Hive详细资料

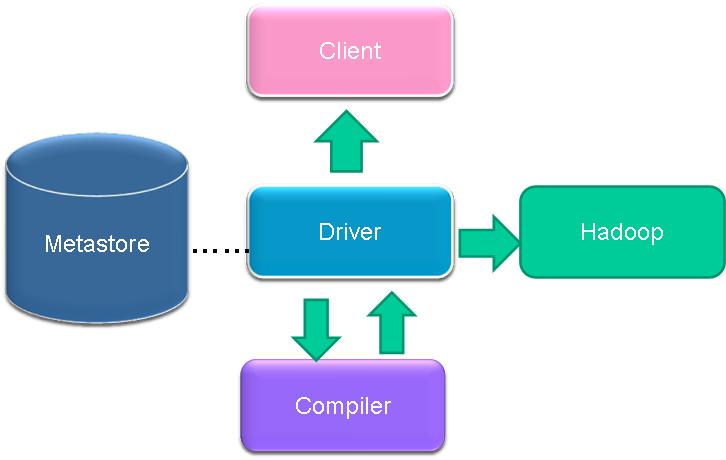
# 什么是hive

hive是基于Hadoop的一个[数据仓库](http://baike.baidu.com/view/19711.htm)工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。 其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合[数据仓库](http://baike.baidu.com/view/19711.htm)的统计分析。

## 数据仓库

[数据仓库](http://baike.baidu.com/view/19711.htm)，英文名称为Data Warehouse，可简写为[DW](http://baike.baidu.com/subview/69890/15906842.htm)或DWH。数据仓库，是为企业所有级别的决策制定过程，提供所有类型数据支持的战略[集合](http://baike.baidu.com/view/15216.htm)。它是单个数据存储，出于分析性报告和决策支持目的而创建。 为需要业务智能的企业，提供指导业务流程改进、监视时间、成本、质量以及控制，简而言之，数据仓库是用来做查询分析的数据库，基本不用来做插入，修改，删除

## Hive架构原理



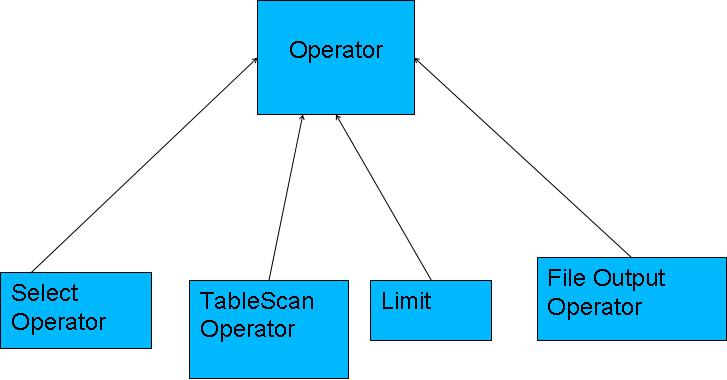
**Hive执行流程**

* 编译器将一个Hive QL转换操作符
* 操作符是Hive的最小的处理单元
* 每个操作符代表HDFS的一个操作或者一道MapReduce作业

**Operator**

* Operator都是hive定义的一个处理过程
* Operator都定义有:
* protected List **<Operator<?**  **extends** Serializable **>>** childOperators;
* protected List **<Operator<?**  **extends** Serializable **>>** parentOperators;
* protected boolean done; // 初始化值为false

所有的操作构成了 Operator图，hive正是基于这些图关系来处理诸如limit, group by, join等操作



**Hive执行流程**

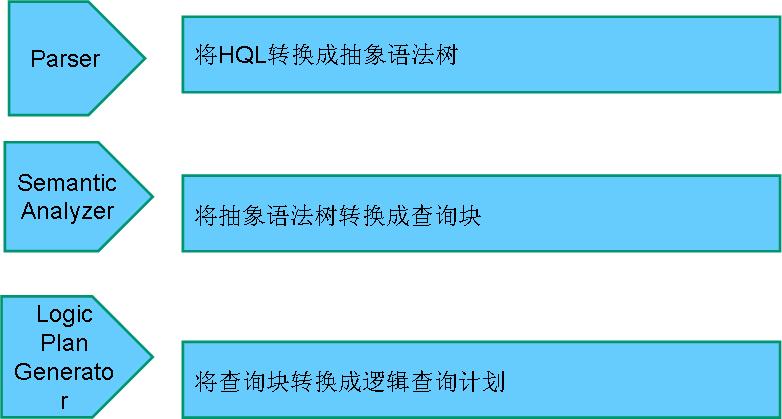
|  |  |
| --- | --- |
| **操作符** | **描述** |
| TableScanOperator | 扫描hive表数据 |
| ReduceSinkOperator | 创建将发送到Reducer端的<Key,Value>对 |
| JoinOperator | Join两份数据 |
| SelectOperator | 选择输出列 |
| FileSinkOperator | 建立结果数据,输出至文件 |
| FilterOperator | 过滤输入数据 |
| GroupByOperator | GroupBy语句 |
| MapJoinOperator | /\*+mapjoin(t) \*/ |
| LimitOperator | Limit语句 |
| UnionOperator | Union语句 |

