# 用户经营项目

--数据来源打点数据—端侧

--云侧数据

--各个华为维修店上传的维修数据

华为的BU分很多：音乐，智慧负一屏，运动健康，华为财经，华为搜索，支付与地图等等，采用的是融合模型的理念，整体采用的是从业务架构到模型架构。

公共模型：

终端设备全量表 dws\_device\_service\_active\_dm pt\_service 业务分区

华为设备信息维表 dim\_common\_oneid\_basic\_ds 货物到达城市名称，货物到达国家名称

华为设备模块信息维表 dim\_common\_device\_basic\_ds 运行内存，机身内存，制造商，尺寸，颜色

Hota设备检测日志小时表 dwd\_evt\_hota\_device\_chk\_log\_hm

设备编号映射 dwd\_eqp\_dev\_id\_mapping\_ds(imei,id)

来源udid(优先取本身last\_date,然后是hota,然后是push)

设备编号映射融合表dwd\_eqp\_dev\_id\_mapping\_nosingle

（华为推送的平台：）

目前所在的项目组是《用户经营项目》：我们的项目主要分为畅联，画像标签，留存，维护解决方案，设备账号，NPS（voc） 等模块，运用的科学的数据作为支撑，服务于产品的销售、售后服务、产品研发改进等，为企业提供精准化运营、从而提高客户留存，以及满意度。

在这个项目中我主要工作职责：

## 在kylin上构建cube（数据立方体）:

Kylin：预计算平台，常用于数据分析和数据索引的技术，可以对原始数据建立多为索引，kylin就是为了将大数据的sql查询速度提高到亚秒级，打破查询时间随数据量增加成线性增加的规律，以空间换时间的做法。

项目组所有cube的构建以及cube的维护（分给我们的资源300TB,使用265TB的时候就会达到预警，我需要和业务沟通删除一些暂时不适用的cube,释放cube的存储空间）。

cube的清理是通过修改retention hold 来实现的

1.disable 2.修改保留周期3.enable

Cube的构建分为3种，1，新增cube。2，增加字段。3字段名称的修改，需要删除model ，删除cube ，然后新建cube，追溯历史数据。

Cube的优化1，清理无用的segemengt 2,联合维度的重新组合，降低膨胀率 3，rowkey调优，基数高的放在基数低的前面，过滤频率高的放在过滤频率低的前面

联合维度的组合是否合理很大程度上决定了cube的膨胀率：一般情况下：age,carer,gender作为一个联合维度

Active\_city ,active\_provice,active\_district,组合成为一个联合维度

设备属性的device\_prod\_sprdname device\_brand,device\_emui\_ver ,device\_catge,device\_series\_name,

Device\_productline 组合成一个联合维度

Device\_launch\_date,device\_launch\_duration组合成一个维度

其他剩下的将来源于同一张表的组合成联合维度

Kylin常常遇到的问题

高基维：维度去重后基数达到百万以上，成为高基维

维度去重后其基数达到千万以上，称为超高基维

处理原则：

高基维直接导致cube膨胀，构建时间长，查询过慢，设计阶段尽可能避免

用户ID，udid,IMEI这些去重指标，用于去重分析

若高基维是维度，那么rowkey设计时，需要将基数高的维度排到rowkey前列，且设置为按期进行重分布（shard by =true）

解决方案

1. 高基维cube失败，extract fact table distinct colums

现象 log :too big dictionary

由于up\_id超大kylin字典超过最大值2G

解决 up\_id有字典编码改为fix\_length定长编码，设置编码长度为64

说明：设置定长编码后，该维度将不再使用字典编码而使用定长编码，将导致数据急剧膨胀，性能下降

去年11月份kylin已经下线，我们现在的按业务划分，各模块的负责人负责把cube任务往clickhouse的搬迁工作。

## 维护解决方案模块脚本的开发以及后续的问题定位

（1，数据异常（比如说某个维度下计算为空，或者数据不对））：

维护解决方案的业务过程包括

1智能检测：我的华为有一个‘服务’内容，做一个基础的检测，自动检测

2智能提醒：关掉流量，手机自动弹出是否要联网，电池老化

3线上服务请求：热线咨询，问题

4远程检测：具体的检测，已经的检测结果

5故障检测：去门店做线下维修的全部检测

6线下维修工单：实体门店维修的手机

7增值权益：保险，屏保，售后

8官修率：去官方门店维修的数目占活跃设备的比列

维护解决方案：留存，故障，留存率，官修率，出险率=出险数/到期量，搭售率=产品销售数/设备总激活数

留存的定义：比如202101月的留存，是指：202001到202012激活的，并且在202101月还活跃，同时满足一线产品,换机周期>20或者二线产品三线产品换机周期>12

## 畅联

是一款场景化APP，可以切换对话场景，以及有共享屏幕的功能。畅联是一款差异化的产品，讲究的是万物互联的概念，可以和冰箱，空调电视机连接。这是目前重点运营的模块，通过这款产品以后可以加快公司向移动支付领域的发展进程以及促进全屋智能系统的构建。

## 7大人群的标签：

爱好音乐人群：按近90天音乐类APP使用天数，时长，安装次数(穿戴-耳机)

商务人群：（办公软件，笔记，邮箱）或（飞机出行爱好者或者火车出行爱好者，或者线上办公爱好者或金融理财爱好者或保险爱好者）（平板，智慧屏）

广义年长人群：55岁以上或简易模式用户（月度使用两天）显示大小设置最大，字体大小设置最大。

游戏爱好人群：p1-p3等级（主打游戏方面的手机）

视觉障碍人群：当月使用屏幕阅读3次以上，并且三方视障碍APP（宇通，慧视助手，解说）

年轻人群：18-34是年轻，年龄1-6个等级，其中2，3等级是年轻人群的标志

爱好运动人群：（华为运动手环）

听觉障碍人群：1讯飞听见月使用5天以上 2.音书月使用天数5天以上 3.周期内使用讯飞与音书月使用总时长30分钟及以上的设备数 4.A1字幕麦克风声源使用天数5天以上。5手之生APP月大于1

**画像的开发流程：**

走隐私流程—提需求—上RAT—开发—配置标签—验证数据—数据OK走上线流程。

隐私流程：

1公共标签：开发团队统一申请，业务团队领域模型和业务标签开发团队无需申请公共标签

2业务标签：由业务部需求方相关人提交电子流发起隐私评审。

3此流程可申请来源表已做黑白名单管理。

数据监控

O层和D层的监控

监控描述：

1. 分区记录数连续3天下降
2. 分区记录数比前一分区记录数差异大于10%
3. 分区记录数比前一分区记录数差异大于20%
4. 分区记录数为空

D层监控：

Moitor\_type 监控类型(数据量的监控，空值的监控)

Abnormal\_indicator 异常指标名

Abnormal\_cnt 异常记录数

Total\_cnt 总记录数

Abnormal\_rate 异常记录数占比

## 数仓架构

DIM

3每天全量，每日一个分区

4魔鬼数字需要关联码值表进行转义

5不管数据源是增量还是全量，都要与昨日累全量的处理。主要是增强模型的健壮性，防止因为数据源未到而导致当天数据为空。

DWD模型设计的方法：

单事务事实表

设计方法：单事务事实表记录事务层面的事实，保存的是原子数据，也称‘原子事实表’。用户可以通过事务事实表对事务进行特别详细的分析。

四步走策略：

1选择业务过程：业务过程是组织完成的操作型活动：注册，登录，使用，付费。明确关键的业务步骤，从而选择与需求有关的业务过程。具体业务过程两个原则，一，基于业务的操作和处理流程，二基于运营的分析场景。播放是一个操作，播放对象分为歌曲和电台。业务过程划分为歌曲播放和电台播放。分享也是客户端上的一个操作，大部分运营对分享的场景不区分对象，所以业务过程为分享。

2声明粒度：业务过程选定后，要针对每个业务过程确定一个粒度，即确定事务事实表每一行所表达的细节层次。一保证所有的事实按照同样的细节层次记录，同一个事实表中不能有多种不同粒度的事实。二尽量学则最细级别的原子粒度，以确保事实表的应用具有最大灵活性。

3确认维度（何人who，何事what，何处where，何时when，如何how）

何人 Who 设备 账号

何处 where 位置

何时 when 日期 时间

何事 what 页面 歌曲集 专辑，艺人，歌曲

如何 how 客户端 本地/在线 ，触发来源，播放品质。

4确定事实

事实表的设计是为了度量业务过程，所以分析哪些事实与业务过程有关是关注点。

一选择事实时，注意只选择与业务过程有关的事实。二在事实表中应该尽量包含所有与业务过程相关的事实。

切歌播放市场，实际播放时长，播放次数

事实可以通过回答“过程的度量时什么”来确定。事实有可加性，半可加性，非可加性。尽量将不可加性事实分解为可加组件，如比率拆解为分子分母

DWS实体汇总表：

1实体是全量即使当天无操作也有记录，例如全量歌曲

2该表已经是最后结果了，不支持上卷精确去重

3累计相关指标都从该表出

主维度：

设备产品传播名，设备EMUI版本，设备APP版本号 ，电台专辑编码

--主维度不建议超过5个

附属维度：

电台专辑名称，电台专辑艺术家名，cp内

--附属维度不限制

军规：必须是全量实体，主维度不易超过5个，高基维不超过2个

多维分析汇总表

多维可加：不存在需要上卷分析的设备数

多维不可加：需要计算设备数和账号数，做上卷分析操作

# 港中旅风险管理驾驶舱

## 项目整体框架

港中旅风险管理驾驶舱项目包括6个业务单元：中旅银行，产业基金，融资租赁，保险经纪，安信信贷，香港中保。风险管理驾驶舱体系主要划分为经营分析，风险分析，项目分析，以及风险预警，大额风险查看等，利用永洪可视化报表工具展现主要指标变化情况，可以让管理者，决策人快速掌握当前经营情况以及各项风险暴露情况。

数据来源

银行部分的数据来源：

1. 中旅银行风控

贷款基本情况月报 ods\_bank\_risk\_base\_info

客户风险集中度明细 ods\_bank\_risk\_concentration\_detail

客户全部风险集中度汇总 ods\_bank\_risk\_conventration\_total

集团客户 ods\_bank\_risk\_group

贷款质量迁徙情况 ods\_bank\_risk\_loan

2.中旅银行投行部

同业业务基本情况月报 ods\_bank\_investment

3银行科技

ods\_bank\_technology\_balance 银行科技负债表

ods\_bank\_technology\_profit 银行科技利润表

ods\_bank\_technology\_business\_poor 银行科技不良分布

4中旅银行计财

优质流动性资产充足率 ods\_bank\_fin\_abundant

经营情况表 ods\_bank\_fin\_operations

中旅银行的报表部分主要按模块划分，对其经营分析，风险分析，项目分析，信用风险，流动性风险，大额风险暴露等进行了展示。

### 经营分析：

资产 ：现金及存放中央储备金，存放同业，各项贷款及垫款，各项投资，其他资产（流动性资产，非流动性资产）

负债：各项存款，同业存放，向央银借款，其他负债

流动性负债：活期存款，短期类到账的拆放/存放同业款项，中央银行借款，短期内到期的定期存款，发行的票据和债券

非流动性负债：长期借款，应付债券，长期应付款，专项应付款

营收：利息收入，手续费收入，投资收益，其它营业收入

净利润：营业利润+营业外收支净额-资产减值损失-所得税

资产负债率：

### 风险分析：

不良总额（贷款和同业）：

不良资产率：不良信用风险资产/信用风险资产 –数据来源（资产五极分类情况表）

不良信用风险资产=次级类资产，可疑类资产，损失类资产

信用风险资产=贷款，拆放同业及买入返售资产，存放同业，银行账户债券投资，应收利息，其他应收款，不可撤销的承诺及或有负债

前十大客户贷款集中度：最大十家客户贷款余额/资本净额

资本净额= 核心资本+附属资本-扣减项

核心资本：实收资本、资本公积、盈余公积、未分配利润以及少数股权

附属资本包括：重估储备、普通准备金、优先股、长期次级债券和可转换债券

扣减项包括：商誉、对未并表银行机构的资本投资、对未并表非银行金融机构的资本投资、对非自用不动产的投资、对工商企业的资本投资、贷款损失准备尚未提足部分。

前十大集团授信集中度：最大十家集团客户授信净额/资本净额

--数据来源（授信集中情况表）

### 项目分析：

项目新增数及累计数

贷款项目 ：

同业项目 限定：投向标准化证券 授信金额>5000万

贷款逾期： 五级分类中的（次级，可疑，损失）正常，关注 贷款余额 >5000万

同业逾期：

### 信用风险：

分析了两个维度，同业和贷款

贷款总额：

不良总额：次级，可疑，损失

不良率：不良余额，不良余额地区，不良余额行业，不良余额业务品种

### 流动性风险：

存款比例：各项贷款余额/各项存款余额 --来源资产负债统计表

流动比率：流动资产/流动负债

超额备付率：在中国人民银行的超额准备金存款+库存现金/人民币各项存款期末余额

核心负债依存度：核心负债/负债总额

核心负债包括距到期日三个月以上(含)定期存款和发行债券以及活期存款的50%

流动性缺口率：90天内到期资产负债间缺口/同期限内到期的资产余额

流动性匹配率：流动性资产/未来30日内资金净流出

优质流动性资产充足率：

优质资产/可能现金流入-可能现金流出

优质流动性资产持仓

优质流动性资产总额（一级资产，二级资产）

可能现金流入

可能现金流出

不良率：小于等于5%

不良资产率：小于等于4%

存贷款比例：小于75%(不能因为超过75%得出流动性不足的问题，还要考虑其他负债的稳定性情况)

流动比率：大于等于25%

超额备付率：大于等于2%

核心负债依存度：大于等于60%

流动性缺口率：大于等于-10%

流动性匹配率：不小于100%

优质流动性资产充足率：不低于100%(资产规模小于2000亿元的商业银行)

### 大额风险暴露：

非同业单一客户大额风险暴露 d\_bank\_conventration\_total

--客户全部风险集中度汇总

非同业集团客户大额风险暴露

同业单一大额风险暴露

同业集团客户大额风险暴露

指的是商业**银行**对单一客户或一组关联客户超过其一级资本净额2.5%**的风险暴露**。

这个项目以看板为主，我们有专门的ui去设计图片，对于每张图片的大小和坐标都做布局。

在这个项目的报表中我们会有一些js语言去控制图片的可见或不可见,或者对某个数据进行判断，来控制另一个图片的可见和不可见

Var a=文本16.data

if (a>0){图片5.visible true ;图片9.visible=false}

颜色变化：

Var color=new color(ox2f57fb);

Var color1=new color();

Var loc=new locator([],Object,Detail);

投决通过项目.set Backgroud(loc.color);

投决通过项目.set Foreground(loc.color1’)

Param[‘投决项目’]=’投决通过’

上框栏 报告>脚本

右边的框栏 设置>脚本

装载时运行: （管理整个页面）

页面直接呈现的情况；打开报表就有

变化时运行

数据变化的时候里面的脚本会运行和装载时运行很像，页面刚加载的时候也会运行

组件中 装载时运行（管理组件，对组件做特殊的处理，一般可以放在编辑栏中的脚本里实现）（页面加载时运行脚本）（页面加载完成以后如果想要有变化，那么在提交时运行中写脚本）

提交时运行（即点击组件时，组件里面的脚本生效运行）

## 银行风险监管指标标准化手册(银行风控指标框架)

### 资本充足

表格

描述已自动生成

### 信用风险(资产质量)

表格

描述已自动生成

### 信用风险(贷款迁徙)

图片包含 信件

描述已自动生成

### 信用风险(集中度风险)

表格

描述已自动生成

### 盈利性

表格

描述已自动生成

### 流动性风险

表格

描述已自动生成

### 市场风险

表格

低可信度描述已自动生成

### 信用卡风险

表格

中度可信度描述已自动生成

#### **呆账率**：

指标解读：1，反映行用卡资产品质的指标，2年度呆账率所反映的资产品质状况与发卡机构风险经营策略实施之间通常出在滞后性。

指标定义：逾期超过180天的透支金额/应收账款余额

#### **不良率（拖欠率）**：

香港0.3-0.4%，国内的3%

指标解读：信用卡不良率沿袭传统银行贷款业务风险指标建立的，按信用卡贷款监管五级分类的定义即次级类，可疑类和损失类

指标计算：信用卡90天以上逾期账款/应收账款余额

#### 延滞率：

指标解读：延滞资产占应收账款的比率，信用卡资产分为M0至M6+，M2至M5资产衡量延滞率。延滞资产的划分

信用卡延滞率=M2至M5的延滞资产/应收账款余额(不含180天以上的呆账金额)

#### 伪冒损失率

信用卡伪冒损失率是指银行的信用卡伪冒损失金额与交易额之间的比率，其中信用卡伪冒是指使用被窃，伪造，取消的信用卡或未经授权使用他人信用卡以获得财产和服务的行为。伪冒损失金额是指统计期间由于各类伪冒原因所形成的伪冒损失金额。交易金额是指统计期间信用卡所有成功交易的总金额。

