11、JDBC连接数据库的步骤？

JDBC连接数据库可以概括为6步，分别是加载JDBC驱动、建立数据库连接、创建一个语句对象、执行一个查询、处理结果集和关闭数据库连接。

1、加载JDBC驱动

使用java反射机制中的方法forName()进行加载，如：Class.forName("com.mysql.jdbc.Driver");

2、建立数据库连接

驱动管理类DriverManager使用特定的驱动程序，通过getConnection(String url)方法建与某个特定数据库的连接。每个JDBC驱动都对应一个URL地址用于自我标识

3、创建一个语句对象

创建一个语句对象则需要调用接口java.sql.Connection中的createStatement()方法创建Statement类的语句对象，

通过创建一个语句对象则可以发送SQL语句到数据库准备执行相应的操作。

4、执行SQL语句

将SQL语句发送到数据库之后，根据发送的SQL语句确定执行executeQuery()方法或executeUpdate()方法。

5、处理结果集

如果需要从返回的结果集中获取数据，那么可以通过结果集对象调用ResultSet接口的getXXX()方法进行获取。

6. 关闭数据库连接

结果集处理完成之后，为了释放资源需要在finally语句块中首先关闭语句对象，再关闭数据库连接，如：

```
stmt.close();     //关闭语句对象。
conn.close();     //关闭数据库连接。
```

12、浮点数运算

用BigDecimal

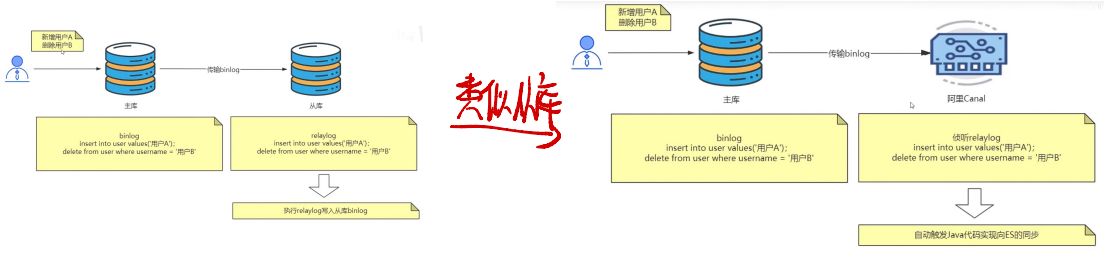
```
public class Test {
public static void main(String[] args) {
   System.out.println((new BigDecimal("2.0")).subtract(new BigDecimal("1.9")).doubleValue());
   }
}
```

```
api:
    1、BigDecimal(T)
    2、add(BigDecimal) BigDecimal对象中的值相加，返回BigDecimal对象
    3、subtract(BigDecimal) BigDecimal对象中的值相减，返回BigDecimal对象
    4、multiply(BigDecimal) BigDecimal对象中的值相乘，返回BigDecimal对象
    5、divide(BigDecimal) BigDecimal对象中的值相除，返回BigDecimal对象
    6、doubleValue() 将BigDecimal对象中的值转换成双精度数
    7、intValue() 将BigDecimal对象中的值转换成整数
    8、int a = bigdemical.compareTo(bigdemical2) java中对BigDecimal比较大小一般用的是bigdemical的compareTo方法
```

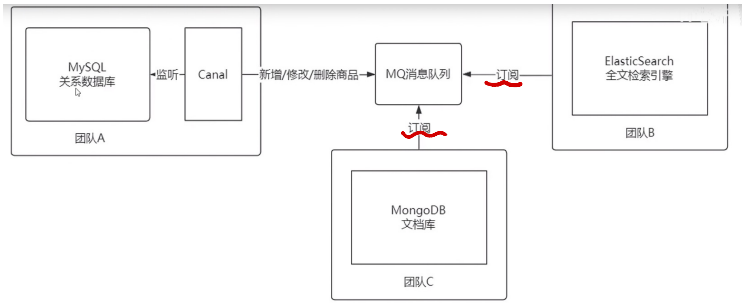
13、如何实现MySQL异构数据同步？

阿里Canal + MQ。主要作用：数据监听 + 解耦。

canal原理：



整体方案：



14、工厂模式?

目的：实现调用者和创建者的解耦，调用者无需了解创建细节（比如相关参数），只需使用

核心本质：

实例化对象不适用new，用工厂方法代替

将选择实现类，创建对象统一管理和控制，从而将调用者跟我们的实现类解耦

三种模式：

简单工厂模式：

用来生产同一等级结构中的任何产品(对于增加新的产品，需要扩展已有代码)

工厂方法模式：

用来生产同一等级结构中的固定产品(支持增加任意产品)