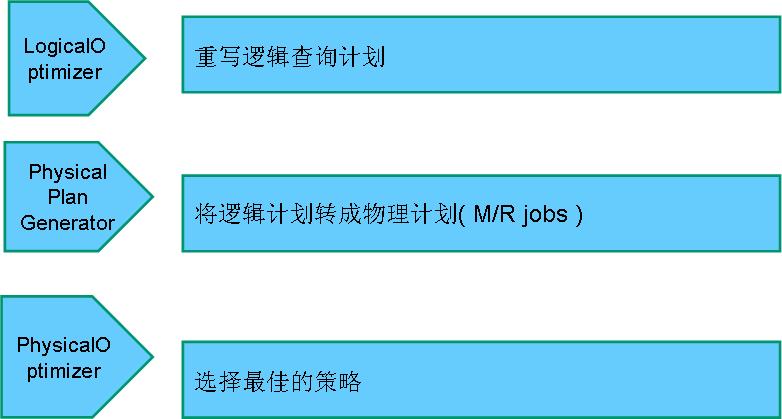
* Hive通过ExecMapper和ExecReducer执行MapReduce任务
* 在执行MapReduce时有两种模式
* 本地模式
* 分布式模式

**ANTLR词法语法分析工具**

* ANTLR—Another Tool for Language Recognition
* ANTLR 是开源的
* 为包括Java，C++，C#在内的语言提供了一个通过语法描述来自动构造自定义语言的识别器（recognizer），编译器（parser）和解释器（translator）的框架
* Hibernate就是使用了该分析工具

**Hive编译器**





# 二.hive三种方式区别和搭建

Hive中metastore（元数据存储）的三种方式：

1. 内嵌Derby方式
2. Local方式
3. Remote方式

## 1.本地derby

这种方式是最简单的存储方式，只需要在hive-site.xml做如下配置便可

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

  <name>javax.jdo.option.ConnectionURL</name>

  <value>jdbc:derby:;databaseName=metastore\_db;create=true</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionDriverName</name>

  <value>org.apache.derby.jdbc.EmbeddedDriver</value>

</property>

<property>

  <name>hive.metastore.local</name>

  <value>true</value>

</property>

<property>

  <name>hive.metastore.warehouse.dir</name>

  <value>/user/hive/warehouse</value>

</property>

</configuration>

注：使用derby存储方式时，运行hive会在当前目录生成一个derby文件和一个metastore\_db目录。这种存储方式的弊端是在同一个目录下同时只能有一个hive客户端能使用数据库，否则会提示如下错误

[html] [view plaincopyprint?](http://blog.csdn.net/reesun/article/details/8556078)

hive> show tables;

FAILED: Error in metadata: javax.jdo.JDOFatalDataStoreException: Failed to start database 'metastore\_db', see the next exception for details.

NestedThrowables:

java.sql.SQLException: Failed to start database 'metastore\_db', see the next exception for details.

FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask

hive> show tables;

FAILED: Error in metadata: javax.jdo.JDOFatalDataStoreException: Failed to start database 'metastore\_db', see the next exception for details.

NestedThrowables:

java.sql.SQLException: Failed to start database 'metastore\_db', see the next exception for details.

FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask

## 2.本地mysql

这种存储方式需要在本地运行一个mysql服务器，并作如下配置（下面两种使用mysql的方式，需要将mysql的jar包拷贝到$HIVE\_HOME/lib目录下）。

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

  <name>hive.metastore.warehouse.dir</name>

  <value>/user/hive\_remote/warehouse</value>

</property>

<property>

  <name>hive.metastore.local</name>

  <value>true</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionURL</name>

  <value>jdbc:mysql://localhost/hive\_remote?createDatabaseIfNotExist=true</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionDriverName</name>

  <value>com.mysql.jdbc.Driver</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionUserName</name>

  <value>hive</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionPassword</name>

  <value>password</value>

</property>

</configuration>

## 3..远端mysql

### 1.remote一体

这种存储方式需要在远端服务器运行一个mysql服务器，并且需要在Hive服务器启动meta服务。

这里用mysql的测试服务器，ip位192.168.1.214，新建hive\_remote数据库，字符集位latine1

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

  <name>hive.metastore.warehouse.dir</name>

  <value>/user/hive/warehouse</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionURL</name>

  <value>jdbc:mysql://192.168.57.6:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionDriverName</name>

  <value>com.mysql.jdbc.Driver</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionUserName</name>

  <value>hive</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionPassword</name>

  <value>password</value>

</property>

<property>

  <name>hive.metastore.local</name>

  <value>false</value>

</property>

<property>

  <name>hive.metastore.uris</name>

  <value>thrift://192.168.1.188:9083</value>

</property>

</configuration>

**注：这里把hive的服务端和客户端都放在同一台服务器上了。服务端和客户端可以拆开**

### 2.Remote分开

将hive-site.xml配置文件拆为如下两部分

1. **服务端配置文件**

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

  <name>hive.metastore.warehouse.dir</name>

  <value>/user/hive/warehouse</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionURL</name>

  <value>jdbc:mysql://192.168.57.6:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionDriverName</name>

  <value>com.mysql.jdbc.Driver</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionUserName</name>

  <value>root</value>

</property>

<property>

  <name>javax.jdo.option.ConnectionPassword</name>

  <value>123456</value>

</property>

</configuration>

1. **客户端配置文件**

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

  <name>hive.metastore.warehouse.dir</name>

  <value>/user/hive/warehouse</value>

</property>

<property>

  <name>hive.metastore.local</name>

  <value>false</value>

</property>

<property>

  <name>hive.metastore.uris</name>

  <value>thrift://192.168.57.5:9083</value>

</property>

</configuration>

启动hive服务端程序

 hive --service metastore

客户端直接使用hive命令即可

root@my188:~$ hive

Hive history file=/tmp/root/hive\_job\_log\_root\_201301301416\_955801255.txt

hive> show tables;

OK

test\_hive

Time taken: 0.736 seconds

hive>

# 三．Hive的数据类型和DDL

具体参见<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL>

重点是hive 的建表语句和分区

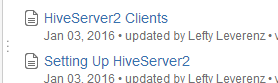
# 四．Hive的数据加载和DML

具体参见<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML>

重点是数据加载和查询插入语法

# 五．Hiveserver2和hive JDBC

企业用hive的-e -f -i三个参数







# 六．Hive的UDF和UDAF UDTF

自定义函数包括三种UDF、UDAF、UDTF

UDF(User-Defined-Function) 一进一出

UDAF(User- Defined Aggregation Funcation) 聚集函数，多进一出。Count/max/min

UDTF(User-Defined Table-Generating Functions)&#160; 一进多出，如lateral view explore()

使用方式 ：在HIVE会话中add 自定义函数的jar文件，然后创建function继而使用函数

**UDF开发**

1、UDF函数可以直接应用于select语句，对查询结构做格式化处理后，再输出内容。

2、编写UDF函数的时候需要注意一下几点：

a）自定义UDF需要继承org.apache.hadoop.hive.ql.UDF。

b）需要实现evaluate函数，evaluate函数支持重载。

3、步骤

a）把程序打包放到目标机器上去；

b）进入hive客户端，添加jar包：hive>add jar /run/jar/udf\_test.jar;

c）创建临时函数：hive>CREATE TEMPORARY FUNCTION add\_example AS 'hive.udf.Add';

d）查询HQL语句：

    SELECT add\_example(8, 9) FROM scores;

    SELECT add\_example(scores.math, scores.art) FROM scores;

    SELECT add\_example(6, 7, 8, 6.8) FROM scores;

e）销毁临时函数：hive> DROP TEMPORARY FUNCTION add\_example;

**UDAF 自定义集函数**

多行进一行出，如sum()、min()，用在group  by时

1.必须继承

         org.apache.hadoop.hive.ql.exec.UDAF(函数类继承)

         org.apache.hadoop.hive.ql.exec.UDAFEvaluator(内部类Evaluator实现UDAFEvaluator接口)

