用户行为分析

一个二手交易平台

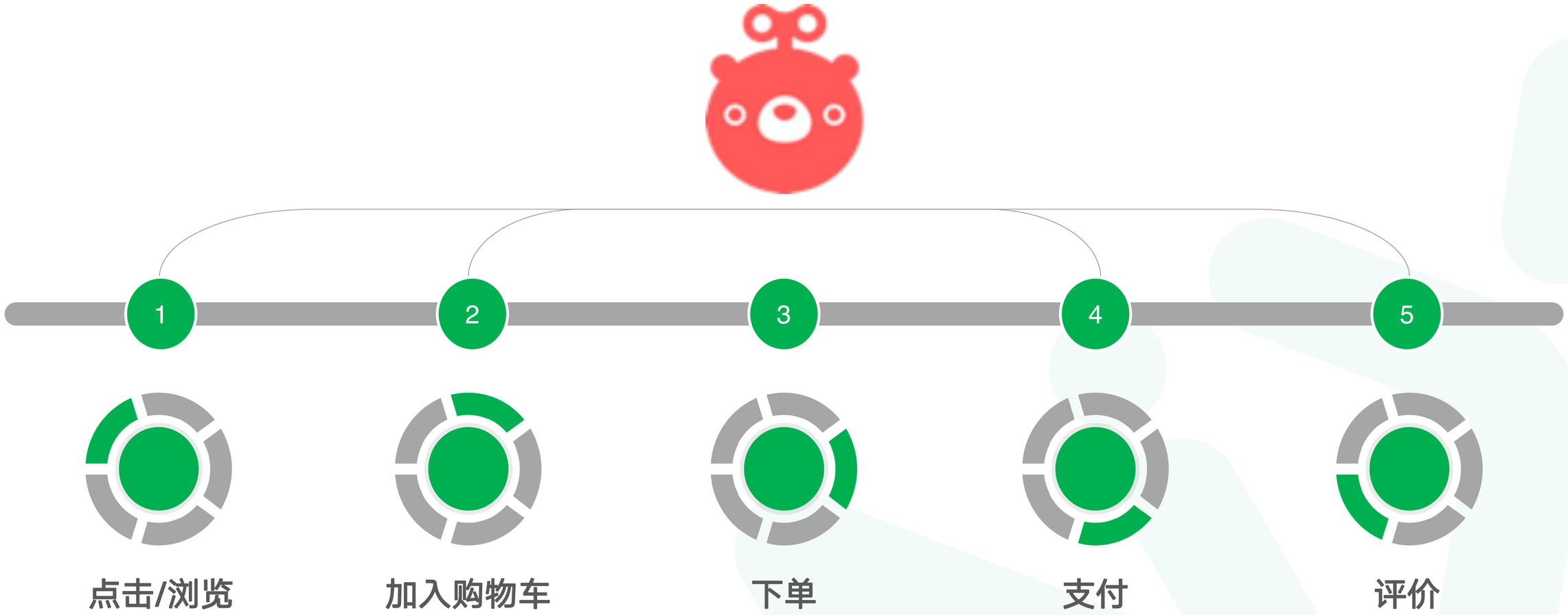
一个帮你赚钱的网站

每年帮助超过1000万用户卖出宝贝


中国联通 4G
下午 5:17
转转二手交易网
华为平板
热搜：
限时加价10%
airpods
苹果6
iphonex
小米
双11Super福利提前抢
预付1元可抵388·爆款手机免费抽
专业质检
已完成68781272次验机服务
专业智能验机
放心严选好货
手机3C保卖
腾讯战略投资
苹果手机
安卓手机
平板
电脑
特价零食
文玩玉翠
自营图书
二手车
限时秒杀
15:42:22
全部秒杀
办公无线鼠标
首页
省心卖
购物车
我的


中国联通 4G
下午 5:31
转转二手交易网
华为平板
热搜：
限时加价10%
小米9
iphonex
airpods
小米
老罗推荐
转转严选二手手机
官方验机 | 1年质保 | 7天无理由退货
专业质检
已完成68781845次验机服务
专业智能验机
放心严选好货
手机3C保卖
腾讯战略投资
苹果手机
安卓手机
平板
电脑
特价零食
文玩玉翠
自营图书
二手车
限时秒杀
15:28:17
全部秒杀
办公无线鼠标
首页
省心卖
购物车
我的


