Hadoop 集群（第 15 期）

——HBase、Hive 与 RDBMS

1、Hive与HBase

1.1 Hive与HBase区别

Hive 是基于 **Hadoop** 的一个**数据仓库工具**，是为**简化编写 MapReduce** 程序**而生**的，使用 MapReduce 做过数据分析的人都知道，很多分析程序**除业务逻辑不同外**，**程序流程基本一样**。在这种情况下就需要 Hive 这样的用户编程接口。**Hive 本身不存储**和**计算数据**，它**完全依赖 HDFS** 和 **MapReduce**，**Hive** 中的**表**是**纯逻辑表**，就是些**表的定义**等，也就是**表的元数据**，这样就可以将**结构化**的**数据文件映射**为为**一张数据库表**，并提供完整的 SQL 查询功能，并将 SQL 语句最终转换为 MapReduce 任务进行运行。之所以使用 SQL 实现 Hive，是因为 SQL 大家都熟悉，转换成本低，不必开发专门的 MapReduce 应用，类似作用的 Pig 就不是 SQL。**Hive 十分适合**数据仓库的**统计分析**。

HBase 是一个**分布式**的、**面向列**的**开源数据库**，该技术来源于 Google 的论文“Bigtable：一个结构化数据的分布式存储系统”。就像 Bigtable 利用了 Google 文件系统所提供的分布式数据存储一样，HBase 在 Hadoop 之上提供了类似于 Bigtable 的能力，**HBase** 不同于一般的关系数据库，它是一个**适合于非结构化数据存储**的**数据库**。另一个不同的是 **HBase 基于列的**而不是基于行的模式。HBase 使用和 Bigtable 非常相同的数据模型，用户存储数据行在一个表里，一个数据行拥有一个可选择的键和任意数量的列。**表**是**疏松**的存储的，因此用户可以给行定义各种不同的列。

HBase 为**查询而生**的，它通过组织起节点内所有机器的内存，提供一个超大的内存 Hash 表，它需要**组织自己**的**数据结构**，**包括磁盘**和**内存**中的，而 **Hive** 是**不做这些**的，表在 **HBase** 中是**物理表**，而**不是逻辑表**，**搜索引擎**使用它来**存储索引**，以满足查询的实时性需求。**HBase 主要用于**需要**随机访问，实时读写**你的**大数据**（Big Data）。

1.2 Hive与HBase整合

1）集群环境

Java 版本：jdk-**6u31**-windows-i586.exe

Win 系统：Windows 7 旗舰版

Eclipse 软件：eclipse-jee-**indigo**-SR1-win32.zip | eclipse-jee-**helios**-SR2-win32.zip

Hadoop 软件：hadoop-1.0.0.tar.gz

HBase 软件：hbase-0.92.0.tar.gz

Hive 软件：hive-0.8.1.tar.gz

2）开场简介

Hive 与 HBase 的整合功能的实现是利用两者本身对外的 API 接口互相进行通信，相互通信主要是依靠 hive_hbase-handler.jar 工具类， 大致意思如图所示：

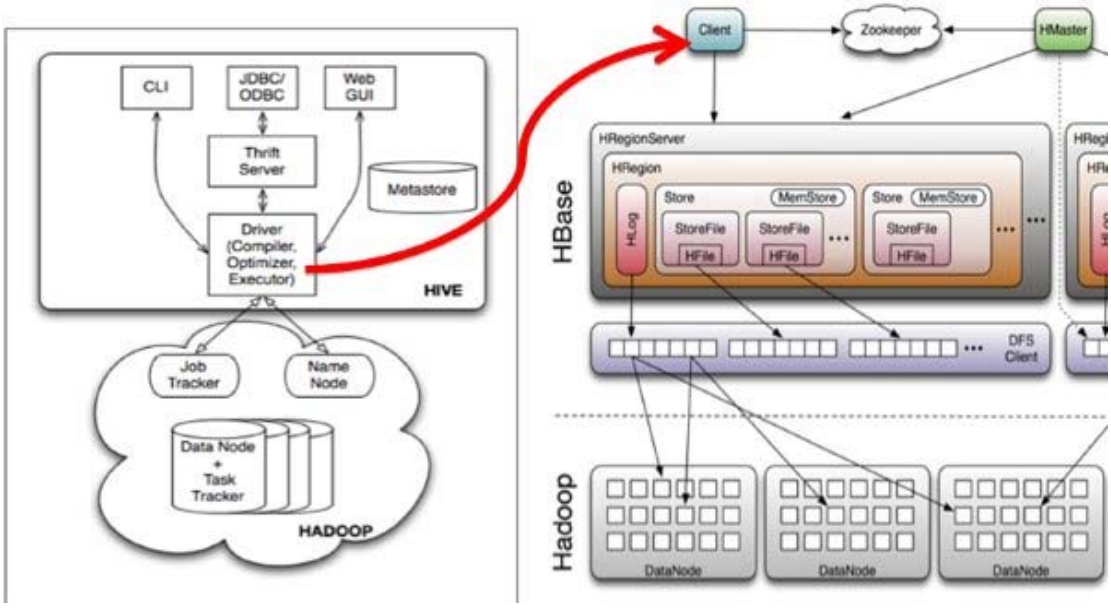


图 1.2-1 Hive 与 HBase 整合

备注：需要的软件有 Hadoop、Hive、Hbase、Zookeeper，Hive 与 HBase 的整合对 Hive 的版本有要求，Hive.0.6.0 的版本才支持与 HBase 对接，在 Hive 的 lib 目录下可以看见有个 hive-hbase-handler.jar 这个 jar 包，它是 Hive 扩展存储的 Handler。