信用卡伪冒损失率=统计期间伪冒损失金额/统计期间成功交易总金额。

行用卡业务的特点:信用卡具有金额小，笔数多，持卡人数量庞大，且分散分布的特点。

## 各类风险指标值

不良率：小于等于5%

不良资产率：小于等于4%

存贷款比例：小于75%(不能因为超过75%得出流动性不足的问题，还要考虑其他负债的稳定性情况)

流动比率：大于等于25%

超额备付率：大于等于2%

核心负债依存度：大于等于60%

流动性缺口率：大于等于-10%

流动性匹配率：不小于100%

优质流动性资产充足率：不低于100%

## 项目痛点：

1.数据获取比较困难，PM需要沟通和交流的部门比较多2，模型设计方面做的不够完善。3，报表的复杂度比较高，许多切图都需要动态的效果(需要写少量的js)。4，移动端的适配性差，当时我们把所有的切图都去掉，导致移动端的报表比较单调。5,本身是项目一期的原因，风险监管的指标不够全面，

# 前面的工作内容编写

深圳市儒道数据分析有限公司2016.4-2017.3

主要做的是工厂智能化的一些项目，当时我进去的时候做的事情和我想的不太一样，公司给我安排的数据分析的岗位，我想做的是那种为企业带来发展，带来业务增长的那种数据分析，但实际上，做的是一个对接交付的工作。

深圳市聚荣鑫科技有限公司2017.3至今2019.3 2018年10月开始休产假，在这个公司干了约1年半左右，做报表开发的部分工作，finereport，公司的业务主要包括两个部分，弱电工程业务和租赁业务，我负责租赁业务的部分数据处理。

# 证券知识总结

证券经纪

证券公司通过旗下的证券营业部，接受客户的委托，按照要求买卖证券，证券公司不垫付资金，不赚差价，只收取一定比例的佣金。

交易对象上市交易的的股票和债券。席位代表了会员（在深圳证券交易所的权益），会员需要拥有席位方可在证券交易所进行交易。

证券投资咨询

证券公司向客户提供投资证券及证券相关产品的投资建议服务。辅助客户作出投资决策，并直接或间接获取经纪利益的相关活动

证券承销与保荐

保荐业务资格

注册资本不低于人民币1亿元，净资本不低于人民币5000万元。

依法承销的，应当聘请具有保荐业务资格的证券公司履行保荐职责

（1）首次公开发行股票

（2）上市公司发行新股，可转换公司债券

（3）公开发行存托凭证

证券承销

是指证券公司代理证券发行人发行证券的行为。

代销 证券公司代发行人发售证券，在承销期结束时，将未售出的证券全部退还给发行人的承销方式。

包销 全额包销 余额报销 是指证券公司将发行人的证券按照协议全部购入或者在承销期结束时将售后剩余证券全部自行购入的承销方式。

证券承销是属于一级市场

## 融资融劵业务账户体系

1证券公司信用账户（以自己的名义证券登记结算机构分别开立账户）

融劵专用证券账户

用于记录证券公司持有的拟向客户融出的证券和客户归还的证券。该账户不能用于证券买卖

客户信用交易担保证券账户

用于记录客户委托证券公司持有、担保证券公司因向客户融资融券所产生债权的证券

信用交易证券交收账户

用于客户融资融劵交易的证券结算

信用交易资金交收账户

用于客户融资融券交易的资金结算

以自己的名义在商业银行分别开立账户

融资专用资金账户

用于存放证券公司拟向客户融出的资金及客户归还的资金

客户信用交易担保资金账户

用于存放客户交存的担保证券公司因向客户融资融券所生债权的资金

2客户信用账户

信用资金台账

信用证券账户

信用资金账户

融资业务（买多）：以资金或证券作为担保，向证券公司借入资金，用于买入证券，并在约定的期限内偿还本金和利息。

融券（卖空）： 以资金或证券作为担保，向证券公司借入证券卖出，在约定的期限内，买入相同数量和品种的证券归还证券公司并支付相应的融劵费用

融资融券业务与其他业务的区别：1具有财务杠杆效应，获取的收益可能更多，但是损失也可能更多。2融资融劵的账户内资产具有担保属性，该资产的划分受限制。

融资融券业务的投资期限为0-1 年，投资品种为融资融券业务，风险等

级为R4，对应投资者风险承受能力为C4( 增长型) 及以上。风险测评结果为

C5( 进取型)、C4( 增长型)、C3( 平衡型) 的客户可申请开展融资融券业务，

并签署确认文件。

1、什么是担保品？

进行融资融券业务，您信用账户的资金及证券作为担保品，为信用交易的负

债进行担保。“担保品”与业务合同中“担保物”是同一概念。

2、担保品的范围是什么，如何查询？

担保品范围包括：

1. 自有资金；②自有证券（可充抵保证金的）；③融资买入的全部证券；④

融券卖出所得全部价款；⑤上述资金、证券所生孳息；⑥您信用账户内所有其他

5、什么是保证金？

保证金是您信用账户内所有担保品，经过折算率计算过后形成的金额，是对

信用交易的担保能力。

（1）以现金作为保证金的，全额计入保证金金额；

（2）以证券充抵保证金的，应当以各证券对应折算率对其市值或净值进行

折算。

图示

中度可信度描述已自动生成

8、什么是保证金比例？

指您提交的保证金与融资、融券交易金额的比例。即您进行每笔融资融券交

易时，证券公司会收取一定比例的保证金，保证金比例也随之变动。

具体可分为融资保证金比例和融券保证金比例。

（1）融资保证金比例

指融资买入时交付的保证金与融资交易金额的比例。

计算公式：融资保证金比例＝保证金/（融资买入证券数量× 买入价格）

×100%。

目前我公司融资保证金比例不得低于1：1，即最高一倍杠杆。

（2）融券保证金比例

指融券卖出时交付的保证金与融券交易金额的比例。

计算公式：融券保证金比例＝保证金/（融券卖出证券数量× 卖出价格）

×100%。

目前我公司融券保证金比例不得低于1：2，即最高两倍杠杆。

5、担保品划转是否实时生效？

信用账户可由现金或证券作为担保品，现金转账实时生效，证券划转需下一

个交易日才能在划入的账户中显示。

6、普通账户当日买入的证券，当日可以划入信用账户吗？

您在普通账户当日买入的证券，当日不能划转至信用账户，次一交易日起方

可划转入信用账户。

## 指标体系：

融资融劵总资产：资金余额+证券市值+权益调整资产总值

融资融劵总资产\_BI口径：资金余额+证券市值+权益调整资产总值-融券负债金额-未还融资所有费用-未还融券所有费

融资融券净资产：资金余额+证券市值+权益调整资产总值-融资负债本金-融劵负债本金-未还融资所有费-未还融券所有费

融资融券净资产\_BI口径：资金余额+证券市值-融资负债本金-融劵负债本金

指标分类

### 1客户汇总类

1.1资产类

1.1.1客户总资产

普通账户总资产；

融资融券净资产；

股票期权总资产；

1.1.2客户总净转出资产

普通账户净转出资产；

融资融券账户净转出资产；

股票期权账户净转出资产；

1.1.3客户总负债

普通账户股票质押负债；

融资融券总负债；

1.2交易类

1.2.1客户总成交金额

普通账户总成交金额；

融资融券账户总成交金额；

股票期权账户总成交金额

1.2.2客户总股基成交金额

普通账户股基成交金额；

信用账户股基成交金额；

1.2.3客户总收益

普通账户总收益

融资融券账户总收益；

股票期权账户总收益；

1.3贡献类

1.3.1客户总佣金

普通账户总佣金；

融资融券账户总佣金；

股票期权账户总佣金

1.3.2客户总利息贡献

普通账户利息贡献；

融资融券利息贡献；

股票期权利息贡献；

### 2普通账户类

2.1资产类

2.1.1普通账户总资产

[普通账户资金总额](#_普通交易资金余额)；

[场外理财总市值](#_场外理财总市值)；

[场内交易总市值](#_场内交易总市值)；

新三板股票市值；

2.1.2普通账户资金总额

普通账户人民币资金余额；

普通账户美元资金余额；

普通账户港币资金余额

2.1.3普通账户场外理财总市值

[场外公募基金市值；](#_普通账户场外公募基金市值)

[资管理财市值；](#_普通账户资管理财市值)

私募基金产品市值；

收益凭证产品市值；

公募专户市值；

信托产品市值；

银行理财产品市值；

保险产品市值；

2.1.4普通账户场内交易总市值

普通账户A股市值；

普通账户B股市值；

普通账户债券市值；

普通账户场内基金市值；

普通账户[天添利](#_天添利资产)市值；

普通账户[港股通市值](#_港股通市值)；

普通账户质押式回购市值；

普通账户股票质押负债；

普通账户场内其他市值；

2.1.5普通账户股基市值

普通账户A股市值 + B股市值\*汇率 + 资管理财市值 + 场内基金市值 + 场外公募基金市值 + 港股通市值

2.2交易类

表格

描述已自动生成

手机屏幕截图

描述已自动生成

2.3贡献类

表格

描述已自动生成

文本, 信件

描述已自动生成

2.4费用类

手机屏幕截图

中度可信度描述已自动生成

### 3信用账户类

#### 3.1资产类

3.1.1融资融券净资产

融资融券总资产 - 融资融券总负债

3.1.2融资融券总资产

资金余额 + 证券市值 + 权益调整资产总值

3.1.3融资融券总负债

融资负债本金 + 融券负债市值+ 未还融资利息 + 未还融资罚息+未还融券利息 + 未还融券罚息 + 管理费 + 未还融券补偿权益金额

3.1.4融资融券总市值

股票收盘价 \* 持仓数量

#### 3.2交易类

3.2.1融资融券保证金净转入

保证金转入-保证金转出

3.2.2融资融券担保品转入市值

担保品转入市值

3.2.3担保品转出市值

3.2.4融资融券成交金额

担保品买入成交金额 + 担保品卖出成交金额 + 融资买入成交金额 + 融券卖出成交金额 + 融资平仓成交金额 + 融券平仓成交金额 + 强制卖出成交金额 + 买券还券成交金额 + 卖券还款成交金额

3.2.5担保品买入成交金额

成交价格 \* 担保品买入数量

3.2.6担保品卖出成交金额

成交价格 \* 担保品卖出数量

3.2.7担保品净转入

担保品转入市值 - 担保品转出市值

3.2.8融资买入成交金额

成交价格 \* 融资买入数量

3.2.9融券卖出成交金额

成交价格 \* 融券卖出数量

3.2.10融资平仓成交金额

成交价格 \* 融资平仓数量

3.2.11融券平仓成交金额

成交价格 \* 融券平仓数量

3.2.12强制卖出成交金额

成交价格 \* 强制卖出数量

3.2.13买券还券成交金额

成交价格 \* 买券还券数量

3.2.14融资融券盈亏

上一交易日盈亏：（股份余额 + 买入在途股份 – 卖出在途股份）\* 上一交易日收盘价 – 盈亏成本。

实时盈亏：（股份余额 + 股份实时买卖差额 + 买入在途股份 – 卖出在途股份）\* 最新价 – 盈亏成本。

3.2.15融资融券成本

融资成本 + 融券成本 + 融资应付利息 + 融券应付利息

3.2.16融资融券收益率

融资融券盈亏 / 融资融券成本

3.2.17融资融券A股成交金额

成交价格 \* A股成交数量

3.2.18融资融券债券成交金额

存在公允价格且净价交易：（成交价+ 应计利息）\* 成交数量

存在公允价格且全价交易：成交价 \* 成交数量

无公允价格：（面值 + 应计利息）\*成交数量

3.2.19融资融券基金成交金额

上一交易日净值 \* 成交数量

3.2.20融资融券维持担保品比例

总资产/总负债

3.2.21融资融券授信额度

3.2.22融资融券已使用授信额度

融资负债本金 + 融券负债市值

3.2.23融资融券融资额度

融资融券授信额度 – 融资融券已用授信额度

3.2.24融资融券融券额度

融资融券授信额度 – 融资融券已用授信额度

#### 3.3贡献类

3.3.1融资融券利息贡献

融资利息;

融券利息;

保证金利息;

罚息

融资利息 + 融券利息 + 保证金利息 + 罚息

3.3.2融资融券融资利息

剩余融资负债本金\* 融资利率 \* 天数 / 360

3.3.3融资融券融券费用

融券负债数量 \* 收盘价 + 应付权益金额）\* 融券年利率 \* 天数 /360

3.3.4融资融券罚息

未还融资罚息 + 未还融券罚息

3.3.5融资融券保证金利息

融资融券资金余额 \* 本日计算利差基数\*利率/360

3.3.6两融买入佣金

普通买入佣金 + 融资买入佣金 + 买券还券佣金

3.3.7两融卖出佣金

普通卖出佣金 + 融券卖出佣金 + 卖券还款佣金

3.3.8两融买入净佣金

两融买入净佣金 – 一级费用

3.3.9两融卖出净佣金

#### 3.4费用类

3.4.1融资融券印花税

成交金额 \* 印花税率

3.4.2融资融券经手费

成交金额 \* 经手费率

3.4.3融资融券过户费

成交金额 \* 过户费率

3.4.4融资融券证管费

成交金额 \* 证管费率

3.4.5融资融券交易规费

成交金额 \* 交易规费率

### 4股票期权类

4.1资产类

文本, 信件

描述已自动生成

4.2交易类

表格

描述已自动生成

4.3和4.4贡献费用类

文本

描述已自动生成

### 5项目类

表格

描述已自动生成

## 招商证券的主题领域划分(有具体的主题表名单)

招商证券的主题划分

当事人（party）

产品（product）

协议（aggrement）

事件(event)

资产(asset)

财务(financial)

机构()

地域()

营销()

渠道()

表格

描述已自动生成

图示

描述已自动生成图示

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

手机屏幕截图

描述已自动生成

## 证券公司的业务类型：

证券经纪业务

接受客户委托，按照客户要求，代理客户买卖证券的业务。不垫付资金，不赚取差价，不承担客户的价格风险，只收取一定比列的佣金作为业务收入。

证券投资咨询

证券投资顾问：接受客户委托，按照约定向客户提供涉及证券及证券相关产品的投资建议服务，辅助客户作出投资决策，并直接或间接获取经济利益的经营活动。

证券研究报告：对证券及证券相关产品的价值，市场走势或者相关影响因素进行分析，形成证券估值、投资评级等投资分析意见，制作报告，并向客户发布的行为。

与证券交易、证券投资活动有关的财务顾问

证券承销与保荐

证券自营

证券自营是指，用自有资金和依法筹集的资金，用自己的名义开设证券账户依法买卖公开发行或者中国证监会认可的其他有价证券。

自营权益类证券及其衍生品合计不超过净资本的100%  
自营非权益类证券及其衍生品不超过净资本的500%

持有一种权益类的证券成本不超过净资本的30%

持有一种权益类证券的市值与其总市值的比列不超过5%

证券资产管理

募集方式：公募产品，私募产品

投资性质：固定收益（用于投资存款，债券，等债权类资产的比例不低于80%），权益类用于投资股票，非上市企业股权等权益类资产比列不低于80%。

投资者人数：私募资产管理产品按照投资者人数不同，可分为单一资产管理产品，集合资产管理。

单一资产管理：为单一投资者设立单一资产管理计划。

集合资产管理计划：投资者人数不少于2人，不超过200人。

----项目痛点：指标多，并且梳理起来复杂，数据底层的业务系统也在进行升级改造。

## 构建领域模型的流程：

1数据调研：分解业务要素（数据怎么产生，数据的应用）

2构建总线矩阵：（业务过程，分析维度）

3设计指标体系：原子指标，派生指标，指标修饰词。

4逻辑模型设计：一致性逻辑维度及维度属性，一致性度量及指标口径设计

5模型开发上线

6持续监控和维护

## 怎么保证数据一致性：

--模型分两层，融合模型和领域模型，融合模型层，公共指标会在融合模型中进行计算

1具有相同含义的字段在不同表中的命名必须相同，必须使用规范定义中的名称

2同一个指标的计算口径必须一致

3保持同一个表中数据的粒度一致

4同一个指标的计算尽量来源同一个表

5上线前的数据核对

# KYLIN架构：

Kylin获取的表是星型模型结构的。目前建模时，只支持一张事实表，多张维表。

如果业务需求比较复杂，就要考虑在Hive中进行进一步处理。（比如生成一张大的宽表，或者采用View代替）

HBase：Kylin中用来存储OLAP分析的Cube数据的地方，实现多维数据集的交互式查询

Cuboid：Kylin中将维度任意组合成为一个Cuboid

Cube：Kylin中将所有维度组合成为一个Cube，即包含所有的Cuboid

假设我们有4个dimension(age,provice,product\_line,device\_pro\_name),这个cube中的每个节点（cuboid）都是这4个维度的组合（(age,provice),( product\_line,device\_pro\_name)），度量的聚合结果保存在每个cuboid上。