中国联通 4G
下午 5:30
转转自营二手书
道德经
¥9.00
原价 36
2.5折
全新书
新人
新人30元大礼包
满49减12元
满29减5元
领取
自营图书
道德经
1人卖过这本书
服务
自营图书
消毒塑封
100%正版
满45包邮
作者
(春秋)老子
更多作品
出版社
华龄出版社 2017-06
更多作品
首页
购物车
加入购物车
立即购买


1
2
3
4
5
点击/浏览
加入购物车
下单
支付
评价

Zeye系统

转转属于电商平台，电商平台的用户行为频繁且较复杂，系统上线后每天收集到大量的用户行为数据， 公司需要一个系统利用大数据技术进行深入挖掘和分析， 得到感兴趣的商业指标用来做数据分析和商业决策，并增强对风险的控制，故转转开发了Zeye系统。

奈学教育

01 实时统计热门商品

02 实时统计热门页面

03 实时统计PV/UV/GMV

04 实时统计广告点击

05 实时风控

01 实时统计热门商品

需求分析

每隔5分钟统计最近1个小时热门商品

数据展示

用户编号	商品编号	品类编号	用户行为	访问时间
0231-/,	-. 0/ 30/ 5	0. 52144	pv	-2, -24411.


1
读取数据
2
添加水位
3
过滤用户行为
4
按商品分组
5
统计窗口数据
6
根据窗口分组
7
商品TopN排序
8
打印输出

实现思路

```
时间: 2020-10-03 09:40:00.0
 商品ID: 21123024 商品浏览量=6
 商品ID: 14188074 商品浏览量=5
 商品ID: 3104010 商品浏览量=5
================================
时间: 2020-10-03 09:45:00.0
 商品ID: 21123024 商品浏览量=8
 商品ID: 1219464 商品浏览量=7
 商品ID: 3104010 商品浏览量=6
================================
时间: 2020-10-03 09:50:00.0
 商品ID: 1219464 商品浏览量=9
 商品ID: 20837616 商品浏览量=8
 商品ID: 21123024 商品浏览量=8
```

02

实时统计热门页面

需求分析

每隔5秒统计最近10分钟热门页面

数据展示

IP地址	用户ID	事件时间	请求方式	URL
24.233.162.179	-	17/05/2020:11:05:31 +0000	GET	/images/jordan-80.png

实现思路

```
时间: 2020-05-20 21:13:25.0
URL:/blog/tags/puppet?flav=rss20 访问量: 6
URL:/projects/xdotool/ 访问量: 4
URL:/favicon.ico 访问量: 4
URL:/images/web/2009/banner.png 访问量: 3
URL:/presentations/logstash-puppetconf-2012/css/reset.css 访问量: 3
====================
时间: 2020-05-20 21:13:30.0
URL:/blog/tags/puppet?flav=rss20 访问量: 6
URL:/projects/xdotool/ 访问量: 4
URL:/favicon.ico 访问量: 4
URL:/images/web/2009/banner.png 访问量: 3
URL:/presentations/logstash-puppetconf-2012/css/reset.css 访问量: 3
====================
```

03 实时统计PV/UV/GVM

需求分析

每小时PV统计

数据展示

用户编号	商品编号	品类编号	用户行为	访问时间
0231-/,	-.0/30/5	0.52144	pv	-2,-24411.