抽象工厂模式：

围绕一个超级工厂创建其他工厂，该超级工厂又被称为其他工厂的工厂

15、建造者模式？

使用多个简单的对象一步一步构建成一个复杂的对象，将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示

指挥者通过指挥不同的工人、按不同的顺序建造，可以得到不同的产品。相比工厂模式的生产零件，建造者模式更像汽车的组装工厂。

允许用户只通过指定复杂对象的类型和内容就可以构建它们，不需要知道内部的具体构建细节

何时使用：一些基本部件不会变，而其组合经常变化的时候。

核心组成：

1Builder：抽象建造者，定义多个通用方法和构建方法

2ConcreteBuilder：具体建造者，可以有多个

3Director：指挥者，控制整个组合过程，将需求交给建造者，由建造者去创建对象

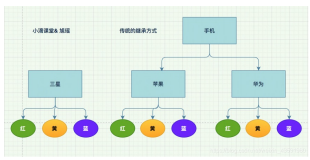
4Product：产品角色

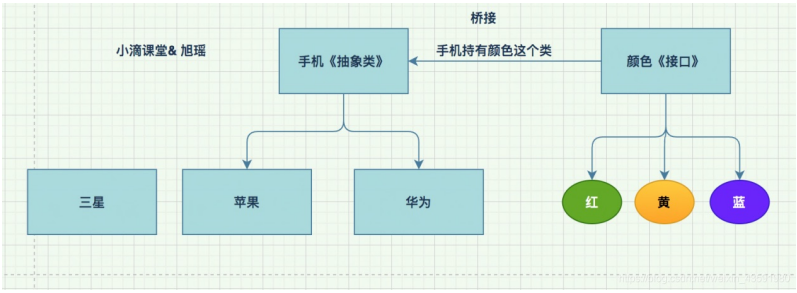
16、桥接模式？

桥接模式提高了系统的可扩展性，在不同的维度中任意扩展一个维度，都不需要修改原系统，负责开闭原则。

就像一座桥，把两个有独立变化的维度组合起来。

例子：JDBC的驱动程序、JAVA虚拟机实现了平台的无关性。





17、单一索引和联合索引优先使用哪个？

当创建**(a,b,c)联合索引时，相当于创建了(a)单列索引**，(a,b)联合索引以及**(a,b,c)联合索引**

复合索引的结构与电话簿类似，人名由姓和名构成，电话簿首先按姓氏对进行排序，然后按名字对有相同姓氏的人进行排序。

如果您知道姓，电话簿将非常有用；如果您知道姓和名，电话簿则更为有用，但如果您只知道名不姓，电话簿将没有用处。

所以说创建复合索引时，应该仔细考虑列的顺序。对索引中的所有列执行搜索或仅对前几列执行搜索时，复合索引非常有用；

仅对后面的任意列执行搜索时，复合索引则没有用处。

> 多个单列索引在多条件查询时优化器会选择最优索引策略，可能只用一个索引，也可能将多个索引全用上！
>
> 但多个单列索引底层会建立多个B+索引树，比较占用空间，也会浪费一定插入修改效率，故如果只有多条件联合查询时最好建联合索引！

18、什么是跨域问题？Springboot中关于跨域问题的解决方法？

URL由协议、域名、端口和路径组成，如果两个URL的协议、域名和端口全部相同，则表示他们同源。

否则，只要协议、域名、端口有任何一个不同，就是跨域。

> 在Spring Boot 2.X应用程序中可以使用注解@CrossOrigin，也可以通过使用WebMvcConfigurer对象来定义全局CORS配置。
>
> 1) 可以通过实现WebMvcConfigurer接口，然后重写addCorsMappings方法解决跨域问题。
>
> 2) 在Controller或者其中业务方法上加上注解@CrossOrigin(origins = "http://localhost:8080");

19、各种MQ优缺点？

Kafka、ActiveMQ、RabbitMQ、RocketMQ 有什么优缺点？

特性	ActiveMQ	RabbitMQ	RocketMQ	Kafka
单机吞吐量	万级，比 RocketMQ、Kafka 低一个数量级	同 ActiveMQ	10 万级，支撑高吞吐	10 万级，高吞吐，一般配合大数据类的系统来进行实时数据计算、日志采集等场景
时效性	ms 级	微秒级，这是 RabbitMQ 的一大特点，延迟最低	ms 级	延迟在 ms 级以内
可用性	高，基于主从架构实现高可用	同 ActiveMQ	非常高，分布式架构	非常高，分布式，一个数据多个副本，少数机器宕机，不会丢失数据，不会导致不可用
消息可靠性	有较低的概率丢失数据	基本不丢	经过参数优化配置，可以做到 0 丢失	同 RocketMQ

20、Redis的使用场景？

1、热点数据的缓存

由于redis访问速度块、支持的数据类型比较丰富，所以redis很适合用来存储热点数据，另外结合expire，

我们可以设置过期时间然后再进行缓存更新操作，这个功能最为常见，我们几乎所有的项目都有所运用。

2、限时业务的运用

redis中可以使用expire命令设置一个键的生存时间，到时间后redis会删除它。利用这一特性可以运用在限时的优惠活动信息、手机验证码等业务场景。

3、计数器相关问题

redis由于incrby命令可以实现原子性的递增，所以可以运用于高并发的秒杀活动、分布式序列号的生成、

具体业务还体现在比如限制一个手机号发多少条短信、一个接口一分钟限制多少请求、一个接口一天限制调用多少次等等。

4、排行榜相关问题

关系型数据库在排行榜方面查询速度普遍偏慢，所以可以借助redis的ZSet进行热点数据的排序。

5、分布式锁

这个主要利用redis的setnx命令进行，setnx："set if not exists"就是如果不存在则成功设置缓存同时返回1，否则返回0，

因为我们服务器是集群的，定时任务可能在两台机器上都会运行，所以在定时任务中首先通过setnx设置一个lock，如果成功设置则执行，

如果没有成功设置，则表明该定时任务已执行。 当然结合具体业务，我们可以给这个lock加一个过期时间，防止死锁的出现。

6、分布式的集中存储

分布式session、分布式id的集中存储之地。