

基于Hadoop的数据仓库Hive



東北大學
Northeastern University



HORIZON
昊宸科技



1 Hive环境搭建

2 Hive简介

3 Hive体系结构

4 Hive应用场景

5 Hive表常用操作

6



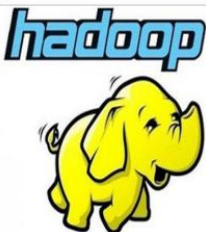



- Hive是构建在Hadoop之上的数据仓库平台。
- Hive是SQL解析引擎，它将SQL语句转译为MapReduce作业，并在Hadoop上运行。
- Hive表是HDFS的文件目录，一个表对应一个目录名，如果有分区的话，则分区值对应子目录。

Hive发展历史



**Hive是Facebook开发的，
构建于Hadoop集群之上的数
据仓库应用。2008年
Facebook将Hive项目贡献
给Apache，成为开源项目。
目前最新版本hive-2.3.1**

	随着数据量增加某些查询需要几个小时甚至几天才能完成。当数据达到1T时，MySQL进程跨掉。
	可以支撑几个T的数据，但每天收集用户点击流数据（每天约400G）时,Oracle开始撑不住。
	有效解决了大规模数据的存储与统计分析的问题，但是MapReduce程序对于普通分析人员的使用过于复杂和繁琐。
	对外提供了类似于SQL语法的HQL语句数据接口，自动将HQL语句编译转化为MR作业后在Hadoop上执行。降低了分析人员使用Hadoop进行数据分析的难度。