实现思路

```
(pv,24164)
(pv,13)
```

01

UV统计

需求分析

每小时UV统计

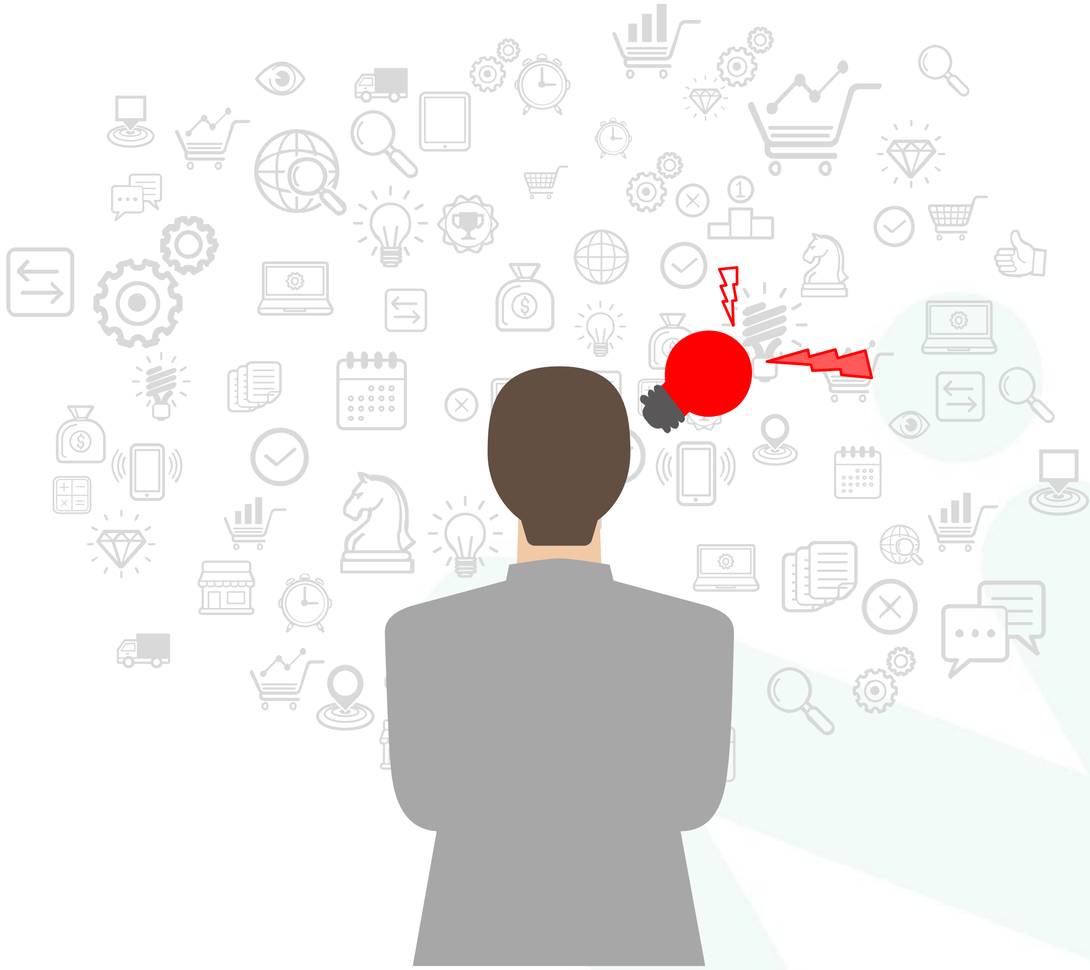
数据展示

用户编号	商品编号	品类编号	用户行为	访问时间
0231-/,	-.0/30/5	0.52144	pv	-2,-24411.

实现思路

```
UvInfo(2020-10-03 10:00:00.0,17416)
UvInfo(2020-10-03 11:00:00.0,13)
```

