**离线计算开发流程**

**1、基本信息**

**1.1 DolphinScheduler平台**

**使用参考**[**使用手册**](https://swkj.yuque.com/staff-ii172u/clci8k/mo597x)  
http://119.3.142.120:12345/dolphinscheduler/ui/#/projects/list

**1.2 Hue在线编辑器**

<http://hw-node5:8889/hue/editor/?type=hive>

hive库名：iwater

用户名密码：wt\_house/wt\_house

**1.3 DataX配置**

DataX通过python脚本执行，鉴于不同的表进行同步需要很多重复的字段匹配工作，因此写了个脚本，配合数据库维护datax的同步配置。

配置数据库：hw-node5:3306/datax\_config\_sw, root/root

**1.4 关联服务器**

#华为云开发用服务器，请加入本地host配置  
121.36.155.193 hw-node1  
122.112.171.139 hw-node2  
119.3.149.192 hw-node3  
119.3.142.120 hw-node4  
122.112.190.101 hw-node5

**1.5 配套学习资料**

[尚硅谷大数据技术之Hive.pdf](https://swkj.yuque.com/attachments/yuque/0/2021/pdf/12946970/1631683445378-f2c99bb1-85f3-4822-8d94-58c6045744af.pdf)

[数仓命名规范.pdf](https://swkj.yuque.com/attachments/yuque/0/2021/pdf/12946970/1631683445719-f9c1f2cb-64dc-471f-8dd5-69e8fe8f2c8e.pdf)

[SQL编写规范.pdf](https://swkj.yuque.com/attachments/yuque/0/2021/pdf/12946970/1631683445875-0824acad-c5a2-457e-8f56-ce002a5c4daf.pdf)

[《大数据之路：阿里巴巴大数据实践》.pdf](https://swkj.yuque.com/attachments/yuque/0/2021/pdf/12946970/1631683446038-08127981-1489-40ab-9d2e-a02eabce6a95.pdf)

**2、流程设计**

在正式进行代码和脚本的编写之前，你需要对整个运行流程有个整体的认识。从业务需求开始，我们从原始表中同步数据到我们的数仓，通过hiveSQL对数据执行了清洗，之后再以特定的方式提供数据应用。

