

2015年6月19日星期五

HUAWEI ENTERPRISE **A BETTER WAY**

FusionInsight HD 培训材料

基础技术— Hive

enterprise.huawei.com

HUAWEI TECHNOLOGIES CO., LTD.



1

Hive应用场景

2

Hive功能与架构

3

Hive关键流程

4

Hive开发接口介绍

5

Hive常用维护

Hive是一个开源的，建立在Hadoop上的数据仓库框架，提供类似SQL的HQL语言操作结构化数据，其基本原理是将HQL语言自动转换成MapReduce任务，从而完成对Hadoop集群中存储的海量数据进行查询和分析。

Hive主要特点如下：

- 通过HQL语言非常容易的完成数据提取、转换和加载（ETL）。
- 通过HQL完成海量结构化数据分析。
- 灵活的数据存储格式，支持JSON，CSV，TEXTFILE，RCFILE，SEQUENCEFILE等存储格式，并支持自定义扩展。
- 多种客户端连接方式，支持JDBC、Thrift等接口。

优点

- 海量结构化数据分析汇总
- 高可靠性，高容错性
- 将复杂的MapReduce编写任务简化为SQL语句。大大提升了开发效率
- 灵活的数据存储TextFile，RCFile，ORC，SequenceFile，CSV，Parquest，自定义格式
- 可扩充UDF/UDAF/UDTF

缺点

- 延迟较高，性能有提升空间
- 不支持事务类操作

Hive提供数据提取、转换、加载功能，并可用类似于SQL的语法，对HDFS海量数据库中的数据进行查询统计等操作。

Hive常用于以下几个方面：

- 数据汇总（每天/每周用户点击数，点击排行）
- 非实时分析（日志分析，统计分析）
- 数据挖掘（用户行为分析，兴趣分区，区域展示）

某地公安抓到了一个犯罪团伙的团员，获取了该员的相关信息。该团员很倔强，拒不招供同伙信息。该地公安想通过该员的相关信息，从公安系统海量的数据中找出同伙的信息来。

- 计算全省旅客空港信息里两次及两次以上与该团员搭乘同一天同一次航班的人员信息
- 计算旅店住房信息中两次及两次以上与该团员同一天（入住时间相差半小时内）同住一家旅店人员信息
- 输入身份证信息及时找出本身份证的所有关系，如果关系中是人员，可以继续逐层的挖掘人员的关系和相关信息

- 利用hive 创建对应的数据表 从公安系统中载入所有历史数据（使用hive客户端工具）

表字段：

姓名 NAME

证件 ZJHM

日期 OFFDAY

航班号 AIRPLANE

- 建立类sql语句来查询出所需要的数据

```
select c.n1,c.n2,count(*) from (select a.name as n1,b.name as n2 from hb_test1 a join hb_test1 b on  
a.offday=b.offday and a.airplane=b. airplane )c where c.n1> c.n2 group by c.n1,c.n2 having  
count(*) > 1;
```

1

Hive应用场景

2

Hive功能与架构

3

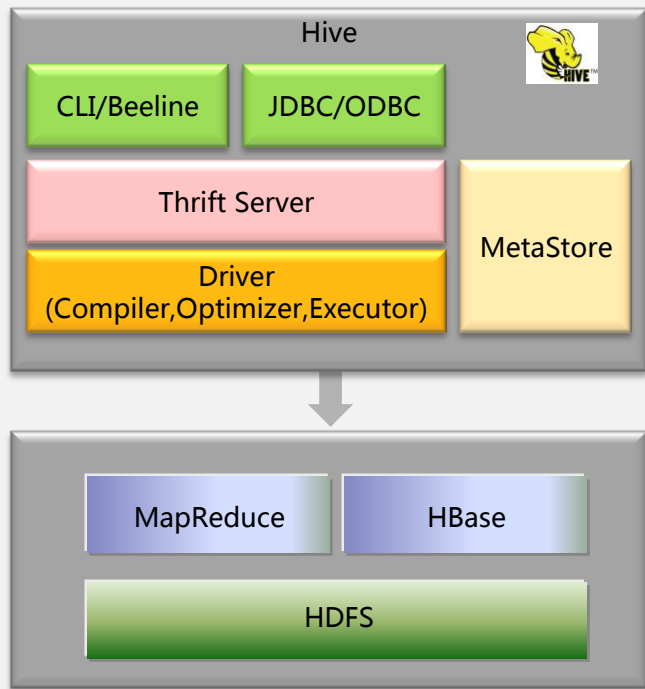
Hive关键流程

4

Hive开发接口介绍

5

Hive常用维护



MetaStore : 存储表, 列和Partition等元数据, 为关系型数据库。

Driver : 管理HiveQL执行的生命周期并贯穿Hive任务整个执行期间。

Compiler : 编译HiveQL并将其转化为一系列相互依赖的Map/Reduce任务。

Optimizer : 优化器, 分为逻辑优化器和物理优化器, 分别对HiveQL生成的执行计划和MapReduce任务进行优化。

Executor : 按照任务的依赖关系分别执行Map/Reduce任务。

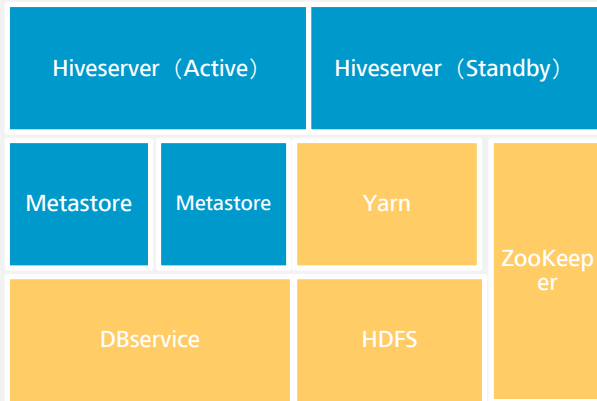
ThriftServer : 提供thrift接口, 作为JDBC和ODBC的服务端, 并将Hive和其他应用程序集成起来。

Clients : 包含命令行接口(CLI/Beeline) 和JDBC/ODBC 接口, 为用户访问提供接口。

	Hive	传统数据仓库
存储	HDFS，理论上有无限拓展的可能。	集群存储，存在容量上限（一般几百TB，不超过PB），而且伴随容量的增长，计算速度急剧下降。只能适应于数据量比较小的商业应用，对于超大规模数据无能为力。
执行引擎	依赖于MapReduce框架，可进行的各类优化较少但是比较简单。	可以选择更加高效的算法来执行查询，也可以进行更多的优化措施来提高速度。
使用方式	HQL（类似SQL）。	SQL。
索引	低效，目前还不完善。	高效。
灵活性	元数据存储独立于数据存储之外，从而解耦合元数据和数据，同样的数据，不同的用户可以有不同的元数据，可以进行不同的操作。	低，数据用途单一。
分析速度	计算依赖于MapReduce和集群规模，易拓展，在大数据量情况下，远远快于普通数据仓库。	在数据容量较小时非常快速，数据量较大时，急剧下降。

