数据库面试题汇总

1. 如何理解几种范式：1NF、2NF、3NF、BCNF

码：设 K 为某表中的一个属性或属性组，若除 K 之外的所有属性都完全函数依赖于 K（这个“完全”不要漏了），那么我们称 K 为候选码，简称为码。在实际中我们通常可以理解为：假如当 K 确定的情况下，该表除 K 之外的所有属性的值也就随之确定，那么 K 就是码。一张表中可以有超过一个码。

主属性：包含在码中的属性。

非主属性：不包含在任一码中的属性。

范式是用来指导数据表设计的，越高级的范式，数据表的拆分也就越合理，会使得数据表的冗余数据减少，使得查询的效率提高，同时避免一些插入和删除的异常。

1NF：数据表中的字段都是单一属性的，不可再分。例如价格(进价，售价)，就需要拆分为两个字段进价和售价，这样每个字段才是不可再分的。

2NF：2NF 在 1NF 的基础之上，**消除了非主属性对于码的部分函数依赖**。如果表中的某个字段只依赖码中的部分字段，那么数据表就需要进一步进行拆分，因为这样可以减少数据的冗余，提高查询效率。

3NF：3NF 在 2NF 的基础之上，**消除了非主属性对于码的传递函数依赖**。意思是非主属性之间不能有相互依赖的关系，如果一个表中的非主属性之间有相互依赖的关系，那么他们需要分拆成一张新表，这样才能降低数据冗余。

BCNF：在 3NF 的基础上消除**主属性**对于码的部分与传递函数依赖。经过 BCNF 的进一步分解，数据表的设计达到最合理的状态，不会有数据冗余和插入删除异常等情况。

2. 视图的概念与游标的概念

视图 view 是一张虚拟的表，并不是真实的表，更多的是一种逻辑上的表，主要的用途是**简化 SQL 逻辑**，使得 SQL 代码更加清晰，例如一段复杂的嵌套查询，可以利用视图进行拆分，使得逻辑更加清晰。同时还可以**便于复用，提高效率**，比如对某张表的查询结果被使用了很多次，就可以使用 view 来进行复用。

游标可以充当指针的作用，用于对查询数据库所返回的记录进行遍历，以便进行相应的操作。

3. drop delete 和 truncate 的区别

drop 是删除一张表。

truncate 是删除表中的全部数据，但是不删除表本身。

delete 是删除表中的数据，可以加 where 条件进行筛选。

4. 数据库索引的数据结构和原理

数据库索引的底层数据结构是 B+树，B+树是在 B-树的基础上演化而来，B-树是多路平衡查找树(是在平衡二叉树的基础上演化而来)，B+树的不同之处在于其所有的数据都储存在叶子结点上，非叶子结点只用来做查找条件，叶子结点由双向链表连接，这样在进行索引查找时先定位到某一叶子结点上，再顺序的通过链表进行查找。

根据叶子节点存储的数据不同又分为聚簇索引(innodb)和非聚簇索引(myiasm)。

索引通过以下语句查询：show index from <table_name>;

5. 聚簇索引和非聚簇索引的区别

聚簇索引叶子节点存放的是具体数据，非聚簇索引叶子节点存储的是指针。

非聚簇索引叶子节点存储的是建立索引用的字段值和指向具体数据的一个指针，因此先通过该索引找到指针，然后定位到具体数据。

聚簇索引叶子节点存储的是具体的数据值，如果是在主键上建立索引的话，则存储的是主键值和一整条数据，如果是在非主键的字段上建立索引的话，则存储的是该字段值和主键值，也就是说聚簇索引(非主键上)在建立的同时会建立主键索引和辅助索引，先在辅助索引中查找到对应的主键值后再根据主键索引查找对应的数据。

聚簇索引的优势：

1. 数据访问更快，虽然需要两次索引，但是主键索引非常快，数据在叶子结点上是可以直接拿到的，而非聚簇索引需要经过多的 IO 操作，因此聚簇索引更快。

聚簇索引的劣势：

1. 因为要维护两个索引，因此插入和删除的代价较大，一般推荐主键采用自增 id，因为这样主键索引在插入和删除时不需要做大的调整。

6. 联合索引的底层数据结构

联合索引的底层存储结构与单值索引类似，只是索引条件变成了多列，例如有索引(a,b,c)，那么索引是先按 a 的值进行排序，a 相同的情况下按照 b 的值排序，b 相同的情况下再按照 c 的值排序。

7. 索引的最左匹配原则

对于联合索引来说，查询的条件中必须包含联合索引中最左边的字段才能命中索引。例如有联合索引(a,b,c)，其实会建立三个索引(a),(a,b),(a,b,c)，查询条件 a=1 可以命中索引(a)，查询条件 a=1,c=1 也只能命中(a)，这就是索引的最左匹配原则。

在复合索引的底层数据结构中我们能找出最左匹配原则的理论依据，索引是在 a 有序的基础上对 b 进行排序，如果单独对 b 进行查询，无法使用索引，因为 b 并不是全局有序的。