Hive体系结构

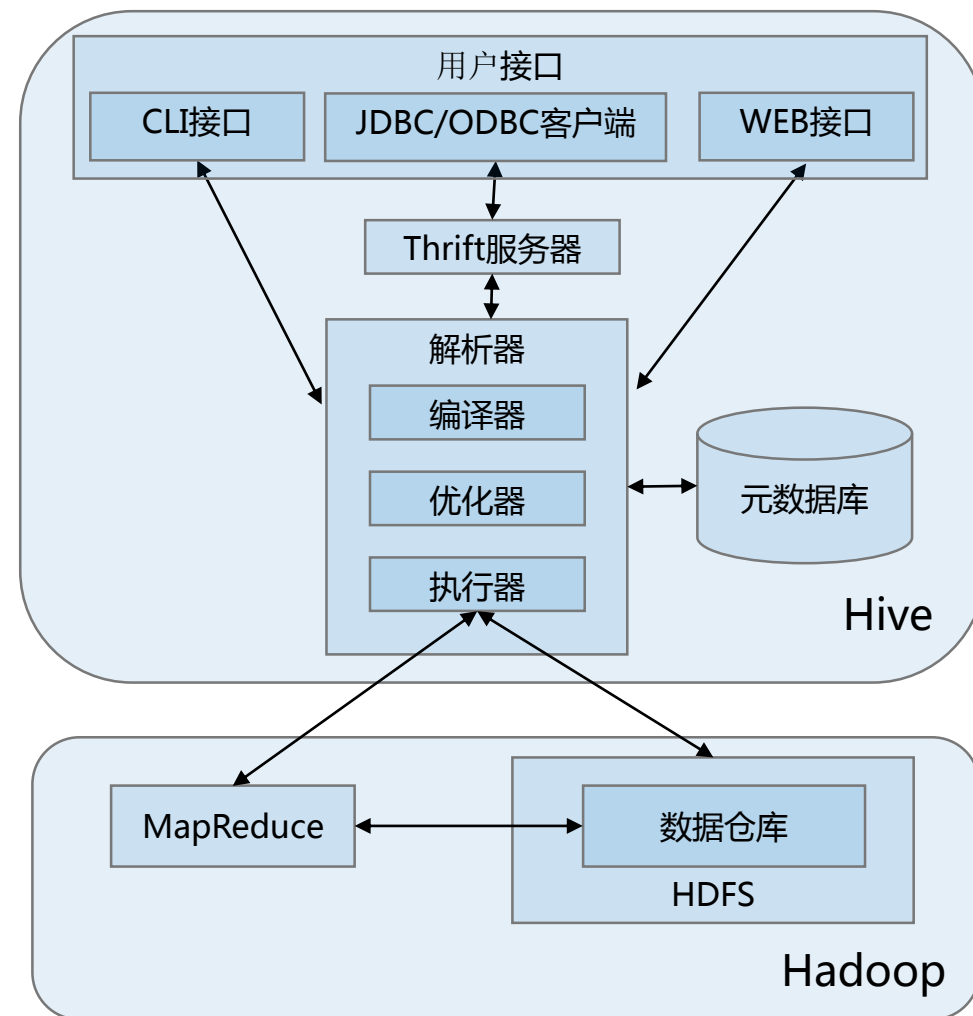


用户接口

- **CLI** : Cli 启动的时候，会同时启动一个 Hive 副本。
- **JDBC客户端** : 封装了Thrift,java应用程序，可以通过指定的主机和端口连接到在另一个进程中运行的hive服务器
- **ODBC客户端** : ODBC驱动允许支持ODBC协议的应用程序连接到Hive。
- **WUI 接口** : 是通过浏览器访问 Hive

Thrift服务器

基于socket通讯，支持跨语言。Hive Thrift服务简化了在多编程语言中运行Hive的命令。绑定支持C++,Java,PHP,Python和Ruby语言。