在整个环节当中，我们可能需要考虑一些问题：

原始表的更新周期及结构什么样，我们需要保持什么样的同步频率、同步策略才可以满足业务需求？

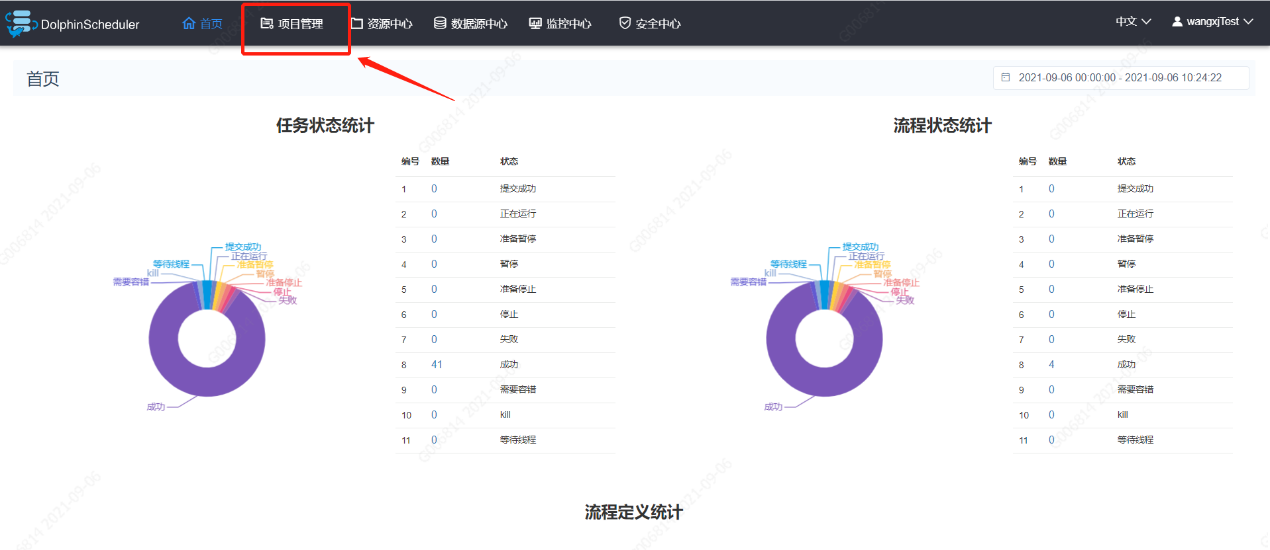
数据量的规模如何，我们需要制定一个合理的存储策略？

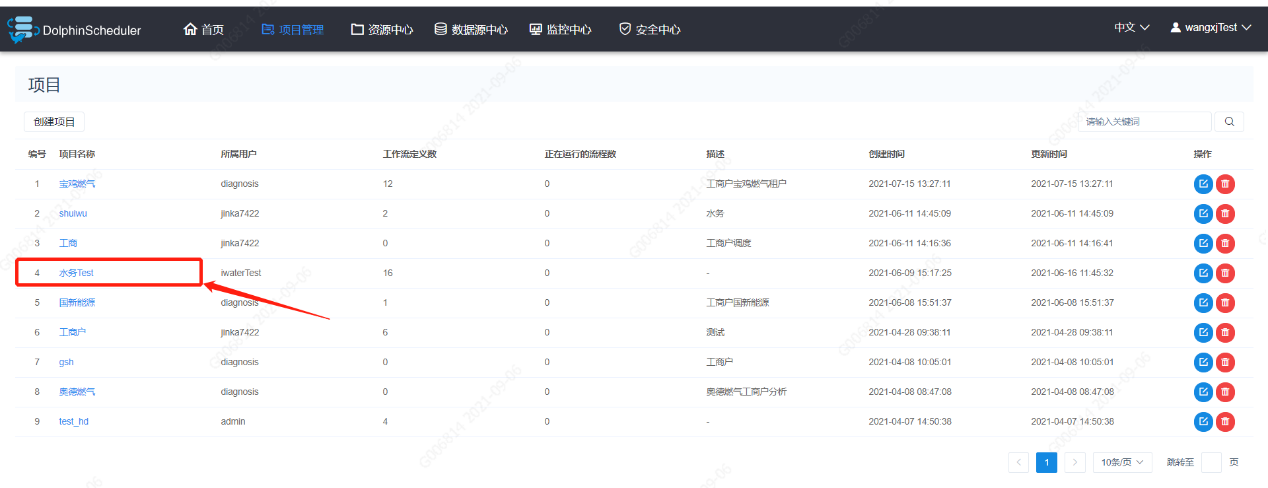
数据的清洗规则是什么样的，如何设计数据分层，能有更好的性能和可维护性？

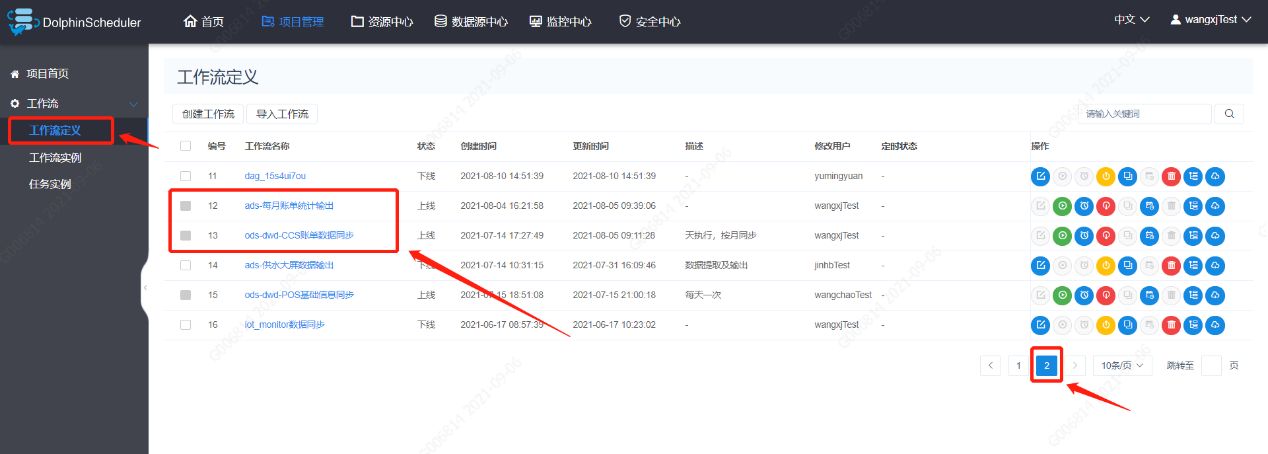
实际使用过程当中，我们需要考虑的问题会更多，这些在实际操作前应尽可能考虑清楚。

**2.1 参考案例**

项目管理->水务test->工作流定义



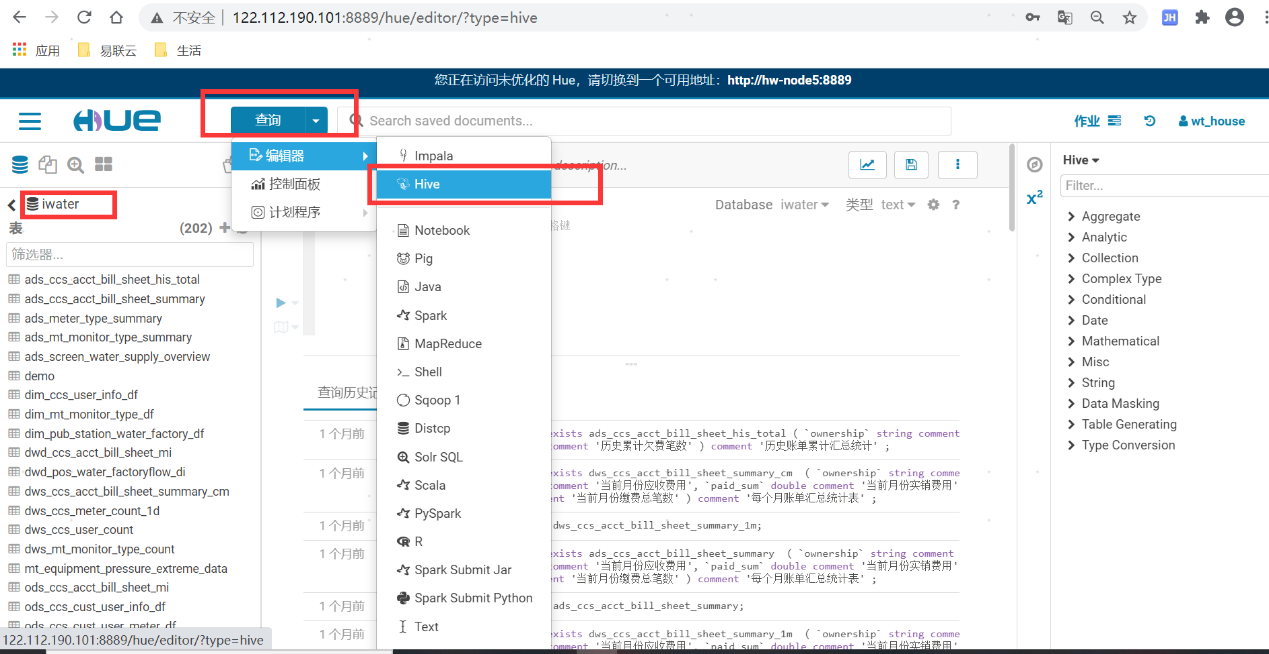


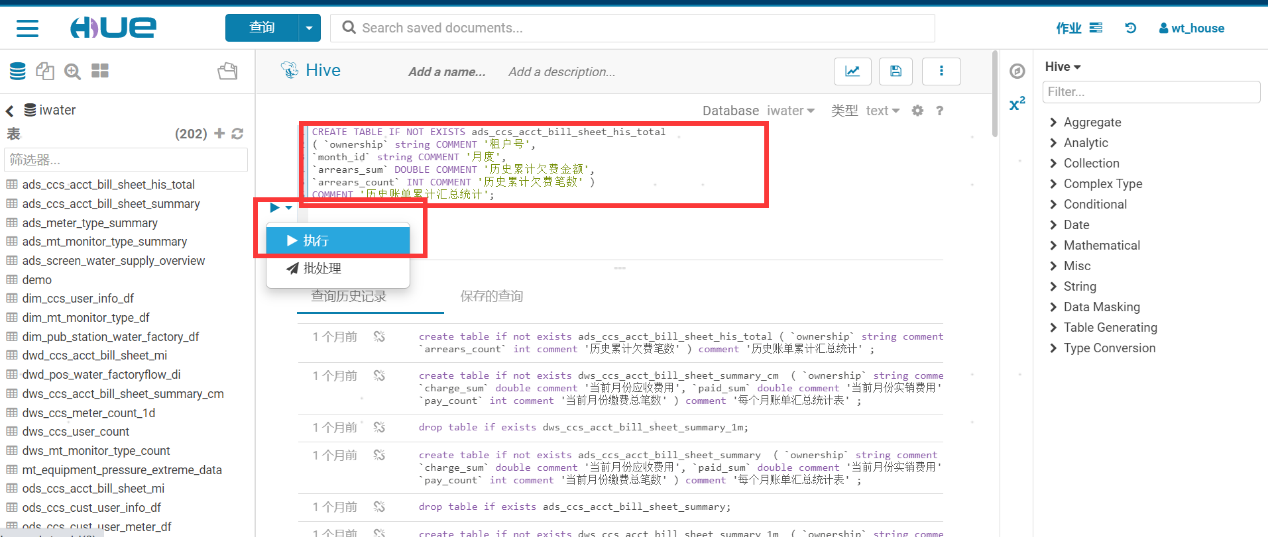


**2.2、开发流程之mysql->hive**