	Hive	传统数据仓库
易用性	需要自行开发应用模型，灵活度较高，但是易用性较低。	集成一整套成熟的的报表解决方案，可以较为方便的进行数据的分析。
可靠性	数据存储在HDFS，可靠性高，容错性高。	可靠性较低，一次查询失败需要重新开始。数据容错依赖于硬件Raid。
依赖环境	依赖硬件较低，可适应一般的普通机器。	依赖于高性能的商业服务器。
价格	开源产品。	商用比较昂贵，开源的性能较低。

- Hive需要启动一主一备两个HiveServer进程，来保证HiveServer服务的高可靠性。
- MetaStore进程可以启一个或两个来提供元数据服务（当提供两个MetaStore进程时Hiveserver会按配置的顺序来选择使用MetaStore 服务）
- 元数据的存储需要依赖DBservice服务





- 该页可以查看hive整体的信息
- More Action中提供了启停服务和健康检查等操作
- 右侧可以通过Customize 来调整需要显示的hive性能指标信息
- 可以下载hive 客户端

FusionInsight Manager

admin

Cluster Time: Tue Jan 13 16:57:48 CST 2015

Alarms: 0 0 0 0

Services > Hive Instances

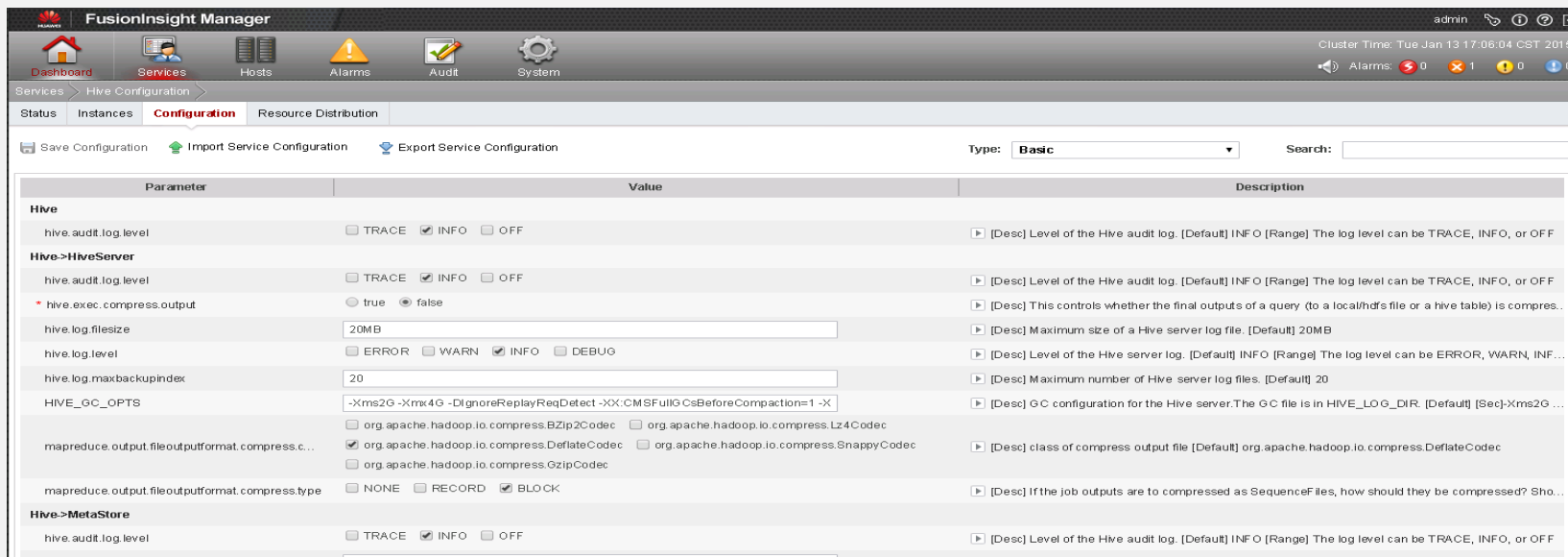
Status **Instances** Configuration Resource Distribution

+ Add Instance More Actions ▾ Refresh every 30 seconds Search:

	Role	Host Name	OM IP	Business IP	Rack	Operating Status	Health Status	Configuration Status
<input type="checkbox"/>	HiveServer(Active)	51-196-23-78	51.196.23.78	51.196.23.78	/default/rack0	✓ Started	✓ Good	✓ Synchronized
<input type="checkbox"/>	HiveServer(Standby)	51-196-23-79	51.196.23.79	51.196.23.79	/default/rack0	✓ Started	✓ Good	✓ Synchronized
<input type="checkbox"/>	MetaStore	51-196-23-78	51.196.23.78	51.196.23.78	/default/rack0	✓ Started	✓ Good	✓ Synchronized
<input type="checkbox"/>	MetaStore	51-196-23-79	51.196.23.79	51.196.23.79	/default/rack0	✓ Started	✓ Good	✓ Synchronized

Page 1 Total: 1 | < > To page Go Items 1 To 4 Total: 4 | 10 Items per page

- 此页可以查看实例主备状态所在节点以及健康状态信息
- More Action 中包含启动删除实例等操作
- Add Instance 可以添加所需要的实例



The screenshot displays the FusionInsight Manager interface for configuring Hive. The top navigation bar includes Dashboard, Services, Hosts, Alarms, Audit, and System. The main content area is titled 'Hive Configuration' and shows a table of parameters. The 'Configuration' tab is selected, and the 'Type' is set to 'Basic'. The table lists parameters for Hive, Hive->HiveServer, and Hive->MetaStore.