8. Hash 索引和 B+树索引的区别

B+树索引如前所述。

Hash 索引底层使用 hash 表，key 是索引字段的值，value 是数据行，冲突机制采用链表法。

Hash 索引优势：

1. 适合等值查询，理论速度是 O(1)的。

Hash 索引劣势：

1. 无法进行范围查询
2. 无法使用联合索引(因为是将所有的索引字段进行 hash，无法只进行部分索引字段的查询
3. 性能不稳定，在索引字段重复率很高的情况下性能较差。

与之相反的即使 B+树索引的优势。

9. 什么是覆盖索引

覆盖索引是用于提高索引查询速度的一种方式，指的是将需要查询的字段作为联合索引的一部分加到索引中，这样便可以直接查询出数据来，而无需经过回表(聚簇索引中第二次进行主键索引查找)或者多余的 IO 操作。

9. 什么是事务、事务的隔离级别

事务的基本属性包括：

原子性：要么全部执行，要么不执行。

一致性：事务执行前后数据一致性不能被破坏。

隔离性：多个事务并发执行要相互隔离。

持久性：事务对数据库的改变是持久的。

并发事务会出现三种问题：脏读，不可重复读，幻读。

脏读：事务 A 读到事务 B 未提交时的数据。

不可重复读：事务 A 只能读到事务 B 提交之后的数据，所以在事 B 提交前后读出的数据不同。

幻读：每次读出的数据以第一次读出的为准，也就解决了不可重复读的问题，但是如果事务 B 插入一条数据，事务 A 按第一次的查询结果看不到这条数据，再进行插入时就会报错已有该条数据。幻读重点在于事务 A 的读写，不可重复读在于事务 A 的读读。

并发事务的四种隔离级别：

读未提交：会出现脏读情况。

读已提交：解决了脏读的情况，会出现不可重复读的情况。

可重复读：解决了不可重复读的情况，会出现幻读的情况。（Mysql 默认隔离级别）

串行化(加锁)：解决了幻读的问题。

10. 数据库的 ACID 怎么保证

ACID 指的是数据库的原子性，一致性、隔离性、持久性。

原子性通过回滚机制(undo log 记录)保证，一个事务中如果某条操作失败了，之前所有操作都会回滚。

隔离性通过锁机制和 MVCC(多版本并发控制，一种并发控制的方法，一些情况下可以代替行锁)。

持久性通过 redo log(适用于崩溃恢复)保证，如果事务提交成功，但是所做的更改在刷回磁盘前挂掉，还可以通过 redo log 进行持久性的保证。

一致性由上面三者来保证。

11. 讲讲 MVCC

MVCC 即多版本并发控制，和锁机制一样用于并发控制，实现数据库的隔离性。不同之处在于：

1. MVCC 用于读已提交和可重复读的实现，锁机制用于串行化。
2. MVCC 的效率更高一些。

MVCC 实现原理：使用 undo log 的版本链和 read view 机制实现。

undo log 中存在版本链，可以将数据行的修改历史记录下来，用于确定当前该事务的 read view。read view 会为每个事务生成一张视图，后续的读操作在该视图上完成，也就实现了隔离，读已提交的策略是在事务每次查询时都生成一张视图，可重复读的策略是事务开始时生成一张视图。

10. 超键、候选键、主键、外键分别是什么？

超键：在关系中能唯一标识数据的属性集称为超键，超键可以有多个

候选键：不含多余元素的超键称为候选键，候选键可以有多个

主键：用户选择的一个候选键作为主键

外键：是另一个关系中的主键的属性集，称为外键

11. SQL 约束有哪几种？

1. 主键：主键用于唯一标识一行数据，要求是唯一且非空。
2. 唯一键：要求该列数据必须唯一。
3. 外键：一张表的外键是另一张表中的主键，必须在另一张表的主键范围内选择。
4. not null：有 not null 约束的字段不能为空。
5. default：默认值约束，如果 insert 数据该字段为空则会赋值为默认值。
6. check：用于约束 SQL 中某一字段的取值范围。

12. MyIASM 和 Innodb 两种引擎的区别

1. Innodb 支持事务，MylASM 不支持事务
2. Innodb 支持外键，MylASM 不支持外键
3. Innodb 是聚簇索引，MylASM 是非聚簇索引
4. Innodb 最小的锁粒度是行锁，MylASM 最小的锁粒度是表锁

13. Mysql 的锁分类

锁机制是并发访问中进行一致性保证的重要方式。

1. 锁的粒度大小可以划分为行锁，表锁和页锁，Innodb 中支持行锁，表锁，MylASM 中支持表锁。

   行锁：对某一行加解锁，优点是锁的粒度小，并发高，缺点是加解锁的时间长，同时会产生死锁。

   表锁：对整个表进行加解锁，优点是加解锁快，不会产生死锁，缺点是锁的粒度大，并发低。

   页锁：行锁和表锁的折中，平衡了加锁时间和并发。
2. 按照锁的性质分可以分为共享锁，排他锁和意向锁。

共享锁和排他锁是人为设置的，而意向锁是 mysql 内部进行操作的，是为了提高效率。

共享锁：即读锁，加锁之后不允许写操作，有行级共享锁和表级共享锁，语句是 lock in share mode。

排他锁：即写锁，加锁之后不允许读写操作，有行级排他锁和表级排他锁，语句是 for update。

意向锁：分为意向共享锁和意向排他锁，意向锁可以理解为是否存在行级锁的标志，解决的问题是加表级锁时，如何高效的判断是否和行级锁冲突的问题，因此意向锁可以理解为行级锁的标志，意向锁和行级锁之间不会有任何冲突。表级别的锁之间的兼容性情况如下图所示：