文本

描述已自动生成

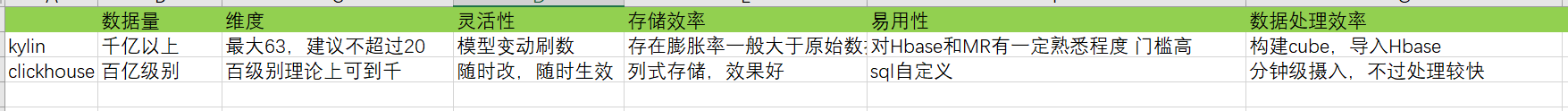
( 5)ODBC驱动程序:为了支持第三方工具与应用程序——例如Tableau,finebi——我们构建起了一套ODBC驱动程序并对其进行了开源。我们的目标是让用户能够更为顺畅地采用这套Kylin平台

# Kylin和clickhouse的区别

Clichouse

Mapp在线计算，向量化执行，存储明细数据，本地磁盘存储，支持复杂数据类型（map,arry数组），现场计算能力强，查询请求比较灵活，借助三方spark connector ,适用于多列且where条件随意组合的用用标签筛选，并发量不大的复杂即席查询。。。灵活查询，现场计算支持灵活查询，容易增加列，不占用FI资源,不依赖其他组件，支持几百亿的数据秒级响应。

Kylin,cube预计算，mapp现场计算，存储聚合数据，HDFS分布式存储入库HBASE，采用预计算技术生成cube,适合有固定模式的聚合查询，sql中的join ,group by where条件模式固定，数据量越大，优势越明显，和hadoop无缝接入。。。cube难变更，需要修改只能重建追溯历史数据，占用资源多，kylin的构建存储约300台机器，架构复杂维护成本高。



# HIVE

Hive是什么：是基于hadoop的数据仓工具，可以将结构化的数据文件映射为一张表，并且提供类sql查询功能，本质是将hql转换成mapreduce程序，数据存储在hdfs，分析底层实现是mapreduce,执行程序运行在yarn上。

### Hive和数据库的区别

相同点：相似的语言查询

不同点：

1数据存储位置，hive建立在hadoop之上，所有hive的数据块都是存储在hdfs中，数据库则可以将数据保存在本地设备或本地文件系统中

2数据更新：hive不支持对数据的改写，所有的数据在加载的时候中确定好，而数据库中的数据通常是经常进行修改，可以使用update…set

3索引：hive在加载数据过程中不会对数据处理，不会对数据中的某些key建立索引。Hive要访问数据中满足条件的特定值时，需要暴力扫描整个数据，因此访问延迟较高。数据库中，通常会针对一个或者n个列建立索引，因此对于少量的特定条件的数据访问，数据库可以有很高的效率，较低的延迟

4执行：hive中大多数查血的执行时通过hadoop提供的mapreduce来实现的，而数据库通常有自己的执行引擎

5执行延迟：hive在查询数据的时候，由于没有索引，需要扫描整个表，因此延迟较高，另一个导致hive执行延迟高的因素是mapreduce框架，mapreduce本身具有较高的延迟。相对的数据库的执行延迟较低，当然这个是有条件的，当数据规模大到超过数据库处理能力的时候，hive的并行计算能力体现优势。

6可扩展性：hive是建立在hadoop上，hive的可扩展性是和hadoop的可扩展性一致。数据库的扩展性非常有限，目前最先进的并行数据库在oracle在理论上扩展能力有100台。

7数据规模：由于hive建立在集群上并可以利用mapreduce进行并计算，因此可以支持很大规模的数据，对应的，数据库可以支持的数据规模小。

### Hive架构

1用户接口：CLI，Client 和 WUI。其中最常用的是CLI，Cli启动的时候，会同时启动一个Hive副本。Client是Hive的客户端，用户连接至Hive Server。在启动 Client模式的时候，需要指出Hive Server所在节点，并且在该节点启动Hive Server。 WUI是通过浏览器访问Hive。

2元数据 metastore，Hive中的元数据包括表的名字，表的列和分区及其属性，表的属性（是否为外部表等），表的数据所在目录等

--3hadoop

--使用hdfs进行存储，用mapreduce进行计算

4驱动器：driver(complier/optimizer/executor)

(1)解释器（sqlparer）:将sql字符转换成抽象语法树AST

(2)编译器（physical plan）将AST编译生成逻辑执行计划

(3)优化器（query optimizer）:对逻辑执行计划进行优化

(4)执行器（execution）把逻辑执行计划转化成可以运行的物理计划

只有map没有reduce的过程：查询，插入表等不需要聚合、分组的操作。

### Hivesql的执行计划：

1对 hivesql进行语法解析,将hivesql解析成抽象语法树ATS

2对ATS进行语义分析，转化成hive内部查询结构（查询块），然后查询块转换成对应操作

对应操作：

Tablescanoperator 扫描指定表数据

Reducesinkoperateor 将数据发给reduce任务操作

Joinoperator 两张表进行join 操作

Select operator 查询操作

Filesinkoperator 写文件操作

3 complier合并所有operator按顺序组成一棵操作数，即逻辑执行计划

4逻辑优化器进行Opertortree变换，合并不必要的Reducesinkoperateor，减少shuffle数据量

5优化逻辑计划，被转化成物理执行计划，也就是真正的map/reduce task

6hive在物理执行计划层面的优化(map join)

Explain+hivesql,查看执行计划

# HDFS

Hadoop 由 HDFS 、 MapReduce 、 spark 、 Hive 和 yarn等成员组成

## HDFS优缺点：

优点：

（1）高容错性

数据自动保存多个副本，通过增加副本的形式，提高容错性。

数据丢失后，自动恢复

（2）适合批处理

移动计算而不是移动数据

数据位置暴露给计算框架

（3）适合大数据处理

GB,TB甚至PB级别数据

能够处理百万规模以上的文件数量

（4）可构建在廉价机器上

通过都副本提高可靠性

缺点：

1HDFS不支持对文件的随机读写，可以APPEND(追加),但是不能修改，原因：文件在HDFS上存储时，以BLOCK为基本单位。一：没有提供对文件的在线寻址（打开）功能，二：文件以块形式存储，修改了一个块中的内容，就会影响当前块之后的所有块，效率低

2不支持并发写入，同一个文件在同一时刻只能有一个客户端写入，不允许多个线程。例如同一个任务onework和TCC不能同时运行。

3不适合低延时数据访问。比如毫秒级的存储数据，是做不到的。

4HDFS不适合存储小文件，会降低NN的服务能力。

举例： 当前运行NN的机器，有64G内存，除去系统开销，分配给NN50G内存！

文件a (1k), 存储到HDFS上，需要将a文件的元数据保存到NN，加载到内存

包括：文件名 创建时间 所属主 所属组 权限 修改时间+ 块的映射(1块)

NN占用内存：150B

最多存储50G/150B个文件a

存储占用磁盘空间：50G/150B \* 1k

文件b (128M), 存储到HDFS上，需要将b文件的元数据保存到NN，加载到内存

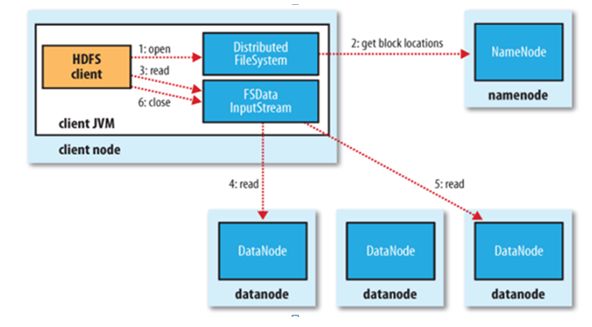
包括：文件名 创建时间 所属主 所属组 权限 修改时间+块的映射(1块)

NN占用内存：150B

最多存储50G/150B个文件b

存储占用磁盘空间：50G/150B \* 128M

## HDFS文件读写：



文本

描述已自动生成

 1.客户端发出读数据请求，Open File指定读取的文件路径，去找namenode要元数据信息。

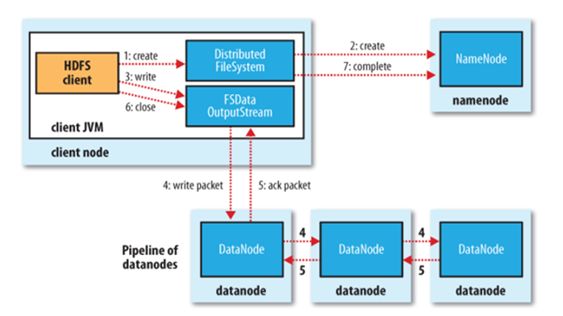
2.namenode将文件的元数据信息返回给客户端。

3.客户端根据返回的元数据信息，去对应的datanode去读块数据。

 假如一个文件特别大，比如1TB，会分成好多块，此时，namenode并是不一次性把所有的元数据信息返回给客户端。

4. 客户端读完此部分后，再去想namenode要下一部分的元数据信息，再接着读。

5.读完之后，通知namenode关闭流



图形用户界面, 文本, 应用程序

描述已自动生成

3.数据块的发送，先发给第一台datanode，然后再有第一台datanode发往第二台datanode，……。实际这里，用到了pipeLine 数据流管道的思想。

## HDFS的Namenode,Datanode

Hdfs分布式存储系统，服务器可以看成载体：

Rack :机架，存储节点放在不同的机架上

Block：数据块数据切分成特定大小的数据块，分发到不同存储节点，默认128MB dfs.blocksize

Replication:数据块在不同的节点之间通过复制方式来拷贝

Namenode (管理节点)名称空间管理，元数据管理，负责文件数据块的映射。

元数据：database，表名，表的列及类型，表类型，存储空间，分区，表数据所在目录

Datanode：向管理节点汇报数据块信息，存储节点之间通过复制操作来实现数据均衡和备份，与客户端交互。

HDFS读写定义不同存储位置的距离，NN随机选择可用blocks时根据距离排序，距离最近的节点位于最前列，客户端读取数据时，选择第一个节点进行读取。

副本放置策略：

NN选择数据块的放置节点，按照机架配置来选择节点，如果是3副本放置策略，优先放入离写入客户端最近的DN节点，然后是该节点同机架上的一个节点，最后写入不在同一机架的节点。

## 切片的理论

切片的形成：

不断迭代节点列表，逐个节点（以数据快为单位）形成切片：

A如果maxsplitsize==0，则整个节点上的block数据形成一个切片

B如果maxsplitsize!==0，遍历累加每个节点上的数据块，如果累加数据块大小>=maxsplitsize,则将这些数据块形成一个切片。继续该过程，直到剩余数据块累加大小<=maxsplitsize.则进行下一步

C如果剩余数据块累加大小>=minsplitsizenode,则将这些剩余数据块形成一个切片。继续该过程，直到剩余数据块累加大小<minsplitsizenode。然后进行下一步，并这些数据块留待后续处理。

不断迭代机架列表，逐个机架（以数据块为单位）形成切片（Rack split）:

A遍历并累加这个机架上所有节点的数据块（这些数据块即上一步遗留下来的数据块），如果累加数据块大小>=maxsplitsize,则将这些数据块形成一个切片。继续该过程，直到剩余数据块累加大小<maxsplitsize。则进行下一步

B如果剩余数据块累加大小>=minsplitsizerack，则将这些剩余数据块形成一个切片。如果剩余数据块累加大小<minsplitsizerack，则这些数据块留待后续处理。

# MAPREDUCE

Shuffle的过程

Map ： 拆分，数据切片

Partition:按照关联字段或者行数进行hash算法转成partiton id即key值， 根据reduce进行分区，分区按key值不同，hash方法保证key 值的唯一性。（分区决定当前key值交给哪个reduce处理，相同的key值必须由同一个reduce处理）

形成缓冲区：对每条（key,value）打标签

Sort： 不同切片，不同的分区排序（将相同分区的数据进行分区内排序）

Spill ：当缓冲区达到阈值80%开始溢写，将分区排序后的溢写数据写到磁盘变成file，产生多个小文件

Combine： 结果送到reduce前进行计算，减少中间结果，降低网路开销（将spill产生的小文件进行合并，合并过程中也要对分区的数据进行分区内排序）

Copy： 把输出结果传给reduce，reduce向不同节点下载自己分区

Merge:

数据从不同的Mapper下载，所以要进行合并

Reduce根据每个内容做最后一次统计。

# Spark的性能优化以及架构

Spark性能调优就是分配资源，即增加和分配更多资源对速度提升显而易见

Spark.driver.memory 4g 内存

Spark.driver.memoryoverhead 2g （除去java推以外暂用空间大小）

Spark.drive.cores 2 cpu核数

Spark executor.memory 执行内存

Spark executor.cores 执行核数

Spark.dynamicallocaton.maxexecutors 500 请求数

Spark.default.parallelism 500 合理的并行度

Spark.sql.adaptive.enable true

Spark.sql.adaptive.skewjoin.enable true 同时配置起效

Spark.shuffle.statistics.verbose true

## Spark架构

RBO（谓词下推）：把过滤条件推到最底层，避免无用的计算，少扫描很多的数据RBO(列裁剪)，RBO（常量折叠）

Spark2.0 rule +cost基于代价的优化（CBO）broadcast拿到统计不准确 导致不会做广播

Spark3.0 rule+cost+runtime 运行时的统计

Driver 进程

Master 进程

Worker 进程

Executor 进程

task 线程

第一步：Driver进程启动之后，会做一些初始化的操作；在这个过程中，就会发送请求到master上，进行spark应用程序的注册，说白了，就是让master知道，有一个新的spark应用程序要运行

第二步：master，在接收到了spark应用程序的注册申请之后，或发送请求给worker，进行资源的调度和分配，OK，说白了资源分配就是executor的分配

第三步：worker接收到master的请求之后，会为spark应用启动executor

第四步：executor启动之后，会向Driver进行反注册，这样，Driver就知道哪些executor是为它进行服务的了

第五步：Driver注册了一些executor之后，就可以开始正式执行我们的spark应用程序了，首先第一个步就是创建初始RDD，读取数据源

第六步：RDD的分区数据放到内存中

第七步：Driver会根据我们对RDD定义的操作，提交一大堆task去executor上

第八步：task会对RDD的partition数据执行指定的算子操作，形成新的RDD的partition

## Application job stage task 的关系

1. Application：初始化一个SparkContext即生成一个Application；

2）Job：一个Action算子就会生成一个Job

3）Stage：Stage等于宽依赖的个数加1；

4）Task：一个Stage阶段中，最后一个RDD的分区个数就是Task的个数

每个stage由多个tasks来组成，这些tasks就表示每个并行计算，并且会在多个执行器上执行。

## RDD特性

RDD可以理解为（resilient distribute dataset）(弹性 分布式 数据集)

一个RDD对应多个分区，每个分区对应HIVE表中的一部分数据或HDFS中的数据，（每个分区对应几块数据）。

1RDD有很多Partiton构成：（在RDD计算中有多少个partition就有多少个task）

2对RDD做计算，相当于对RDD的每个SPLIT或partition做计算。Partition=split

3rdd之间存在一系列的依赖关系

4如果RDD里面存的数据式key-value形式，则可以传递一个自定义的partitioner进行重新分区，比如可以按key的hash值分区

5对于每个计算分片有用最佳计算位置

RDD基本操作 Transformation

惰性执行 输入和输出均是RDD只有在执行Action的时候才真正执行。

Narrow transformation

父RDD的每个分区只被一个子RDD分区使用一对一的依赖，即rangedependency(union)支持不同算子在一个executor上以pipeline形式并行执行。

分区丢失情况下，只需要计算一个parent rdd partition 即可

Wide transformation

父RDD的每个分区都有可能被多个子RDD分区使用，需要等待所有parent rdd partition 都完成后才能开始计算涉及shuffle，分区丢失的情况下，需要重新计算依赖的失落与partition

RDD常用transformation

Map filter flatmap distinct groupbykey reducebykey aggregatebykey

Coalesce (descrease the number of partitons in the rdd to numpartition .useful for running operations more efficiently after filtering down a large dataset)

RDD操作-Action

返回值不是RDD结果返回给driver或者写入存储系统，真正触发依赖的transformation的执行，RDD中的数据在transformation后被拉到各个计算节点。

RDD常用Action

Reduce collect count first take takesample takeordered saveastextfile saveassequencefile saveasobjectfile countbykey foreach

## Spark执行过程

Sql通过解析生成AST（抽象语法树）（unresolved logical plan ），通过分析器分析（analyer ）元数据生成logical plan （逻辑计划），通过优化器生成最佳执行计划（physical plan）,由执行引擎执行物理计划定义的计算逻辑，生成RDD。

Analyzer使用预先定义好的Rule及sessioncatalog 等信息对unresolved logical plan 绑定数据源以及数据类型等。

Optimizer包括RBO/CBO,其中spark catalyst属于RBO Planner 在生成physical plan 时候，还会基于cost代价做进一步优化等。

## Spark申请资源过程：

1因为是Client模式，所以当我们 Spark-Submit 提交Spark任务的时候，会直接走到我们的main方法，进行Spark Context 的初始化。

2Spark Context 初始化的时候会生成两个比较重要的对象DAGSchedule 和 TaskSchedule,

TaskSchedule 会进行任务资源的申请，因为我们这里是用 Yarn 作为资源调度器，

所以 TaskSchedule 会向 ResourceManager(RM) 进行资源申请。

3接下来就是 Yarn 的资源调度了

4Yarn 首先会启动一个 ApplicationMaster(AM) 来管理本次申请，所以 Yarn 的第一步是选一台空闲的 NodeManager 启动 AM

5AM 启动后，会根据我们提交任务时申请的资源向 RM 进行资源申请用来启动 Container，

当然这里用来处理的是Spark任务，实际上启动的是 Excutor.