3）整合步骤

- **第一步：整合前提**

在整合前必须保证已经正确安装 Hadoop、HBase 以及 Hive，如果没有安装，参考下面文章：

Hadoop 集群安装：Hadoop 集群_第 5 期_Hadoop 安装配置

HBase 集群安装：Hadoop 集群_第 11 期_HBase 简介及安装

Hive 软件安装：Hadoop 集群_第 13 期_Hive 简介及安装

- **第二步：准备 Jar 包**

在 Hive 的安装目录的 lib 子目录中，已经存在了 HBase 和 ZooKeeper 的相关 Jar，但是版本不是很一致，需要把 HBase 安装目录中的相关 Jar 拷贝到 Hive 的 lib 下面。

原 Hive 下面的相关 Jar 包版本：

```
[hadoop@Master lib]$ ll | grep hbase
-rw-rw-r--. 1 hadoop hadoop 2158756 11月  5 2010 hbase-0.89.0-SNAPSHOT.jar
-rw-rw-r--. 1 hadoop hadoop  842077 11月  5 2010 hbase-0.89.0-SNAPSHOT-tests.jar
-rw-rw-r--. 1 hadoop hadoop   48829  1月 26 08:16 hive-hbase-handler-0.8.1.jar
[hadoop@Master lib]$ ll | grep zookeeper
-rw-rw-r--. 1 hadoop hadoop  596183  5月  8 2010 zookeeper-3.3.1.jar
[hadoop@Master lib]$
```

HBase 的相关 Jar 包版本：

```
[hadoop@Master hbase]$ ll | grep jar
-rw-r--r--. 1 hadoop hadoop 3103099  1月 16 23:03 hbase-0.92.0.jar
-rw-r--r--. 1 hadoop hadoop 1483740  1月 16 23:03 hbase-0.92.0-tests.jar
[hadoop@Master hbase]$ ll lib | grep zookeeper
-rw-r--r--. 1 hadoop hadoop  764555  1月 16 23:03 zookeeper-3.4.2.jar
[hadoop@Master hbase]$
```

用下面命令进行拷贝相关的 HBase 的 Jar 包。（**备注**：用 **hadoop** 用户进行操作）

```
cp　/usr/hbase/hbase-0.92.0.jar　/usr/hive/lib
cp　/usr/hbase/hbase-0.92.0-tests.jar　/usr/hive/lib
cp　/usr/hbase/lib/zookeeper-3.4.2.jar　/usr/hive/lib
```

重点：重新编译“hive-hbase-handler”这个 Jar 包
在网上下了一个已经编译好的。
参考文献：http://blog.csdn.net/fullofwindandsnow/article/details/7331403
下载地址：http://download.csdn.net/download/xia520pi/4177324

接着，把“**/usr/hive/lib**”下面的几个**旧**的**删除**或者**重命名**，我们这里采取重命名。

```
[hadoop@Master lib]$ ll | grep hbase
-rw-rw-r--. 1 hadoop hadoop 2158756 11月  5 2010 hbase-0.89.0-SNAPSHOT.jar
-rw-rw-r--. 1 hadoop hadoop  842077 11月  5 2010 hbase-0.89.0-SNAPSHOT-tests.jar
-rw-r--r--. 1 hadoop hadoop 3103099  3月 27 05:56 hbase-0.92.0.jar
-rw-r--r--. 1 hadoop hadoop 1483740  3月 27 05:56 hbase-0.92.0-tests.jar
-rw-rw-r--. 1 hadoop hadoop   48829  1月 26 08:16 hive-hbase-handler-0.8.1.jar
-rw-r--r--. 1 hadoop hadoop   48958  3月 27 21:30 hive-hbase-handler-0.9.0-SNAPSHOT.jar
[hadoop@Master lib]$ mv hbase-0.89.0-SNAPSHOT.jar hbase-0.89.0-SNAPSHOT.jar.back
[hadoop@Master lib]$ mv hbase-0.89.0-SNAPSHOT-tests.jar hbase-0.89.0-SNAPSHOT-tests.jar.back
[hadoop@Master lib]$ mv hive-hbase-handler-0.8.1.jar hive-hbase-handler-0.8.1.jar.back
[hadoop@Master lib]$ ll | grep hbase
-rw-rw-r--. 1 hadoop hadoop 2158756 11月  5 2010 hbase-0.89.0-SNAPSHOT.jar.back
-rw-rw-r--. 1 hadoop hadoop  842077 11月  5 2010 hbase-0.89.0-SNAPSHOT-tests.jar.back
-rw-r--r--. 1 hadoop hadoop 3103099  3月 27 05:56 hbase-0.92.0.jar
-rw-r--r--. 1 hadoop hadoop 1483740  3月 27 05:56 hbase-0.92.0-tests.jar
-rw-rw-r--. 1 hadoop hadoop   48829  1月 26 08:16 hive-hbase-handler-0.8.1.jar.back
-rw-r--r--. 1 hadoop hadoop   48958  3月 27 21:30 hive-hbase-handler-0.9.0-SNAPSHOT.jar
```

```
-rw-rw-r--. 1 hadoop hadoop  596183  5月  8 2010 zookeeper-3.3.1.jar
-rw-r--r--. 1 hadoop hadoop  764555  3月 27 05:57 zookeeper-3.4.2.jar
[hadoop@Master lib]$ mv zookeeper-3.3.1.jar zookeeper-3.3.1.jar.back
[hadoop@Master lib]$ ll
```