2.Evaluator需要实现 init、iterate、terminatePartial、merge、terminate这几个函数

         init():类似于构造函数，用于UDAF的初始化

         iterate():接收传入的参数，并进行内部的轮转，返回boolean

         terminatePartial():无参数，其为iterate函数轮转结束后，返回轮转数据，                                 类似于hadoop的Combiner

         merge():接收terminatePartial的返回结果，进行数据merge操作，其返回类型为boolean

         terminate():返回最终的聚集函数结果

开发一个功能同：

Oracle的wm\_concat()函数

Mysql的group\_concat()

# 七．Hive嵌入Python

Python的输入输出都是\t为分隔符，否则会出错，python脚本输入print出规定格式的数据

用法为先add file，使用语法为TRANSFORM (name, items) USING 'python test.py' AS (name string, item1 string,item2 string,item3 string)，这里后面几个字段对应python的类型

create table test (name string,items string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

LOAD DATA local INPATH '/opt/data/tt.txt' OVERWRITE INTO TABLE test ;

create table test2 (name string,item1 string,item2 string,item3 string) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

INSERT OVERWRITE TABLE test2

SELECT

TRANSFORM (name, items)

USING 'python test.py'

AS (name string, item1 string,item2 string,item3 string)

FROM test;

#!/usr/bin/python

import sys

for line in sys.stdin:

line = line.strip()

name,it = line.split('\t')

count = it.count(',')+1

for i in range(0,3-count):

it = it+',NULL'

result = it.split(',')[0:3]

print '%s\t%s'%(name,'\t'.join(result))

# 八．Hwi环境搭建

HWI是Hive Web Interface的简称，是hive cli的一个web替换方案。

需要下载Hive的源码文件，然后将hwi/web目录下的文件用 jar cvf hive-hwi-0.13.1.war ./\*

其实war包也是zip包，可以通过

cd hwi/web  
zip hive-hwi-1.2.1.zip ./\*     //打包成.zip文件。

将zip包后缀改成war

mv hive-hwi-1.2.1.zip hive-hwi-1.2.1.war  
cp hive-hwi-1.2.1.war /opt/sxt/soft/apache-hive-1.2.1-bin/lib/

命令来打包成一个war包，然后放到Hive的lib目录下即可

配置conf/hive-site.xml

<property>

<name>hive.hwi.war.file</name>

<value>lib/hive-hwi-1.2.1.war</value>

</property>

<property>

<name>hive.hwi.listen.host</name>

<value>0.0.0.0</value>

</property>

<property>

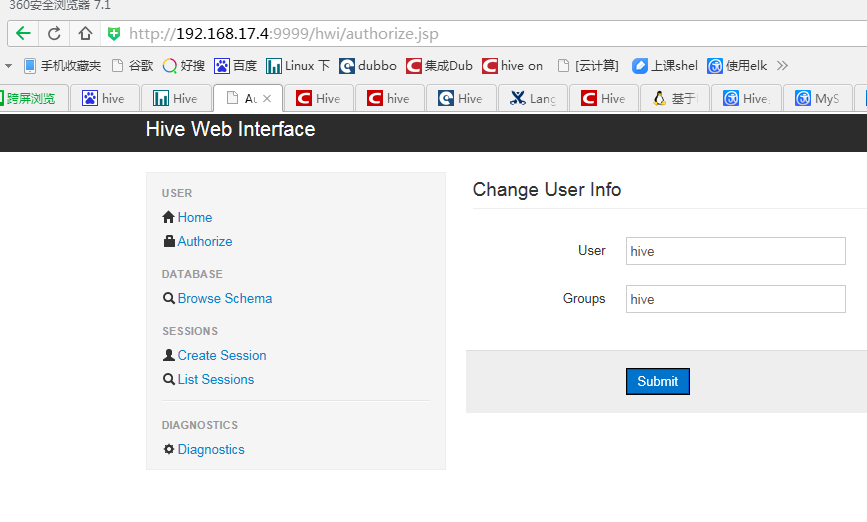
<name>hive.hwi.listen.port</name>

<value>9999</value>

</property>

执行命令hive --service hwi

访问<http://192.168.17.4:9999/hwi>



可参照<http://blog.csdn.net/wulantian/article/details/38271803>

# 九．Hive的优化

见文档hive优化