6当我们的Excutor 启动之后，他们会向Driver 端的 TaskSchedule 进行注册。

7这个时候我们的 Spark Context 的初始化基本完成。接下来就是根据我们的代码，

生产Task 进行任务调度了

区别就在于Driver程序所在的位置。

因为是Cluster模式，所以当我们 Spark-Submit 提交Spark任务的时候，

首先是直接去向 RM 申请启动Driver的资源

Yarn 还是会首先选一台空闲的 NodeManager 来启动 AM管理本次申请，

不过在AM启动的时候，AM也会对Spark Context 进行初始化，

所以在 Cluster 模式下，AM 还扮演着另外一个角色，那就是 Driver。

Driver启动之后，那就是开始申请 Excutor的资源了，所以AM 就开始向RM申请资源了，RM 进行资源申请用来启动 Container，实际上启动的是 Excutor。

## Spark和mapreduace区别

[Spark](https://so.csdn.net/so/search?q=Spark&spm=1001.2101.3001.7020" \t "_blank)和MapReduce都是可以处理海量数据，但是在处理方式和处理速度上存在着差异，**1.spark处理数据是基于内存的，而MapReduce是基于磁盘处理数据的。**

[MapReduce](https://so.csdn.net/so/search?q=MapReduce&spm=1001.2101.3001.7020" \t "_blank)是将中间结果保存到磁盘中，减少了内存占用，牺牲了计算性能。

 Spark是将计算的中间结果保存到内存中，可以反复利用，提高了处理数据的性能。

**2.Spark在处理数据时构建了DAG有向无环图，减少了shuffle和数据落地磁盘的次数**

Spark 计算比 MapReduce 快的根本原因在于 DAG 计算模型。一般而言，DAG 相比MapReduce 在大多数情况下可以减少 shuffle 次数。Spark 的 DAGScheduler 相当于一个改进版的 MapReduce，如果计算不涉及与其他节点进行数据交换，Spark 可以在内存中一次性完成这些操作，也就是中间结果无须落盘，减少了磁盘 IO 的操作。但是，如果计算过程中涉及数据交换，Spark 也是会把 shuffle 的数据写磁盘的！Spark 支持将需要反复用到的数据给 Cache 到内存中，减少数据加载耗时。

3.Spark是粗粒度资源申请，而MapReduce是细粒度资源申请

粗粒度申请资源指的是在提交资源时，spark会提前向资源管理器（yarn，mess）将资源申请完毕，如果申请不到资源就等待，如果申请到就运行task任务，而不需要task再去申请资源。

MapReduce是细粒度申请资源，提交任务，task自己申请资源自己运行程序，自己释放资源，虽然资源能够充分利用，但是这样任务运行的很慢。

# YARN

Yarn是hadoop的资源管理器，可为上层应用提供统一的资源管理和调度，它的引入可是方便监控程序的运行状态以及日志

Resource manager:整个集权只有一个，负责集权资源的统一管理和调度，处理客户端的请求，启动/监控applicationmaster,监控nodemanager汇总上报资源。

Node manager :整个集群有多个（每个从属节点一个）

单个节点的资源管理和任务管理

监控资源使用情况（cpu,memory,disk,network）并向resource manage汇报

Container

计算资源抽象为container,任务运行资源（节点，内存，cpu）

Yarn资源调度过程：

1Yarn申请资源，客户端提交作业到resourcemanager

2resourcemanager实时要求nodemanager上报节点使用情况

3如果节点处于空闲状态resourcemanager会在节点上创一个applicationmaster

4applicationmaster向resourcemanager申请资源，resourcemanager向nodemanager分配资源

5nodemanager将资源封装成container交给application去运行

# 大数据性能优化

优化的整体思路：

四个方面考虑：模型涉及（遵守模型设计的规则）

代码优化

参数优化

业务侧优化

模型设计：例如：做累计的时候是增量累计还是全量累计，一般的我们会

查看执行日志：times token 就能知道单个job的执行时间，重点对消耗时间比较长的进行优化。

## sql优化写法

1.表join前尽量先过滤数据，因为提前过滤会比在on和where时过滤减少IO

2.消灭子查询中的group by 例如 ccp1.ccp2中数据集并列取去重后的did，

(Select did from ccp1 group by did union all select did from ccp2 group by did)group by did

可以写成 (Select did from ccp1 union all select did from ccp2)group by did

3.第一xx的优化

SELECT ROW\_NUMBER()OVER(PARTITION BY IMEI ORDER BY device\_active\_time)

优化后：SPLIT(MIN(CONCAT(device\_active\_time,’#’,device\_name)),’#’)[1]

4.job并行的写法

Select \* from logs where log\_date between 20210801 and 20210803

Select \* from logs where log\_date=20210801

Union all select \* from logs where log\_date=20210802

Union all select \* from logs where log\_date=20210803

5.count(distinct)优化

Select count(distict imei ) from a

Select count(1) from (select imei from a group by imei)

6.大表关联，尽量一次得出所有可能的数据，避免大表多次重复关联

A left join b union all c left join b

优化写法 (a union all c )left join b

7.WITH AS 语句的使用（重点展开讲一讲）

A left join b on a.hw\_did=b.fault\_did

Left join b on a.hw\_did=n.hw\_new\_did

With as 语句的特点：对同样子查询可以做一个子表，减少一次查询

在长语句中避免出现多层嵌套select,便于阅读和维护

8.查询A表的记录在B表不存在的情况

左连接方案：写成not exists

Select \* from a left join b on a.id=b.id

Where b.id is null

Not exists

Select \* from a where not exists (select 1 from b where b.id=a.id)

In 和exist区别:

当外层查询记录大于内层查询时用in/not in ，当内层查询记录数远大于外层查询记录数时用exists/not exists,两者都差不多时用exists/not exists

Not in 和 not exists 区别：

通常情况下可以划等号，但是假如内层查询的结果中有null时不能直接用not in ,not in相当于 where t.name!=’jack’ and t.name!=null，而null是不能进行‘=’操作的，所有的结果都是false,所以假如要用not in要在内层查询中加入非空过滤（and name is not null）

1宏观优化

Set mapreduce.map.memory.mb=2048

Set mapreduce.reduce.memory.mb=2408

Set mapreduce.reduce.cpu.vcores=1

Set mapreduce.map.cpu.vcores=1

2参数调优

## 大小表关联

Set hive.auto.convert.join=true

Set hive.mapjion.smalltable=25MB 25600KB

或者在select 后面加上 /\*+mapjob(table)\*/

## 数据倾斜：

Reduce节点大部分执行完，但有一个或者几个reduce节点运行很慢，导致整个程序处理时间长，因为某一个key的条数比其他key的很多，从而某个节点迟迟不运行完。

1.调整参数法

Set hive.optimize.skewjoin=true;

Set hive.skewjion.key=5000000;不超过skewjion.key的key,走正常join

Set hive.map.aggr=true; --开启map端聚合

Set hive.groupby.mapaggr.checkinternal=100000;

Set hive.groupby.skewindata=true;--负载均衡，防止数据倾斜，job会增加，执行时间可能变长，

1. 业务优化

将倾斜的key过滤或者将倾斜的key单独join，最后的结果进行union 合并

1. join优化
2. mapjoin

set hive.auto.convert.join=True

set hive.mapjoin.smalltable.filesize=25000000

(2)关联值为空的时候

ON IF (isempty(T1.did),concat(‘userper’,rand()),did)=T2.did

On coalesce(t1.did,concat(‘hive’,rand()))=t2.did

## 数据倾斜（华为提供的方案）

：表关联会导致相同key的数据都分配到一个reduce中执行，job在完成99%后，长时间不能执行完。个别值集中占据大部分数据量，加上Hadoop的计算模式。

手段一：

关联的时候判断为空，映射成随机数

手段二：

将倾斜的key过滤或者将倾斜的key单独进行join，最后结果进行union合并。

手段三：

脚本通过mapjoin方式执行（小表大小一般不大于100M）

手段四：

减小切片大小，减少一个map处理的数据量（范围合理，避免map过多造成资源浪费）

手段五：

增加数据倾斜参数 set hive.optimize.skewjoin=true;

Job数会增加执行时间可能变长

手段六：

增加Map和reduce内存---临时规避，浪费资源，不一定能跑过

大小表关联：一般小表放前面大表放后面效率会快一些

## 合并小文件（输出合并）（为什么要合并小文件）

使用hive过程中经常会遇到小文件的情况：

1. 在执行插入数据操作过程中，可能会产生小文件；

②map-only作业，可能会产生小文件；

③map-reduce作业，每个reduce输出一个文件，可能产生小文件

Hive迁移数据时发现大量小文件，严重影响hive执行效率。

执行hive任务时，每个文件执行一个MAPtask,小文件数据过多，会导致大量maptask的不断启动和初始化，这样花费的时间大于逻辑处理时间，导致hive任务时间太长，造成资源浪费。

hdfs存储：存储过多小文件会产生大量元数据，会增加NameNode占有的空间，影响集群健康和拓展。

调整map的个数 以下方法来在map执行前合并小文件，减少map数

输入合并,hive读取文件时进行小文件合并，减少map个数

set mapreduce.input.fileinputformat.split.maxsize=300000000;

set mapreduce.input.fileinputformat.split.minsize=100000000;

set mapreduce.input.fileinputformat.split.minsize.per.node=100000000;

set mapreduce.input.fileinputformat.split.minsize.per.rack=100000000;

set hive.input.format = org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;

调整reduce的个数

Set.mapred.reduce.tasks 参数直接指定reduce个数

Set hive.exec.reducers.bytes.per.reducer根据输入数据量大小来决定reduce个数

--Map输入合并大小（输入合并）

Set mapred.max.split.size=256\*10的6次方

每个Map最大输入大小，这个值决定了合并后文件的数量

Set mapred.mix.split.size.per.node=256\*10的6次方

一个节点上split的至少的大小，这个值决定了多个DataNode上的文件是否需要合并

Set mapred.min.split.size.per.rack=256\*10的6次方

设置：一个交换机下split的至少的大小，这个值决定了多个交换机上的文件是否需要合并；

Sethive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;设置：输入端map合并小文件

## 输出合并，在mapreduce输出结果时进行小文件合并

Set hive.merge.mapfiles=true;

Set hive.merge.mapredufiles=true;

Set hive.merge.size.per.task=256\*10的6次方

Set hive.smallfiles.avgsize=128\*10的6次方

## job并发的设置

set hive.exec.parallel=true;

set hive.exec.paralle.thread.number=8设置并发数

shuffle过程中copy阶段开启

set mapred.reduce.slowstart.completed.maps=0.5

如果5%或者100%的map完成情况下reduce开始copy这个时候有点浪费资源

这个调优参数最好不要开启有可能发生map回退的风险。

Job的整体优化

执行模式

本地

hive> set hive.exec.mode.local.auto=true;   
hive> set hive.exec.mode.local.auto.inputbytes.max=50000000;  
hive> set hive.exec.mode.local.auto.tasks.max=4;

1.job的输入数据大小必须小于参数：hive.exec.mode.local.auto.inputbytes.max(默认128MB)

2.job的map数必须小于参数：hive.exec.mode.local.auto.tasks.max(默认4)

3.job的reduce数必须为0或者1

Join算法

1把其中一个表复制到所有节点，这样另一个表在每个节点上的分片就可以跟这个完整的表join （map side join）

2repartiton join 把两份数据按照join key 进行hash重分布，让每个节点处理hash值相同的join key (reduce side join)

## 开启动态分区参数：

hive.exec.dynamic.partition =true

hive.exec.dynamic.partition.mode = nonstrict

在每个执行MR的节点上，最大可以创建多少个动态分区

hive.exec.max.dynamic.partitions.pernode =100

在所有执行MR的节点上，最大一共可以创建多少个动态分区。

hive.exec.max.dynamic.partitions =1000

整个MR Job中，最大可以创建多少个HDFS文件

hive.exec.max.created.files = 100000

当有空分区生成时，是否抛出异常。

hive.error.on.empty.partition

使用静态分区时，必须指定分区的值

**insert** overwrite **table** employees partitions (country = 'US', state=’china’)

**select** ...,se.cnty, se.st

**from** staged\_employees se

**where** se.cnty = 'US'

where se.state=’china’;

使用动态分区

**insert** overwrite **table** employees partitions (country, state)

**select** ...,se.cnty, se.st

**from** staged\_employees se;

**动静分区结合(**注意：静态分区需要出现在动态分区字段之前。**)**

**insert** overwrite **table** employees partitions (country = 'US', state)

**select** ...,se.cnty, se.st

**from** staged\_employees se

**where** se.cnty = 'US';

## 切片的形成：

不断迭代节点列表，逐个节点（以数据快为单位）形成切片：

A如果maxsplitsize==0，则整个节点上的block数据形成一个切片

B如果maxsplitsize!==0，遍历累加每个节点上的数据块，如果累加数据块大小>=maxsplitsize,则将这些数据块形成一个切片。继续该过程，直到剩余数据块累加大小<=maxsplitsize.则进行下一步

C如果剩余数据块累加大小>=minsplitsizenode,则将这些剩余数据块形成一个切片。继续该过程，直到剩余数据块累加大小<minsplitsizenode。然后进行下一步，并这些数据块留待后续处理。

不断迭代机架列表，逐个机架（以数据块为单位）形成切片（Rack split）:

A遍历并累加这个机架上所有节点的数据块（这些数据块即上一步遗留下来的数据块），如果累加数据块大小>=maxsplitsize,则将这些数据块形成一个切片。继续该过程，直到剩余数据块累加大小<maxsplitsize。则进行下一步

B如果剩余数据块累加大小>=minsplitsizerack，则将这些剩余数据块形成一个切片。如果剩余数据块累加大小<minsplitsizerack，则这些数据块留待后续处理。

遍历并累加所有rack上的剩余数据块，如果累加数据块大小>=maxsplitsize,则将这些数据块形成一个切片。继续该过程，直到剩余数据块累加大小<maxsplitsize.则进行下一步

数据是怎么提交上来的？按照分片的形式提交上来的，一个片的数据存储在一个Map上，数据按列存储。读取元数据的时候就会提前进行分片，然后读取的是每个分片。

Hive是一个计算引擎，通过语法解析然后编译，生成Block查询块，来读取meta store 数据，提交到yarn上，yarn运行MR任务。Hive决定map reduce的个数，因为在编译的过程中会读取hdfs的元数据。

只有map没有reduce的过程：查询，插入表等不需要聚合、分组的操作。

## 常见报错：

Return code 1

常见来源：一般是语法问题，hive在编译过程中出错，未提交到yarn上

Return code 2

常见来源：一般是MR运行过程，需要yarn上看MR日志，无明显报错需要看hiveserver日志

Return code3

一般不是代码有逻辑或语法问题，而是数据层面的问题，例如，数据倾斜执行超时的问题。

UDF函数

实现方法：1，重写evaluate()方法 2，打包 ，3添加jar包 4使用自定义函数

1--使用自定义临时函数

Package com.ruozedata.java;

import org.apache.hadoop,hive.ql.exec.UDF

public class Addprefix extends UDF {

/\*这里我们实现将任一输入添加自定义前缀信息\*/

public string evaluate(string srt){return “Hive UDF prefix:”+str;

}

} --创建类

--打包成jar

--添加jar包

Hive > add jar /usr/wh/Addprefix.jar

--创建自定义临时函数

Hive > create temporary function add\_prefix as ‘com.test.Addprefix’;()

2,永久

(1)上传到hdfs

--hadoop fs -put AddPrefix.jar /wh/test

--创建永久函数

Create function add\_prefix as ‘com.test.Addprefix’ using jar ‘hdfs:///wh/test/Addprefix.jar’

--与自定义临时函数不同，使用永久函数只能在当前库

--这里我们放在test

--销毁自定义永久函数

Drop function add\_prefix;