- **第三步：修改 hive-site.xml**

```
<property>
    <name>hive.aux.jars.path</name>
    <value>file:///usr/hive/lib/hive-hbase-handler-0.9.0-SNAPSHOT.jar,file:///usr/hive/lib/hbase-0.92.0.jar,file:///usr/hive/lib/zookeeper-3.4.2.jar</value>
</property>

<property>
    <name>hbase.zookeeper.quorum</name>
    <value>Slave1.Hadoop,Slave2.Hadoop,Slave3.Hadoop</value>
</property>
```

备注：“**hive.aux.jars.path**”的**值**中间**必须连接**，**不能**有**空格**，不然在执行 Hive 查询操作时会出错。

```
        <property>
          <name>hive.hwi.listen.port</name>
          <value>9999</value>
        </property>
        <property>
          <name>hive.aux.jars.path</name>
          <value>file:///usr/hive/lib/hive-hbase-handler-0.9.0-SNAPSHOT.jar,file:///usr/hive/lib/hbase-0.92.0.jar,
file:///usr/hive/lib/zookeeper-3.4.2.jar</value>
        </property>
        <property>
          <name>hbase.zookeeper.quorum</name>
          <value>Slave1.Hadoop,Slave2.Hadoop,Slave3.Hadoop</value>
        </property>
</configuration>
"hive-site.xml" 45L, 1290C 已写入
[hadoop@Master conf]$
```

备注：当出现下面错误信息时，请执行下面操作，**我们**这里**没有出现**，所以**没有执行**，留着以备无患。

当运行 hive 时，出现如下错误：

org.apache.hadoop.hbase.**ZooKeeperConnectionException**: HBase is able to connect to ZooKeeper but the connection closes immediately. This could be a sign that the server has too many connections (30 is the default). Consider inspecting your ZK server logs for that error and then make sure you are reusing HBaseConfiguration as often as you can. See HTable's javadoc for more information. at org.apache.hadoop. hbase.zookeeper.ZooKeeperWatcher.

解决方案：拷贝 hbase-0.92.0.jar 到所有 hadoop 节点（包括 master）的 hadoop/lib 下，拷贝 hbase/conf 下的 hbase-site.xml 文件到所有 hadoop 节点（包括 master）的 hadoop/conf 下。

4）启动 Hive

- **单个 hbase 启动**

hive　-hiveconf hbase.master=**Master.Hadoop**:60000

- **hbase 集群启动**

hive　-hiveconf hbase.zookeeper.quorum=Slave1.Hadoop,Slave2.Hadoop,Slave3.Hadoop

```
[hadoop@Master ~]$ hive -hiveconf hbase.zookeeper.quorum=Slave1.Hadoop,Slave2.Hadoop,Slave3.Hadoop
Logging initialized using configuration in file:/usr/hive/conf/hive-log4j.properties
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201203271806_1331585161.txt
hive>
```

备注：先保证 Hadoop 集群和 HBase 集群已经启用。

如果 **hive-site.xml** 文件中**没有**配置 **hive.aux.jars.path**，则可以按照如下方式启动。

bin/hive　--auxpath　/usr/hive/lib/hive-hbase-handler-0.8.1.jar, /usr/hive/lib/hbase-0.92.0.jar, /usr/hive/lib/zookeeper-3.4.2.jar　-hiveconf　hbase.zookeeper.quorum=Slave1.Hadoop, Slave2.Hadoop,Slave3.Hadoop

```
[hadoop@Master ~]$ hive --auxpath /usr/hive/lib/hive-hbase-handler-0.9.0-SNAPSHOT.jar,/usr/hive/lib/hbase-0.92.0.jar,/usr/hive/lib/zo
okeeper-3.4.2.jar  -hiveconf hbase.zookeeper.quorum=Slave1.Hadoop,Slave2.Hadoop,Slave3.Hadoop
Logging initialized using configuration in file:/usr/hive/conf/hive-log4j.properties
Hive history file=/tmp/hadoop/hive_job_log_hadoop_201203280037_1420766928.txt
```

备注：这条命令要一气合成，中间不能有回车换行，而是自动换行。

5）举例测试

- **创建 hbase 识别的数据库**

```
CREATE TABLE hbase_table_1(key int, value string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cf1:val")
TBLPROPERTIES ("hbase.table.name" = "xyz");
```

备注：

hbase.table.name 定义在 hbase 的 table 名称

hbase.columns.mapping 定义在 hbase 的列族

英文原文：

The **hbase.columns.mapping** property is required and will be explained in the next section. The **hbase.table.name** property is optional; it controls the name of the table as known by HBase, and allows the Hive table to have a different name. In this example, the table is known as **hbase_table_1** within **Hive**, and as **xyz** within **HBase**. If not specified, then the Hive and HBase table names will be identical.

Hive 端查看结果：

```
hive> create table hbase_table_1(key int,value string)
    > stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > with serdeproperties ("hbase.columns.mapping"=":key,cf1:val")
    > tblproperties ("hbase.table.name"="xyz");
OK
Time taken: 4.271 seconds
hive> show tables;
OK
btest2
choice
hbase_table_1
loginfo
mutil2
ptest
testhivedrivertable
userinfo
xp
Time taken: 0.094 seconds
```

HBase 端查看结果：

```
[hadoop@Master lib]$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.92.0, r1231986, Mon Jan 16 13:16:35 UTC 2012

hbase(main):001:0> list
TABLE
heroes
heroes-name
heroes-power
scores
wordcount
xyz
6 row(s) in 0.4900 seconds
```

备注：如果出现如下错误，是因为 hive-hbase-handler 的版本与 hbase 和 hive 的不一致造成的，且记得**重命名**或**删除旧**的版本。

```
hive> create table hbase_table_1(key int,value string)
    > stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > with serdeproperties ("hbase.columns.mapping"=":key,cf1:val")
    > tblproperties ("hbase.table.name"="xyz");
FAILED: Error in metadata: java.lang.IllegalArgumentException: Not a host:port pair: ♦14507@Master.HadoopMaster.Hadoop,60000,1332851145712
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask
```

