**源自 阿伟玩代码和Frankiee**

**1. 主键 外键**

**主 键**：

一个数据列只能有一个主键，且主键的取值不能缺失，即不能为空值（Null）。

**外 键：**

在一个表中存在的另一个表的主键称此表的外键。

**2.数据库事务的四个特性及含义**

数据库事务transanction正确执行的四个基本要素。

ACID,原子性(Atomicity)、一致性(Correspondence)、隔离性(Isolation)、持久性(Durability)。

**原子性:** 一个事务必须视为一个不可分割的最小工作单元，整个事务中的所有操作要么全部提交成功，要么全部失败回滚，对于一个事务来说，不可能只执行其中的一部分操作，这就是事务的原子性。

**一致性:**在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。

**隔离性:** 一个事务所做的修改在最终提交以前，对其他事务是不可见的

**持久性:** 一旦事务提交，则其所做的修改就会永久保存到数据库中。此时即使系统崩溃，修改的数据也不会丢失。

**3. 事务的隔离级别，mysql默认的隔离级别是什么？**

**读未提交(Read uncommitted)**，一个事务可以读取另一个未提交事务的数据，最低级别，任何情况都无法保证。

**读已提交(Read committed)**，一个事务要等另一个事务提交后才能读取数据，可避免脏读的发生。

**可重复读(Repeatable read)**，就是在开始读取数据（事务开启）时，不再允许修改操作，可避免脏读、不可重复读的发生。

**串行(Serializable)，**是最高的事务隔离级别，在该级别下，事务串行化顺序执行，可以避免脏读、不可重复读与幻读。但是这种事务隔离级别效率低下，比较耗数据库性能，一般不使用。Mysql的默认隔离级别是Repeatable read。

**4. 说一说脏读、不可重复读、幻读脏读：**

事务A读取了事务B更新的数据，然后B回滚操作，那么A读取到的数据是脏数据；

不可重复读：事务 A 多次读取同一数据，事务 B 在事务A多次读取的过程中，对数据作了更新并提交，导致事务A多次读取同一数据时，结果因此本事务先后两次读到的数据结果会不一致；

幻读：幻读解决了不重复读，保证了同一个事务里，查询的结果都是事务开始时的状态（一致性）；

**5. 数据表类型有哪些**

答：MyIASM、InnoDB、HEAP、ISAM、MERGE、DBD以及Gemeni(一般只知道前两者即可)

**6. innodb引擎的4大特性**

答：插入缓冲(insert buffer)；二次写(double write)；自适应哈希索引(ahi)；预读(read ahead)

**7. nnoDB引擎的行锁是通过加在什么上实现的？**

答：基于索引；

**8. Mysql中的myisam与innodb的区别，至少五点**

InooDB支持事务，而MyISAM不支持事务；

InnoDB支持行级锁，而MyISAM支持表级锁；

InnoDB支持MVCC，而MyISAM不支持；

InnoDB支持外键，而MyISAM不支持；

InnoDB不支持全文索引，而MyISAM支持；

InnoDB不能通过直接拷贝表文件的方法拷贝表到另外一台机器， myisam 支持；InnoDB表支持多种行格式， myisam 不支持；

InnoDB是索引组织表， myisam 是堆表；

**9. myisam与innodb select count(\*)哪个更快，为什么**

答：myisam更快，因为myisam内部维护了一个计数器，可以直接调取

**10. 数据库三大范式**

**第一范式**：数据库表中的字段都是单一属性的，不可再分(保持数据的原子性)；

**第二范式**：第二范式必须符合第一范式，非主属性必须完全依赖于主键。

**第三范式**：在满足第二范式的基础上，确保数据表中的每一列数据都和主键直接相关，而不能间接相关

**11. 说一说你能想到的sql语句优化，至少五种**

避免select \*，将需要查找的字段列出来；使用连接（join）来代替子查询；

拆分大的delete或insert语句；使用limit对查询结果的记录进行限定；

用 exists 代替 in 是一个好的选择；

用Where子句替换HAVING 子句 因为HAVING 只会在检索出所有记录之后才对结果集进行过滤；

不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引

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；

尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描；

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；

**12. 说几个mysql中你常用的函数答：**

sum、count 、avg、min、max

**13. 说几个你除了增删改查之外在mysql中常用的命令**

答：

Explain、describe、show、truncate

**14. 说几个除了增删改查之外常用的关键字**

答：distinct、limit、offset、order by、union、union all、between、group by；

**15. union、union all的区别**

**对重复结果的处理：**

UNION在进行表链接后会筛选掉重复的记录，

Union All不会去除重复记录；

**对排序的处理：**

Union将会按照字段的顺序进行排序；

UNION ALL只是简单的将两个结果合并后就返回；

从**效率**上说，UNION ALL 要比UNION快很多

**16. varchar(100)和varchar(200)的区别**

答：varchar(100)最多存放100个字符，varchar(200)最多存放200个字符，varchar(100)和(200)存储hello所占空间一样，但后者在排序时会消耗更多内存，因为order by col采用fixed\_length计算col长度(memory引擎也一样)