行列转换

Hive中的行列转换

列转成行

Union all

Max +Case when+group by

Collect\_set(x) 列转行 没有重复，组装多列的数据结构体

Collect\_list(x) 列转行函数—可以用重复，组装多列的数据结构体

Concat\_ws 拼接函数，用于多列转成同一行字段后，间隔符

结合使用concat\_ws(‘,’,collect\_set(address.addr))+group by

行转列

Later\_view

Lateral view explode(split(addr,’,’)) as addr1;

去重的方法

1，distinct 2，group by 3,row\_number() 4,Max(split(‘,’,concat(a1,’,’,a2)))[1] group by a2

# oracle性能优化

## 索引建立的原则：

4.经常做函数计算的列可以做基于函数的索引；

3.重复率比较高的列，可以建立位图索引

1. 建在where 子句经常用到的列上。如果某个大表经常使用某个字段进行查询，并且检索的行数小于总表行数的5%。应该考虑
2. 对于两个表连接的字段，应该建立索引。如果经常在某表的一个字段进行order by，则也应该建立索引。
3. 不应该在小表上建立索引

## 索引优缺点

1. 索引主要进行提高数据的查询速度。当进行DML时，会更新索引。因此索引越多，则DML越慢，其需要维护索引。
2. 当一个表的索引达到4个以上时,oracle性能还是改善不了，

为什么不多个索引，在表上建立的每个索引都会增加存储开销，索引对于插入，删除，更新操作也会增加处理上的开销。过多的符合索引，在有单字段索引的情况下，一般都是没有存在价值的，相反会降低数据 增加 删除 时的性能 。

## 性能优化一：建索引

单一索引

Create index <index\_name> on <table\_name> (column\_name);

复合索引(组合索引)

当两个或多个列经常一起出现在where条件中时

Create index<index\_name>on <table\_name> (c1,c2);

Select \* from table\_name where c1=66 and c2=sal 走索引

Select \* from table\_name where c1=66 or c2=sal 进行全表扫描，不走索引

Select \* from table\_name where c1=66 走索引

Select \* from table\_name where c2=sal 进行全表扫描，不走索引

基于函数的索引

Create index emp\_upper\_idx on <表名>（函数名（列名））;

1、何时创建：在WHERE条件语句中包含函数或者表达式时

2、函数包括：算数表达式、PL/SQL函数、程序包函数、SQL函数、用户自定义函数。

位图索引

Create bitmap index index\_name on 表名（字段名）

1. 重复率比较高的列，可以建立位图索引

## 性能优化二：使用表分区

表分区：将一个表或索引物理的分解为多个更小更容易管理的部分，分而治之。

范围分区：

创建表空间

Create tablespace tbsp\_1 datafile ‘路径/tbsp\_1.dbf’ size 10;

Create tablespace tbsp\_2 datafile ‘路径/tbsp\_2.dbf’ size 10;

Create table score(name,class,grade)partition by range(grade)

(partition bujige values less than(60), partition jige values less than(85), partition youxiu values less than(maxvaule))

Select \* from score partition(bujige);

散列分区：

根据hash值进行均匀分布，尽可能实现各分区所散列的数据相等，删表重建分区

Oracle采用hash技术分区，具体分区oracle说了算，可能下一次查询就不是这个数据

Create table() partition by hash(sno)(partition p1,partition p2,partition p3);

Select \* from graderecored partition(p1);

列表分区：

列表分区明确指定了根据莫格字段的某个具体值进行分区，该分区特点，某列值只有几个

Create table() partition by list(dormitory)

(partition d229 values(‘229’),partition d228 values(‘228’),partition d240 values(‘240’) )

复合分区：

--范围-散列分区

Create table() partition by range(grade) subpartition by hash(sno,sname)

(partition p1 values less than(75) (subpartiton sp1 ,subpartiton sp2)),

(partition p2 values less than(maxvalue)(subpartition sp3,subpartition sp4));

--范围列表

Create table ()partition by range(acct\_month,area\_no) subpartition by list(day\_id)

(partition p1 values less than(‘200705’,’012’) (subpartition shangxun values(‘01’,,’10’),subpartition zhongxun values(‘11’…’15’), subpartition zhongxun values(‘16’…’31’))),

(partition p2 values less than(‘200709,’014’) (subpartition shangxun values(‘01’,,’10’),subpartition zhongxun values(‘11’…’15’), subpartition zhongxun values(‘16’…’31’)));

范围分区 range(A,B)通常以A 为标准小于A的不用考虑B值直接插入，等于A的考虑B，满足B的也插进去。

## 修改执行计划

1、/\*+ PARALLEL(表名1,并行数)[(表名2,并行数)……] \*/ --指定开启多少个并行|并发（一般为2、4、8……）

2、/\*+ INDEX(表名,索引名) \*/ --指定索引

3、/\*+ FULL(表名) \*/ --指定全表扫描

4、/\*+ USE\_NL(表名1，表名2) \*/ --指定用NESTED LOOP连接

5、/\*+ USE\_HASH(表名1，表名2) \*/ --指定用HASH连接

6、/\*+ USE\_MERGE(表名1，表名2) \*/ --指定用SORT MERGE JOIN

7、/\*+ LEADING(表名1，表名2) \*/ --指定表1作为驱动表

8、/\*+ APPEND \*/ --数据直接插入到高水位上面(与insert连用)直接往后面插，无视前面的空

## 执行计划：

执行计划时一条查询语句在oracle中的执行过程或访问路径的描述：

按F5可以查看刚刚的执行计划

执行计划常用列字段解释：

基数（rows）:oracle估计的当前操作返回结果集行数

字节（Bytes）:执行该步骤后返回的字节数

耗费（cost）、CPU耗费：oracle估计的该步骤的执行成本，用于说明sql执行的代价，理论上越小越好（该值可能与实际值有出入）

时间（time）：oracle估计的当前操作所需的时间

执行顺序：

根据operation缩进来判断，缩进最多的最先执行；（缩进相同时，最上面的最先执行）

同一级如果某个动作没有子ID就最先执行

同一级的动作执行时遵循最上最右先执行的原则

Sql执行顺序：

Index range scan →table access by global index rowid → index unique scan →table access by index rowid → nested loops outer → sort group by → select statement,goal=all\_rows。

表访问的几种方式：

Table access full(全表扫描)：查询出该表所有的数据，获取的数据执行where语句

Table access by rowed:通过rowid获取表数据

Table access by index scan :索引扫描

索引扫描分为5中：

Index unique scan (索引唯一扫描)：查询条件中包含主键，因为主键时右唯一性约束的特殊索引。

Index range scan(索引范围扫描)

Index full scan（索引全扫描）进行全索引扫描时，查询出的数据必须从索引中可以直接得到，且查询条件不包含索引前导列。

Index fast full scan(索引快速扫描)扫描索引中的所有数据块，与index full scan 类似，但是一个显著的区别是它不对查询出的数据进行排序（数据不是以排序顺序被返回）。

Index skip scan (索引跳跃扫描)

## 索引失效

NOT NULL/NULL如果某列建立索引，当进行select \* from hum\_resouce where hu\_name is not null / is null

索引列上不要使用函数，select col from tbl1 where sbustr(name,1,3)=’ABC’

Select col from table1 where name like ‘%ABC%’ --失效

Select col from table1 where name like ‘ABC%’—使用索引

索引列上不能进行计算 select col from table where col/10>10 会使索引失效，应该改成

Select col from table1 where col>10\*10

索引列上不要使用not (!=,<>) select col from table1 where col!=10

应该改写成select col from table1 where col>10 or col<10;

Sql语句优化

Cost\_based\_optimizer(基于成本的查询优化器)

Sql执行顺序

1. from from字句是从右往左进行的，选择数据少的表放右边
2. where 执行顺序是从后向前执行，因此可以过滤最大数量记录的条件写在where子句的末尾，对于多表之间的连接写在之前。这样进行连接时，可以去掉大多不重复的项
3. group by
4. agg\_fun聚合函数
5. with cube|rollup
6. having
7. select
8. order by
9. limit

用union 替换or (适用于索引列)

高效

Select id ,name

From location

Where id=1

Union all

Select id ,name

From location

Where name =’sub123’

低效率

Select id ,name

From location

Where id=1 or name=’sub123’

用exists代替in ,用not exists 代替not in

建立临时表

# Mysql优化（数据结构算法）

Mysql中常用的引擎1.InnoDB 2.MYIASM

Oracle中不存在引擎的概念，联机事务处理OLTP,日常事务处理，如银行交易，强调内存各种指标的命令率，情调绑定变量，强调并发操作。

联机分析处理OLAP，强调数据分析，强调SQL执行市场，情调磁盘I/O,强调分区。

Mysql常见索引：B+数索引，全文索引，哈希索引

InnoDB存储引擎会根据表的使用情况自动为表生成哈希索引，B+树索引就是传统意义上的索引，B+树种的B不代表二叉（binary）而是代表（balance）B+树索引并不能找到一个给定键值的具体行，能找到的只是被查找数据行所在的页，然后通过数据库把页读入到内存，再在内存种进行查找。

## 数据结构与算法

二分查找法（折半查找法）

用来查找一组有序的记录数组中的某一记录，基本思想：将记录有序化（递增或递减），在查找过程中跳跃式方式查找，即先以有序数列中点位置为比较对象，如果要找的元素小于该中点元素，则将待查序列缩小为左半部分，否则为右半部分。

顺序查找发：

二叉查找树

二叉查找树的数字代表每个节点的键值，在二叉查找树中，左子树的键值总是小于根的键值总是小于根的键值，右子树的键值总是大于根的键值。二叉查找树可以任意构造，效率低的二叉查找树。

平衡二叉树

首先符合二叉查找树的定义：任何节点两个子树的高度最大差为1，平衡二叉树查找性能比较高，接近最高，最优二叉树的建立和维护需要大量的操作。

白板上的字

描述已自动生成

白板上的字

描述已自动生成

**B+树**

B+树，二叉树，平衡二叉树都是经典的数据结构，B+树是一种平衡查找树，为直接存取辅助设备和磁盘设计的。

B+树中，所有记录节点都是按键值大小顺序放在同一层叶子节点，右各叶子节点指针进行连接，每页存4条记录，扇出为5

图示

描述已自动生成

B+树插入三种情况

B+树插入必须保证插入后叶子节点中的记录依然排序，同时需要考虑插入到B+树三种情况

Leaf page index page 都不满 插入 leaf page

Leaf page 满 index page 不满

根据中间值拆分叶子节点

Leaf page 满 index page 满 需要做两次拆分

为了保持平衡对于新插入的键值可能需要做大量拆分页（split）操作。B+树的作用于磁盘，页的分页意味着磁盘操作，在可能情况下尽量结案是页的拆分操作，B+树同样提供了类似于平衡二叉树的旋转。

B+树索引

B+树索引的本质就是B+树在数据库中的实现，B+树索引在数据库中有高扇出性，因此在数据库中B+树高度一般为2-4层，查找某一键值的行记录最多需要2到4次IO。

B+树索引可以分为聚集索引和辅助索引（非聚集索引）

聚集索引就是按照每一张表的主键构造一棵B+树，同时叶子节点中存放的即为整张表的行记录数据，聚集索引的叶子节点称为数据页。每张表只能有一个聚集索引，实际数据页只能按照一个B+树进行排序，聚集索引的存储并不是物理上连续的，而是逻辑上连续的，页通过双向链表链接，页按照主键的顺序排序，每个页中的记录通过双向链表进行维护，物理存储上可以同样不按照键存储。

范围查询：如果要查找主键某一范围内的数据，通过叶子节点的上层中间节点就可以得到页的范围，之后直接读取。

## 关联机制

嵌套循环关联（NESTED LOOPS JOIN（NL））：是指依次从驱动表中提取一条记录，遍历被探查表，将匹配的记录放入待展示的缓存区中。

优点：适用广，占用内存小，展现快

缺点：需要不停地从硬盘中读取扫描表，性能不好

**注意**：把两张表最终需要关联的数据对比，大表适合做被探查表，因为可以减少从硬盘读取扫描表的次数。

**哈希关联**（Hash Join（HJ））：计算出整张被探查表关联字段的哈希值，这些哈希值和整张被探查表一起放入缓存区，然后从驱动表逐条取记录，计算出关联字段对应的哈希值，再与被探查表的哈希值匹配,匹配上了再精准匹配每一条记录。

优点：性能好，匹配次数大大减少

缺点：只适用于等值关联，占用内存较大

**注意**：把两张表最终需要关联的数据对比，小表适合做被探查表，因为怕缓存不够。如果缓存足够的前提下，大表适合做被探查表。

**排序合并连接(Sort Merge Join (SMJ) )：**是指将关联的a表跟b表分别进行排序，生成临时的两张表后，随机取一张表逐条抽取记录与另一张表匹配。

优点：适合有索引的两张表或者不等关联

缺点：排序性能消耗大，占用内存大

# 特殊脚本以及hive的各种函数使用：

## --连续签到的最大天数(凡是涉及到连续签到参考这个答案回答)

思路：1找出每个id最早登录的时间 2date\_diff用登录时间减去最早登录时间得到(day\_interval) 3对每个登录时间进行排序

文本

描述已自动生成

脚本(hivesql版本)

文本

描述已自动生成

结果展示：在这个的结果上可以思考很多不一样的版本



## --累计的方法

1.Sum(a)over(partition by year（etl\_date）order by etl\_date)

2. select s1.date,s1.sales,

(select sum(s2.sales) from sales s2 where date<=s1.date)as cum\_sales

from sales s1

order by date asc;

1. SET @csum := 0;

SELECT 日期, 净利润, (@csum := @csum + 净利润) AS 累计利润  
FROM daily\_pnl\_view;

## --计算次日留存：

文本

描述已自动生成

SELECT a.date, ROUND(COUNT(b.user\_id) \* 1.0/COUNT(a.user\_id), 3) AS p

FROM (

SELECT user\_id, MIN(date) AS date

FROM login

GROUP BY user\_id) a

LEFT JOIN login b

ON a.user\_id = b.user\_id

AND b.date = date(a.date, '+1 day')

GROUP BY a.date

UNION

SELECT date, 0.000 AS p

FROM login

WHERE date NOT IN (

SELECT MIN(date)

FROM login

GROUP BY user\_id)

ORDER BY date;

## --行转列

行转列

表格

描述已自动生成

表格

描述已自动生成

方法一：采用union all的形式

方法二：采用lateral\_view 和str\_to\_map

select

a.dt\_month

,add\_t.type

,add\_t.num

from temp.temp\_xw\_rowtocol a

lateral view explode(str\_to\_map(concat('valid\_num=',valid\_num

,'&unvalid\_num=',unvalid\_num

),'&','='

)

) add\_t as type,num

lateral view侧视图即产生一张虚拟表

explode([split](https://so.csdn.net/so/search?q=split&spm=1001.2101.3001.7020" \t "_blank)(hobby,’,’))temp as x此与句本身返回的就是一张虚拟表，记为temp，可以为temp表的列指定字段名X

列转行：(将多行（一列）数据融为一行)

通过group by +sum+case when

select

a.dt\_month

,sum(case when type = 'valid\_num' then num end) as valid\_num

,sum(case when type = 'unvalid\_num' then num end) as unvalid\_num

from temp.temp\_xw\_coltorow a

group by dt\_month

简单的列转行

collect\_list不去重  
collect\_set(column)将某一列所有的数据转化为一个集合（去重）  
concat\_ws(’,’,collect\_set(column))将集合中的所有元素以,号分割连接为字符串

形状, 矩形

描述已自动生成

形状, 矩形

描述已自动生成

Select

Name

,concat\_ws(‘,’,collect\_set(hobby))

From stu

Group by name

## --hive中order by sort by ，distribute by,cluster by的区别

**Order by 全局排序**，只有一个Reducer,所以当数据量很大的时候用order by会比较慢

**sort by**:**区内排序**，每个Reducer内部进行排序，对全局结果集来说不是排序。  
(使用sort by的话前提要设置一下[reduce](https://so.csdn.net/so/search?q=reduce&spm=1001.2101.3001.7020" \t "_blank)个数,set mapreduce.job.reduces=n,n为reduce的个数，n>1)

**Distribute By分区排序**

DISTRIBUTE BY是控制map的输出在reducer是如何划分的

Distribute By：类似MR中partition，进行分区，并且Distribute By一般情况下都是结合sort by使用，partition by结合order by使用。这两组先分区在排序的效果是一样的，注意，Hive要求DISTRIBUTE BY语句要写在SORT BY语句之前。

**Cluster By**：当distribute by和sorts by字段相同时，可以使用cluster by方式。cluster by除了具有distribute by的功能外还兼具sort by的功能。但是排序只能是降序排序，不能指定排序规则为ASC或者DESC。

以下两种等价的写法：

select \* from emp cluster by deptno;

select \* from emp distribute by deptno sort by deptno;