- **查看各表的描述**

Hive 端查看结果：

```
hive> describe hbase_table_1;
OK
key     int     from deserializer
value   string  from deserializer
Time taken: 0.059 seconds
hive>
```

HBase 端查看结果：

```
hbase(main):003:0> describe 'xyz'
DESCRIPTION                                                                                                  ENABLED
 {NAME => 'xyz', FAMILIES => [{NAME => 'cf1', BLOOMFILTER => 'NONE', REPLICATION_SCOPE => '                  true
 0', VERSIONS => '3', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => '2147483647', BLOC
 KSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}]}
1 row(s) in 0.0330 seconds
```

- **使用 sql 导入数据**

第一步：新建 hive 的数据表

```
create table pokes (foo int, bar string);
```

第二步：批量插入数据

```
load data local inpath '/usr/hive/examples/files/kv1.txt' overwrite into table pokes
```

“**'/usr/hive/examples/files/kv1.txt**”的内容：

```
[hadoop@Master files]$ pwd
/usr/hive/examples/files
[hadoop@Master files]$ more kv1.txt
238 val_238
86 val_86
311 val_311
27 val_27
165 val_165
409 val_409
255 val_255
278 val_278
98 val_98
484 val_484
265 val_265
193 val_193
401 val_401
```

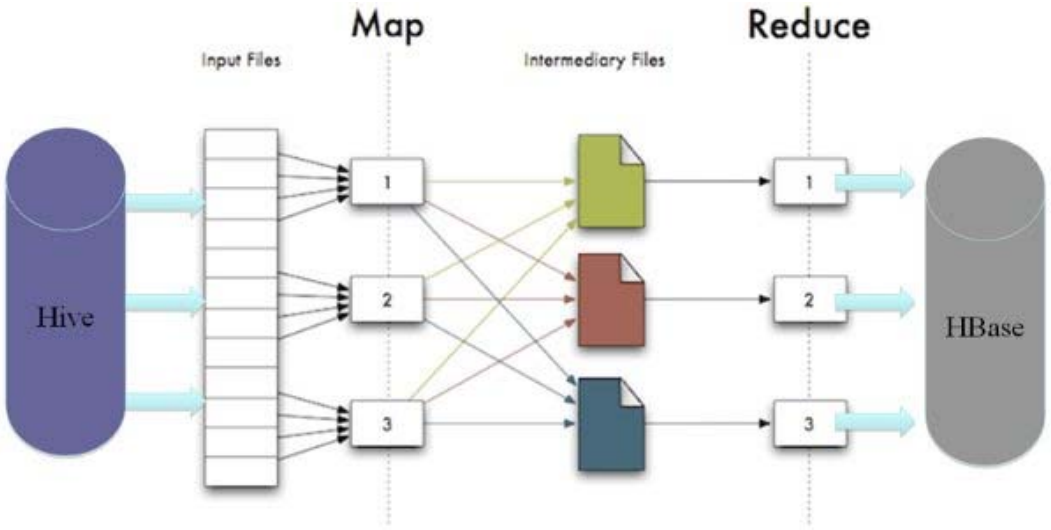
第三步：使用 sql 导入 hbase_table_1

```
insert overwrite table hbase_table_1 select * from pokes where foo=86;
```

```
hive> create table pokes(foo int,bar string);
OK
Time taken: 2.805 seconds
hive> load data local inpath '/usr/hive/examples/files/kv1.txt' overwrite into table pokes;
Copying data from file:/usr/hive/examples/files/kv1.txt
Copying file: file:/usr/hive/examples/files/kv1.txt
Loading data to table default.pokes
Deleted hdfs://192.168.1.2:9000/user/hive/warehouse/pokes
OK
Time taken: 0.414 seconds
hive>
```

```
hive> insert overwrite table hbase_table_1 select * from pokes where foo=86;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_201203271804_0014, Tracking URL = http://Master.Hadoop:50030/jobdetails.jsp?jobid=job_201203271804_0014
Kill Command = /usr/hadoop/libexec/../bin/hadoop job  -Dmapred.job.tracker=http://192.168.1.2:9001 -kill job_201203271804_0014
Hadoop job information for Stage-0: number of mappers: 1; number of reducers: 0
2012-03-28 01:01:03,843 Stage-0 map = 0%,  reduce = 0%
2012-03-28 01:01:09,862 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2012-03-28 01:01:10,866 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2012-03-28 01:01:11,870 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2012-03-28 01:01:12,874 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2012-03-28 01:01:13,878 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2012-03-28 01:01:14,881 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 1.15 sec
2012-03-28 01:01:15,887 Stage-0 map = 100%,  reduce = 100%, Cumulative CPU 1.15 sec
MapReduce Total cumulative CPU time: 1 seconds 150 msec
Ended Job = job_201203271804_0014
1 Rows loaded to hbase_table_1
MapReduce Jobs Launched:
Job 0: Map: 1   Accumulative CPU: 1.15 sec   HDFS Read: 6023 HDFS Write: 0 SUCESS
Total MapReduce CPU Time Spent: 1 seconds 150 msec
OK
Time taken: 23.354 seconds
hive>
```

插入数据时采用了 MapReduce 的策略算法，并且同时向 HBase 写入，如图所示：



- **查看数据**

Hive 端查看结果：

```
hive> select * from hbase_table_1;
OK
86      val_86
Time taken: 1.242 seconds
hive>
```