在hive创建一张与mysql中要同步的表相同结构的表，ods的数据类型都用String，表名一般以**ods\_业务名（例如ccs）\_原表名**，在datax的mysql配置库配置两个库的地址和表名，再在DolphinScheduler平台上配置同步任务，通过python脚本将mysql的数据同步到hive上

**A.在hive数据库创建表**





建表语句

create table if not exists ods\_ccs\_sys\_dict\_value\_df

(

`id` string comment '主键字段',

`dict\_type\_id` string comment '字典分类',

`dict\_id` string comment '字典代码',

`dict\_name` string comment '字典名称',

`status` string comment '状态0：无效 1：有效',

`sort\_no` string comment '排序',

`seq\_no` string comment '序号',

`filter1` string comment '业务过滤1',

`filter2` string comment '业务过滤2',

`tenant\_id` string comment '租户分公司或者总公司'

) comment '数据字典配置表' partitioned by (`dt` string) ;

**B DataX配置修改**

**配置mysql库来源表信息、配置hive库目标表信息**

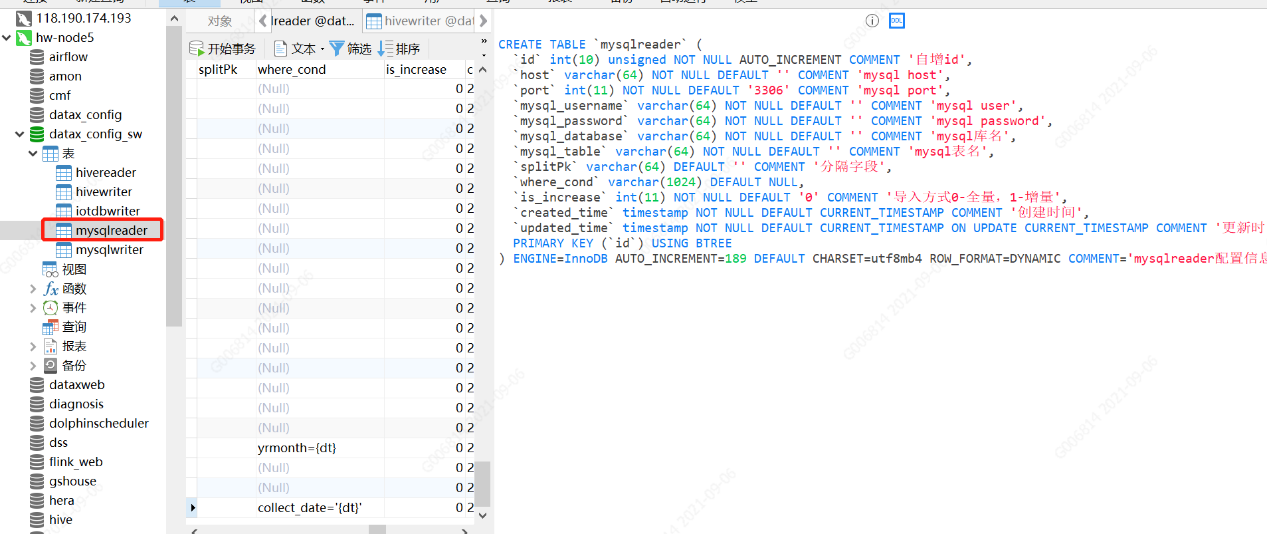
INSERT INTO `mysqlreader`( `host`, `port`, `mysql\_username`, `mysql\_password`, `mysql\_database`, `mysql\_table`, `splitPk`, `where\_cond`, `is\_increase`, `created\_time`, `updated\_time`)

VALUES ( '192.168.1.173', 3306, 'eslink123', 'Eslink@123', 'iot\_monitor', 'mt\_monitor\_type', '', NULL, 0,NOW(), NOW());