**group by :**

图形用户界面, 文本

描述已自动生成

**Partition by:**

**图形用户界面, 文本, 应用程序

描述已自动生成**

## --分区表分桶表的区别

桶是以文件的形式存在，而不是像分区表一样以文件夹的形式存在

分区表和分桶表区别如下：　1、分区使用的是表外字段，需要指定字段类型；分桶使用的是表内字段，已经知道字段类型，不需要再指定。　2、分区通过关键字partitioned by(partition\_name string)声明，分桶表通过关键字clustered by(column\_name) into 3 buckets声明。3、分区划分粒度较粗，分桶是更细粒度的划分、管理数据，可以对表进行先分区再分桶的划分策略。4、分区是个伪列，只对应着文件存储路径上的一个层级。

*文本

描述已自动生成*

*图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成*

*图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成*

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

文本

中度可信度描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

## --and和where的区别

表格

描述已自动生成

## --hive分析函数的使用

### --时间函数

to\_date(string timestamp):返回时间字符串中的日期部分,如to\_date('1970-01-01 00:00:00')='1970-01-01'

current\_date:返回当前日期

year(date)：返回日期date的年,类型为int如year('2019-01-01')=2019

month(date)：返回日期date的月,类型为int,如month('2019-01-01')=1

day(date): 返回日期date的天,类型为int,如day('2019-01-01')=1

weekofyear(date1)：返回日期date1位于该年第几周。如weekofyear('2019-03-06')=10

datediff(date1,date2):返回日期date1与date2相差的天数，如datediff('2019-03-06','2019-03-05')=1

date\_add(date1,int1):返回日期date1加上int1的日期,如date\_add('2019-03-06',1)='2019-03-07'

date\_sub(date1,int1):返回日期date1减去int1的日期,如date\_sub('2019-03-06',1)='2019-03-05'

months\_between(date1,date2):返回date1与date2相差月份,如months\_between('2019-03-06','2019-01-01')=2

add\_months(date1,int1):返回date1加上int1个月的日期，int1可为负数。如add\_months('2019-02-11',-1)='2019-01-11'

last\_day(date1):返回date1所在月份最后一天。如last\_day('2019-02-01')='2019-02-28'

next\_day(date1,day1):返回日期date1的下个星期day1的日期。day1为星期X的英文前两字母如next\_day('2019-03-06','MO') 返回'2019-03-11'

trunc(date1,string1):返回日期最开始年份或月份。string1可为年(YYYY/YY/YEAR)或月(MONTH/MON/MM)。如trunc('2019-03-06','MM')='2019-03-01'，trunc('2019-03-06','YYYY')='2019-01-01'

unix\_timestamp():返回当前时间的unix时间戳，可指定日期格式。如unix\_timestamp('2019-03-06','yyyy-mm-dd')=1546704180

from\_unixtime():返回unix时间戳的日期，可指定格式。如select from\_unixtime(unix\_timestamp('2019-03-06','yyyy-mm-dd'),'yyyymmdd')='20190306'

### --条件函数

if(boolean,t1,t2):若布尔值成立，则返回t1,反正返回t2。如if(1>2,100,200)返回200

case when boolean then t1 else t2 end:若布尔值成立，则t1,否则t2,可加多重判断

coalesce(v0,v1,v2):返回参数中的第一个非空值,若所有值均为null,则返回null。如coalesce(null,1,2)返回1

isnull(a):若a为null则返回true，否则返回false

### --分析函数

row\_number() over(partitiion by .. order by .. ):根据partition排序,相同值取不同序号，不存在序号跳跃

rank() over(partition by .. order by .):根据partition排序，相同值取相同序号，存在序号跳跃（1224）

dense\_rank() over(partition by .. order by ..):根据partition排序，相同值取相同序号，不存在序号跳跃（12234）

sum() over(partition by .. order by ..)

count() over(partition by .. order by ..)

lag(col,n) over(partition by .. order by ..) ：查看当前行的上第n行

lead(col,n) over(partition by .. order by ..)：查看当前行的下第n行

first\_value() over(partition by .. order by ..):满足partition及排序的第一个值

last\_value() over(partition by .. order by ..):满足partition及排序的最后值

ntile(n) over(partition by .. order by ..):满足partition及排序的数据分成n份

### --字符串函数

length(string1):返回字符串长度

concat(string1,string2):返回拼接string1及string2后的字符串

concat\_ws(sep,string1,string2):返回按指定分隔符拼接的字符串

lower(string1):返回小写字符串，同lcase(string1)。upper()/ucase()：返回大写字符串

trim(string1):去字符串左右空格，ltrim(string1):去字符串左空格。rtrim(string1):去字符串右空格

repeat(string1，int1)：返回重复string1字符串int1次后的字符串

reverse(string1):返回string1反转后的字符串。如reverse('abc')返回'cba'

rpad(string1,len1,pad1):以pad1字符右填充string1字符串，至len1长度。如rpad('abc',5,'1')返回'abc11'。lpad()：左填充

split(string1,pat1):以pat1正则分隔字符串string1,返回数组。如split('a,b,c',',')返回["a","b","c"]

substr(string1,index1,int1):以index位置起截取int1个字符。如substr('abcde',1,2)返回'ab'

## --SQL的执行顺序

图形用户界面, 文本, 应用程序

描述已自动生成

(1) FROM:对FROM子句中的左表<left\_table>和右表<right\_table>执行笛卡儿积，产生虚拟表VT1;

(2) ON: 对虚拟表VT1进行ON筛选，只有那些符合<join\_condition>的行才被插入虚拟表VT2;

(3) JOIN: 如果指定了OUTER JOIN(如LEFT OUTER JOIN、RIGHT OUTER JOIN)，那么保留表中未匹配的行作为外部行添加到虚拟表VT2，产生虚拟表VT3。如果FROM子句包含两个以上的表，则对上一个连接生成的结果表VT3和下一个表重复执行步骤1~步骤3，直到处理完所有的表;

(4) WHERE: 对虚拟表VT3应用WHERE过滤条件，只有符合<where\_condition>的记录才会被插入虚拟表VT4;

(5) GROUP By: 根据GROUP BY子句中的列，对VT4中的记录进行分组操作，产生VT5;如果应用了group by，那么后面的所有步骤都只能得到的vt5的列或者是聚合函数（count、sum、avg等）。原因在于最终的结果集中只为每个组包含一行。这一点请牢记。

(6) CUBE|ROllUP: 对VT5进行CUBE或ROLLUP操作，产生表VT6;

(7) HAVING: 对虚拟表VT6应用HAVING过滤器，只有符合<having\_condition>的记录才会被插入到VT7;

(8) SELECT: 第二次执行SELECT操作，选择指定的列，插入到虚拟表VT8中;

(9) DISTINCT: 去除重复，产生虚拟表VT9;

(10) ORDER BY: 将虚拟表VT9中的记录按照<order\_by\_list>进行排序操作，产生虚拟表VT10;

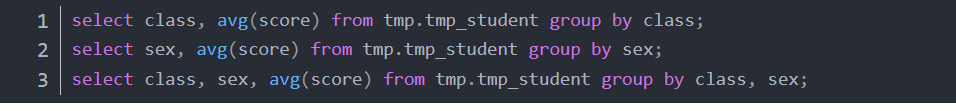
(11) LIMIT: 取出指定街行的记录，产生虚拟表VT11，并返回给查询用户

## --grouping sets的应用(多维不可加表)

1、grouping sets 只能用于 group by 之后。

2、grouping sets 中可以包含多种粒度，粒度之间用逗号连接。

Grouping sets(把单个group by 逻辑中没有参与group by 的那一列置为null)相当于把不同维度的group by 结果union all一起



文本

描述已自动生成

不可加维度：检测用户复购不可加汇总表

Detc\_type:智能提醒，智能检测，现场检测，远程检测

## --拉链表

拉链表是针对数据仓库设计中表存储数据的方式而定义的，顾名思义，所谓拉链，就是记录历史。记录一个事物从开始，一直到当前状态的所有变化的信息。

拉链表实现方式一：每条记录的生命周期

表格

描述已自动生成 表格

描述已自动生成

表格

描述已自动生成

表格

描述已自动生成

每日用户跟新表：

图片包含 表格

描述已自动生成

我们需要一张ODS层的用户全量表。至少需要用它来初始化。每日的用户更新表。

每日用户跟新表可以用主键关联 （b,a表），where concat(a所有维度)<>concat(b表所有的维度)

图片包含 表格

描述已自动生成

ods层的user表

现在我们来看一下我们ods层的用户资料切片表的结构：

1. CREATE EXTERNAL TABLE ods.user (
2. user\_num STRING COMMENT '用户编号',
3. mobile STRING COMMENT '手机号码',
4. reg\_date STRING COMMENT '注册日期'
5. COMMENT '用户资料表'
6. PARTITIONED BY (dt string)
7. ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
8. STORED AS ORC
9. LOCATION '/ods/user';
10. )
11. ods层的user\_update表
12. 然后我们还需要一张用户每日更新表，前面已经分析过该如果得到这张表，现在我们假设它已经存在。
13. CREATE EXTERNAL TABLE ods.user\_update (
14. user\_num STRING COMMENT '用户编号',
15. mobile STRING COMMENT '手机号码',
16. reg\_date STRING COMMENT '注册日期'
17. COMMENT '每日用户资料更新表'
18. PARTITIONED BY (dt string)
19. ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
20. STORED AS ORC
21. LOCATION '/ods/user\_update';
22. )
23. 拉链表
24. 现在我们创建一张拉链表：
25. CREATE EXTERNAL TABLE dws.user\_his (
26. user\_num STRING COMMENT '用户编号',
27. mobile STRING COMMENT '手机号码',
28. reg\_date STRING COMMENT '用户编号',
29. t\_start\_date ,
30. t\_end\_date
31. COMMENT '用户资料拉链表'
32. ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
33. STORED AS ORC
34. LOCATION '/dws/user\_his';
35. )
36. 实现sql语句
37. 然后初始化的sql就不写了，其实就相当于是拿一天的ods层用户表过来就行，我们写一下每日的更新语句。
38. 现在我们假设我们已经已经初始化了2017-01-01的日期，然后需要更新2017-01-02那一天的数据，我们有了下面的Sql。
39. 然后把两个日期设置为变量就可以了。
40. INSERT OVERWRITE TABLE dws.user\_his
41. SELECT \* FROM
42. (
43. SELECT A.user\_num,
44. A.mobile,
45. A.reg\_date,
46. A.t\_start\_time,
47. CASE
48. WHEN A.t\_end\_time = '9999-12-31' AND B.user\_num IS NOT NULL THEN '2017-01-01'
49. ELSE A.t\_end\_time
50. END AS t\_end\_time
51. FROM dws.user\_his AS A
52. LEFT JOIN ods.user\_update AS B
53. ON A.user\_num = B.user\_num
54. UNION
55. SELECT C.user\_num,
56. C.mobile,
57. C.reg\_date,
58. '2017-01-02' AS t\_start\_time,
59. '9999-12-31' AS t\_end\_time
60. FROM ods.user\_update AS C
61. ) AS T

# 数据治理

资产总览：

全部表，表中文名，英文名，文件数，总存储，（建议下线的表）

数据地图：

明细信息（字段信息，分区信息，变更记录）

影响分析（受影响的任务，受影响的表）

元数据中心：

Hive元数据（租户，数据框，表名，模型类型，元数据更新时间）详细的：字段信息，数据量监控

Clickhouse元数据：字段信息，血缘关系，数据量监控：分区文件大小，数据行数

血缘关系：

数仓血缘：依赖和被依赖关系

影响分析：被依赖的任务数

数据质量：

文件质量

数据质量：

数据质量总览（运行任务，失败任务，成功任务数）

任务管理（详细的任务更新时间，负责人）

执行效果：成功，等待中

生命周期管理：

Hive生周期（偏移量设置，分区的清除）

数据同步：

手工维表的导入：

现在直接现网导入，以前式用脚本归档的方式。

# 维度建模

基于kimlball理论构建领域模型，从根本上解决了一致性，拓展性，重复计算的问题。

构建领域模型，Kyllin,tableau,/finebi整体的多为分析解决方案。

---维度建模

将数据结构逻辑化设计的方法，它将客观世界划分为度量和上下文。度量常常是以数值的形式出现，事实周围有上下文包裹，这种上下文被直观地分成独立的逻辑块，称之为维度。它与实体—--关系建模有很大的区别，实体—--关系建模是面向应用，遵循第三范式，以消除数据冗余为目标的设计技术。维度建模面向分析，为了提高查询性能可以增加数据冗余，反规范化的设计技术。

## 数据架构（数据分层的意义）:

Ods（operational data store）

整体思路为贴源设计，目的在于提供业务系统接口数据（增量，全量）的存储、屏蔽数据源系统的差异，可以利用数据仓库资源进行数据处理。

接入原则：

数据结构，数据之间关系与数据源系统保持一致；（一致性）

数据粒度原则上与数据源系统最细粒度保持一致；（一致性）

不抽取存储数据源系统的汇总数据；

数据抽取为增量、全量两种方式；

DWD(data warehouse detail)

目的在于完成数据的清洗，转换和集成，提供细节数据的长期沉淀，为未来的可能需求提供历史数据支持，为DWS层提供数据结构统一，业务语义标准的基础明细数据。通常存储最明细粒度的事实数据，需要面向分析对象或概念实体划分主题域。

DWD完成如下事项：

1. 将不和规的数据空置或过滤
2. 统一数据的格式：如日期格式YYYYMMDD
3. 字段加密操作SHA256
4. 复合字段解析
5. 通过后端码值维表转移魔鬼数字（举列子）
6. 事务事实表，必须添加唯一主键，可以通过UUID实现

DIM

维度是度量的环境，用来反映业务的一类属性，这类熟悉的集合构成一个维度，也可以称为实体对象。维度属于一个数据域（地理维度：国家，省，地区）时间维度（年，季，月）

维度是维度建模的基础和灵魂，在维度建模中，将度量称为‘事实’，将环境描述为‘维度’，维度是用于分析事实所需要的环境。例如在分析交易过程中，可以通过买家，卖家，商品和时间等维度描述交易发生的环境。

维度所包含的表示维度的列，称为维度属性。维度属性是查询约束条件，分组和报表标签生成的基本来源，是数据易用性的关键。

DWS—(data warehouse summary)

对DWD层数据做预汇总处理（时间粒度变粗，概念实体粒度汇总等），压缩数据，提高访问性能，扩展核心业务实体的基础特征信息，支持便捷，灵活的多为分析，为ADS层提供便捷，可复用的汇总事实数据。

ADS（application data summary）

针对报表展示的定制开发报表，以分析需求为第一要务，力求快速产出，直面应用，模型设计应尽量收敛以减少物理表数量。

为保证不同报表同一指标的数据一致性，ADS层设计时应尽量仅通过DWS层数据简单汇总计算得出，可对汇总结果做简单关联，复杂逻辑判断应尽量下沉至DWD,DWS层实现，以达到同源复用的目的。

## 模型设计方法:

### DIM模型设计方法

单实体维度表

1只有一个自然主键

2只有一个代理主键

3每天全量，每日一个分区

4魔鬼数字需要关联码值表进行转义

5不管数据源是增量还是全量，都要与昨日累全量的处理。主要是增强模型的健壮性，防止因为数据源未到而导致当天数据为空。

混合实体维度表

1具有双主键代表A和B的组合关系

2无代理主键

3每天全量，每日一个分区，魔鬼数字需要关联码值表进行转义

4为了保障数据的一致性，混合实体维度除了双主键之外，其余的属性只能来自相关联的单实体维度表。

### DWD模型设计方法：

单事务事实表

设计方法：单事务事实表记录事务层面的事实，保存的是原子数据，也称‘原子事实表’。用户可以通过事务事实表对事务进行特别详细的分析。

四步走策略：

1选择业务过程：业务过程是组织完成的操作型活动：注册，登录，使用，付费。明确关键的业务步骤，从而选择与需求有关的业务过程。具体业务过程两个原则，一，基于业务的操作和处理流程，二基于运营的分析场景。播放是一个操作，播放对象分为歌曲和电台。业务过程划分为歌曲播放和电台播放。分享也是客户端上的一个操作，大部分运营对分享的场景不区分对象，所以业务过程为分享。

2声明粒度：业务过程选定后，要针对每个业务过程确定一个粒度，即确定事务事实表每一行所表达的细节层次。一保证所有的事实按照同样的细节层次记录，同一个事实表中不能有多种不同粒度的事实。二尽量学则最细级别的原子粒度，以确保事实表的应用具有最大灵活性。

