05

Hive

- Hive简介
- Hive体系架构
- Hive数据模型
- Hive 命令

什么是Hive

Hive是基于Hadoop构建的一套**数据仓库**分析系统，它提供了丰富的SQL查询方式来分析存储在Hadoop分布式文件系统中的数据：

- 可以将结构化的数据文件映射为一张数据库表，并提供完整的SQL查询功能；
- 可以将SQL语句转换为MapReduce任务运行，通过自己的SQL查询分析需要的内容，这套SQL简称Hive SQL，使不熟悉MapReduce的用户可以很方便地利用SQL语言查询、汇总和分析数据。
- MapReduce开发人员可以把自己写的mapper和reducer作为插件来支持Hive做更复杂的数据分析。

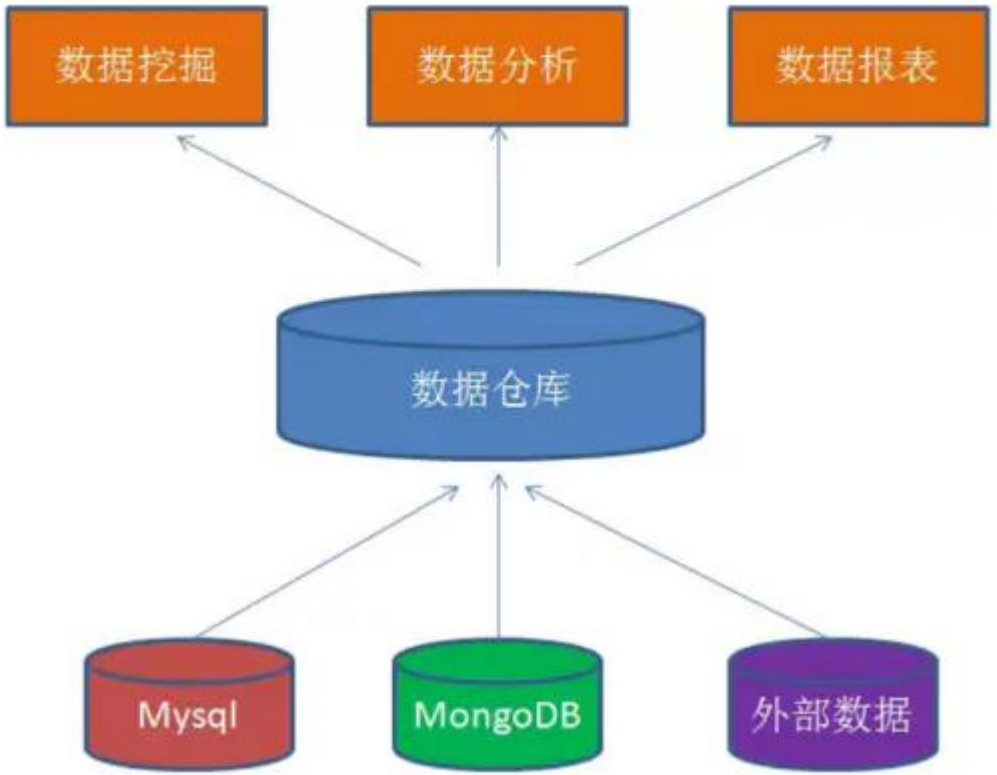
数据仓库(Data Warehouse，DW)

数据仓库是一个面向主题的(Subject Oriented)、集成的(Integrated)、相对稳定的(Non-Volatile)、反映历史变化(Time Variant)的数据集合，用于支持**管理决策**。

- **面向主题**，即数据仓库中表的设计是按照一个个主题进行组织的而非按照业务流程设计；
- **集成性**，是指将企业中各大业务系统进行数据集中、整合、加工从而形成全局统一的数据视图；
- **相对稳定**，则是指数据仓库中的数据不会做频繁的增删改操作，相对于业务系统中频繁的事务处理，其数据变化相对稳定；
- **反应历史变化**，表明数据仓库通常会保存数据的历史备份，因此就可以从中获取数据历史变化情况。