Parameter	Value	Description
Hive		
hive.audit.log.level	<input type="checkbox"/> TRACE <input checked="" type="checkbox"/> INFO <input type="checkbox"/> OFF	[Desc] Level of the Hive audit log. [Default] INFO [Range] The log level can be TRACE, INFO, or OFF
Hive->HiveServer		
hive.audit.log.level	<input type="checkbox"/> TRACE <input checked="" type="checkbox"/> INFO <input type="checkbox"/> OFF	[Desc] Level of the Hive audit log. [Default] INFO [Range] The log level can be TRACE, INFO, or OFF
hive.exec.compress.output	<input type="radio"/> true <input checked="" type="radio"/> false	[Desc] This controls whether the final outputs of a query (to a local/hdfs file or a hive table) is compressed.
hive.log.filesize	20MB	[Desc] Maximum size of a Hive server log file. [Default] 20MB
hive.log.level	<input type="checkbox"/> ERROR <input type="checkbox"/> WARN <input checked="" type="checkbox"/> INFO <input type="checkbox"/> DEBUG	[Desc] Level of the Hive server log. [Default] INFO [Range] The log level can be ERROR, WARN, INFO, or DEBUG
hive.log.maxbackupindex	20	[Desc] Maximum number of Hive server log files. [Default] 20
HIVE_GC_OPTS	-Xms2G -Xmx4G -DignoreReplayReqDetect -XX:CMSFullGCsBeforeCompaction=1 -X	[Desc] GC configuration for the Hive server. The GC file is in HIVE_LOG_DIR. [Default] [Sec]-Xms2G -Xmx4G -DignoreReplayReqDetect -XX:CMSFullGCsBeforeCompaction=1 -X
mapreduce.output.fileoutputformat.compress.class	<input type="checkbox"/> org.apache.hadoop.io.compress.BZip2Codec <input type="checkbox"/> org.apache.hadoop.io.compress.Lz4Codec <input checked="" type="checkbox"/> org.apache.hadoop.io.compress.DeflateCodec <input type="checkbox"/> org.apache.hadoop.io.compress.SnappyCodec	[Desc] class of compress output file [Default] org.apache.hadoop.io.compress.DeflateCodec
mapreduce.output.fileoutputformat.compress.type	<input type="checkbox"/> NONE <input type="checkbox"/> RECORD <input checked="" type="checkbox"/> BLOCK	[Desc] If the job outputs are to be compressed as SequenceFiles, how should they be compressed? Should they be compressed as SequenceFiles or as Hadoop archives?
Hive->MetaStore		
hive.audit.log.level	<input type="checkbox"/> TRACE <input checked="" type="checkbox"/> INFO <input type="checkbox"/> OFF	[Desc] Level of the Hive audit log. [Default] INFO [Range] The log level can be TRACE, INFO, or OFF

- 此页可以查看和修改Hiveserver和Metastore的配置信息
- 可以通过导出导入配置信息来修改配置信息

1

Hive应用场景

2

Hive功能与架构

3

Hive关键流程

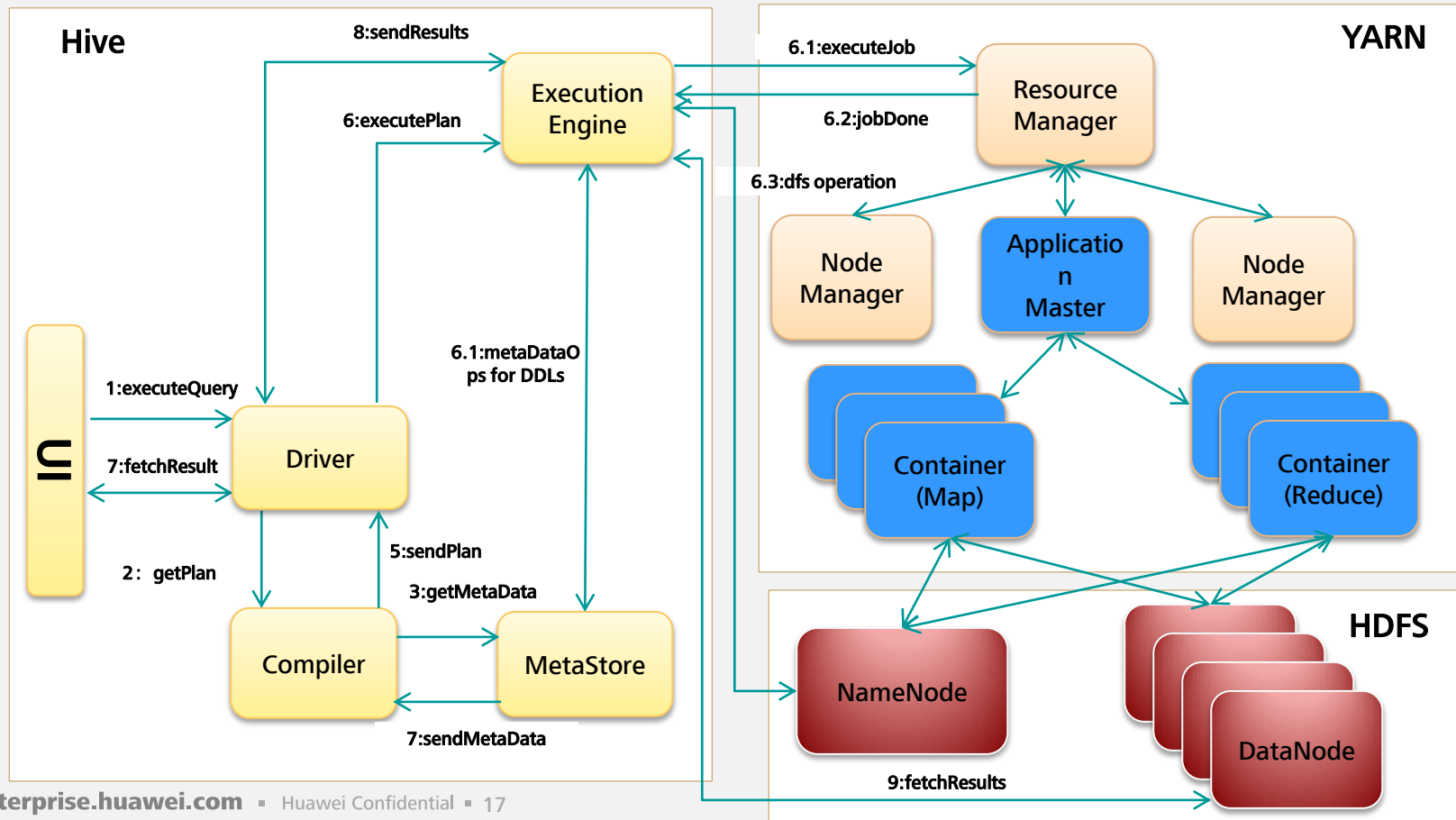
4

Hive开发接口介绍

5

Hive常用维护

任务执行流程





- Data Definition Statements
 - DDL Statements
 - Bucketed Tables
 - Indexes
 - Archiving
- Data Manipulation Statements
 - DML: Load
 - Import/Export
 - Data Retrieval: Queries
 - Select
 - Group By
 - Sort/Distribute/Cluster/Order By
 - Operators and User-Defined Functions (UDFs)
 - Joins
 - Join Optimization
 - Union
 - Sub Queries
 - Sampling
- Explain Execution Plan

Create Database:

```
CREATE (DATABASE|SCHEMA) [IF NOT EXISTS] database_name  
  [COMMENT database_comment]  
  [LOCATION hdfs_path]  
  [WITH DBPROPERTIES (property_name=property_value, ...)];
```

Drop Database:

```
DROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];
```

Use Database:

```
USE database_name;  
USE DEFAULT;
```

Create table:

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,  
    page_url STRING, referrer_url STRING,  
    ip STRING COMMENT 'IP Address of the User')  
COMMENT 'This is the page view table'  
PARTITIONED BY(dt STRING, country STRING)  
STORED AS SEQUENCEFILE;
```

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,  
    page_url STRING, referrer_url STRING,  
    ip STRING COMMENT 'IP Address of the User')  
COMMENT 'This is the page view table'  
PARTITIONED BY(dt STRING, country STRING)  
ROW FORMAT DELIMITED  
    FIELDS TERMINATED BY '\001'  
STORED AS SEQUENCEFILE;
```

Create table:

```
CREATE EXTERNAL TABLE page_view(viewTime INT, userid BIGINT,  
    page_url STRING, referrer_url STRING,  
    ip STRING COMMENT 'IP Address of the User',  
    country STRING COMMENT 'country of origination')  
COMMENT 'This is the staging page view table'  
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\054'  
STORED AS TEXTFILE  
LOCATION '<hdfs_location>';
```

```
CREATE TABLE new_key_value_store  
    ROW FORMAT SERDE "org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe"  
    STORED AS RCFile  
    AS  
SELECT (key % 1024) new_key, concat(key, value) key_value_pair  
FROM key_value_store  
SORT BY new_key, key_value_pair;
```

```
CREATE TABLE empty_key_value_store  
LIKE key_value_store;
```

Create table:

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,  
    page_url STRING, referrer_url STRING,  
    ip STRING COMMENT 'IP Address of the User')  
COMMENT 'This is the page view table'  
PARTITIONED BY(dt STRING, country STRING)  
CLUSTERED BY(userid) SORTED BY(viewTime) INTO 32 BUCKETS  
ROW FORMAT DELIMITED  
    FIELDS TERMINATED BY '\001'  
    COLLECTION ITEMS TERMINATED BY '\002'  
    MAP KEYS TERMINATED BY '\003'  
STORED AS SEQUENCEFILE;
```

Drop table:

```
DROP TABLE [IF EXISTS] table_name
```

Alter Table:

Rename Table

```
ALTER TABLE table_name RENAME TO new_table_name;
```

Alter Table Properties

```
ALTER TABLE table_name SET TBLPROPERTIES table_properties;
```

table_properties:

: (property_name = property_value, property_name = property_value, ...)

```
ALTER TABLE table_name SET TBLPROPERTIES ('comment' = new_comment);
```

Add SerDe Properties

```
ALTER TABLE table_name SET SERDE serde_class_name [WITH SERDEPROPERTIES  
serde_properties];
```

```
ALTER TABLE table_name SET SERDEPROPERTIES serde_properties;
```

serde_properties:

: (property_name = property_value, property_name = property_value, ...)

Loading files into tables

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

Inserting data into Hive Tables from queries

Standard syntax:

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]]  
select_statement1 FROM from_statement;
```

```
INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM  
from_statement;
```

Hive extension (multiple inserts):

```
FROM from_statement
```

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]]  
select_statement1
```

```
[INSERT OVERWRITE TABLE tablename2 [PARTITION ... [IF NOT EXISTS]] select_statement2]
```

```
[INSERT INTO TABLE tablename2 [PARTITION ...] select_statement2] ...;
```

```
FROM from_statement
```

```
INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1
```

```
[INSERT INTO TABLE tablename2 [PARTITION ...] select_statement2]
```

```
[INSERT OVERWRITE TABLE tablename2 [PARTITION ... [IF NOT EXISTS]] select_statement2] ...;
```

Inserting data into Hive Tables from queries

Hive extension (dynamic partition inserts):

```
INSERT OVERWRITE TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select_statement  
FROM from_statement;
```

```
INSERT INTO TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select_statement FROM  
from_statement;
```

Writing data into the filesystem from queries

Standard syntax:

```
INSERT OVERWRITE [LOCAL] DIRECTORY directory1  
[ROW FORMAT row_format] [STORED AS file_format] (Note: Only available starting with Hive 0.11.0)  
SELECT ... FROM ...
```

Hive extension (multiple inserts):

```
FROM from_statement
```

```
INSERT OVERWRITE [LOCAL] DIRECTORY directory1 select_statement1
```

```
[INSERT OVERWRITE [LOCAL] DIRECTORY directory2 select_statement2] ...
```

```
row_format
```

```
: DELIMITED [FIELDS TERMINATED BY char [ESCAPED BY char]] [COLLECTION ITEMS TERMINATED BY  
char]
```

```
[MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char]
```

```
[NULL DEFINED AS char] (Note: Only available starting with Hive 0.13)
```

Joins

```
SELECT a.* FROM a JOIN b ON (a.id = b.id)
```

```
SELECT a.* FROM a JOIN b ON (a.id = b.id AND a.department = b.department)
```

```
SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2)
```

Hive converts joins over multiple tables into a single map/reduce job if for every table the same column is used in the join clauses e.g.

```
SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key1)
```

is converted into a single map/reduce job as only key1 column for b is involved in the join. On the other hand