INSERT INTO `datax\_config\_sw`.`hivewriter`( `hive\_host`, `hive\_port`, `hive\_database`, `hive\_table`, `writeMode`, `hive\_username`, `hive\_password`, `is\_partition`, `channel`, `created\_time`, `updated\_time`)

VALUES ('hw-node4', 10001, 'iwater', 'ods\_pos\_mt\_monitor\_type\_df', 'overwrite', 'hdfs', '', 1, 4,NOW(), NOW());

**配置表常用字段说明**



**splitPk**

MysqlReader进行数据抽取时，如果指定splitPk，表示用户希望使用splitPk代表的字段进行数据分片，DataX因此会启动并发任务进行数据同步，这样可以大大提供数据同步的效能。推荐splitPk用户使用表主键，因为表主键通常情况下比较均匀，因此切分出来的分片也不容易出现数据热点。

目前splitPk仅支持整形数据切分，不支持浮点、字符串、日期等其他类型。如果用户指定其他非支持类型，MysqlReader将报错！如果splitPk不填写，包括不提供splitPk或者splitPk值为空，DataX视作使用单通道同步该表数据

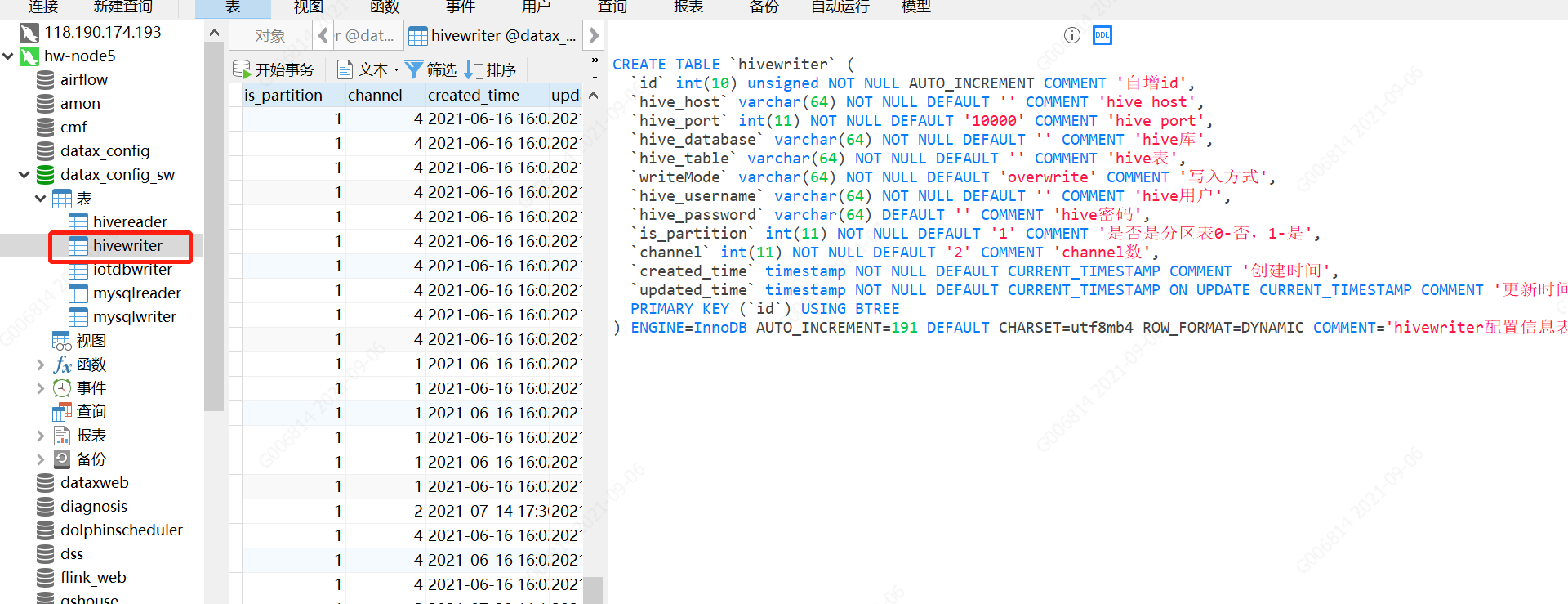
**where\_cond**

筛选条件，MysqlReader根据指定的column、table、where条件拼接SQL，并根据这个SQL进行数据抽取。在实际业务场景中，往往会选择当天的数据进行同步，可以将where条件指定为gmt\_create > $bizdate 。注意：不可以将where条件指定为limit 10，limit不是SQL的合法where子句。  
where条件可以有效地进行业务增量同步。如果不填写where语句，包括不提供where的key或者value，DataX均视作同步全量数据。