数据仓库

可以对多种业务数据进行筛选和整合，用于数据分析、数据挖掘、数据报表。

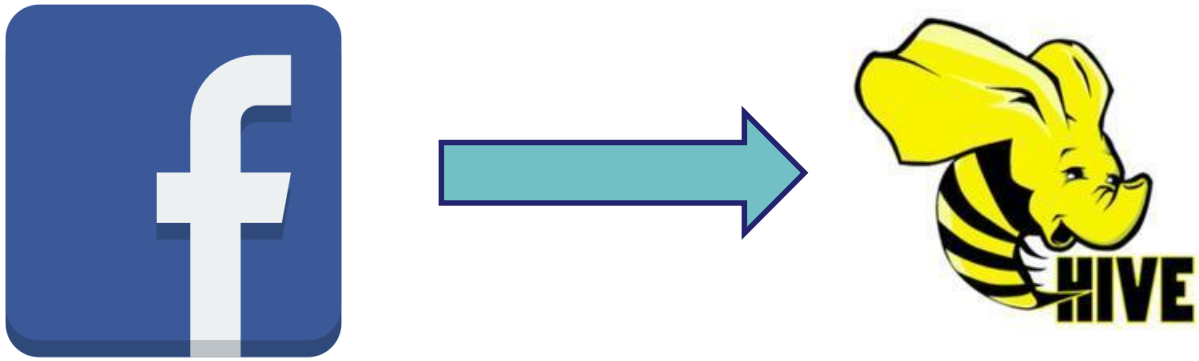


Hive与传统数据库

查询语言	HiveQL	SQL
数据存储位置	HDFS	本地FS
数据格式	用户定义	系统决定
数据更新	不支持	支持
索引	新版本有，但弱	有
执行	MapReduce	Executor
执行延迟	高	低
可扩展性	高	低
数据规模	大	小

