但是发送方会定期的向接收方发送窗口探测报文以得到窗口的大小。

补充

TCP传输协议中，流量控制是使用滑动窗口（Sliding Window）来实现的。滑动窗口是一种基于数据流的、动态调整的、可变大小的窗口，它通过协商双方的接收窗口和发送窗口大小，控制数据的传输速率。

在TCP协议中，每个数据包都有一个序号，接收方通过序号来确认是否收到了正确的数据包。发送方将数据分成若干个数据段，每个数据段的大小不超过发送窗口的大小，然后将这些数据段发送给接收方。接收方会确认已经收到的数据，同时告诉发送方自己的接收窗口大小。发送方根据接收方的窗口大小，动态调整自己的发送窗口大小，从而控制数据的传输速率。

滑动窗口的大小是可以动态调整的，它可以根据网络状况和双方的能力来自适应地调整，从而实现流量控制的功能。如果接收方的接收窗口变小，发送方会相应地减小自己的发送窗口，以避免过多的数据堆积在网络中导致拥塞。如果接收方的接收窗口变大，发送方会相应地增加自己的发送窗口，以提高数据传输速率。