**querySql**

当用户配置querySql时，MysqlReader直接忽略table、column、where条件的配置，querySql优先级大于table、column、where选项。

描述：在有些业务场景下，where这一配置项不足以描述所筛选的条件，用户可以通过该配置型来自定义筛选SQL。当用户配置了这一项之后，DataX系统就会忽略table，column这些配置型，直接使用这个配置项的内容对数据进行筛选，例如需要进行多表join后同步数据，使用select a,b from table\_a join table\_b on table\_a.id = table\_b.id



**writeMode**

描述：hdfswriter写入前数据清理处理模式：

append，写入前不做任何处理，DataX hdfswriter直接使用filename写入，并保证文件名不冲突。

nonConflict，如果目录下有fileName前缀的文件，直接报错。

**channel**

通道数

**C 脚本**

python3 wc/dataX\_mysql\_to\_hive\_by\_auto\_increment\_id.py iot\_monitor.mt\_monitor\_type iwater.ods\_pos\_mt\_monitor\_type\_df ${dt}

dataX\_mysql\_to\_hive\_by\_auto\_increment\_id.py 同步用到的python脚本

iot\_monitor.mt\_monitor\_type myslq库名+表名

iwater.ods\_pos\_mt\_monitor\_type\_df hive库名+表名



注意dt分区参数为dolphinScheduler调度时生成的活动变量。

租户选择root



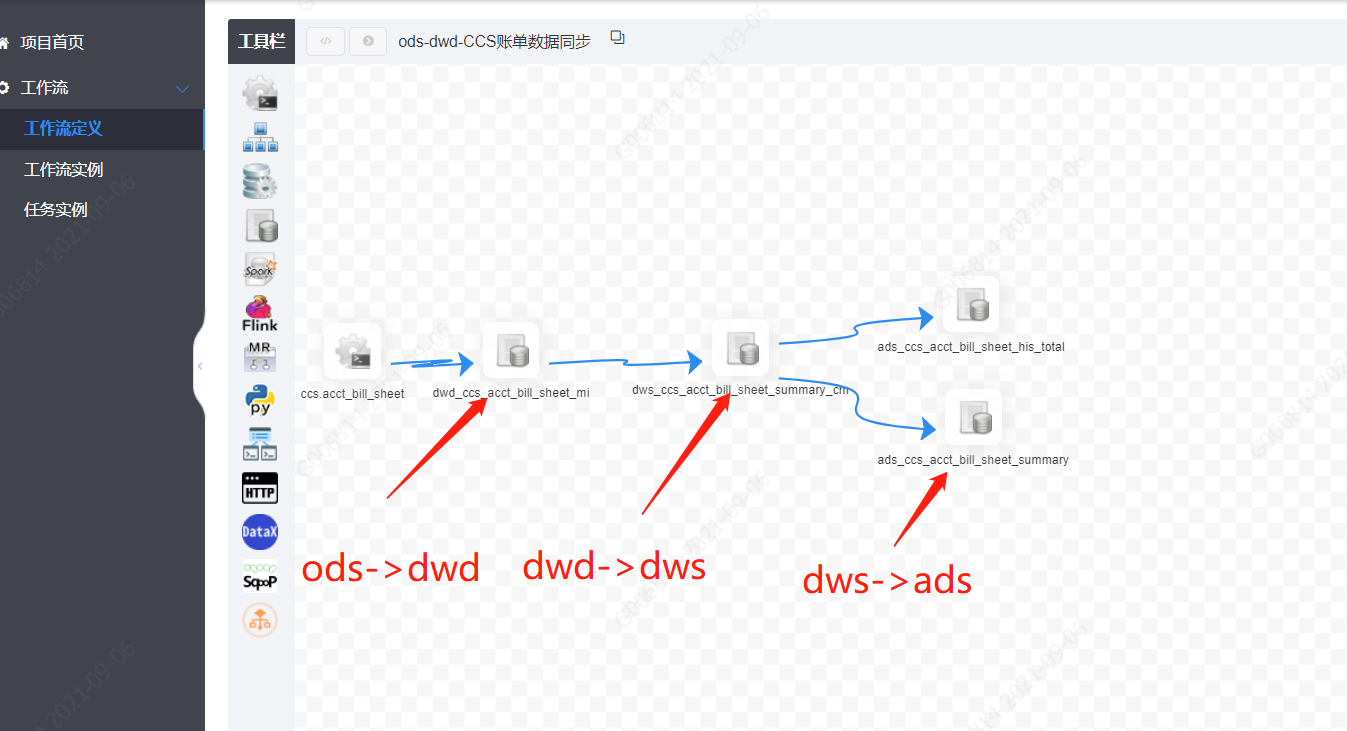
**2.3、数仓流程**

数仓流程就是通过一层层数据统计处理，将数据汇总成最总业务需要的数据结构，即最终的ads层。

**A 数仓的分层**

ods->sdd->dim->dwd->dws->ads 根据实际业务部分层级可省略。







**B 数据流程及分区规划**

根据来源表的特性，我们制定的特定的同步频率和同步条件，落盘到数仓中，应该考虑数据的存储方式：按照全量、周期快照，累积快照等方式。

举个例子：如果我们需要对CCS的账单数据进行同步，账单生成的规则是每月生成一次账单，月字段上有索引，但是账单的状态会持续的进行更新，这种情况下我们同步账单数据可以按照月建立分区，因为账单数据每天都可能更新，所以我们会每天执行同步任务，将对应的分区进行更新。

**C hiveSQL编写**

整个业务的数据计算，依赖于hiveSQL，你需要熟悉hiveSQL的基本语法以及其相较于常规sql语句的不同之处，以及一些hiveSQL的黑科技，如更新表：