3确认维度（何人who，何事what，何处where，何时when，如何how）

何人 Who 设备 账号

何处 where 位置

何时 when 日期 时间

何事 what 页面 歌曲集 专辑，艺人，歌曲

如何 how 客户端 本地/在线 ，触发来源，播放品质。

4确定事实

事实表的设计是为了度量业务过程，所以分析哪些事实与业务过程有关是关注点。

一选择事实时，注意只选择与业务过程有关的事实。二在事实表中应该尽量包含所有与业务过程相关的事实。

切歌播放市场，实际播放时长，播放次数

事实可以通过回答“过程的度量时什么”来确定。事实有可加性，半可加性，非可加性。尽量将不可加性事实分解为可加组件，如比率拆解为分子分母。

为保证全局分析中，大家都有一致性的维度，事实表中涉及维度的数据都要跟维表进行关联，进行数据的准确性矫正。

事务事实表军规

1每个事务事实表都有一个etl代理ID，通过UUID产生。

2必须要和维表关联，特殊如果没有维表推动业务去解决

3审计字段，主要用于问题定位，保留码值转义前的原始值，所有关键维表和关联码值表字段都需要把对应的原始信息放入审计字段中，用于问题定位。

### DWS实体汇总表

1实体是全量即使当天无操作也有记录，例如全量歌曲

2该表已经是最后结果了，不支持上卷精确去重

3累计相关指标都从该表出

主维度：

设备产品传播名，设备EMUI版本，设备APP版本号 ，电台专辑编码

--主维度不建议超过5个

附属维度：

电台专辑名称，电台专辑艺术家名，cp内

--附属维度不限制

军规：必须是全量实体，主维度不易超过5个，高基维不超过2个

多维分析汇总表

多维可加：不存在需要上卷分析的设备数

多维不可加：需要计算设备数和账号数，做上卷分析操作

多维不可加

表格

描述已自动生成

军规：

维度（可加，不可加）数量不易过多（10个）

如果一定哟啊单独计算去重用户数（设备，账号）指标，要命名为不可加（XA）

前端码值维度表

设计方法

事件编码（主键）

事件名称

Key编码 主键

Key名称

Value编码 主键

Value名称

军规：

1每个码值都要额外增加两个value

2-1111标识如果原始上报数据为空，则填充-1111，如果业务有定义默认值，则填充默认值

3-9999标识如果有原始上报与码值都关联不上则填充-9999

## 建模的过程:

1. 选择业务过程，（都是对操作型活动），维修来说1,线上。2，线下
2. 声明粒度，（一条数据代表的意思）。
3. 确定维度5W1h 线下服务门店维表
4. 确认事实 ：可加事实 ：charge维修金额 ，优惠金额，dicount\_charge

不可加事实：维修状态

事实，实体的区别

## 缓慢纤维化：

缓慢变化维：缓慢变化维的提出是因为在现实世界中，维度的属性并不是静态的，它会随着时间的流失发生缓慢的变化。

1保留原始值（维度属性值不做更改，保留原始值）

例如：一个商品上架售卖后可能由于缺货下架，补充库存后又再次上架，此情况西产生了多个商品上架售卖时间，如果重点关注的是商品首次上架售卖的时间，则采用方式0

2直接覆盖

--修改维度属性为最新值，直接覆盖，不保留历史信息

--比如商品属于哪个品类：当商品品类发生改变时，直接重写为新品类，如果业务值关系最--新的品类

3增加新的行（大型维表）

在维度表中增加一行，新行中采用新的属性值。此种方式需要借助代理键，需要为新行分配新的代理键，将其作为事实表的外键。采用此方式一般维度表中有额外3列：行生效时间，行失效时间，以及当前标识

4增加新的属性列

在维度表中增加新的一列，原先属性列存放上一版本的属性值

四种类型的维表

前端码值表：ODS

后端码值表：SKD

单事实维表:

多事实维表：

对于码值表：每份码值表都有一份备份表--加bak命名，将变化前的数据同步到备份表中，然在正式表中将变化后的维度属性直接覆盖原始值。

事实维表：（累全量，防止当天的数据没有推送，数据跑空，或者防止关联后丢数据）

用当天分区和前一天的分区进行累全量，然后用用主键进行去重，直接取最新的维度

拉链表：（弄清楚拉链表的写法）

## 模型设计的意义：

1性能良好的模型帮助快速查询

2成本，帮助减少不必要的数据冗余，实现结果的可复用性

3效率，改善用户使用属于的体验

## OLAP和OLTP系统区别看模型方法选择：

OLTP系统面向的是数据操作是随机读写，主要采用3F实体关系模型，在事物处理中解决数据冗余和一致性问题。

OLAP系统面向数据批量读写，事物处理中的一致性不是OLAP所关注的，主要关注数据的整合以及复查，大数据查询和处理中的性能。

OLAP联机分析处理：数据按不同维度聚合，上钻下卷

OLTP联机事务处理，对数据增删改查

Kylin和clickhouse 是最流行的OLAP引擎

## ER模型：（自上向下）

从企业的的高度设计一个3F，用实体关系（entity relationship）描述企业业务，站在企业的角度面向主题的抽象，而不是针对某个具体业务流程的实体对象关系，为数据分析决策服务。

1需要全面了解企业的业务和数据

2实施周期长

3对建模人员的要求高

Erwin模型建设数据仓库的出发点是整合数据，将各个系统中的数据以整个企业的角度按照主题进行相似性合并和组合，并进行一致处理

FS-LDM(financial service logical data)

1. 高层模型：一个高度抽象的模型，描述主要主题以及主题之间的相互关系
2. 中层模型：在高层模型的基础上，细化主题的数据项
3. 底层模型：在中层模型的基础上，考虑物理存储，基于平台和系统特点进行物理模型设计，也可能做一些表的合并，分区设计。

## Kimball维度模型

维度建模从分析决策的需求出发构建模型，为分析需求服务，关注如何快速的完成需求分析，同时具体较好的大规模复杂查询的响应性能，是一种自下而上的建模方式，典型代表是星模型，在一些特殊的场景下使用雪花模型。

星模型和雪花模型

星型所有的维度表都可以和事实表直接关联 数据冗余 查询效率高

雪花模型 一个或者多个维度表没有直接和事实表关联，有些统计需要通过外部连接才能产生

名词术语：

维度：度量的环境

维度属性：隶属于维度，如地理维度当中国家名称，国家id

度量/原子指标：明确含义的名词，访问次数，维修次数，支付金额，活跃设备数

数据域：面向业务过程，将业务过程或维度进行抽象集合，数据区域需要长期维护和更新，但是不轻易变动，在划分数据域时，要涵盖当前业务需求又要在新业务进入时无影响的包含在已有的数据当中。

修饰词：维度统计以为指标的业务场景限制，如pc端、业务端。

派生指标：一个原子指标+多个修饰词（可选）+时间周期 （最近一天+8设备活跃设备数）

# 8个数据分析的模型

模型：视角，框架，帮助我们理清分析逻辑，提高分析准确性

1,AARRR模型（acquisition,activation,retention,revenue,referal）

获客，激活，留存，收益，传播，形成用户什么周期的闭环模式

指标体系：典型互联网业务数据驱动运营的指标体系

2转化漏斗模型

主要通过转化率分析整个业务流程中转化和流失的情况，通过转化数据对每个环节的流失用户再进行精准营销（浏览到下单）

3 RFM模型

RFM模型是一种使用的客户分析法，主要是通过对R（最近一次消费时间），F（最近一段时间内消费频次）以及M(最近一段时间内消费金额)这三个关键指标对客户进行观察和分类，从而得出每类细分用户的价值，根据不同的用户价值去做不同的营销，这个模型对实现精准营销和节约成本有很大作用。

4波士顿矩阵

波士顿矩阵主要通过销售增长率（反映市场引力的指标）和市场占有率（反映企业实力的指标）两个指标来对公司产品进行四象限分类得出每一个产品所处的时期和特征，便于公司整体产品布局，合理投资

白板上的文字

描述已自动生成

5 购物篮分析

通过研究用户消费数据，将不同商品进行关联，并挖掘二者之间的联系。

6 KANO模型

对用户需求分类和排序，根据用户需求对用户的满意的影响，得出产品性能和用户满意之间的非线性关系。

7 ABC分析法

ABC 分析法核心：少数项目贡献了大部分价值，通过统计事务对于目标达成的贡献度，进行分类排队，分清重点和一般，从而有区别的确定管理方式

8 杜邦分析法

财务分析常用的模型，主要通过对ROE进行分解，从盈利能力，运营能力和偿债能力3个方面去衡量企业的经营业绩，帮助投资者了解一家公司的核心竞争力是什么

# 用户经营维表分类：

1人口信息维表（手工表2873）城市，区，人口数

2产品线维表

设备产品传播名(华为matex,xs)；

设备系列名(mate,p,麦芒，nova)；

产品线（一线，二线，三线）；

产品系列(mate,p);

设备品牌(华为，荣耀);

3产品线信息维表

New\_product\_line,new\_produce\_line\_series\_name

4设备日期维表

First\_active\_date,last\_active\_date,activate\_dates

5华为设备维表

从融合模型接入，包含了dim\_oneid,hwdevice\_lable(设备标签维度表)(年龄，职业，性别，激活省份，城市)

6用户竞品分析维表：产品传播名Hisense,F30S,HTC,VIVO,OPPO,1623

7 Dim\_useroper\_tablet\_health\_device\_ds(用户经营平叛类设备维表) 504

设备品牌(华为，荣耀)

内外部型号

传播名(matepad pro)

8 Dim\_useroper\_health\_device\_ds用户经营健康类设备维表133

事实维表(累全量处理)

用户经营门店信息维表（门店属性，门店状态，门店等级，负责人）

# 智慧维表以及业务过程矩阵拆分:

业务过程：

1 进入搜索

2 发起搜索

3 热词点击

4 热词曝光

5 搜索历史

6结果点击

7更多点击

8 活跃

9 搜索日志

10 云侧活跃

11 跳转

--维度：公共维度

(地域，行政区划编码，操作时间，操作类型，范围，数据来源明细，包名，审计日志)

--设备

华为设备编号，内部型号，外部型号，产品传播名，品牌，品类，系类名称，EMUI版本，价格范围，上市日期，APP版本号

--事件

事件编号，事件名称

--国家

编号，中文名

--进入搜索

进入类型

--发起搜索

搜索类型，搜索引擎

--搜索日志

请求来源，客户端请求时间，客户端语种，搜索关键字，搜索关键字字数，搜索内容类型，请求接收，请求响应时间，搜索状态，搜索结果，搜索场景

--热词

热词，热词位置，搜索类型

--APP信息

APP包名，APP名称，APP位置

# 音乐的总线矩阵:

功能块 业务过程 一致性内容

播放 歌曲播放 APP版本，歌曲，内容集，专辑，MV，艺术家，商品，年龄，性别，日期

电台播放

MV播放

添加歌单

分享 内容分享

收藏 内容收藏 设备维度表

下载 歌曲下载 账号维表

电台下载 歌曲

评论/点赞 内容评论 歌曲专辑

加入话题圈 日期

点赞 时间

支付 套餐支付 艺术家维度

歌曲支付

登录

退出

# 基金知识点总结

## 基金是一种共享经济

通过发售基金份额，将众多投资者的资金汇集起来，形成独立的财产，由基金管理人进行投资管理，基金投资人共享投资收益，共担投资风险的集合投资方式。

股票：直接投资工具，投资实业，是所有权关系，是一种所有权凭证

基金：信托关系，是一种受益凭证，间接的投资工具。主要投向证券，股票。

## 根据运作方式的不同划分

封闭式基金

开放式基金 封闭式基金存续期固定，通常在5年以上，一般为10年或15年，交易场所在证券交易所。开放式基金无特定存续期，不固定，一般不上市，向基金管理公司和代销基金进行申购，赎回。以基金份额净值为基础，不受市场供求关系影响。

## 根据组织形式的不同划分

契约型基金

公司型基金 契约型基金 不具有法人资格 ，基金持有人大会赋予基金持有者的权力相对较小， 组织方式：信托 公司型基金，基金持有者的权力较大

## 根据募集的方式不同

公募基金 ：基金募集对象不固定，投资金额相对较低，适合中小投资者参与 可以面向社会公开发售基金份额和宣传推广

私募基金：采取非公开的方式，面向特定的投资者

## 根据投资标的不同

股票基金：基金资产80%以上投资于股票

债券基金：基金资产80%以上投资于债券

货币市场基金：仅投资于货币市场工具的为货币市场基金

混合基金：偏股型 ，偏债型，股债平衡，灵活配置型基金

## 特殊基金类型

伞形基金： 多个基金公用一个合同，子基金独立运作，子基金之间可以进行相互转换的基金结构

基金中的基金： 以基金为主要投资标的

交易型开放式指数基金 ：（ETF）特点：1，被动操作的指数基金 2.独特的实物申购机制，etf指定的的一篮子证券或者商品来换取，赎回也是相应的一篮子证券或者商品。3.实行一级市场与二级市场并存的交易制度

上市开放式基金（lof）： 即可以在场外进行申购赎回，又可以在交易所（场内市场）进行交易的。可以是指数型，主动型基金，申购和赎回以现金进行，并且没有规模和场所限制

QDII ：在一国设立的经该国有关部门批准从事境外证券市场的股票,债券等有价证券投资的基金，其投资目的地为与中国证监会签署了双边监管合作谅解备忘录的国家或者地区，投资工具为股票，基金，债券，金融衍生品。

## 资金来源和用途的不同划分

在案基金：本国募集资金并投资于本国证券市场的证券投资基金

离岸基金：一国的证券投资基金组织在他国发售证券投资基金份额，用于投资本国或第三国证券市场的证券投资基金

## 根据投资目标不同

成长型基金(长期增值为目标)

收入型基金（主要用于投资可带来现金收入的有价证券）

平衡型基金

## 根据投资理念的不同

主动型基金 ：力图取得超越基准组合表现的基金

被动型基金 ：（指数基金）并不主动寻求取得超越市场的表现。而是试图复制指数的表现。一般选择特定的指数作为跟踪对象。

指数增强型基金 ：介于主动型基金和被动型基金之间的，获得高于标的指数回报水平

## 基金的投资者结构

投资者结构国际化和机构化，其中个人投资者可能已经降至50%左右，

外资

企业年金

信托

私募金

券商

社保基金

保险

公募基金

## 公募基金评价体系

图示

描述已自动生成

## 基金管理人为核心的评价体系

日程表

描述已自动生成

**换手率：**是衡量持股周期的主要指标，换手率高低反映出基金在报告期内买卖股票的频繁程度，换手率低说明基金的投资风格偏稳健，换手率高说明基金的投资风格更主动。

**持股集中度：**基金持股集中是指基金前十大重仓股市值占股票投资市值比例。

持股集中度越高可能会加大组合波动风险，但一定程度上也反映出基金管理人的持股信心和看好程度。我们的研究结果显示，高集中度管理人倾向于获得更高的超额收益。

**风险偏好：**不同管理人的风险偏好不尽相同，高风险偏好的基金管理人倾向于持有高Beta进攻性的股票，低风险偏好的基金管理人倾向于持有低Beta防御性的股票。

我们的研究结果显示，高风险偏好管理人历史业绩往往表现更差

**行业/风格偏好：**行业偏好可通过持仓分析或回归方法计算主动管理人的行业持股比例，之后观察主动管理人相对于市场基准的行业偏离度。市场上晨星风格箱是较为著名的基金持股风格分析方法，该方法主要衡量基金大小盘股与价值成长股的持股偏好。

**近年来主动管理更加强调行业/股票风格机会。**相较10年产品策略而言，19年策略描述更加强调对行业或股票风格的偏好，19年词云中除了“行业”、“主题”等较宽泛词汇外，不乏一些具体行业或风格词汇，如“新兴”、“消费”、“科技”等。

## A股B股H股的划分：

### A股：

全称是：人民币普通股票。

是由中国境内注册公司发行，在境内上市，以人民币标明面值，供境内机构、组织或个人（2013年4月1日起，境内港澳台居民可开立A股账户）以人民币认购和交易的普通股股票。

公司在A股上市，主要是为了募集境内资金；

### B股：

全称是：人民币特种股票。

是以人民币标明面值，以外币认购和买卖，在中国境内（上海、深圳）证券交易所上市交易的外资股。B股公司的注册地和上市地都在境内。

公司在B股上市，主要是为了募集境外资金；

B股名字中会直接带一个B字，比如京粮B、深中华B、绿庭B股等。

### H股：

也称：国企股。

是注册地在内地、上市地在香港的中资企业股票。因香港英文——HongKong首字母，而称得名H股。

H股无涨跌幅限制。

中国大陆地区机构投资者和个人投资者均可以投资H股。

**股票重要的板块有五个：主板、中小板、创业板、科创板和新三板。**

对于新手来说，开通股票账户后能直接交易的板块只有两个：主板和中小板

### 主板：

主板的上市企业多为大型成熟企业，资本规模大，基础好，盈利稳定；

主板能在一定程度上展现全国的经济发展状态，素有“国民经济晴雨表”之称。

上交所和深交所都设有主板。上交所主板的股票代码以“60”开头，深交所主板的股票代码以“000”或“001”开头。

### 中小板：

中小板是相对于主板而言的，主要是中小型稳定发展的企业。

这些公司的规模不够大，达不到主板市场的要求，只能选择在中小板上市。

中小板只设在深交所，所以股票代码以“002”“003”和“004”开头。

**A股+B股=所有在上交所和深交所上市的公司股票。**

A股采取人民币交易，而沪市B股采取美元交易，深市B股采取港币交易。

场内交易（深沪主板市场，中小板市场，科创板市场，创业板市场，新三板（全国中小企业股份转让系统））

场外交易市场（OTC或柜台市场）

# 各种报表工具的优缺点

FINEBI

帆软旗下的自助BI产品，轻量化的BI工具，部署方便，走多维分析方向。

优势：报价优势，性价比高。简单易学，自助取数前端分析。轻型BI，提供数据清洗/管理/分析/可视化一体方案,PC端、phone、LED大屏展示等多种展示方案。

--通过大数据引擎fineindex,可以自动建模，傻瓜式操作，用户只需在dashboard中简单拖曳操作，便能制作出丰富多样的数据可视化信息，进行数据钻取，联动和顾虑等操作，自由分析数据。Finebi面向企业IT部门，业务人员，tableau面向数据分析师。

