面试题：
主编：范传奇
更新时间：2023年3月19日

手写SpringBoot

1. HTTP请求结构？HTTP请求头中有什么？

   - 一个HTTP请求报文由四个部分组成：请求行、请求头部、空行、请求数据
   - 请求头

        1.Accept：浏览器可接受的mime类型。

        2.Accept-Charset：浏览器可接受的字符集。

        3.Accept-Encoding：浏览器能够进行解码的方式。

        4.Content-Length：表示请求消息的长度。

        5.Host： 客户端告诉服务器，想访问的主机名。

        6.Cookie：客户端可以向服务器带数据，只是非常重要的信息之一。

2. GET和POST的区别

   - GET方式是通过请求行传递用户所输入的内容，其内容会全部显示的浏览器的地址栏中；
   - GET提交具有长度限制
   - GET是从服务器上获取数据
   - GET请求没有HTTP消息体
   - POST提交将用户所输入数据放到HTTP消息体中发送到服务器端
   - POST没有提交长度限制
   - POST是向服务器传送数据

3. 如何用反射获取方法？
      1 首先找到类型的Class对象
            Class.forName(类型)
            类型.class
            对象.getClass()
      2 然后在类型上查找方法
            getMethods();//获取包括自身和继承（实现）过来的所有的public方法
            getDeclaredMethods();//获取自身所有的方法
            getMethod(方法签名);//表示获取指定的一个公共的方法，包括继承的
            getDeclaredMethod(方法签名);//表示获取本类中的一个指定的方法，不包括继承的方法
            方法签名包括：方法的名字，参数的Class类型列表

4. 什么是反射

   - ava反射（Java Reflection）是指在运行时获取程序的类型信息并可以操作对象的机制。通过反射，可以在运行时获取类的成员变量、方法、构造函数等信息，并可以在运行时调用对象的方法，创建对象的实例，操作对象的属性等。
   - 在Java中，反射机制主要通过以下几个类实现：
     - Class类：表示类的类型，通过该类可以获取类的构造函数、成员变量、方法等信息。
     - Constructor类：表示类的构造函数类型，通过该类可以创建类的实例。
     - Field类：表示类的成员变量类型，通过该类可以获取、设置类的成员变量的值。
     - Method类：表示类的方法类型，通过该类可以调用类的方法。
   - 反射机制在Java中具有广泛的应用，例如：
     - 动态代理：可以通过反射生成实现了某个接口的代理类，实现动态代理。
     - 依赖注入：可以通过反射获取类的构造函数、成员变量等信息，实现依赖注入。

- 注解处理器：可以通过反射获取类的注解信息，实现注解处理器。
- 配置文件解析器：可以通过反射获取类的属性信息，实现配置文件解析器。
- 尽管反射机制可以方便地获取类的信息并进行操作，但是它也具有一定的缺点，如性能较差、代码可读性差、安全性差等。因此，在实际开发中，应该根据实际情况谨慎使用反射机制。

数据库

1. 数据库中常见的聚合函数？

   - 在关系型数据库中，聚合函数（Aggregate Functions）是用于计算数据集合中的统计值的函数。常见的聚合函数包括以下几个：
     - COUNT：计算数据集合中的记录数。
     - SUM：计算数据集合中某个字段的总和。
     - AVG：计算数据集合中某个字段的平均值。
     - MAX：获取数据集合中某个字段的最大值。
     - MIN：获取数据集合中某个字段的最小值。
     - 这些聚合函数可以与 SQL 语句的 SELECT 语句结合使用，对数据集合进行统计和汇总。

2. 现有数据库tedu用于保存一所学校的组织结构。具体的表结构如下图所示

   ![tedu数据库结构图.png](tedu数据库结构图.png)

3. 教"英语"的老师都是谁?

   ```sql
   SELECT t.name,su.name
   FROM teacher t,subject su
   WHERE t.subject_id=su.id
   AND su.name='英语'
   ```

4. 老师"范传奇"代班的学生共多少人?

   ```sql
   SELECT COUNT(*)
   FROM teacher t,class c,student s
   WHERE t.id=c.teacher_id
   AND c.id=s.class_id
   AND t.name='范传奇'
   ```

5. 查询教语文的老师的平均工资是多少?

   ```sql
   SELECT AVG(t.salary)
   FROM teacher t,subject su
   WHERE t.subject_id=su.id
   AND su.name='语文'
   ```

6. 查看每门课老师的平均工资分别是多少?结果集中要体现:平均工资及科目名称

   ```sql
   ```

```sql
SELECT AVG(t.salary),su.name
FROM teacher t,subject su
WHERE t.subject_id=su.id
GROUP BY su.name
HAVING AVG(t.salary)>6000
ORDER BY AVG(salary) DESC
```

7. 查看所有教**英语**的老师带的来自**上海**或**南京**的学生都有谁?

```sql
SELECT s.name,s.age,c.name,t.name,l.name
FROM student s,class c,teacher t,location l,subject su
WHERE s.class_id=c.id
AND c.teacher_id=t.id
AND t.subject_id=su.id
AND s.location_id=l.id
AND su.name='英语'
AND l.name IN ('上海','南京')
```