数据量大了，内存不够？

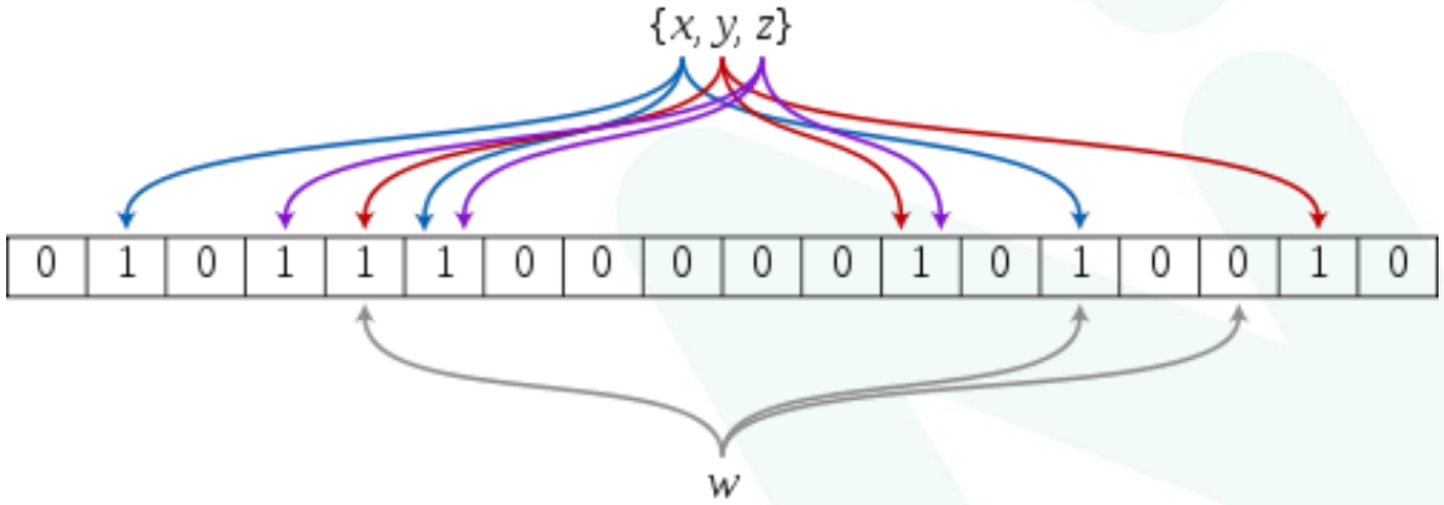


直观的说，bloom算法类似一个hash set，用来判断某个元素（key）是否在某个集合中。

和一般的hash set不同的是，这个算法无需存储key的值，对于每个key，只需要k个比特位，每个存储一个标志，用来判断key是否在集合中。

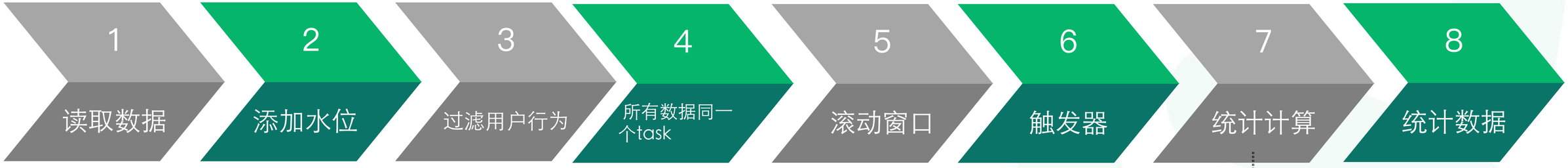
算法：

1. 首先需要k个hash函数，每个函数可以把key散列成为1个整数
2. 初始化时，需要一个长度为n比特的数组，每个比特位初始化为0
3. 某个key加入集合时，用k个hash函数计算出k个散列值，并把数组中对应的比特位置为1
4. 判断某个key是否在集合时，用k个hash函数计算出k个散列值，并查询数组中对应的比特位，如果所有的比特位都是1，认为在集合中。



在我们平时开发过程中，会有一些 bool 型数据需要存取，比如用户一年的签到记录，签了是 1，没签是 0，要记录 365 天。如果使用普通的 key/value，每个用户要记录 365个，当用户上亿的时候，需要的存储空间是惊人的。为了解决这个问题，Redis 提供了位图数据结构，这样每天的签到记录只占据一个位，**365 天就是 365 个位**，46 个字节 (一个字节有8位) 就可以完全容纳下，**这就大大节约了存储空间**。

位图不是特殊的数据结构，它的内容其实就是普通的字符串，也就是 byte 数组。我们可以使用普通的 get/set 直接获取和设置整个位图的内容，也可以使用位图操作 getbit/setbit等将 byte 数组看成「位数组」来处理。



实现思路

1. 写一个布隆过滤器
2. 计算当前用户编号hash值
3. 计算在布隆过滤器的位置
4. 根据上个步骤更新结果

```
UvInfo(2020-10-03 10:00:00.0,17416)
UvInfo(2020-10-03 11:00:00.0,13)
```