HBase 端查看结果：

```
hbase(main):002:0> scan 'xyz'
ROW                                   COLUMN+CELL
 86                                   column=cf1:val, timestamp=1332869049357, value=val_86
1 row(s) in 0.0670 seconds
```

- **HBase 端插入数据**

```
put   'xyz','100','cf1:val','xiapi'
```

HBase 端查看结果：

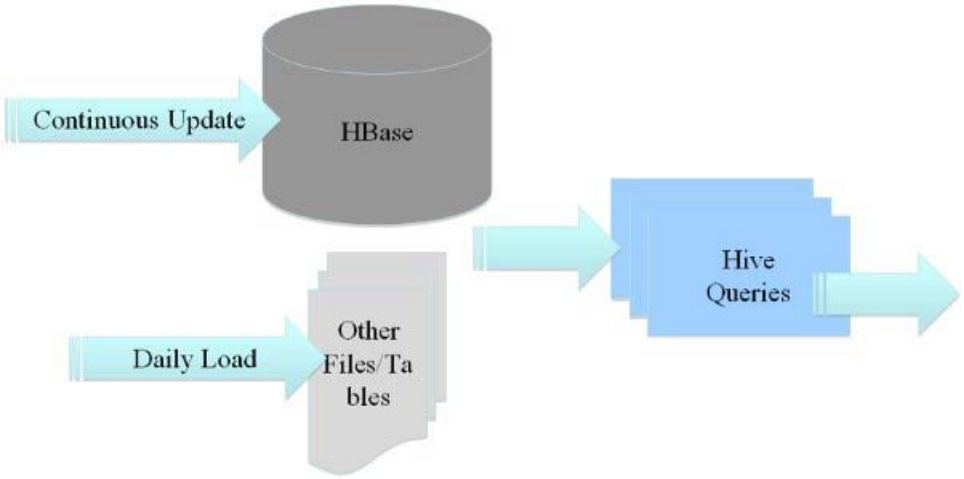
```
hbase(main):004:0> put 'xyz','100','cf1:val','xiapi'
0 row(s) in 0.0080 seconds

hbase(main):005:0> scan 'xyz'
ROW                                   COLUMN+CELL
 100                                  column=cf1:val, timestamp=1332869668231, value=xiapi
 86                                   column=cf1:val, timestamp=1332869049357, value=val_86
2 row(s) in 0.0180 seconds
```

Hive 端查看结果：

```
hive> select * from hbase_table_1;
OK
100     xiapi
86      val_86
Time taken: 2.93 seconds
hive>
```

现在 Hive 中添加记录 HBase 中有记录添加，同样在 HBase 中添加记录 Hive 中也会添加， 表示 Hive 与 HBase 整合成功，对海量级别的数据我们是不是可以在 HBase 写入，在 Hive 中查询呢？因为 HBase 不支持复杂的查询，但是 HBase 可以作为基于 key 获取一行或多行数据，或者扫描数据区间，以及过滤操作。而复杂的查询可以让 Hive 来完成，一个作为存储的入口(HBase)，一个作为查询的入口(Hive)。如下图示。



- **hive 访问已经存在的 hbase**

使用 CREATE EXTERNAL TABLE。

```
CREATE EXTERNAL TABLE hbase_table_2(key int, value string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = "cf1:val")
TBLPROPERTIES("hbase.table.name" = "some_existing_table");
```

2、Hive与RDBMS

由于 **Hive 采用了 SQL** 的**查询语言** HQL，因此**很容易**将 **Hive 理解**为**数据库**。**其实从结构上来看**，**Hive** 和**数据库除了拥有类似**的**查询语言**，**再无类似之处**。本节将从多个方面来阐述 Hive 和数据库的差异。数据库可以用在 Online 的应用中，但是 **Hive** 是为**数据仓库**而设计的，清楚这一点，有助于从应用角度理解 Hive 的特性。

表 2-1 Hive 与 RDBMS 比较

	Hive	RDBMS
查询语言	HQL	SQL
数据库的存储位置	HDFS	Raw Device 或 Local FS
数据格式	用户定义	系统决定
数据更新	不支持	支持
索引	无	有
执行	MapReduce	Executor
执行延迟	高	低
可扩展性	高	低
数据规模	大	小

- **查询语言**

由于 SQL 被广泛的应用在数据仓库中，因此，专门针对 Hive 的特性设计了类 SQL 的查询语言 HQL。熟悉 SQL 开发的开发者可以很方便的使用 Hive 进行开发。

- **数据存储位置**

Hive 是建立在 Hadoop 之上的，所有 Hive 的数据都是存储在 HDFS 中的。而数据库则可以将数据保存在块设备或者本地文件系统中。

- **数据格式**

Hive 中没有定义专门的数据格式，数据格式可以由用户指定，用户定义数据格式需要指定三个属性：列分隔符（通常为空格、“\t”、“\x001”）、行分隔符（“\n”）以及读取文件数据的方法（Hive 中默认有三个文件格式 TextFile，SequenceFile 以及 RCFile）。由于在加载数据的过程中，不需要从用户数据格式到 Hive 定义的数据格式的转换，因此，Hive 在加载的过程中不会对数据本身进行任何修改，而只是将数据内容复制或者移动到相应的 HDFS 目录中。而在数据库中，不同的数据库有不同的存储引擎，定义了自己的数据格式。所有数据都会按照一定的组织存储，因此，数据库加载数据的过程会比较耗时。

- **数据更新**

由于 Hive 是针对数据仓库应用设计的，而数据仓库的内容是读多写少的。因此，Hive 中不支持对数据的改写和添加，所有的数据都是在加载的时候中确定好的。而数据库中的数据通常是需要经常进行修改的，因此可以使用 INSERT INTO ... VALUES 添加数据，使用 UPDATE ... SET 修改数据。