```
SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2)
```

Return Type	Name(Signature)	Description
DOUBLE	round(DOUBLE a)	Returns the rounded BIGINT value of a.
DOUBLE	round(DOUBLE a, INT d)	Returns a rounded to d decimal places.
BIGINT	floor(DOUBLE a)	Returns the maximum BIGINT value that is equal to or less than a.
BIGINT	ceil(DOUBLE a), ceiling(DOUBLE a)	Returns the minimum BIGINT value that is equal to or greater than a.
DOUBLE	abs(DOUBLE a)	Returns the absolute value.
INT or DOUBLE	positive(INT a), positive(DOUBLE a)	Returns a.
INT or DOUBLE	negative(INT a), negative(DOUBLE a)	Returns -a.
DOUBLE	e()	Returns the value of e.
DOUBLE	pi()	Returns the value of pi.

Return Type	Name(Signature)	Description
string	from_unixtime(bigint unixtime[, string format])	Converts the number of seconds from unix epoch (1970-01-01 00:00:00 UTC) to a string representing the timestamp of that moment in the current system time zone in the format of "1970-01-01 00:00:00".
bigint	unix_timestamp()	Gets current Unix timestamp in seconds.
bigint	unix_timestamp(string date)	Converts time string in format yyyy-MM-dd HH:mm:ss to Unix timestamp (in seconds), using the default timezone and the default locale, return 0 if fail: unix_timestamp('2009-03-20 11:30:01') = 1237573801
bigint	unix_timestamp(string date, string pattern)	Convert time string with given pattern (see [http://docs.oracle.com/javase/tutorial/i18n/format/simpleDateFormat.html]) to Unix time stamp (in seconds), return 0 if fail: unix_timestamp('2009-03-20', 'yyyy-MM-dd') = 1237532400.
string	to_date(string timestamp)	Returns the date part of a timestamp string: to_date("1970-01-01 00:00:00") = "1970-01-01".

Return Type	Name(Signature)	Description
int	year(string date)	Returns the year part of a date or a timestamp string: year("1970-01-01 00:00:00") = 1970, year("1970-01-01") = 1970.
int	month(string date)	Returns the month part of a date or a timestamp string: month("1970-11-01 00:00:00") = 11, month("1970-11-01") = 11.
int	day(string date) dayofmonth(date)	Returns the day part of a date or a timestamp string: day("1970-11-01 00:00:00") = 1, day("1970-11-01") = 1.
int	hour(string date)	Returns the hour of the timestamp: hour('2009-07-30 12:58:59') = 12, hour('12:58:59') = 12.
int	minute(string date)	Returns the minute of the timestamp.
int	second(string date)	Returns the second of the timestamp.
timestamp	from_utc_timestamp(timestamp, string timezone)	Assumes given timestamp is UTC and converts to given timezone (as of Hive 0.8.0). For example, from_utc_timestamp('1970-01-01 08:00:00', 'PST') returns 1970-01-01 00:00:00.
timestamp	to_utc_timestamp(timestamp, string timezone)	Assumes given timestamp is in given timezone and converts to UTC (as of Hive 0.8.0). For example, to_utc_timestamp('1970-01-01 00:00:00', 'PST') returns 1970-01-01 08:00:00.

Return Type	Name(Signature)	Description
string	<code>concat(string binary A, string binary B...)</code>	Returns the string or bytes resulting from concatenating the strings or bytes passed in as parameters in order. For example, <code>concat('foo', 'bar')</code> results in 'foobar'. Note that this function can take any number of input strings.
string	<code>concat_ws(string SEP, string A, string B...)</code>	Like <code>concat()</code> above, but with custom separator SEP.
int	<code>length(string A)</code>	Returns the length of the string.
string	<code>lower(string A)</code> <code>lcase(string A)</code>	Returns the string resulting from converting all characters of B to lower case. For example, <code>lower('fOoBaR')</code> results in 'foobar'.
string	<code>regexp_replace(string INITIAL_STRING, string PATTERN, string REPLACEMENT)</code>	Returns the string resulting from replacing all substrings in INITIAL_STRING that match the java regular expression syntax defined in PATTERN with instances of REPLACEMENT. For example, <code>regexp_replace("foobar", "oo ar", "")</code> returns 'fb.' Note that some care is necessary in using predefined character classes: using '\s' as the second argument will match the letter s; '\\s' is necessary to match whitespace, etc.

Key	Value	Description
mapred.reduce.tasks	Default Value: -1 Typically set to a prime close to the number of available hosts.	Hadoop set this to 1 by default, whereas Hive uses -1 as its default value. By setting this property to -1, Hive will automatically figure out what should be the number of reducers.
hive.map.aggr	Default Value: true	Whether to use map-side aggregation in Hive Group By queries.
hive.groupby.skewindata	Default Value: false	Whether there is skew in data to optimize group by queries.
hive.optimize.groupby	Default Value: true	Whether to enable the bucketed group by from bucketed partitions/tables.
hive.optimize.ppd	Default Value: true	Whether to enable predicate pushdown.
hive.join.cache.size	Default Value: 25000	How many rows in the joining tables (except the streaming table) should be cached in memory.
hive.optimize.skewjoin	Default Value: false	Whether to enable skew join optimization.

Key	Value	Description
hive.exec.compress.output	Default Value: false	This controls whether the final outputs of a query (to a local/hdfs file or a Hive table) is compressed. The compression codec and other options are determined from Hadoop configuration variables <code>mapred.output.compress*</code> .
hive.exec.compress.intermediate	Default Value: false	This controls whether intermediate files produced by Hive between multiple map-reduce jobs are compressed. The compression codec and other options are determined from Hadoop configuration variables <code>mapred.output.compress*</code> .
hive.exec.parallel	Default Value: false	Whether to execute jobs in parallel.
hive.exec.parallel.thread.number	Default Value: 8	How many jobs at most can be executed in parallel.
hive.merge.mapfiles	Default Value: true	Merge small files at the end of a map-only job.
hive.merge.mapredfiles	Default Value: false	Merge small files at the end of a map-reduce job.

Key	Value	Description
hive.auto.convert.join	Default Value: true	Whether Hive enables the optimization about converting common join into mapjoin based on the input file size.
hive.exec.dynamic.partition	Default Value: true	Whether or not to allow dynamic partitions in DML/DDL.
hive.exec.mode.local.auto	Default Value: false	Let Hive determine whether to run in local mode automatically.
hive.exec.drop.ignorenonexistent	Default Value: true	Do not report an error if DROP TABLE/VIEW specifies a non-existent table/view.
hive.mapred.mode	Default Value: nonstrict	The mode in which the Hive operations are being performed. In strict mode, some risky queries are not allowed to run.

Key	Value	Description
hive.metastore.server.max.threads	Default Value:100000	MetaStore内部线程池中能启动的，最大的用于处理连接的线程数
hive.metastore.server.min.threads	Default Value: 200	MetaStore启动的用于处理连接的线程数，如果超过设置的值之后，MetaStore就会一直维护不低于设定值的线程数，即常驻MetaStore线程池的线程会维护在指定值之上。
hive.exec.mode.local.auto.parallel.max	Default Value: 4	HiveServer在执行本地任务时可以并行的任务数量，如果本地任务之间没有依赖行，那么对于有很多本地任务的场景，该参数能加快任务的执行效率（因为并行化）
hive.server.session.control.maxconnections	Default Value:500	HiveServer支持的最大连接数
hive.server.session.control.maxconnection.peruser	Default Value: 500	每个用户独占HiveServer连接的数量，不能大于hive.server.session.control.maxconnections设置的值；另外，如果是多个用户，则多个用户共享hive.server.session.control.maxconnections设置的值，即所用用户使用的连接数的和不能超过hive.server.session.control.maxconnections设置的值。
hive.server.timewindow.delaytime	Default Value: 60	时间窗的时间间隔

Key	Value	Description
hive.server.timewindow.maxsessions.in.delaytime	Default Value:500	一个时间窗内，最多能允许的连接数
metastore.log.maxbackupindex	Default Value: 20	日志份数
metastore.log.filesize	Default Value: 20M	每份日志的大小
hive.server2.idle.session.timeout	Default Value:4320m	HiveServer空闲会话的超时时间，单位分钟
hive.server2.session.check.interval	Default Value: 3000ms	HiveServer会话心跳时间，单位毫秒
hive.server2.thrift.max.worker.threads	Default Value: 1000	HiveServer内部线程池，最大能启动的线程数量
hive.server2.thrift.min.worker.threads	Default Value: 5	HiveServer内部线程池，初识化时启动的线程数量
mapreduce.output.fileoutputformat.compress.type	Default Value: block	HiveServer提交给YARN的Job，如果输出文件的格式是SequenceFiles,那么该参数就控制在写SequenceFile时，是按数据大小还是按记录数量flush，默认是按数据大小，即BLOCK.
hive.server.close.wile.socket	Default Value: true	HiveServer的Socket异常时，是否关闭Server transport

- HA
- 基于HDFS Colocation特性建表
- 列加密
- HBase表批量记录删除功能
- 支持在HBase上建索引
- 流控特性
- 指定行分隔符
- 支持CSV Serde

FusionInsight HD提供的Hive支持双机冷备，即基于主备切换方式的服务器热备来保证Hive服务的高可用性，在同一时间内只有一个Hive Server运行，当主Hive Server出现故障无法继续提供服务时，备Hive Server会被激活，保证业务在短时间内完全恢复正常使用。

其基本原理如下：

通过Zookeeper的Master Election机制保证总有一个Hive Server正常运行并提供服务。主备两个HiveServer以Ephemeral方式分别注册到Zookeeper中，使得其中一个Hive Server出现故障时，能够及时选举并恢复服务。同时，Zookeeper中存储有主Hive Server的IP地址。Hive客户端可通过Zookeeper获取主Hive Server IP地址，来连接Hive Server获得服务。

在Hive双机环境下，需要知道Zookeeper的IP地址和端口才能获取主Hive Server的地址；在安全版本下，除了需要Zookeeper的IP地址和端口信息外，还需要连接Zookeeper的相关认证信息。

概要说明：

HDFS Colocation (同分布) 是HDFS提供的数据分布控制功能, 利用HDFS Colocation接口, 可以将存在关联关系或者可能进行关联操作的数据存放在相同的存储节点上。

Hive支持HDFS的Colocation功能, 即在创建Hive表时, 通过设置表文件分布的locator信息, 可以将相关表的数据文件存放在相同的存储节点上, 从而使后续的多表关联的数据计算更加方便和高效。

使用：

- 通过hdfs接口创建groupid
hdfs colocationadmin -createGroup -groupId groupId -locatorIds locatorid1,locatorid2,locatorid3
关于hdfs创建groupid的介绍可以参看hdfs的相关说明, 这里不做赘述。
- Hive使用colocation
假设tbl_1和tbl_2是相关联的两张表, 创建两表的语句如下:
CREATE TABLE tbl_1 (id INT, name STRING) stored as RCFILE
TBLPROPERTIES("groupId"="group1","locatorId"="locator1");
CREATE TABLE tbl_2 (id INT, name STRING) row format delimited fields terminated by '\t' stored as TEXTFILE **TBLPROPERTIES("groupId"="group1","locatorId"="locator1");**

约束：

- 1、必须使用insert语句分别向该类型表导入数据, HDFS Colocation特性才能生效。
- 2、文件格式仅支持TEXTFile和RCFile。

概要说明：

Hive支持对表的某一列或者多列就行加密；在创建hive表时，可以指定要加密的列和加密算法。
当使用insert语句向表中插入数据时，即可实现将对应列加密。

Hive列加密机制目前支持的加密算法有两种：

- (1) AES (对应加密类名称为：org.apache.hadoop.hive.serde2.AESRewriter)
- (2) SMS4 (对应加密类名称为：org.apache.hadoop.hive.serde2.SMS4Rewriter)

使用：

- 1、在创建表时指定相应的加密列和加密算法