01 实时广告点击统计

需求分析

1. 实时生成黑名单(同一个用户，同一个广告点击)
2. 每隔5秒统计最近1小时的各省份的广告点击

数据展示

用户ID	广告ID	省份	城市	时间
543462	1715	广东	深圳	1511658600

3 用户ID，广告ID分组

4 一天之内点击超过100次的进入黑名单

5 根据侧输出黑名单

1 读取数据

2 添加水位

3 省份分组

4 滑动窗口

5 结果统计

实现思路

```
BlackListWarning(937166,1715,Click over100 times)
```

```
CountByProvince(2017-11-26 09:06:05.0,上海,1)
CountByProvince(2017-11-26 09:06:05.0,北京,2)
CountByProvince(2017-11-26 09:06:05.0,广东,4)
CountByProvince(2017-11-26 09:06:10.0,上海,1)
CountByProvince(2017-11-26 09:06:10.0,北京,2)
CountByProvince(2017-11-26 09:06:10.0,广东,4)
CountByProvince(2017-11-26 09:06:15.0,上海,1)
CountByProvince(2017-11-26 09:06:15.0,北京,2)
CountByProvince(2017-11-26 09:06:15.0,广东,4)
```

01

实时用户检测

需求分析

2秒之内连续失败2次，则进行风控

数据展示

用户ID	IP	用户行为	时间
5692	66.249.3.15	fail	1558430844

1. 设置state，存储失败任务
2. 2s之内超过2次，告警

实现思路

-* 乱序的数据不好处理

.* 很难应对复杂多变风控的需求

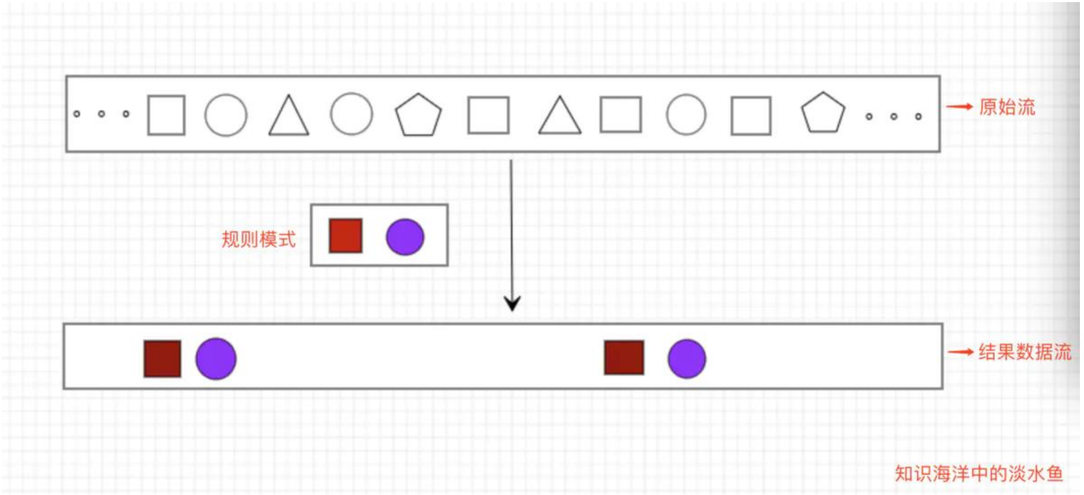
1、什么是CEP

复杂事件处理(Complex Event Processing, CEP)

2、什么是复杂事件

我们先来给这样的业务来定这样一个描述：检测和发现无界事件流中多个记录的关联规则，也就是从无界事件流中得到满足规则的复杂事件。

CEP(Complex Event Processing)就是在无界事件流中检测事件模式，让我们掌握数据中重要的部分。flink CEP是在flink中实现的复杂事件处理库。



1.读取数据

val loginEventStream = env.addSource(.....)

2.定义匹配规则

```
val loginFailPattern = Pattern.begin[LoginEvent]( name = "begin").where(_.eventType == "fail")
  .next( name = "next").where(_.eventType == "fail")
  .within(Time.seconds( seconds = 3))
```

3. 在事件流上应用匹配规则

```
val patternStream = CEP.pattern(loginEventStream, loginFailPattern)
```

4. 提取匹配事件

```
val loginFailDataStream = patternStream.select( new LoginFailMatch() )
```

需求分析

- 个体模式(Individual Patterns)

  -组成复杂规则的每一个单独的模式定义，就是“个体模式”

  start.times (3).where(_.behaviar<startsWith(“av”))

- 序列模式(Combining Patterns)