- **索引**

之前已经说过，Hive 在加载数据的过程中不会对数据进行任何处理，甚至不会对数据进行扫描，因此也没有对数据中的某些 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。由于 MapReduce 的引入， Hive 可以并行访问数据，因此即使没有索引，对于大数据量的访问，Hive 仍然可以体现出优势。数据库中，通常会针对一个或者几个列建立索引，因此对于少量的特定条件的数据的访问，数据库可以有很高的效率，较低的延迟。由于数据的访问延迟较高，决定了 Hive 不适合在线数据查询。

- **执行**

Hive 中大多数查询的执行是通过 **Hadoop** 提供的 **MapReduce** 来实现的（类似 select * from tbl 的查询不需要 MapReduce）。而数据库通常有自己的执行引擎。

- **执行延迟**

之前提到，**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。另外一个导致 **Hive** 执行延迟高的因素是 **MapReduce** 框架。由于 MapReduce 本身具有较高的延迟，因此在利用 MapReduce 执行 Hive 查询时，也会有较高的延迟。相对的，数据库的执行延迟较低。当然，这个低是有条件的，即数据规模较小，当数据规模大到超过数据库的处理能力的时候，Hive 的并行计算显然能体现出优势。

- **可扩展性**

由于 Hive 是建立在 Hadoop 之上的，因此 Hive 的可扩展性是和 Hadoop 的可扩展性是一致的（世界上最大的 Hadoop 集群在 Yahoo!，2009 年的规模在 4000 台节点左右）。而数据库由于 ACID 语义的严格限制，扩展行非常有限。目前最先进的并行数据库 Oracle 在理论上的扩展能力也只有 100 台左右。

- **数据规模**

由于 Hive 建立在集群上并可以利用 MapReduce 进行并行计算，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较小。

3、HBase与RDBMS

HBase 是一个基于列模式的映射数据库，它只能表示很简单的键——数据的映射关系，这大大简化了传统的关系数据库。与关系数据库相比，它有如下特点：

- **数据类型**

HBase 只有简单的字符串类型，所有的类型都是交由用户自己处理的，它只保存字符串。而关系数据库有丰富的类型选择和存储方式。

- **数据操作**

HBase 操作只是简单的插入、查询、删除、清空等操作，表与表之间是分离的，没有复杂的表与表之间的关系，所有不能也没有必要实现表和表之间的关联等。而传统的关系数据库常有各种各样的函数、连接操作。

- **存储操作**

HBase 是基于列存储的，每个列族都由几个文件保存，不同列族的文件是分离的。传统的关系型数据库是基于表格结构和行模式保存的。

- **数据维护**

确切来说，**HBase** 的更新操作应该不叫更新，虽然一个主键或列会对应新的版本，但它的旧版本仍然会保留，所以它实际上是插入了新数据，而不是传统关系数据库里面的替换修改。

- 可伸缩性

Hbase 这类分布式数据库就是为了这个目的而开发出来的，所以它能够轻易地增加或减少（在硬件错误的时候）硬件数量，并且对错误的兼容性比较高。而传统的关系数据库通常需要增加中间层才能实现类似的功能。

当前的关系型数据库通常都是从 20 世纪 70 年代发展而来，它们都具有 ACID 特性，并且拥有丰富的 SQL 语言。关系数据库由包含数据记录的多个数据表组成，用户可在有相关数据的多个表之间建立相互联系。在关系数据库中，数据被分散到不同的数据表中，以便使每一个表中的数据只记录一次，从而避免数据的重复输入，减少冗余。

除了具有面向磁盘存储、带有索引结构、多线程访问、基于锁的同步访问机制、基于 log 记录恢复机制之外，还有以下几个特点：

- 关系中的每个属性必须是不可分割的数据单元（即表中不能再包含表）。
- 关系中的每一列元素必须是类型相同的数据。
- 同一个关系中不能有相同的字段（属性），也不能有相同的记录。
- 关系的行、列次序可以任意交换，不影响其信息内容。

而 Bigtable 和 HBase 这些基于列模式的分布式数据库，更适合海量存储和互联网应用的需求，灵活的分布式架构可以使其利用廉价的硬件设备组建一个大的数据仓库。互联网应用是以字符为基础的，而 Bigtable 和 HBase 就是针对这些应用开发出来的数据库。

由于 HBase 具有时间戳特性，所以它生来就特别适合开发 wiki、archiveorg 之类的服务，并且它原本就是作为一个搜索引擎的一部分开发出来的。

4、数据库与数据仓库

4.1 数据库相关概念

数据（data）是对客观事物的符号表示，是用于表示客观事物的未经加工的原始素材，如图形符号、数字、字母等。或者说，数据是通过物理观察得来的事实和概念，是对现实世界中的地方、事件、其它对象或概念的描述。在计算机科学中数据是指所有能输入到计算机并被计算机程序处理的符号介质的总称。

数据库技术是数据管理的最新技术，它是研究如何科学地组织和存储数据，如何高效地检索和处理数据的实用技术，它是当代信息系统的基础。

数据库管理系统（DBMS）是计算机系统的一个重要组成部分。数据库技术的产生并不是偶然的，而是数据库管理的必然产物。数据管理方法经历了人工管理阶段、文件系统阶段和数据库系统阶段。而在数据库系统阶段，计算机技术迅猛发展，数据处理量急剧增大，这时的数据管理要求数据有更高的独立性，更高的共享性。