Hive体系结构

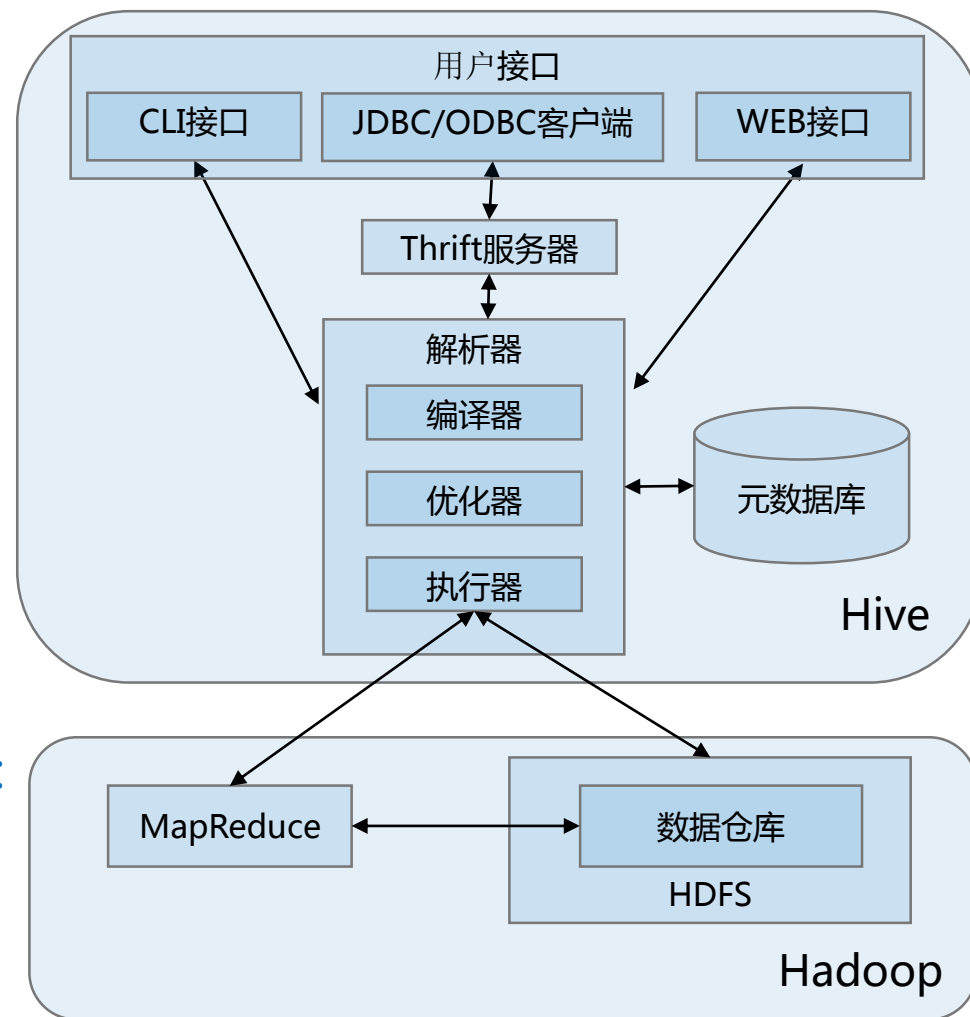


解析器

- **编译器**:完成 HQL 语句从词法分析、语法分析、编译、优化以及执行计划的生成。
- **优化器**:是一个演化组件，当前它的规则是：列修剪，谓词下压。
- **执行器**会顺序执行所有的Job。如果Task链不存在依赖关系，可以采用并发执行的方式执行Job。

元数据库

Hive的数据由两部分组成：数据文件和元数据。元数据用于存放Hive库的基础信息，它存储在关系数据库中，如 mysql、derby。元数据包括：数据库信息、表的名称，表的列和分区及其属性，表的属性，表的数据所在目录等。