**17.varchar(20)和int(20)中的20含义一样吗**

答：不一样，前者表示最多存放20个字符，后者表示最多显示20个字符，但是存储空间还是占4字节存储，存储范围不变；

**18. 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。

**19. 什么是触发器？触发器的作用？**

答：触发器是一种特殊的存储过程，主要是通过事件来触发而被执行的。它可以强化约束，来维护数据的完整性和一致性，可以跟踪数据库内的操作从而不允许未经许可的更新和变化。可以联级运算。如，某表上的触发器上包含对另一个表的数据操作，而该操作又会导致该表触发器被触发。

**20. 存储过程与触发器的区别**

答：触发器与存储过程非常相似，触发器也是SQL语句集，两者唯一的区别是触发器不能用EXECUTE语句调用，而是在用户执行Transact-SQL语句时自动触发（激活）执行。触发器是在一个修改了指定表中的数据时执行的存储过程。通常通过创建触发器来强制实现不同表中的逻辑相关数据的引用完整性和一致性。由于用户不能绕过触发器，所以可以用它来强制实施复杂的业务规则，以确保数据的完整性。触发器不同于存储过程，触发器主要是通过事件执行触发而被执行的，而存储过程可以通过存储过程名称名字而直接调用。当对某一表进行诸如UPDATE、INSERT、DELETE这些操作时，SQLSERVER就会自动执行触发器所定义的SQL语句，从而确保对数据的处理必须符合这些SQL语句所定义的规则

**21.索引的作用？和它的优点缺点是什么？**

答：索引就一种特殊的查询表，数据库的搜索引擎可以利用它加速对数据的检索。它很类似与现实生活中书的目录，不需要查询整本书内容就可以找到想要的数据。索引可以是唯一的，创建索引允许指定单个列或者是多个列。缺点是它减慢了数据录入的速度，同时也增加了数据库的尺寸大小。

**22.MySQL主要的索引类型**

**普通索引：**是最基本的索引，它没有任何限制；

**唯一索引：**索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一；

**主键索引：**是一种特殊的唯一索引，一个表只能有一个主键，不允许有空值；

**组合索引：**指多个字段上创建的索引，只有在查询条件中使用了创建索引时的第一个字段，索引才会被使用。使用组合索引时遵循最左前缀集合；

**全文索引**：主要用来查找文本中的关键字，而不是直接与索引中的值相比较，mysql中MyISAM支持全文索引而InnoDB不支持；

**23. 使用like 'a%' 、like'%a'、like'%a%'查询时是否会使用索引**

答：'a%'会，其他两个不会；

**24. 使用索引注意事项**

索引不会包含有NULL的列，复合索引中只要有一列含有NULL值，那么这一列对于此符合索引就是无效的；

使用短索引，对串列进行索引，如果可以就应该指定一个前缀长度；短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作；

mysql查询只使用一个索引，因此数据库默认排序可以符合要求的情况下不要使用排序操作，尽量不要包含多个列的排序，如果需要最好给这些列建复合索引；注意like，上文已经提到；不要在列上进行运算；不使用NOT IN 、<>、！=操作，但<,<=，=，>,>=,BETWEEN,IN是可以用到索引的；

索引要建立在经常进行select操作的字段上；索引要建立在值比较唯一的字段上；对于那些定义为text、image和bit数据类型的列不应该增加索引；

在where和join中出现的列需要建立索引；

如果where字句的查询条件里使用了函数(如：where DAY(column)=…),mysql将无法使用索引；

在join操作中(需要从多个数据表提取数据时)，mysql只有在主键和外键的数据类型相同时才能使用索引，否则及时建立了索引也不会使用；

**25. 说一说什么是外键，优缺点**

答：外键指的是外键约束，目的是保持数据一致性，完整性，控制存储在外键表中的数据。使两张表形成关联，外键只能引用外表中列的值；

**优点**：由数据库自身保证数据一致性，完整性，更可靠，因为程序很难100％保证数据的完整性，而用外键即使在数据库服务器当机或者出现其他问题的时候，也能够最大限度的保证数据的一致性和完整性。有主外键的数据库设计可以增加ER图的可读性，这点在数据库设计时非常重要。外键在一定程度上说明的业务逻辑，会使设计周到具体全面。