with new as (

select

ownership, -- 租户号

'${dt}' as month\_id, -- 月度

substring('${dt}',0,4) as year\_id, -- 年度

sum(nvl(charge\_sum,0)) as charge\_sum,

sum(nvl(paid\_sum,0)) as paid\_sum, -- 当前月份实销费用

sum(nvl(charge\_sum,0))-sum(nvl(paid\_sum,0)) as arrears\_sum, -- 当前月份欠费金额

sum(

case when

(nvl(charge\_sum,0)-nvl(paid\_sum,0))>0

then 1 else 0 end

) as arrears\_count,

count(1) as pay\_count -- 当前月份缴费总笔数

from dwd\_ccs\_acct\_bill\_sheet\_mi

WHERE dt = '${dt}'

group by ownership

),

old as (select \* from dws\_ccs\_acct\_bill\_sheet\_summary\_cm )

insert overwrite table dws\_ccs\_acct\_bill\_sheet\_summary\_cm

select

nvl(new.ownership, old.ownership),

nvl(new.month\_id, old.month\_id),

nvl(new.year\_id, old.year\_id),

nvl(new.charge\_sum, old.charge\_sum),

nvl(new.paid\_sum, old.paid\_sum),

nvl(new.arrears\_sum, old.arrears\_sum),

nvl(new.arrears\_count, old.arrears\_count),

nvl(new.pay\_count, old.pay\_count)

from new

full outer join old on new.ownership = old.ownership and new.month\_id = old.month\_id

with 创建临时表，new、old 为临时表表名

nvl 相当于mysql的 IFNULL()

**2.4、开发流程之hive->mysql**

**A.在mysql对应库中手动创建需要的表结构**

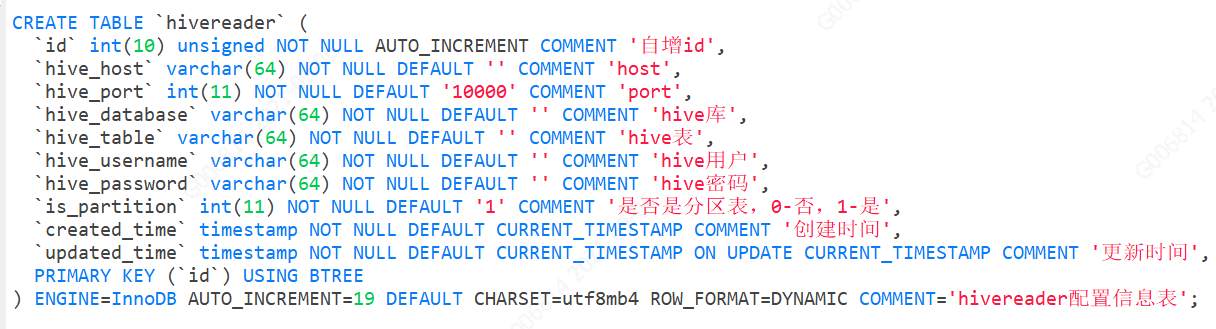
**B 在mysql配置表 配置mysql数据库地址库名表名和hive数据库地址库名表名**