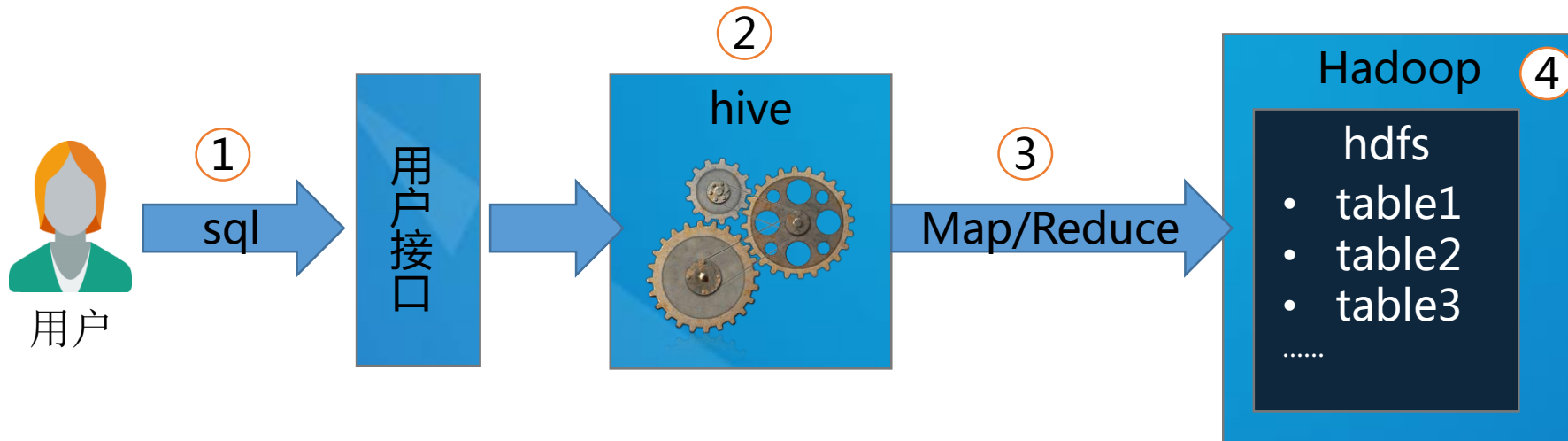
Hive体系结构-Hive元数据库表



表名	说明	关联键
DBS	元数据库信息，存放HDFS路径信息	DB_ID
TBLS	所有hive表的基本信息	TBL_ID,SD_ID,DB_ID
TABLE_PARAM	表级属性，如是否外部表，表注释等	TBL_ID
COLUMNS_V2	Hive表字段信息(字段注释，字段名，字段类型，字段序号)	CD_ID
SDS	所有hive表、表分区所对应的hdfs数据目录和数据格式	SD_ID,SERDE_ID
SERDES	Hive表的序列化类型	SERDE_ID
SERDE_PARAM	序列化反序列化信息，如行分隔符、列分隔符、NULL的表示字符等	SERDE_ID
PARTITIONS	Hive表分区信息	PART_ID,SD_ID,TBL_ID
PARTITION_KEYS	Hive分区表分区键	TBL_ID
PARTITION_KEY_VALS	Hive表分区名(键值)	PART_ID
SEQUENCE_TABLE	保存Hive对象的下一个可用ID,包括数据库，表，字段，分区等对象的下一个ID。默认ID每次+5	SEQUENCE_NAME,NEXT_VAL

Hive体系结构-Hive的运行机制

- 用户通过用户接口连接Hive,发布Hive SQL。
- Hive解析查询并制定查询计划。
- Hive将查询转换成MapReduce作业。
- Hive在Hadoop上执行MapReduce作业。





Hive应用场景-优势/缺点

- 解决了传统关系数据库在大数据处理上的瓶颈。适合大数据的批量处理。
- 充分利用集群的CPU计算资源、存储资源，实现并行计算。
- Hive支持标准SQL语法，免去了编写MR程序的过程，减少了开发成本。
- 具有良好的扩展性，拓展功能方便。
- Hive的HQL表达能力有限：有些复杂运算用HQL不易表达。
- Hive效率低：Hive自动生成MR作业，通常不够智能；HQL调优困难，粒度较粗；可控性差。
- 针对Hive运行效率低下的问题，促使人们去寻找一种更快，更具交互性的分析框架。SparkSQL 的出现则有效的提高了Sql在Hadoop 上的分析运行效率。

```
/**
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership. The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 *
 * The main driver
 * Invokes this
 * throws IOException
 */
public int run(
    JobConf conf,
    Configuration conf) throws IOException {
    // the keys
    conf.setOutFormat(
    // the value
    conf.setOutFormat(

    conf.setMapOutputKeyComparator(
    conf.setMapOutputValueComparator(
    conf.setReducer(

    List<String>
    for(int i=0;
    try {
        if ("m"
        conf.set
        } else {
        conf.set
        } else {
        other_
        }
    } catch (Exception e) {
        System.out.println(e);
        return 1;
    } catch (Exception e) {
        System.out.println(e);
        return 1;
    }
    return 0;
}

// Make sure
if (other_args != null) {
    System.out.println("other_args: " + other_args);
}

return printUsage();
}

FileInputFormat.setInputPaths(conf);

/**
 * This is an example Hadoop Map/Reduce application.
 * It reads the text input files, breaks each line into words
 * and counts them. The output is a locally sorted list of words and the
 * count of how often they occurred.
 *
 * To run: bin/hadoop jar build/hadoop-examples.jar wordcount
 * [-m <mapargs>/>] [-r <reduceargs>/>] <input-dir> <output-dir>
 */
public class WordCount extends Configured implements Tool {

    /**
     * Counts the words in each line.
     * For each line of input, break the line into words and emit them as
     * {<word>/>, <count>/>}.
     */
    public static class MapClass extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, IntWritable> {

        private final static Text word = new Text();

        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

            String line = value.toString();
            StringTokenizer itr = new StringTokenizer(line);
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, new IntWritable(1));
            }
        }
    }

    private MapClass mapClass = new MapClass();

    public void configure(Configuration conf) throws IOException {
        mapClass.setConf(conf);
    }

    public void run(Configuration conf, boolean verbose) throws IOException {
        JobConf job = new JobConf(conf, WordCount.class);
        FileInputFormat.setInputPaths(job, getArgs());
        FileOutputFormat.setOutputPath(job, new Path(job.getOutputDir()));
        job.setMapperClass(MapClass.class);
        job.setReducerClass(WordCount.class);
        job.setOutputKeyComparator(new Text.Comparator());
        job.setOutputValueComparator(new IntWritable.Comparator());
        job.setMapOutputKeyComparator(new Text.Comparator());
        job.setMapOutputValueComparator(new IntWritable.Comparator());
        job.setCombinerClass(WordCount.class);
        job.setDriverClass(WordCount.class);
        job.setJarClass(WordCount.class);
        job.setJarResourcePath(new Path(WordCount.class.getResource("wordcount.jar").toURI()));
        job.setJarResourceFilter(new PathFilter() {
            public boolean accept(Path path) {
                return path.getName().equals("wordcount.jar");
            }
        });
        job.waitForCompletion(true);
        return 0;
    }
}
```

```
select word, count(*)
from (
select
explode(split(sentence, ' '))
word
from article
) t
group by word
```