**缺点：**可以用触发器或应用程序保证数据的完整性；过分强调或者说使用外键会平添开发难度，导致表过多，更改业务困难，扩展困难等问题；不用外键时数据管理简单，操作方便，性能高（导入导出等操作，在insert, update, delete 数据的时候更快）；

**26. 在什么时候你会选择使用外键，为什么**

答：在我的业务逻辑非常简单，业务一旦确定不会轻易更改，表结构简单，业务量小的时候我会选择使用外键。因为当不符合以上条件的时候，外键会影响业务的扩展和修改，当数据量庞大时，会严重影响增删改查的效率。

**27. 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。

**28.mysql有没有rowid？**

答：没有，InnoDB如果没有定义主键，内部会生成一个主键编号rowid ，但是无法查询到。在平时InnoDB建表的时候我们最好自己确定主键，防止每次插入数据前数据库会去生成rowid。

**29.mysql怎么在查询时给查出来的数据设置一个自增的序号？**

set @i=0;SELECT (@i:=@i+1) 别名 FROM table, (SELECT @i:=0) AS 别名 ;

**30.视图的作用，视图可以更改么？**

视图是虚拟的表，与包含数据的表不一样，视图只包含使用时动态检索数据的查询；不包含任何列或数据。使用视图可以简化复杂的sql操作，隐藏具体的细节，保护数据；视图创建后，可以使用与表相同的方式利用它们。

视图不能被索引，也不能有关联的触发器或默认值，如果视图本身内有order by 则对视图再次order by将被覆盖。

创建视图：create view XXX as XXXXXXXXXXXXXX;

对于某些视图比如未使用联结子查询分组聚集函数Distinct Union等，是可以对其更新的，对视图的更新将对基表进行更新；但是视图主要用于简化检索，保护数据，并不用于更新，而且大部分视图都不可以更新。

**31.drop,delete与truncate的区别**

drop直接删掉表 truncate删除表中数据，再插入时自增长id又从1开始 delete删除表中数据，可以加where字句。

（1） DELETE语句执行删除的过程是每次从表中删除一行，并且同时将该行的删除操作作为事务记录在日志中保存以便进行进行回滚操作。TRUNCATE TABLE 则一次性地从表中删除所有的数据并不把单独的删除操作记录记入日志保存，删除行是不能恢复的。并且在删除的过程中不会激活与表有关的删除触发器。执行速度快。

（2） 表和索引所占空间。当表被TRUNCATE 后，这个表和索引所占用的空间会恢复到初始大小，而DELETE操作不会减少表或索引所占用的空间。drop语句将表所占用的空间全释放掉。

（3） 一般而言，drop > truncate > delete

（4） 应用范围。TRUNCATE 只能对TABLE；DELETE可以是table和view

（5） TRUNCATE 和DELETE只删除数据，而DROP则删除整个表（结构和数据）。

（6） truncate与不带where的delete ：只删除数据，而不删除表的结构（定义）drop语句将删除表的结构被依赖的约束（constrain),触发器（trigger)索引（index);依赖于该表的存储过程/函数将被保留，但其状态会变为：invalid。

（7） delete语句为DML（data maintain Language),这个操作会被放到 rollback segment中,事务提交后才生效。如果有相应的 tigger,执行的时候将被触发。

（8） truncate、drop是DLL（data define language),操作立即生效，原数据不放到 rollback segment中，不能回滚

（9） 在没有备份情况下，谨慎使用 drop 与 truncate。要删除部分数据行采用delete且注意结合where来约束影响范围。回滚段要足够大。要删除表用drop;若想保留表而将表中数据删除，如果于事务无关，用truncate即可实现。如果和事务有关，或老师想触发trigger,还是用delete。

（10） Truncate table 表名 速度快,而且效率高,因为:

truncate table 在功能上与不带 WHERE 子句的 DELETE 语句相同：二者均删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。DELETE 语句每次删除一行，并在事务日志中为所删除的每行记录一项。TRUNCATE TABLE 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

（11） TRUNCATE TABLE 删除表中的所有行，但表结构及其列、约束、索引等保持不变。新行标识所用的计数值重置为该列的种子。如果想保留标识计数值，请改用 DELETE。如果要删除表定义及其数据，请使用 DROP TABLE 语句。

（12） 对于由 FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE，而应使用不带 WHERE 子句的 DELETE 语句。由于 TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器。