4.2 数据仓库相关概念

数据仓库（Data Warehouse）简称 DW。最早被誉为数据仓库之父的 W.H.Inmon 将数据仓库明确地定义为：数据仓库是集成的面向主题的数据库集合。它是用来支持决策、支

持功能的。其中每个数据单位都与时间相关。这些数据应该是良好定义的、一致的、不变的，并且支持数据分析、查询、报表生成和与长期积累的历史数据的对比。数据仓库系统是一种专为联机分析应用和决策支持系统（DDS）提供数据分析和决策工具的结构化数据环境。它涉及数据的抽取、转换、装载、存取、元数据管理、查询、报表、分析工具及相应的开发方法等。

4.3 数据仓库体系结构

数据仓库的体系结构如图 4.3-1 所示：

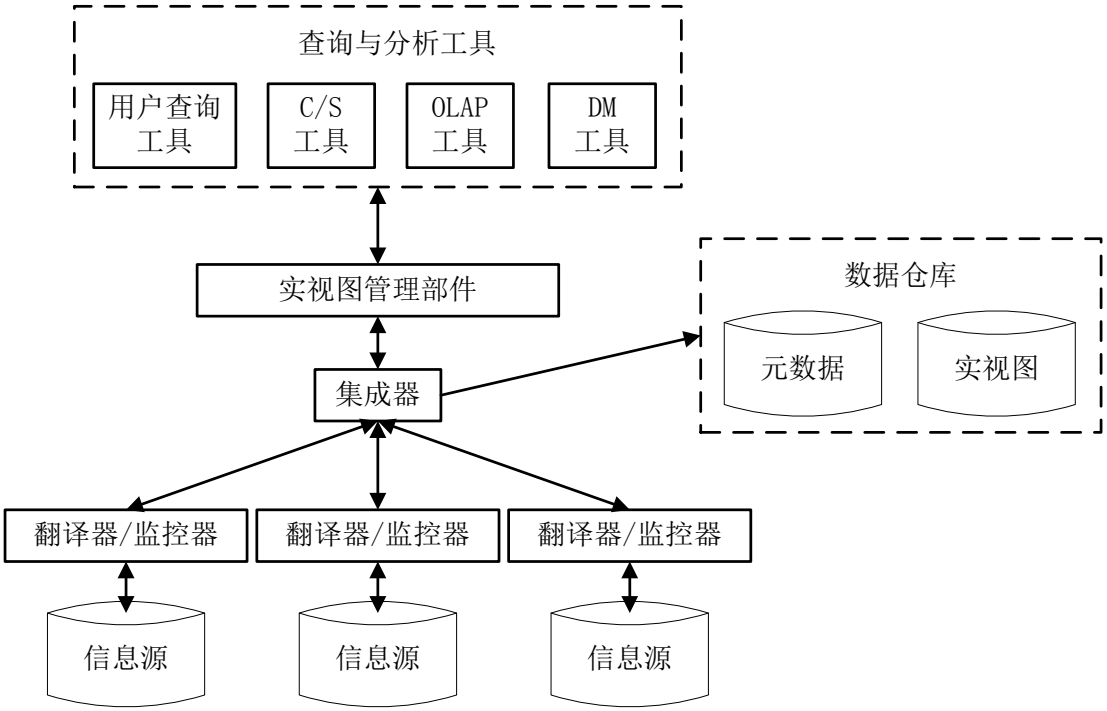


图 4.3-1 数据仓库系统的体系结构

- **信息源**

可以是传统数据库， 也可以是文件、HTML 文件、知识库等。

- **翻译器**

翻译工作包括数据结构的翻译和数据类型的翻译，必要时还需对数据之间的语义联系进行重新构造。

- **监控器**

数据仓库中的实视图是一种“离线”数据，与源数据在一致性上存在着时间差。监控器负责监视数据仓库与信息源之间的数据误差或者说是数据增量， 并报告给上层的集成器。

- **集成器**

集成器除了负责进行数据仓库初始化之外，还负责接受监控器的变化报告，并将信息源

的新变化根据有关定义转化到数据仓库中。

- **实现图管理部件**

负责数据共享、安全保密、归档、备份、恢复及元数据的管理等工作。

- **查询与分析工具**

完成决策问题所需的各种查询与分析工具，包括用户查询工具、C/S 工具、用于多维数据的 OLAP 分析工具、数据挖掘（DW）工具等，以实现决策支持系统的各种要求。

4.5 数据仓库基本特征

- **面向主题**

数据仓库是围绕一些主题的，主题是一个抽象的概念，是在较高层次上将信息系统中的数据按不同类别、不同侧面进行综合、归类和分析利用。从逻辑意义上讲，它是对应某一宏观分析领域所涉及的分析对象，是针对某一决策问题而设置的。数据仓库中的数据面向主题，与传统数据库面向应用相对应。面向主题的数据组织方式，就是在较高层次上对分析对象的数据做一个完整、一致的描述，并统一分析对象所涉及的数据项及数据项之间的关系。

- **数据的集成性**

数据仓库中存贮的数据是集成化的。数据仓库的集成性是指在数据进入数据仓库之前，必须通过数据加工集成，这是建立数据仓库的关键步骤。要清除数据中的不一致性、矛盾性和进行数据综合运算，还要将原始数据结构做一个从面向应用到面向主题的转变。

- **相对稳定性**

数据仓库中的数据是相对稳定的。它不进行实时更新，一旦数据进入数据仓库中就不能由用户进行更新。也就是说，经加工和集成进入数据仓库的数据是极少修改和更新的，但从数据仓库存贮的数据内容上，可分为当前数据和历史数据。在一定时间间隔后，当前数据需要按一定的方法转换成历史数据。对年代久远的、查询率低的数据需要从数据仓库剥离到慢速存贮设备上，对分析处理不再有用的数据需要从数据仓库中删除，这些工作是由系统管理员或由系统自动完成的。因此，可以说数据仓库在一定时间间隔内是稳定的。