8. 查询工资最低的老师带的学生共多少人?

```sql
SELECT COUNT(*)
FROM student s,class c,teacher t
WHERE s.class_id=c.id
AND c.teacher_id=t.id
AND t.salary=(SELECT MIN(salary) FROM teacher)
```

9. 查看学生"张三"每门课程的成绩是多少?

```sql
SELECT s.name,su.name,sss.score
FROM student s,subject su,t_stu_subject_score sss
WHERE s.id=sss.stu_id
AND sss.subject_id=su.id
AND s.name='张三'
```

10. 查看1年级1班所有同学的语文成绩是多少?

```sql
SELECT s.name,su.name,sss.score,c.name
FROM student s,class c,subject su,t_stu_subject_score sss
WHERE s.class_id=c.id
AND s.id=sss.stu_id
AND su.id=sss.subject_id
AND c.name='1年级1班'
AND su.name='语文'
```

```

11. 统计1年级1班数学成绩的平均值?

```sql
SELECT AVG(sss.score)
FROM student s,class c,subject su,t_stu_subject_score sss
WHERE s.class_id=c.id
AND s.id=sss.stu_id
AND su.id=sss.subject_id
AND c.name='1年级1班'
AND su.name='数学'
```

12. 统计6年级的英语成绩的平均值?

```sql
SELECT AVG(sss.score)
FROM student s,class c,subject su,t_stu_subject_score sss
WHERE s.class_id=c.id
AND s.id=sss.stu_id
AND su.id=sss.subject_id
AND c.name LIKE '6年级%'
AND su.name='英语'
```

13. 查看4年级数学成绩最高的同学名字?

```sql
SELECT s.name,c.name,sss.score
FROM student s,class c,subject su,t_stu_subject_score sss
WHERE s.class_id=c.id
AND s.id=sss.stu_id
AND su.id=sss.subject_id
AND c.name LIKE '4年级%'
AND su.name='数学'
AND sss.score=(SELECT MAX(sss.score)
                  FROM student s,class c,subject su,t_stu_subject_score sss
                     WHERE s.class_id=c.id
                     AND  s.id=sss.stu_id
                     AND su.id=sss.subject_id
                     AND c.name LIKE '4年级%'
                     AND su.name='数学')
```

14. 查看老师"刘苍松"所带班级的英语平均分?

```sql
SELECT AVG(sss.score)
FROM student s,class c,teacher t,subject su,t_stu_subject_score sss
WHER s.class_id=c.id
```
```

```
AND s.id=sss.stu_id
AND sss.subject_id=su.id
AND c.teacher_id=t.id
AND t.name='刘苍松'
AND su.name='英语'
```

15. 查看工资最高的老师所带班级的各科成绩的平均分，最高分和最低分分别是多少?

```sql
SELECT MAX(sss.score),MIN(sss.score),AVG(sss.score),su.name
FROM student s,class c,t_stu_subject_score sss,subject su,teacher t
WHERE s.class_id=c.id
AND s.id=sss.stu_id
AND sss.subject_id=su.id
AND c.teacher_id=t.id
AND t.salary=(SELECT MAX(salary) FROM teacher)
GROUP BY su.name
```

16. 查看所有大队长的5门成绩平均分是多少？

```sql
SELECT s.name,AVG(sss.score) avg_score
FROM student s,t_stu_subject_score sss,subject su
WHERE s.id=sss.stu_id
AND sss.subject_id=su.id
AND s.job='大队长'
GROUP BY s.name
ORDER BY avg_score DESC
```

17. 关联查询有几种？列举常见的关联查询。

内联查询：inner join 和 join，两者效果相同
左外连接：left join
右外连接：right join
全连接：mysql不支持 full join，使用union替代

自链接

18. mysql查询出来的结果分页展示一般怎么做？

```sql
方式1：

select * from table order by id limit m, n;
```

很简单，该语句的意思就是查询m+n条记录，去掉前m条，返回后n条。无疑该查询能够实现分页，但m越大，查询性能就越低，因为MySQL需要扫描全部m+n条记录。

方式2：

```
select * from table where id > #max_id# order by id limit n;
```

该查询同样会返回后n条记录，却无需像方式1扫描前m条记录，但必须在每次查询时拿到上一次查询（上一页）的最大id（或最小id），是比较常用的方式。

当然该查询的问题也在于我们不一定能拿到这个id，比如当前在第3页，需要查询第5页的数据，就不行了。

方式3：

为了避免方式2不能实现的跨页查询，就需要结合方式1。

性能需要，m得尽量小。比如当前在第3页，需要查询第5页，每页10条数据，且当前第3页的最大id为#max_id#，则：

```
select * from table where id > #max_id# order by id limit 10, 10;
```

该方式就部分解决了方式2的问题，但如果当前在第2页，要查第1000页，性能仍然较差。

方式4：

```
select * from table as a inner join (select id from table order by id limit m, n) as b on a.id = b.id order by a.id;
```

