**Log4j日志输出格式**

（1）为避免不同需求的日志同时输出到HDFS文件或输出到Kafka时相互干扰，需要为每一个需求设计”日志类别标识“来过滤。

热门商品日志格式（离线）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 日志类别标识 | 商品编号 | 商品名称 | 商品类别 | 计数 |
| “hot\_goods" | orderList.getOrderDetail().get(i).getGoods\_id() | goodsinfo.getGoods\_name() | goodsinfo.getGoods\_category() | 1 |

订单信息日志格式（离线）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 日志类别标识 | 订单编号 | 订单合计金额 | 订单状态 | 修改时间 | 收货人姓名 |
| "order" | orderInfo.getOrder\_id() | orderInfo.getOrder\_sum() | orderInfo.getOrder\_status() | orderInfo.getUpdate\_time() | orderInfo.getConsignee\_name() |

hive离线 临时日志汇总表，第一步将HDFS上的日志文件load进此表，热门商品表和订单信息表的数据从此表中取。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 日志类别标识 | 编号 | 商品名称 | 商品类别 | 计数 | 订单合计金额 | 订单状态 | 修改时间 | 收货人姓名 |
| logid | id | goods\_name | goods\_category | one | order\_sum | order\_status | update\_time | consignee\_name |
| "hot\_goods" | orderList.getOrderDetail().get(i).getGoods\_id() | goodsinfo.getGoods\_name() | goodsinfo.getGoods\_category() | 1 | 0 | 0 | 0000-00-00 00:00:00 | 0 |
| "order" | orderInfo.getOrder\_id() | 0 | 0 | 0 | orderInfo.getOrder\_sum() | orderInfo.getOrder\_status() | orderInfo.getUpdate\_time() | orderInfo.getConsignee\_name() |

活跃人数日志（实时），不放进Hive表中，直接进Kafka。

|  |  |
| --- | --- |
| 日志类别标识 | 用户编号 |
| "active\_count" | userInfo.getUserId() |

（2）初步解决方案

离线日志使用log.warn()打印，实时日志使用log.error()打印。

这样，离线日志不会进入到实时日志中，但实时日志还是会进入离线日志，所以需要在离线日志中过滤掉实时日志。

（3）最佳解决方案

对于Kafka有一个专门的类用于写实时日志，这样可以完全与Log4j2打印的日志独立开来。

**需求2：订单信息统计**

**Hive 订单表order\_info**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 订单编号 | 订单合计金额 | 订单状态 | 修改时间 | 收货人姓名 |
| order\_id | order\_sum | order\_status | update\_time | consignee\_name |

**Mysql 订单统计表 order\_count**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **约束** |
| **order\_count\_id** | int | 订单统计表id，自增 | 主键 |
| **order\_sum\_count** | int | 总的订单个数 |  |
| **order\_pay\_count** | int | 成功支付的订单个数 |  |
| **order\_refund\_count** | int | 退款的订单个数 |  |
| **order\_trade\_sum** | decimal(10,2) | 订单交易总额 |  |
| **order\_refund\_sum** | decimal(10,2) | 订单退款总额 |  |
| **order\_count\_date** | date | 当前日期 |  |

从Hive表 order\_count 生成Mysql表的语句：

select

(success.success\_num+fail.fail\_num) order\_sum\_count,

success.success\_num order\_pay\_count,

fail.fail\_num order\_refund\_count,

success.success\_sum order\_trade\_sum,

fail.fail\_sum order\_refund\_sum,

success.day order\_count\_date

from

(**select** day,count(1) success\_num,sum(order\_sum) success\_sum **from** order\_info **where** day='2019-12-25' and order\_status='2' group by order\_status,day) success,

(**select** count(1) fail\_num,sum(order\_sum) fail\_sum **from** order\_info **where** day='2019-12-25' and order\_status='3' group by order\_status) fail

上面的语句存在的问题：

1、当记录全为order\_status='2'，或全为order\_status='3'时，select就没有结果。即订单全为确认支付或全为取消支付时，将导致报表无当天的数据。

2、多表连接没有连接条件，使用的笛卡尔积，导致查询效率下降。

解决方法：

设计一张只有两条数据的临时表，表中有所有的订单状态（确认支付、取消支付）

|  |
| --- |
| 订单状态 |
| 2 |
| 3 |

与订单表order\_info进行左连接，形成order\_info\_tmp表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 订单编号 | 订单合计金额 | 订单状态 | 修改时间 | 收货人姓名 |
| order\_id | order\_sum | order\_status | update\_time | consignee\_name |

当只有order\_status='2'的记录时：（2有若干条记录，3只有一条记录）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 订单状态 | 订单编号 | 订单合计金额 | 订单状态 | 修改时间 | 收货人姓名 |
| 2 | order\_id | order\_sum | 2 | update\_time | consignee\_name |
| 3 | NULL | NULL | NULL | NULL | NULL |

当只有order\_status='3'的记录时：（3有若干条记录，2只有一条记录）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 订单状态 | 订单编号 | 订单合计金额 | 订单状态 | 修改时间 | 收货人姓名 |
| 2 | NULL | NULL | NULL | NULL | NULL |
| 3 | order\_id | order\_sum | 3 | update\_time | consignee\_name |

然后就可以使用函数将NULL替换成0了。

左连接保证了2或3至少出现一次。

接下来就能根据order\_info\_tmp表来生成Mysql表的语句。