- **随时间而改变**

数据仓库的数据不可更新是相对用户进行分析处理时不对数据进行更新操作而言的。但不是说数据从进入数据仓库以后就永远不变。数据仓库中的数据随时间而定期地被更新。每隔一段（固定的）时间间隔，数据系统中产生的数据就会经抽取、转换后集成到数据仓库中。随着时间的变化，数据以更高的层次不断被综合，以适应趋势分析的要求。

- **数据模型**

数据仓库中数据模型有两种形式：星型（图 4.5-1）和雪花型（图 4.5-2）。星型的核心是事实表，围绕事实表的是维度表，通过事实表将各种不同的维度表连接起来，各个维度表都连接到中央事实表。雪花模型是对星型模型的扩展，每一个维度表都可以向外连接到多个详细类别表。

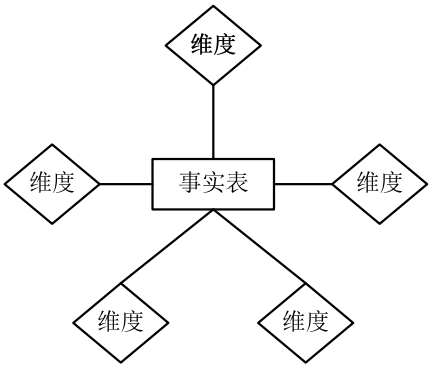


图 4.5-1 星型

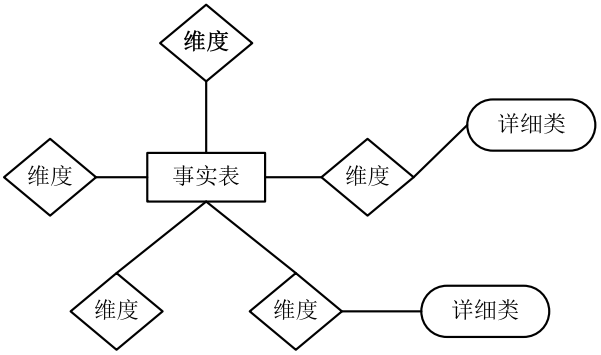


图 4.5-2 雪花型

4.6 数据库与数据仓库比较

- **数据处理方式不同**

数据库系统用于**事务处理**，是联机事务处理（OLTP）的应用。**数据仓库**用于**分析型处理**，是联机分析处理（OLAP）的应用。

操作型数据与**分析型数据**的**根本区别**在于**事务处理**与**分析处理**的**差异**。**传统的数据库**系统主要用于日常事务处理工作，存放的数据符合操作型数据的特点，而**数据仓库**用于数据的分析处理，存放的是分析型数据。

- **系统结构不同**

数据库系统由 3 个部分组成：**数据库（DB）**、**数据库核心**和**数据库工具**。数据库系统处理的是操作型数据，需要进行频繁更新、删除等操作。数据库核心功能非常强大，数据库系统主要面向应用，**数据库工具**有**两类**：数据开发工具和数据查询工具。**数据仓库系统**也是由 3 个部分组成：**数据仓库**、**数据仓库管理系统**和**数据仓库工具**，3 个部分的功能是不同的。**数据仓库**是进一步挖掘信息的基础；**数据仓库管理**系统相对于数据库核心而言，要简单得多；**数据仓库工具**不仅需要具有强大分析功能的工具，而且需要具有一般查询功能的工具。

- **数据组织方式不同**

与**传统数据库面向应用**进行数据组织的特点相对应，**数据仓库**中的数据是**面向主题**进行组织的。**面向应用**进行数据组织，**需充分**了解**企业或机构**的**部门组织结构**，考虑各部门的业务活动特点及数据流动情况，并按照实际应用即业务处理流程来组织数据。而**面向主题**进行数据组织是**根据分析要求**将数据组织成一个**完备**的**分析领域**，即**主题域**，是在**较高层**次上将企业信息系统中的**数据综合**、**归类**并进行**分析**和**利用**。

参考文献

感谢以下文章的编写作者，没有你们的铺路，我或许会走得很艰难，参考不分先后，贡献同等珍贵。

【1】Hadoop 实战——陆嘉恒——机械工业出版社

【2】实战 Hadoop——刘鹏——电子工业出版社

【3】Hive 与数据库的异同

地址：http://cloud.csdn.net/a/20101128/282617.html

【4】Hive HBase 整合

地址：http://running.iteye.com/blog/898399

【5】hive 与 hbase

地址：http://hi.baidu.com/wuhanjzl/blog/item/3d3c4412d85dd7e8c2ce7978.html

【6】Hive 与 Hbase 的区别

地址：http://hi.baidu.com/gcpopo/blog/item/fe8c25add0efe6d07dd92a68.html

【7】Hive 和 HBase 整合

地址：http://hotdog.iteye.com/blog/1203488

【8】Hive HBase 整合使用

地址：http://datalife.iteye.com/blog/910300

【9】Hadoop Hive 与 Hbase 整合

地址：http://blog.csdn.net/hguisu/article/details/7282050

【10】伪分布式环境下，Hive0.8.1 与 HBase0.92.0 集成的配置说明

地址：http://blog.csdn.net/fullofwindandsnow/article/details/7331403

【11】Hbase 与 RDBMS 的区别

地址：http://qa.taobao.com/?p=11852

【12】Hive HBase Integration

地址：https://cwiki.apache.org/confluence/display/Hive/HBaseIntegration