该查询同方式1一样，m的值可能很大，但由于内部的子查询只扫描了id字段，而非全表，所以性能要强于方式1，并且能够解决跨页查询问题。

方式5：

```
select * from table where id > (select id from table order by id limit m, 1) limit n;
```

该查询同样是通过子查询扫描字段id，效果同方式4。但方式5的性能会略好于方式4，因为它不需要进行表的关联，而是一个简单的比较，在不知道上一页最大id的情况下，是比较推荐的用法。

```

## 第二教学月涉及不到的数据库面试题，后续可能需要

1. 什么是事务，以及事务的应用，举例说明

事务：指构成单个逻辑工作单元的操作集合

事务是指是程序中一系列严密的逻辑操作，而且所有操作必须全部成功完成，否则在每个操作中所作的所有更改都会被撤消。可以通俗理解为：就是把多件事情当做一件事情来处理，好比大家同在一条船上，要活一起活，要完一起完

```
原子性（Atomicity）：操作这些指令时，要么全部执行成功，要么全部不执行。只要其中一个指令执行失败，所有的指令都执行失败，数据进行回滚，回到执行指令前的数据状态。

一致性（Consistency）：事务的执行使数据从一个状态转换为另一个状态，但是对于整个数据的完整性保持稳定。
```
```

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。

即要达到这么一种效果：对于任意两个并发的事务T1和T2，在事务T1看来，T2要么在T1开始之前就已经结束，要么在T1结束之后才开始，这样每个事务都感觉不到有其他事务在并发地执行。

持久性（Durability）：当事务正确完成后，它对于数据的改变是永久性的。
```

2. 对SQL的优化方式有哪些

```
减少数据访问： 设置合理的字段类型，启用压缩，通过索引访问等减少磁盘IO
返回更少的数据： 只返回需要的字段和数据分页处理 减少磁盘io及网络io
减少交互次数： 批量DML操作，函数存储等减少数据连接次数
减少服务器CPU开销： 尽量减少数据库排序操作以及全表查询，减少cpu 内存占用
利用更多资源： 使用表分区，可以增加并行操作，更大限度利用cpu资源

总结到SQL优化中，就三点：

最大化利用索引；
尽可能避免全表扫描；
减少无效数据的查询；

分析语句，是否加载了不必要的字段/数据。

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

查看 SQL 涉及的表结构和索引信息。

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。
```

3. 创建索引的条件是什么

- 字段的数值有唯一性的限制

  业务上具有唯一特性的字段，即使是组合字段，也必须建成唯一索引。

  说明：不要以为唯一索引影响了 `insert` 速度，这个速度损耗可以忽略，但提高查找速度是明显的。

- 频繁作为 WHERE 查询条件的字段

- 经常 GROUP BY 和 ORDER BY 的列
  索引就是让数据按照某种顺序进行存储或检索，因此当我们使用 GROUP BY 对数据进行分组查询，
```

或者使用 ORDER BY 对数据进行排序的时候，就需要 对分组或者排序的字段进行索引 。如果待排序的列有多个，那么可以在这些列上建立 组合索引 。

- UPDATE、DELETE 的 WHERE 条件列
  对数据按照某个条件进行查询后再进行 UPDATE 或 DELETE 的操作，如果对 WHERE 字段创建了索引，就能大幅提升效率。原理是因为我们需要先根据 WHERE 条件列检索出来这条记录，然后再对它进行更新或删除。如果进行更新的时候，更新的字段是非索引字段，提升的效率会更明显，这是因为非索引字段更新不需要对索引进行维护。

- DISTINCT 字段需要创建索引

- 多表 JOIN 连接操作时，创建索引注意事项
  对数据按照某个条件进行查询后再进行 UPDATE 或 DELETE 的操作，如果对 WHERE 字段创建了索引，就能大幅提升效率。原理是因为我们需要先根据 WHERE 条件列检索出来这条记录，然后再对它进行更新或删除。如果进行更新的时候，更新的字段是非索引字段，提升的效率会更明显，这是因为非索引字段更新不需要对索引进行维护。

- 使用列的类型小的创建索引

- 使用字符串前缀创建索引

- 区分度高(散列性高)的列适合作为索引

- 使用最频繁的列放到联合索引的左侧

- 在多个字段都要创建索引的情况下，联合索引优于单值索引

4. 说说事务的ACID和隔离级别

```
原子性（Atomicity）：操作这些指令时，要么全部执行成功，要么全部不执行。只要其中一个指令执行失败，所有的指令都执行失败，数据进行回滚，回到执行指令前的数据状态。

一致性（Consistency）：事务的执行使数据从一个状态转换为另一个状态，但是对于整个数据的完整性保持稳定。

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。

持久性（Durability）：当事务正确完成后，它对于数据的改变是永久性的。
为了达到事务的四大特性，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中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该级别 （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。

MySQL 默认采用的 REPEATABLE_READ 隔离级别。
```

5. 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候的内容是一致，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，就需要进入锁等待。

    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，但当前读的时候会有幻读。
```