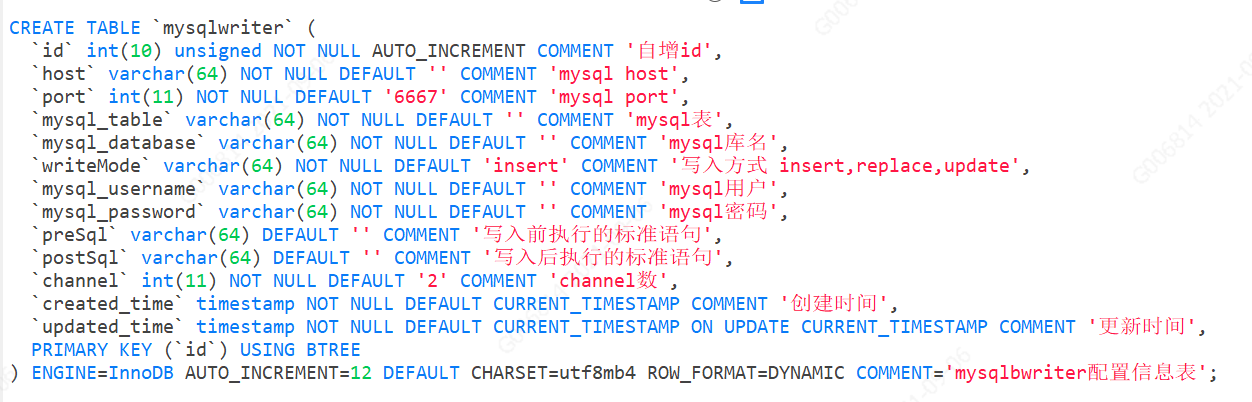
INSERT INTO `hivereader`( `hive\_host`, `hive\_port`, `hive\_database`, `hive\_table`, `hive\_username`, `hive\_password`, `is\_partition`, `created\_time`, `updated\_time`) VALUES ('hw-node4', 10000, 'iwater', 'ads\_ccs\_acct\_bill\_sheet\_summary', 'hdfs', '', 1, NOW(),NOW());

INSERT INTO `hivereader`(`id`, `hive\_host`, `hive\_port`, `hive\_database`, `hive\_table`, `hive\_username`, `hive\_password`, `is\_partition`, `created\_time`, `updated\_time`) VALUES ('hw-node4', 10000, 'iwater', 'ads\_ccs\_acct\_bill\_sheet\_his\_total', 'hdfs', '', 1, NOW(), NOW());

INSERT INTO `mysqlwriter`( `host`, `port`, `mysql\_table`, `mysql\_database`, `writeMode`, `mysql\_username`, `mysql\_password`, `preSql`, `postSql`, `channel`, `created\_time`, `updated\_time`) VALUES ( '192.168.1.173', 3306, 'ads\_ccs\_acct\_bill\_sheet\_summary', 'iwater\_screen', 'replace', 'eslink123', 'Eslink@123', '', '', 2, NOW(), NOW());

INSERT INTO `mysqlwriter`( `host`, `port`, `mysql\_table`, `mysql\_database`, `writeMode`, `mysql\_username`, `mysql\_password`, `preSql`, `postSql`, `channel`, `created\_time`, `updated\_time`) VALUES ( '192.168.1.173', 3306, 'ads\_ccs\_acct\_bill\_sheet\_his\_total', 'iwater\_screen', 'replace', 'eslink123', 'Eslink@123', '', '', 2, NOW(), NOW());





**writeMode：insert/replace/update**

insert 插入数据，重复插入会出错

replace 如果数据已经存在，就会用新值替换旧的值，没有存在相同的id,就新插入一条

update 如果已存在，就不会进行插入，跳过这个

**postSql ：**

写入数据到目的表后，会执行这里的标准语句

**preSql ：**

写入数据到目的表前，会先执行这里的标准语句

**channel：**

通道数

**C 脚本**

python3 wc/datax\_hive\_to\_mysql.py iwater.ads\_ccs\_acct\_bill\_sheet\_summary iwater\_screen.ads\_ccs\_acct\_bill\_sheet\_summary ${dt}

python3 wc/datax\_hive\_to\_mysql.py iwater.ads\_ccs\_acct\_bill\_sheet\_his\_total iwater\_screen.ads\_ccs\_acct\_bill\_sheet\_his\_total ${dt}



**D 保存时参数配置**

租户选择root

****

**3、项目目录结构**

**git地址**

[http://10.200.1.145/server/pos/eslink-data-scripts/](http://10.200.1.145/server/pos/eslink-data-scripts/tree/dev)

**目录结构说明**