```
create table encode_test (id INT, name STRING, phone STRING, address STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
    'column.encode.columns'='phone,address',
    'column.encode.classname'='org.apache.hadoop.hive.serde2.AESRewriter')
STORED AS TEXTFILE;
```

- 2、使用insert语法向设置列加密的表中导入数据（假设test表已存在且有数据）

```
insert into table encode_test select id, name, phone, address from test;
```

约束：

- 1、Serde必须使用 “org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe”
- 2、必须使用insert语句分别向该类型表导入数据，列加密特性才能生效
- 3、文件格式仅支持TEXTFile和SequenceFile

概要说明：

社区0.13版本的Hive并不能支持对单条表数据进行删除操作，但在Hive on HBase功能中，FusionInsight HD Hive提供了对HBase表的单条数据的删除功能，通过特定的语法，Hive可以将自己的HBase表中符合条件的一条或者多条数据清除。

使用：

如果要删除某张HBase表中的某些数据，可以执行HQL语句：

```
remove table hbase_table where expression;
```

其中expression规定要删除数据的筛选条件。

概要说明：

Hive中可以给表建立索引，记录表中键值对应表文件及在文件中的偏移量，从而提高 Hive表的单点查询效率。

但Hive索引具有一定的局限性：以Compact索引为例，为对应表创建的索引是索引字段名、文件、偏移址这三个字段组成的表。这就导致了首先索引会占用大量的存储空间，根据测试，表中一个108G的分区，索引需要32G左右空间。其次，这些数据以Hive表的形式保存，在查询时，遍历索引会消耗一定时间，使得在很多场景下索引带来的效果并不明显，甚至会使查询效率降低。

FusionInsight HD Hive提供了为Hive数据表创建HBase索引的功能，即创建索引时指定索引生成的表为HBase表，将索引数据存放在HBase中，从而大大提高索引的查询效率。

使用：

1、创建索引表

```
create index index_ransom on table hedata(imsi)
as 'org.apache.hadoop.hive.ql.index.hbase.HBaseIndexHandler'
with deferred rebuild
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
TBLPROPERTIES (
    'hbase.table.name'='index_ransom',
    'hbase.composite.key.factory'=
    'org.apache.hadoop.hive.hbase.HBaseBinaryKeyFactory',           --生成rowkey的工厂类
    'index.column.dispatcher.expression'='substr(imsi,length(imsi))', --生成rowkey的某个值
    'index.column.dispatcher.values'='0,1,2,3,4,5,6,7,8,9');        --划分region的枚举值
```

2、更新索引信息

```
alter index index_ransom on hedata rebuild;
```

- (1) 当前已经建立的总连接数阈值控制；
- (2) 每个用户已经建立的连接数阈值控制；
- (3) 单位时间内所建立的连接数阈值控制。

Services > Hive Configuration >

Status Instances **Configuration** Resource Distribution

Save Configuration Import Service Configuration Export Service Configuration Type: All

Hive

- HiveServer
 - HiveHA
 - JVM
 - Log
 - MetaDB
 - MRClient
 - Performance**
 - Security
 - ServerInit

Parameter	Value	
hive.default.fileformat	<input type="checkbox"/> TextFile <input type="checkbox"/> SequenceFile <input checked="" type="checkbox"/> RCFile	[Desc] Format of tal
hive.exec.compress.output	<input type="radio"/> true <input checked="" type="radio"/> false	[Desc] This controls
hive.exec.reducers.max	999	[Desc] Maximum nu
hive.mapred.mode	<input checked="" type="checkbox"/> nonstrict <input type="checkbox"/> strict	[Desc] Mode in whi
hive.server.close.while.socketexception	<input checked="" type="radio"/> true <input type="radio"/> false	[Desc] Specifies wt
hive.server.session.control.enable	<input checked="" type="radio"/> true <input type="radio"/> false	[Desc] Specifies wt
hive.server.session.control.maxconnection.pe...	800	[Desc] Maximum nu
hive.server.session.control.maxconnections	800	[Desc] Maximum nu
hive.server.timewindow.delaytime	60	[Desc] Delay time (
hive.server.timewindow.maxsessions.in.delay...	200	[Desc] Maximum nu

概要说明：

通常情况下，Hive以文本文件存储的表会以回车作为其行分隔符，即在查询过程中，以回车符作为一行表数据的结束符。

但某些数据文件并不是以回车分隔的规则文本格式，而是以某些特殊符号分割其规则文本。

FusionInsight HD Hive支持指定不同的字符或字符组合作为hive文本数据的行分隔符，既在创建表的时候，指定inputformat为SpecifiedDelimiterInputFormat，然后在每次查询前，都设置如下参数，来指定分隔符。*set hive.textinput.record.delimiter= '\${DELIMITER}';*

使用：

- 1、创建表时指定inputFormat和outputFormat

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name  
[(col_name data_type [COMMENT col_comment], ...)]  
[ROW FORMAT row_format]  
STORED AS  
inputformat 'org.apache.hadoop.hive.contrib.fileformat.SpecifiedDelimiterInputFormat'  
outputformat 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat';
```