type	IS	IX	S	X
IS	兼容	兼容	兼容	不兼容
IX	兼容	兼容	不兼容	不兼容
S	兼容	不兼容	兼容	不兼容
X	不兼容	不兼容	不兼容	不兼容

3. 按照思想来分可以分为乐观锁和悲观锁

乐观锁和悲观锁更多的是一种思想，在其他的服务中也有相应的体现，比如 redis。

乐观锁：假定其他操作不会对数据进行修改，等到真正访问数据时再进行判断，如果有修改则放弃操作。是一种 CAS 的思想(check and set)，适合多读的场景。在 mysql 中可以通过使用版本号的方式实现乐观锁。

悲观锁：假定其他操作会对数据进行修改，因此每次操作都会加锁，mysql 中的共享锁和排他锁都是悲观锁的体现。

15. mysql 中的几种 join

1. inner join：等值连接，相当于取两个集合的交集，如果不写 on 条件则是笛卡尔积，在 mysql 中 inner join 的用法与 corss join 和 join 完全相同。
2. left join：左表产生完整的记录，如果右表中没有能进行 join 的数据则补 null。
3. right join：右表产生完整的记录，如果左表中没有能进行 join 的数据则补 null。

4. full join(outer join)：相当于两个集合的并集，如果某个集合没有匹配的数据则补 null。在 mysql 中没有内置的实现，可以通过左连接 union 右连接来实现。

16. 查询语句不同元素(select--from--where--group by--having--order by--join--on--limit)执行先后顺序

执行顺序为：from->join->on->where->group by->having->order by->limit->select

17. MySQL 慢查询怎么解决

打开慢查询日志，定位到具体的 SQL 并进行优化：

1. SQL 语句优化

   SELECT 子句中避免使用*号 、过滤掉最大数量记录的条件最好写在 WHERE 子句的最右、删除全表数据用 TRUNCATE 替代 DELETE、用 IN 替代 OR。

2. 索引优化

   最左前缀匹配原则、尽量选择区分度高的列作为索引、索引列不能参与计算(否则无法使用索引)、避免 LIKE '%名称'(无法使用索引)，避免不等于符号(无法使用索引)。

3. 并发导致的锁超时问题，判断锁的设置是否合理等

4. 表结构设计问题，是否满足 BCNF，如果不满足则进一步拆分表结构，使之更加合理

18. 数据库的主从复制

1. 什么是主从复制

   建立和主数据库完全一样的数据库环境，称为从数据库。

2. 主从复制的作用

   a. 做容灾备份，如果主库挂掉可以立刻切换至从库，这样可以保证线上服务正常运行，并且不会丢失新写入的数据。

   b. 读写分离，提高数据库的并发能力。主库负责写操作，从库负责读操作。

3. 主从复制的原理

   a. 新增操作写入主库的 binlog 文件，该文件中存放了所有操作的记录。

   b. 从库发起连接，连接到主库

   c. 主库创建一个线程，把新增 binlog 内容发送到从库

   d. 从库创建一个线程，将发送来的内容写入本地 relay log 文件

   e. 从库创建一个 SQL 线程把新增操作更新到从库中

19. varchar 和 char 的使用场景

char 的长度是不可变的，而 varchar 的长度是可变的，因此 varchar 的空间效率更多，char 的时间效率更高。

20. count(*)、count(1)、count(column)的区别

count(*)和 count(1)都是对行数进行统计，包括为 null 的数据，两者走相同的优化流程，效率一样。count(column)是对 column 这一列进行统计，不包括为 null 的数据。

21. LIKE 声明中的%和_是什么意思

%对应于 0 个或更多字符，_只是 LIKE 语句中的一个字符。

24. SQL 执行时间怎么查询

show profiles

25. 某列有 null 值，索引是否依然生效

是的

26. 讲讲 sql 绑定变量

where a = :xx，sql 每次查询都需要经过解析优化等，sql 绑定变量可以提高复用，大大

减少查询时间。不应使用 sql 绑定变量的情况包括：单一查询时间很长，复用优化的部分可以忽略不计；绑定变量会使得优化器不准确，如果影响较大的情况下不应使用。

参考文章：

1. https://www.jianshu.com/p/4e3edbedb9a8
2. https://database.51cto.com/art/201912/608325.htm
3. https://www.cnblogs.com/fengzheng/p/12557762.html
4. https://juejin.cn/post/6844903569632526343#heading-38
5. https://www.zhihu.com/search?type=content&q=%E9%9D%9E%E8%81%9A%E7%B0%87%E7%B4%A2%E5%BC%95