Hive应用场景-优势/缺点



◆ 适用场景

- 海量数据的存储处理
- 数据挖掘
- 海量数据的离线分析

◆ 不适用场景

- 复杂的机器学习算法
- 复杂的科学计算
- 联机交互式实时查询

◆ HiveServer2

目前Hive的Thrift服务端通常使用HiveServer2,它是HiveServer改进版本,它提供了新的ThriftAPI来处理JDBC或者ODBC客户端,可以进行Kerberos身份验证,支持多个客户端并发。

◆ Beeline

HiveServer2还提供了新的CLI: BeeLine,它是Hive 0.11引入的新的交互式CLI,基于SQLLine,可以作为Hive JDBC Client 端访问HiveServer2。

通过beeline连接hive : jdbc:hive2://192.168.88.104:10000/hiveTest

◆ cli

Hive数据类型



基本类型	大小	描述	文字示例
TINYINT	1个字节	有符号整数	1
SMALLINT	2个字节	有符号整数	1
INT	4个字节	有符号整数	1
BIGINT	8个字节	有符号整数	1
STRING	最大2GB	字符串，类似SQL中的VARCHAR类型	'a','a'
FLOAT	4个字节	单精度浮点型	1.0
DOUBLE	8个字节	双精度浮点型	1.0
BOOLEAN	~	TRUE/FALSE	TRUE/FALSE
Map	不限	一组无序键值对。键的类型必须是源自的，值可以是任何类型。同一个映射的键的类型必须相同，值的类型也必须相同。	Map('a',1,'b',2)
Array	不限	一组有序字段，字段的类型必须相同。	array(1,2)
Struct	不限	一组命名的字段，字段的类型可以不同。	Struct('a',1,1.0)

Hive与传统数据库比



对比项	Hive	传统数据库
数据插入	支持批量导入	支持单条和批量导入
数据更新	不支持	支持
索引	有索引功能，不想RDBMS有键的概念，可以在某些列上建索引，加速一些查询操作。创建的索引数据会保存在另外的表中。	支持
分区列	支持，hive表的分区根据分区列的值对表进行粗略划分，加快查询速度。	支持，提供分区表来改善大型表以及具有各种访问模式的表的可伸缩性、可管理性、以及提高数据库的效率。
执行延迟	高，构建在HDFS和MR之上，比传统数据库延迟要高。	低，传统SQL延迟一般小于1s。
扩展性	好，基于Hadoop有很好的横向扩展性。	有限，RDBMS非分布式，横向扩展难实现。
数据类型	整数、浮点数、布尔型、文本和二进制串、时间戳、数组、映射、结构	整数、浮点数、定点数、文本和二进制串、时间

Hive表-托管表&外部表



存储的数据+元数据组成。

Hive也有数据库，可以通过CREATE DATABASE创建数据库。默认库是default库。

托管表和外部表

➤ 数据存储：

托管表：数据存储存储在仓库目录下。

外部表：数据存储在任何HDFS目录下。

➤ 数据删除

托管表：删除元数据和数据。

外部表：只删除元数据。

➤ 创建表语句

外部表：CREATE **EXTERNAL** TABLE table_name(attr1 STRING) **LOCATION** '/usr/tom/tablea;