- 2、查询之前指定配置项

```
set hive.textinput.record.delimiter='!@!';
```

- 3、则hive会以 '!@!' 为行分隔符查询数据。

概要说明：

CSV是一种常见的文本文件格式，其文件以纯文本形式存储表格数据（数字和文本），并以逗号作为文本分隔符。

CSV文件具有较强的通用性，有许多应用程序允许用户查看和编辑CSV文件，可以方便的在windows office或者一些传统数据库中使用。

FusionInsight HD Hive增加了对CSV文件的支持，可以将用户的CSV文件导入hive表中，或者将用户的hive表数据以CSV文件格式导出，从而方便在其他应用中使用。

使用：

1、创建CSV格式的表

```
create table tabname
(
  clt_nbr string,
  crd_act_nbr1 smallint,
  crd_nbr string,
  crd_blk_cod string,
  crd_blk_cod_mem string
)
row format serde 'com.bizo.hive.serde.csv.CSVSerde'
stored as textfile;
```

2、将CSV格式的文件导入到创建的表中，或者以insert方式向表中插入数据。

1

Hive应用场景

2

Hive功能与架构

3

Hive关键流程

4

Hive开发接口介绍

5

Hive常用维护

- **Beeline**
- **JDBC**
- **Python Client**

HiveServer2提供一个新的命令行工具，基于SQLLine的JDBC客户端。

Command	Description
!quit !exit	Use quit or exit to leave the interactive shell.
set <key>=<value>	Sets the value of a particular configuration variable (key). Note: If you misspell the variable name, the CLI will not show an error.
dfs <dfs command>	Executes a dfs command from the Hive shell.
<query string>	Executes a Hive query and prints results to standard output.

- Sample Usage:**

```
beeline> set mapred.reduce.tasks=32;  
beeline > select * from tab1;  
beeline > dfs -ls /;
```

参数介绍：

格式	语义
-e <query>	执行一条语句，只能是一条
-f <file>	执行一个文件，若文件中某一行有错，以后的都不再执行
--hivevar name=value	设置当前连接可以使用的变量
--color=[true/false]	以绿色显示表格的框，以红色显示错误
--showHeader=[true/false]	是否显示表头
--headerInterval=ROWS	隔多少行显示一下表头
--fastConnect=[true/false]	按tab键提示表名和表的列

参数介绍：

格式	语义
--verbose=[true/false]	显示详细异常堆栈日志信息
--maxWidth=MAXWIDTH	不指定maxWidth时可能导致结果显示不全
--silent=[true/false]	显示beeline的信息和执行的时间，返回的行数
--autosave=[true/false]	自动保存用户设置的属性，下次启动beeline时会读取
--outputformat=[table/vertical/csv/tsv]	指定查询结果的输出格式，默认是table
--nullemptystring=[true/false]	False时字段为空时显示为NULL
--help	显示帮助信息

```
# beeline
scan complete in 3ms
Connecting to jdbc:hive2://ha-
cluster/default;zk.quorum=51.196.0.34,51.196.0.35,51.196.0.36;zk.port=24002;sasl.qop=auth-
conf;auth=KERBEROS;principal=hive/hadoop.huawei.com@huawei.com
Debug is true storeKey false useTicketCache true useKeyTab false doNotPrompt false ticketCache is
null isInitiator true KeyTab is null refreshKrb5Config is false principal is null tryFirstPass is false
useFirstPass is false storePass is false clearPass is false
Acquire TGT from Cache
Principal is hive/hadoop.huawei.com@huawei.com
Commit Succeeded
```

```
Connected to: Apache Hive (version 0.13.1)
Driver: Hive JDBC (version 0.13.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 0.13.1 by Apache Hive
0: jdbc:hive2://ha-cluster/default> show tables;
```

```
+-----+
| tab_name |
+-----+
+-----+
```

确认待连接的集群是否为安全版本。本例中连接的集群为安全版本。

// 所连接的集群是否为安全版本

boolean isSecureVer = true;设置krb5文件路径。本例中krb5文件为Eclipse工程HiveExample的“conf/krb5.conf”。

// 设置krb5文件路径

System.setProperty("java.security.krb5.conf", "conf/krb5.conf");

设置ZooKeeper的IP列表和端口，IP以“,”分隔。本例中ZooKeeper部署在三个节点上，使用的端口为“2181”。

// Hive Server为HA模式，指定Zookeeper的ip和端口号来查询当前主HiveServer

// zkQuorum为集群中Zookeeper所在节点的IP

String zkQuorum = "10.1.131.17,10.1.131.18,10.1.131.19";

// zkPort为集群中Zookeeper使用的端口号

String zkPort = "2181";

定义HQL。HQL必须为单条语句，注意HQL不能包含“;”。

// 定义HQL，HQL为单条语句，不能包含“;”

String sql = "SELECT COUNT(*) FROM employees_info";

拼接JDBC URL。如果是安全版本需要拼接kerberos用户以及keytab文件路径等信息。

```
// 拼接JDBC URL
StringBuilder sBuilder = new StringBuilder(
    "jdbc:hive2://ha-cluster/default;zk.quorum=").append(zkQuorum)
    .append(";zk.port=").append(zkPort);
if (isSecureVer) {
    // 客户端的kerberos用户
    String userPrincipal = "hive/hadoop";

    // 客户端的keytab文件路径
    String userKeyTab = "conf/hive.keytab";

    sBuilder.append(";user.principal=")
        .append(userPrincipal)
        .append(";user.keytab=")
        .append(userKeyTab)
        .append(";sas.l.qop=auth-
conf;auth=KERBEROS;principal=hive/hadoop@HADOOP.COM;zk.principal=zookeeper/hadoop");
}
String url = sBuilder.toString();
```