--界面符合中国人的使用习惯，有权限管理，有对部门，人员，数据权限的设置，finebi更适合企业使用，tableau权限管理功能要安装sever版，功能局限于分享，没有部门，人员，数据权限的设置。数据挖掘：支持对数据进行时序预测，数据聚类，数据分类（神经网络，决策树算法），移动端，手机可查看分析。

YONGHONG

产品稳定性不高。利用sql处理数据。

缺点：适配性差，移动端开发的过程中，我们需要将PC端展示的组件大小缩小2倍，并且不能添加任何图片效果，降低了视觉效果。自适应性差，每个组件和图片的结合都需要设置好长宽尺度和坐标。动态效果需要对各组件进行js语言控制，操作复杂。永洪本身自带的函数需要花时间学习，学习成本高，界面设置不够灵活

大数据平台拥有自主研发的基于内存的高性能计算引擎，实现从千万到百亿级数据分析的秒级响应，这是永洪BI性能出众的基础。很强的交互性。

FINEREPORT

优势：类excle的设计模式，降低了学习成本。支持多sheet和跨sheet计算，完美兼容excle公式，用户可以所见即所得的设计出任意复杂的表样。

普通模板设计，依靠单元格的拓展与父子各的关系来实现模板效果，可进行参数查询，填报报表，图表设计（交叉报表，行式报表，分组报表）

聚合报表设计，适用于一张模板中显示多个独立模块的报表，几个报表块汇总在一起的复杂报表

数据处理

Sql可视化语句编辑，报表工具开发人员不需要一行一行写sql，只需要通过简单的拖拽，就可以生成sql语句

零代码开发，尤其数据字典，实际值和显示值等的设计更是切入快速开发的要害，这些放在别的语言和体系下往往要大量代码才能实现。

数据填报功能，避免了excle收集数据混乱，还能开发各种各样的流程表单。

决策报表设计，用于弥补普通报表工具分页预览不能展示空间的问题，同时表单可以进行自由拖拽设计，自适应页面大小显示，可以在各种大小类型的屏幕上展示，移动端、可视化大屏，finereport做大屏有先天开发优势。

TABLEAU

快速分析，可视化，分享数据。借助投放功能，无需深入了解也可快速将数据可视化。同时，它还提供分享功能，以方便与同事分享可视化图表。自身定位是一款可视化工具，可视化功能强大，对计算机的硬件要求较高，部署复杂。

优势：学习成本低，可以快速上手，对于不太掌握统计的人也能完成数据分析工作。

在数据源抓取技术方面还有很长的路要走。

---tableau可以连接数据库，或者导入excel文件数据。通过拖拽数据的维度和度量到工作区，来形成可视化图表，基本不需要代码，还会自动推荐合适的图表。

# 正则表达式

|  |  |  |
| --- | --- | --- |
| $ | 匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，那么 $ 还匹配 \n 或 \r 前面的位置。 | 若要匹配 $ 字符本身，请使用 \\$ |
| ( ) | 标记子表达式的开始和结束。可以捕获子表达式以供以后使用。 | 若要匹配这两个字符，请使用 \\( 和 \\) |
| \* | 零次或多次匹配前面的字符或子表达式。 | 若要匹配 \* 字符，请使用 \\\* |
| + | 一次或多次匹配前面的字符或子表达式。 | 若要匹配 + 字符，请使用 \\+ |
| . | 匹配除换行符 \n 之外的任何单个字符。 | 若要匹配 .，请使用 \\. |
| [ ] | 标记中括号表达式的开始。 | 若要匹配这些字符，请使用 \\[ 和 \\] |
| ? | 零次或一次匹配前面的字符或子表达式，或指示“非贪心”限定符。 | 若要匹配 ? 字符，请使用 \\? |
| \ | 将下一字符标记为特殊字符、文本、反向引用或八进制转义符。 | 若要匹配 \ 字符，请使用 \\\ |
| / | 表示文本正则表达式的开始或结束。 | 若要匹配 / 字符，请使用 \\/ |
| ^ | 匹配输入字符串开始处的位置，但在中括号表达式中使用的情况除外，在那种情况下它对字符集求反。 | 若要匹配 ^ 字符本身，请使用 \\^ |
| { } | 标记限定符表达式的开始。 | 若要匹配这些字符，请使用 \\{ 和 \\} |
| | | 指出在两个项之间进行选择。 | 若要匹配 | ，请使用 \\| |

\d 匹配单个数字

\w 匹配单个字符 ，在ASCII码中等价于 *[a-zA-Z0-9\_]* ，在Unicode中则更加匹配内容更加广泛

\s 匹配单个空白字符，包含space、tab以及换行符等

[abc] 匹配范围内的单个字符串，这个例子可以匹配a、b或者c；这个例子也可以表示为[a-c]匹配可选择的内容一次。

regexp\_extract(string subject, string pattern, int index)

 select regexp\_extract('IloveYou','I(.\*?)(You)',1) from test1 limit 1

love

 select regexp\_extract('IloveYou','I(.\*?)(You)',2) from test1 limit 1

you

select regexp\_extract('IloveYou','(I)(.\*?)(You)',1) from test1 limit 1;  
I

select regexp\_extract('IloveYou','(I)(.\*?)(You)',0) from test1 limit 1;  
IloveYou

regexp\_replace(string subject, string pattern, int index)

select regexp\_replace("IloveYou","You","") from test1 limit 1;  
Ilove

select regexp\_replace("IloveYou","You","lili") from test1 limit 1;  
Ilovelili

# PYTHON常用的模块(看书写内容)

## Numpy(做科学计算)

图形用户界面, 文本, 应用程序

描述已自动生成

## Pandas

Pandas模块的核心操作对象就是序列（series）和数据框(dataframe),序列可以理解维数据集中的一个字段，数据框是指含有至少两个字段（或序列）的数据集。

构造序列：（列表，元组，字典，numpy中的一维数组，数据框某一列）

Df1=pd.series([2.8,3.01,8.99,8.59,5.18])

Df2=pd.series({‘北京’：2.8，‘上海’：3.01，‘广东’：8.99，‘江苏’：8.59，‘浙江’：5.18})

Df3=pd.series(np.array(2.8,3.01,8.99,5.18))

数组构造：列表，元组，（嵌套的列表和元组）

数据框的构造：（列表，元组，字典，二维数组，外部数据读取）

Pd.DataFrame([[‘张三’，23，‘男’],[‘李四’，27，‘女’]]) →嵌套列表构造数据

Pd.DataFrame({‘姓名’：[‘张三’,’李四’]，‘年龄’：[23,27],’性别’：[‘男’,’女’]}) →字典构造

Pd.DataFrame(np.array([[张三’，23，‘男’],[ ‘李四’，27，‘女’]])) →二维数组构造数据框

Pd.read\_table 读取txt pd.read\_excel pd.read\_csv

**数据库数据的读取**pymysql.connect(host=None,user=,password=’’,database=’’,port=,charset=’)

数据统计描述：

Sec\_cars.newprice= Sec\_cars.newprice.str[:-1].astypr(‘float’)

Sec\_cars.dtypes 每列的数据类型

Df.shape

Df.coulumns

Df.header() 前5行

Sec\_cars.describe() 数值型基本计量统计

Sec\_cars.describe(include=[‘object’])

Sec\_cars.brand.describe()指定的离散型列

x.skew()偏度 偏度值大于0 右偏

x.kurt()峰度 峰度值大于0尖峰

df.reset\_index（inplace=true）把标签设置为数据框中的列

常用数据清洗方法:

重复观测处理

Df.duplicated()检测数据中是否存在重复观测

Df.drop\_duplicates(inplace=True)

缺失值处理（删除法和替换法）

Any(df.isnull()) 缺失观测的检测

Df.dropna() 删除缺省值的行

Df.drop(‘age’,axis=1) 删除列

Df.fillna(method=’ffill’) 向前替换

Df.fillna(method=’bfill’) 向后替换

Df.fillna(value=0)

Df.fillna(value={‘gender’:df.gender.mode()[0],’age’:df.age.mean(),’income’:df.income.median()})

异常值检测（标准差法）：

Xbar=df.counts.mean()

Xstd=df.counts.std()

Any(df.counts > Xbar+2\*xstd) 上限值检测

Any(df.counts < Xbar+2\*xstd) 下限值检测

异常值检测（箱线图法）这个待看

数据子集的获取：（iloc,loc,ix）

如果是基于条件的记录筛选，只能使用loc和ix两种方法。

Df.iloc[1:4,[0,2]]

Df1.loc[1:3,[‘name’,’age’]]

Df.loc[[‘李四’,’王二’,’丁一’],:]

Df.loc[df.gender==’男’，[‘name’,’age’]]

Df.ix[1:3,[0,2]]

Df.ix[df.gender==’男’, [‘name’,’age’]]

Df.set\_index(‘name’)将员工的姓名作为行标签

透视表功能：

Pd.pivot\_table(data,values,index,columns,aggfunc)

合并与连接

Pd.concat([df1,df2],keys=[‘’,’’]) 纵向合并

Pd.merge(left=df1,right=df2,how=’left’,left\_on=’id’,right\_on=’id’) 横向合并

分组聚合操作：

Select

Color

Cult

Count(\*) as counts

Min(carat) as min\_weight

Avg(price) as avg\_price

Max(face\_width) as max\_face\_width

From diamonds

Group by color ,cut

Grouped=diamonds.groupby(by=[‘color’,’cut’])

Result=grouped.aggregate({‘color’:np.size,’carat’:np.min,’price’:np.mean,’table’:np.max})

Result=pd.dataframe(result,columns=[‘color’,’carat’,’price’,’table’])

Result.rename(columns={‘color’:’counts’,’carat’:’min\_weight’,’price’:’avg\_price’,’table’:’max\_face\_width’},inplace=True)

Result.reset\_index(inplace=True)

## Re

查findall(pattern,string,flags=0)

替sub(patten,repi,string,count=0,flags=0)

分割split(patten,string,maxsplit)

## 线性回归模型（误差平方和最小（最小二乘法的思路））:

要想得到理想的拟合线，就必须使误差项达到最小。误差项是y与a+bx的差，结果可能为正值或负值，因此转换为误差平方和最小的问题。

为了使模型更加健壮，进一步对模型进行诊断，完成模型诊断工作：

线性回归模型的假设前提：

1误差项服从正态分布

2无多重共线性

3线性相关性

4误差项的独立性

5方差齐性

正态性检验：定性的图形法：直方图，PP图，QQ图

定量的非参数法：shapiro和K-S

多重共线性检验：模型中自变量之间存在较高的线性相关关系，它的存在给模型带来后果，方差膨胀因子VIF鉴定，如果VIF>10说明存在多重共线性：弱国VIF大于100.则表明变量间存在严重的多重共线性。

线性相关性检验：确保用于建模的自变量和因变量之间存在线性相关，可用数据框的corrwith

异常值检验：帽子矩阵，DFFITS准则，学生化残差，或cook

方差齐性检验：方差齐性要求模型残差项的方差不随着自变量的变动而呈现某种趋势

回归模型的假设诊断：

模型的显著性检验-F检验

回归系数的显著性检验-T检验

模型拟合

Ols(formular,data)

Statsmodels.api as sm

Sm.formula.ols(‘y-x’,data).fit

模型的优缺点：

优点：在小数据上表现好，结果可以解释，容易说明，训练速度和预测速度较快，不需要参数调整。

缺点：数据冗余的话模型不稳定，预测精确度较低，不适用于非线性，可能会出现过度拟合。

## Kmeans和DBSCAN聚类分析:

Kmeans思想:

文本, 信件

描述已自动生成

Kmeans和DBscan的区别:

文本, 信件

描述已自动生成

DBscan思想：

图示, 文本

描述已自动生成

文本, 信件

描述已自动生成

cluster.DBSCAN(eps=0.5,min\_samples=5,metric='euclidean',metric\_params=None,algorithm='auto',leaf\_size=30,p=none,n\_job=1)

eps ：用于设置密度聚类中的领域，即半径，默认为0.5

min\_samples:用于设置领域内最少的样本量，默认为5

cluster.KMeans(n\_clusters=8,init='k-means',n\_init=10,max\_iter=300,tol=0.0001,precompute\_distances='auto',verbose=0,random\_state=none,copy\_x=true)

n\_clusters:用于指点聚类的簇数

init：用指定初始簇中心设置方法 k-means表示初始簇中心之间的相距较远， 如果为 'random'表示随机挑选K个样本作为初始的簇中心

## Logistic回归模型:

Logistic回归是一种非线性的回归模型，但是它又和线性回归模型有关，称其为广义的线性回归分析模型。

手机屏幕截图

描述已自动生成

文本, 信件

描述已自动生成

白板上的文字

描述已自动生成

分类模型的评估方法

1混淆矩阵(可以通过pandas模块中的crosstab,也可以借助于sklearn子模块metrics中的confusion\_matrix)

准确率指标ACCURACY，正例覆盖率指标SENSITIVITY，负例覆盖率SPECIFICITY指标，这３个指标越高，说明模型越理想

２ＲＯＣ曲线

AUC,当AUC在０.８以上时，模型基本可以接受。

3Ｋ-Ｓ曲线

通常情况下ｋｓ值大于０.４时，模型基本可以接受。

Ｌｏｇｉｓｔｉｃｒｅｇｒｅｓｓｉｏｎ（ｐｅｎａｌｔｙ＝＇１２＇）

from sklearn　 import 　linear\_model

linear\_model．Ｌｏｇｉｓｔｉｃｒｅｇｒｅｓｓｉｏｎ

## 线性回归和逻辑回归的区别:

文本

描述已自动生成

手机屏幕截图

描述已自动生成

## 岭回归LASSO回归模型

实际中自变量间存在多重共线性的情况，或者自变量个数多于样本量。处理原则在线性回归模型的目标函数上添加惩罚项。

可视化的方法确定lambda

Ridge(alpha=,fit\_intercept=true,normalize=,copy\_x,tol=0.001)

Lasso(alpha=,fit\_intercept=true,random\_state=none)

交叉验证方法确定lambda

from sklearn import model\_selection

from sklearn.linear\_model import Ridge,RidgeCV , Lasso, LassoCv

from sklearn　 import 　linear\_model

RidgeCv(alphas=(0.1,1.0,10.0),cv) cv交叉验证的重数。

LassoCv(alpha=,fit\_intercept=true,random\_state=none)

区别：岭回归不管怎么样缩减，都会始终保留建模时的所有变量，无法降低模型的复杂程度。LASSO回归，回归系数的缩减过程中，可以将一些不重要的回归系数值直接缩减为0，即达到变量筛选的功能，LASSO中惩罚项由平方和改成了绝对值。

## 决策树与随机森林树:

类似于流程图的树结构，其规则是ＩＦ．．．ＴＨＥＮ的思想可用于数值型因变量的预测和离散型因变量的分类。

信息增益：信息量越大（分类越不‘纯净’），对应的熵值越大。即信息增益会偏向于取值较多的字段。

信息增益率：信息增益会偏向于取值较多的字段

基尼指数：信息增益率指标实现根节点或中间节点的字段选择，都只是针对离散型CART算法，称为分类回归树，字段选择指标是基尼指数

文本, 信件

描述已自动生成

误差降低剪枝法：自底向上

悲观剪枝法：　自顶向下

代价复杂度剪枝法：

决策树：

from sklearn.model\_selection import GridSearchCV --选择最佳的参数组合

from sklearn import tree

cart\_class=tree.DecisionTreeClassifier(max\_depth,min\_samples\_leaf=4,min\_samples\_split=2)

构建随机森林模型

from sklearn import ensemble

RF\_class=ensemble.RandomForestClassifier(n\_estimator=200,random\_state=1234)

RF\_class.fit(x\_train,y\_train)

## 朴素贝叶斯模型:

## GBDT模型:

梯度提升树，属于有监督的继承学习算法，可以用于分类问题的识别和预测问题的解决。