Hive表-托管表&外部表



练习：

- 1.创建10min数据托管表。
- 2.导入数据到10min数据托管表。
- 3.创建10min数据外部表。
- 4.导入数据到10min外部表。
- 5.删除10min数据托管表，并查看10min数据表目录和数据。
- 6.删除10min数据外部表，并查看10min数据表目录和数据。

一种将数据分片的方式，可以加快查询速度。表->分区->桶。

分区

- 分区列并非实际存储的列数据，分区只是表目录下嵌套的目录。
- 可以按照各种维度对表进行分区。
- 分区可以缩小查询范围。
- 分区是在创建表的时候用PARTITION BY子句定义的。
- 加载数据到分区使用LOAD语句，且要显示的指定分区值。
- 使用SHOW PARTITIONS语句查看Hive表下有哪些分区。
- 使用SELECT语句中指定分区，Hive只扫描指定分区数据。

Hive表-分区和桶



Hive表分区分为两种，静态分区和动态分区。静态分区和动态分区的区别在于导入数据时，是手动输入分区名称，还是通过数据来判断数据分区。对于大数据批量导入来说，显然采用动态分区更为简单方便。

动态分区需要配置：

`hive.exec.dynamic.partition=true`

`hive.exec.dynamic.partition.mode=nostrict`(允许所有的分区都为动态的)。

动态分区只能通过`insert overwrite table table_name select ...`这样的语句导入数据。

静态分区可以通过多种方式导入数据：`load/ITS/hdfs`拷贝等。

静态分区导入不灵活，导入前要预先知道导入的分区。

练习

- 1.创建10min数据表，按照风场，风机分区。
- 2.为10min数据表创建分区。
- 3.观察10min数据表建好的分区结构。
- 4.加载数据到10min数据表指定分区中。
- 5.使用show partitions查看10min数据表下的分区有哪些？
- 6.使用SELECT 语句查询10min数据指定风场、风机下的数据。
- 7.创建未分区的10min数据表。
- 8.导入数据到未分区的10min数据表中。
- 9.使用SELECT语句查询指定风场风机的10min数据。
- 10.对比有分区的10min数据和未分区的10min数据表查询效率。

桶

➤ 桶是表上附加的额外结构，可以提高查询效率。有利于做map-side-join。

➤ 使“取样”更方便高效。

➤ 使用CLUSTER BY子句来指定划分桶所用的列和要划分桶的个数。

```
create table bucketed_user(id int,name string) clustered by (id) into 4 buckets;
```

➤ 对桶中的数据可以做排序。使用SORTED BY子句。

```
create table bucketed_user(id int, name string) clustered by(id) sorted by (id asc) into 4 buckets;
```

➤ 不建议我们自己分桶，建议让Hive划分桶。

➤ 向分桶中填充数据前，需要设置hive.enforce.bucketing设置为true。

```
insert overwrite table bucket_users select * from users;
```

➤ 实际上桶对应于MapReduce的输出文件分区：一个作业产生的桶和reduce任务个数相同。

练习

- 创建10min数据表，按照风场分区，按照风机分桶，按照时间排序。
- 向10min数据表填充数据。
- 查看10min数据表下文件分布情况，并查看文件内容。
- 查询10min数据表中指定风场，风机某段时间内的数据。
- 使用TABLESAMPLE子句对表进行取样。

Hive表-存储格式



Hive从两个维度对表的存储进行管理：“行格式”（row format）和“文件格式”（file format）。

- 行格式：一行中数据的存储格式。按照hive的术语，行格式的定义由SerDe定义，即序列化和反序列化。也就是查询数据时，SerDe将文件中字节形式的数据行反序列化为Hive内部操作数据行时使用的对象形式。Hive向表中插入数据时，序列化工具会将Hive的数据行内部表示形式序列化为字节形式并写到输出文件中去。
- 文件格式：最简单的文件格式是纯文本文件，但是也可以使用面向列的和面向行的二进制文件格式。二进制文件可以是顺序文件、Avro、RCFile、ORC、parquet文件。