加载Hive JDBC驱动。

// 加载Hive JDBC驱动

Class.forName("org.apache.hive.jdbc.HiveDriver").newInstance();获取JDBC连接，执行HQL，输出查询的列名和结果到控制台，关闭JDBC连接。

Connection connection = null;

PreparedStatement statement = null;

ResultSet resultSet = null;

ResultSetMetaData resultMetaData = null;

```
try {
    connection = DriverManager.getConnection(url, "", "");
    statement = connection.prepareStatement(sql);
    resultSet = statement.executeQuery();
    resultMetaData = resultSet.getMetaData();
    int columnCount = resultMetaData.getColumnCount();
    for (int i = 1; i <= columnCount; i++) {
        System.out.print(resultMetaData.getColumnLabel(i) + '\t');
    }
    System.out.println();
    while (resultSet.next()) {
        for (int i = 1; i <= columnCount; i++) {
            System.out.print(resultSet.getString(i) + '\t');
        }
        System.out.println();
    }
}finally {
    if (null != statement) { statement.close(); }
    if (null != connection) { connection.close(); }
}
```



```
from pyhs2.haconnection import HAConnection
hosts = ["10.1.131.17", "10.1.131.19"]
conf = {"krb_host":"hadoop", "krb_service":"hive"}
try:
    with HAConnection(hosts = hosts,
                      port = 10000,
                      authMechanism = "KERBEROS",
                      configuration = conf) as haConn:
        with haConn.getConnection() as conn:
            with conn.cursor() as cur:
                print cur.getDatabases()
                cur.execute("SELECT COUNT(*) FROM employees_info")
                print cur.getSchema()
                for i in cur.fetch():
                    print i
except Exception, e:
    print e
```

1

Hive应用场景

2

Hive功能与架构

3

Hive关键流程

4

Hive开发接口介绍

5

Hive常用维护

安装阶段，依次调用如下脚本，如果安装失败，可以从以下日志中排查：

a、hive-clean.sh

- 删除DBService里面已经建立好的Hive元数据表
- 删除Zookeeper里面的Hive节点
- 脚本日志 /var/log/Bigdata/hive/cleanupDetail.log
- 程序执行日志 /var/log/Bigdata/hive/hive.log
- 初次安装集群时，由于DBService、Zookeeper均未启动，所以在hive.log中看到是执行错误的日志，不影响hive的安装

b、hive-postinstall.sh

- 创建hiveinstallmarker文件，标记Hive已经安装完成
- 脚本日志 /var/log/Bigdata/hive/postinstallDetail.log

c、hive-prestart.sh

- 启动Hive前调用的脚本，会检查是否有hiveinstallmarker文件，如果有，则在DBService中创建hive元数据表
- 脚本日志 /var/log/Bigdata/hive/prestartDetail.log
- 程序执行日志 /var/log/Bigdata/hive/hive.log
- 由于有两个HiveServer节点，只有后启动的这个HiveServer会检查是否有hiveinstallmarker文件，并进行建表操作，先启动的HiveServer会将hiveinstallmarker文件删除，并进行后续查询操作

启动阶段，依次调用如下脚本

a、hive hivestop (停止/重启时会调用)

- 检查pid文件是否存在，并用kill命令杀死pid文件中记录的进程号对应的进程
- 脚本日志 /var/log/Bigdata/hive/stopDetail.log

b、hive-prestart.sh

c、hive hiveserver

卸载阶段，依次调用如下脚本

a、hive hivestop

b、hive-clean.sh

- 检查是否是Zookeeper连接超时导致，若是，请查看Zookeeper的日志；
- 检查是否是网络连接失败导致，请确认主HiveServer与Zookeeper各个节点的连接是否正常，若否，请查看网络连接是否正常；
- 检查是否是Hive响应慢导致，若是，请具体分析/var/log/Bigdata/hive/hive.log，确认HiveServer进程是否在进行FullGC，可以查看/var/log/Bigdata/hive/hive-omm-gc.log日志；或者确认是否死锁导致，这个需要通过jstack命令dump出文件进行确认；
- 查看/var/log/Bigdata/hive/hive.log 日志。HiveServer每次进行主备倒换时，都会打印相关日志，例如此外还需要查看最近的SQL是否执行成功；

- 一、在OMM上查看Services，如果发现Hive的HealthStatus为“**Bad**”（下图为“Good”，说明Hive服务可用），说明Hive服务不可用，可以根据如下步骤进行排查：
- 检查Hive服务依赖的其他组件服务是否可用，包括：LdapServer,KrbServer,HDFS,MapReduce和Zookeeper;
 - 检查是否有Hive服务的告警产生，查看告警描述；
 - 如果以上都没有问题，则需要具体去分析HiveServer的运行日志，如何分析，可以参考后面的“Hive典型问题”

Service	Operating Status	Health Status	Configuration Status	
BookKeeper	Started	Good	Synchronized	3 BookieServer
DBService	Started	Good	Synchronized	2 DBServer
FTP-Server	Started	Good	Synchronized	3 FTP-Server
HBase	Started	Good	Synchronized	2 HMaster , 3 ThriftServer , 3 RegionServer
HDFS	Started	Good	Synchronized	2 Zkfc , 2 NameNode , 3 DataNode
Hive	Started	Good	Synchronized	2 HiveServer
Hue	Started	Good	Synchronized	1 Hue
KrbServer	Started	Good	Synchronized	2 KerberosServer , 2 KerberosAdmin
LdapServer	Started	Good	Synchronized	2 SlapdServer
Loader	Started	Good	Synchronized	2 LoaderServer
MapReduce	Started	Good	Synchronized	1 JobHistoryServer , 3 NodeManager , 1 ProxyServer ,
ZooKeeper	Started	Good	Synchronized	3 quorumpeer

二、在OMM上查看Services，如果发现Hive的HealthStatus为“Concerning”，说明HiveServer主备进程有问题，Hive服务不可用，基本原因是OMM的Controller已经将HiveServer拉起来，但是健康检查现场去连接HiveServer的时候，发现该HiveServer进程不存在。出现该问题时，可以根据如下步骤进行排查：

- 登录出现该“Concerning”状态的HiveServer所在的那个节点（可能是主，也可能是备），使用ps查看HiveServer进程是否存在：
- `>ps -ef|grep HiveServer`
- 如果不存在，检查/var/log/Bigdata/hive/postinstallDetail.log、/var/log/Bigdata/hive/startDetail.log等日志中是否有错误信息；
- 如果没有，则查看hive.log文件，由于连接不上DBService、HDFS、Zookeeper等而无法启动时，均可以在此处查看到相应日志。
- 如果仍然没有日志，则说明进程完全没有启动起来，可以查看nodeagent日志
/var/log/Bigdata/nodeagent/agentlog/agent.log，Hive启动过程中，输出到console上的内容，会记录在此日志中。



HUAWEI ENTERPRISE ICT SOLUTIONS **A BETTER WAY**

Copyright©2013 Huawei Technologies Co., Ltd. All Rights Reserved.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.