  -很多个体模式组合起来，就形成了整个的模式序列

  -模式序列必须以一个“初始模式”开始：

  val start = Pattern.begin("start")

- 模式组(Groups of patterns)

  -将一个序列模式作为条件嵌套在个体模式里，成为一组模式

个体模式

包括“单例(singleton)模式”和“循环(looping)模式” 单例模式只接收一个事件，而循环模式可以接收多个 ，通过量词(Quantifier)指定。

量词

可以在一个个体模式后追加量词，也就是指定循环次数

```
// expecting 4 occurrences
start.times(4)

// expecting 0 or 4 occurrences
start.times(4).optional()

// expecting 2, 3 or 4 occurrences
start.times(2, 4)
```

```
// expecting 1 or more occurrences
start.oneOrMore()

// expecting 0, 2 or more occurrences and repeating as many as possible
start.timesOrMore(2).optional().greedy()
```

（-）pie ao：指定固定的循环执行次数
（.）kl pekj]h：通过Kl pekj]h关键字指定要么不触发要么触发指定的次数
（/）cnaa`u：贪婪模式，在l]pajnj匹配成功的前提下会尽可能多的触发
（0）kj aKnl kna：指定触发一次或多次
（1）pie aoKnl kna：指定触发固定次数以上

```
// expecting 2, 3 or 4 occurrences and repeating as many as possible
start.times(2, 4).greedy()
```

条件

每个模式都需要指定触发条件，作为模式是否接受事件进入的判断依据，CEP中的个体模式主要通过调用.where() .or()和.until。来指定条件按不同的调用方式，可以分成以下几类：简单条件，组合条件，迭代条件，终止条件

简单条件(Simple Condition)

通过.where()方法对事件中的字段进行判断筛选，决定是否接受该事件

```
start.where(event => event.getName.startsWith("foo"))
```

组合条件(Combining Condition)

将简单条件进行合并；.or()方法表示或逻辑相连，where的直接组合就是AND

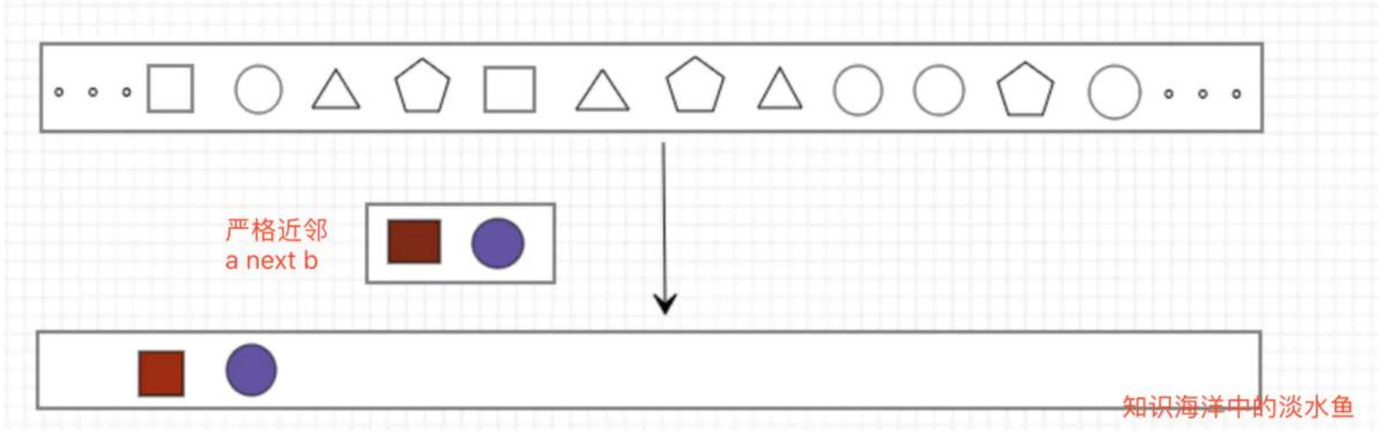
```
pattern.where(event => ... /* some condition */).or(event => ... /* or condition */)
```

终止条件(Stop Condition)