默认存储格式-分隔的文本。

Create table ...等价于create table ... ROW FORMAT DELIMITED FIELDS TERMINATED BY '\001'

COLLECTION ITEMS TERMINATED BY '\002' MAP KEYS TERMINATED BY '\003' LINES TERMINATED BY '\n' STORED AS TEXTFILE;

Hive内部默认使用LazySimpleSerDe处理分隔文本。

Hive现有的文件格式

	TEXTFILE	SEQUENCEFILE	RCFILE
数据类型	text only	text/binary	text/binary
内部存储结构	Row-based	Row-based	Column-based
压缩	File-based	Block-based	Block-based
可拆分	YES	YES	YES
压缩后可拆分	NO	YES	YES

Hive现有的SerDe

	LazySimpleSerDe	LazyBinarySerDe (HIVE-640)	BinarySortable SerDe
serialized format	delimited	proprietary binary	proprietary binary sortable*
deserialized format	LazyObjects*	LazyBinaryObjects*	Writable
	ThriftSerDe (HIVE-706)	RegexSerDe	ColumnarSerDe
serialized format	Depends on the Thrift Protocol	Regex formatted	proprietary column-based
deserialized format	User-defined Classes, Java Primitive Objects	ArrayList<String>	LazyObjects*

※lazyObject:指只有在访问数据时,才去做反序列化操作。

参照官方网站,有更详细的SerDe信息。

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-RowFormats&SerDe>

练习：

使用store as指定文1.创建10min数据表，使用row format指定SerDe分别为：

avro/orc/parquet/rcfile/sequencefile/textfile

件格式为：avrofile/orcfile/parquetfile/rcfile/sequencefile/textfile

2.向新建的10min数据表中灌入数据。查看文件具体格式。

3.查询10min数据表。

向Hive表中导入数据有多种方式。

➤ INSERT导入数据。

参照：<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML#LanguageManualDML-InsertingdataintoHiveTablesfromqueries>

Standard syntax:

```
INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select_statement1 FROM from_statement1;
INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from_statement1;
INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] (z,y) select_statement1 FROM from_statement1;
```

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] ...;
```

多表插入

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;
```

动态分区插入

向Hive表中导入数据有多种方式。

➤ Load方式导入

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

➤ CREATE TABLE ... AS SELECT (CATS)

Example:

```
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;
```

练习:

1. 创建10min数据表，按照风场风机分区。
2. 使用insert overwrite select注入数据。
3. 使用From的方式插入数据到tenmindata4.
4. 使用动态分区插入数据。全部动态如何实现？即：风场为动态，风机为动态插入。
5. 使用CATS创建tenmindata5;
6. 使用CREATE TABLE LIKE创建tenmindata6;

Hive表-表的修改、删除



由于Hive使用“读时模式”，所以才创建表之后，它非常灵活的支持对表定义的修改。但一般需要警惕，在很多情况下，要由你来确保修改数据以符合新的结构。

➤ 重命名表

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

更新元数据+移动表目录(托管表)

➤ 修改列定义

```
ALTER TABLE table_name ADD COLUMNS(col1 STRING);
```

详细参考：<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-AlterTable>

➤ 表删除

DROP TABLE-删除元数据+表数据(托管表)。

➤ 截断表

```
TRUNCATE TABLE table_name [PARTITION partition_spec];
```

- 1.了解Avro、RCFile、ORC、parquet存储结构，比较异同。
- 2.练习课上所有的例子。(看到tenmindata1~tenmindata5)
- 3.描述一下Hive的特点，Hive和关系数据库的区别。
- 4.每人启动自己机器上的hiveServer2，并使用beeline连接Hive，本地写java代码通过JDBC创建表。
- 5.全动态分区的方式向10min数据表中导入数据(贴出SQL语句和运行结果截图)。

THANKS