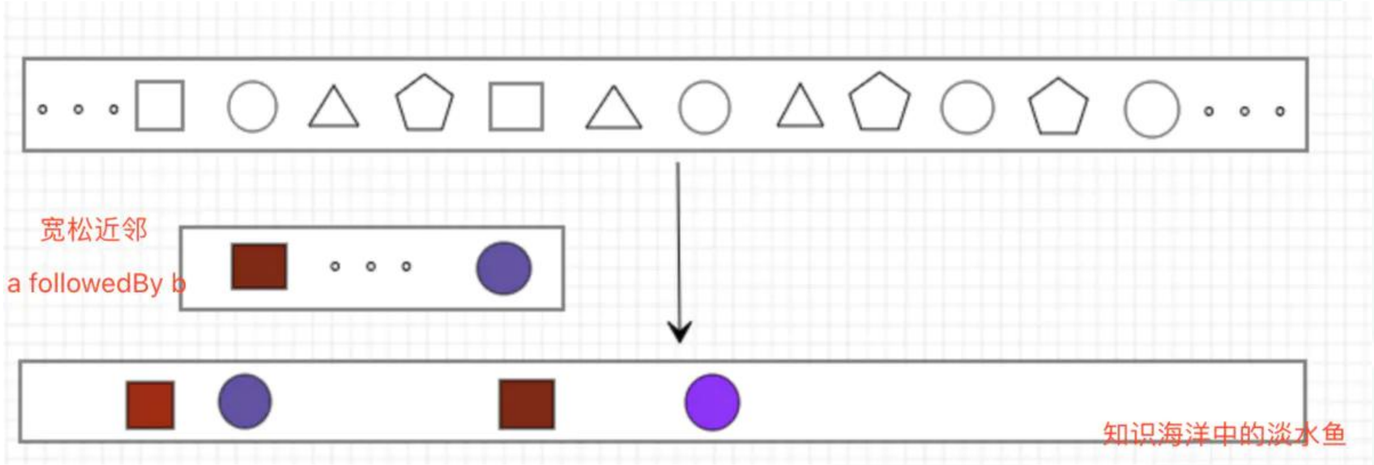
如果使用了 oneOrMore或者oneOrMore.optional

建议使用.until()作为终止条件

严格近邻



宽松近邻



严格近邻(Strict Contiguity)

所有事件按照严格的顺序出现，中间没有任何不匹配的事件，由.next。指定

例如对于模式” a next b” ,事件序列［a, c, bl, b2］没有匹配

宽松近邻(Relaxed Contiguity)

允许中间出现不匹配的事件，由.followedBy()指定

例如对于模式” a followed By b",事件序列［a,c,bl,b2］匹配为{a, bl}

非确定性宽松近邻(Non-Deterministic Relaxed Contiguity)

进一步放宽条件，之前已经匹配过的事件也可以再次使用，由.followedByAny()指定

例如对于模式” a followedByAny b” ,事件序列［a, c, bl, b2］匹配为{a, bl), {a, b2)

注意

1. 如果不希望出现某种近邻关系：

   notNext()——不想让某个事件严格紧邻前一个事件发生

   notFollowedByO——不想让某个事件在两个事件之间发生

2. 所有模式序列必须以.begin。开始

3. 模式序列不能以.notFollowedBy()结束

4. “not”类型的模式不能被0ptional所修饰

5. 可以为模式指定时间约束，用来要求在多长时间内匹配有效

```
val loginFailPattern = Pattern.begin[LoginEvent]( name = "begin").where(_.eventType == "fail")
  .next( name = "next").where(_.eventType == "fail")
  .within(Time.seconds( seconds = 3))
```

指定要查找的模式序列后，就可以将其应用于输入流以检测潜在匹配

```
val patternStream = CEP.pattern(loginEventStream, loginFailPattern)
```

```
val loginFailDataStream = patternStream.select( new LoginFailMatch() )
```

```
class LoginFailMatch() extends PatternSelectFunction[LoginEvent, Warning]{
  override def select(map: util.Map[String, util.List[LoginEvent]]): Warning = {
    // 从map中按照名称取出对应的事件
    val firstFail = map.get("begin").iterator().next()
    val lastFail = map.get("next").iterator().next()
    Warning( firstFail.userId, firstFail.eventTime, lastFail.eventTime, "login fail!" )
  }
}
```

奈学教育，一个有干货更有温度的教育品牌

出品 ： 奈学教育