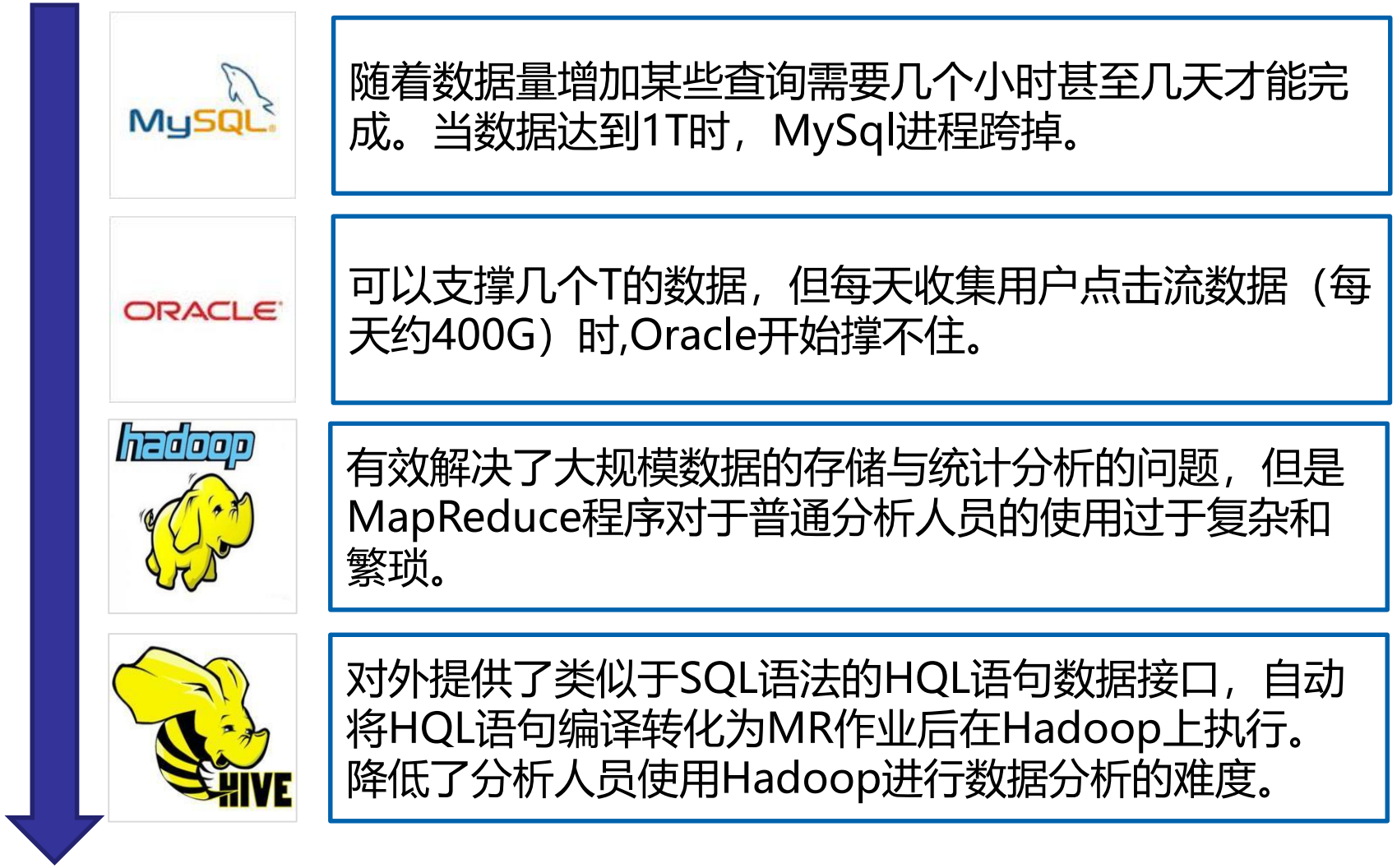
Hive的由来

- 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-2.3.7。



Hive的由来

- **Hadoop和Hive组建成为Facebook数据仓库的发展史**



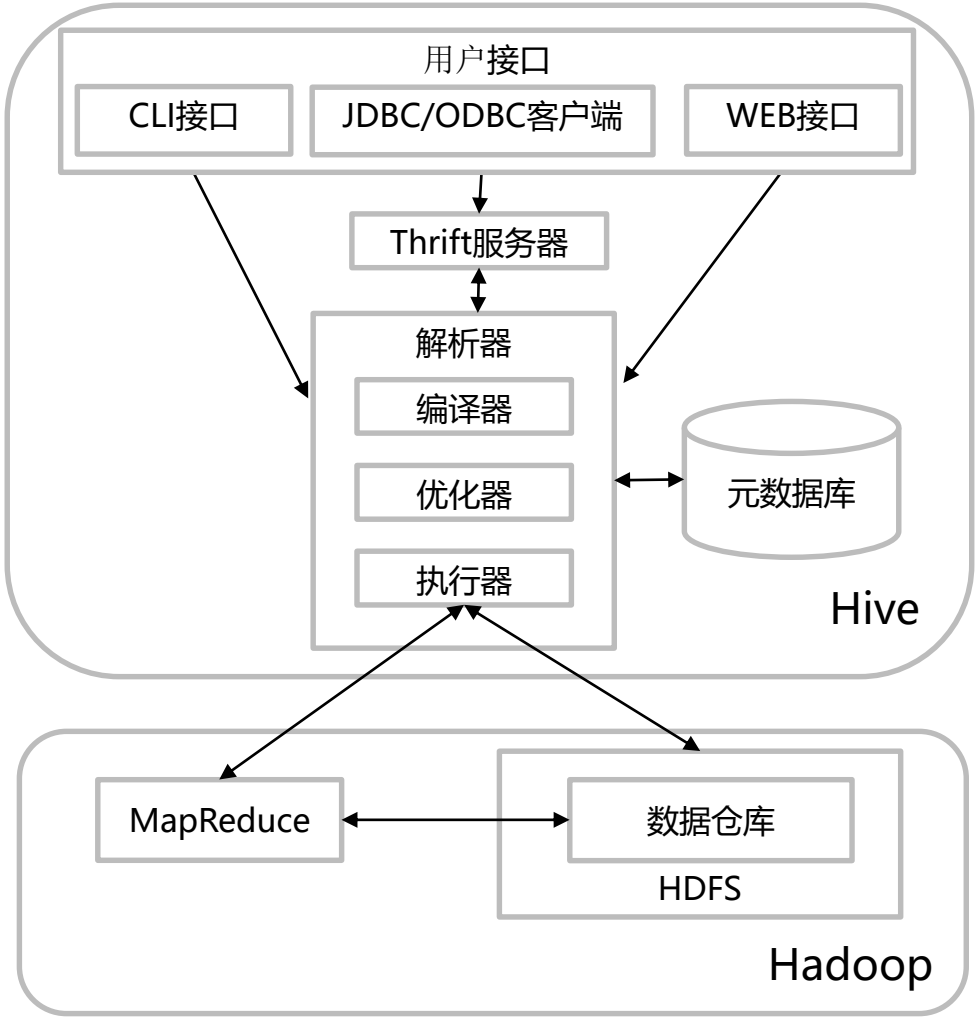
Hive体系结构-Hive设计特征

Hive 做为Hadoop 的数据仓库处理工具，它所有的数据都存储在Hadoop 兼容的文件系统中。Hive 在加载数据过程中不会对数据进行任何的修改，只是将数据移动到HDFS 中Hive 设定的目录下，因此，Hive 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确定的。Hive 的设计特点如下。

- 支持索引，加快数据查询。
- 不同的存储类型，例如，纯文本文件、HBase 中的文件。
- 将元数据保存在关系数据库中，减少了在查询中执行语义检查时间。
- 可以直接使用存储在Hadoop文件系统中的数据。
- 内置大量用户函数UDF 来操作时间、字符串和其他的数据挖掘工具，支持用户扩展UDF 函数来完成内置函数无法实现的操作。
- 类SQL 的查询方式，将SQL 查询转换为MapReduce 的job 在Hadoop集群上执行。
- 编码跟Hadoop同样使用UTF-8字符集。

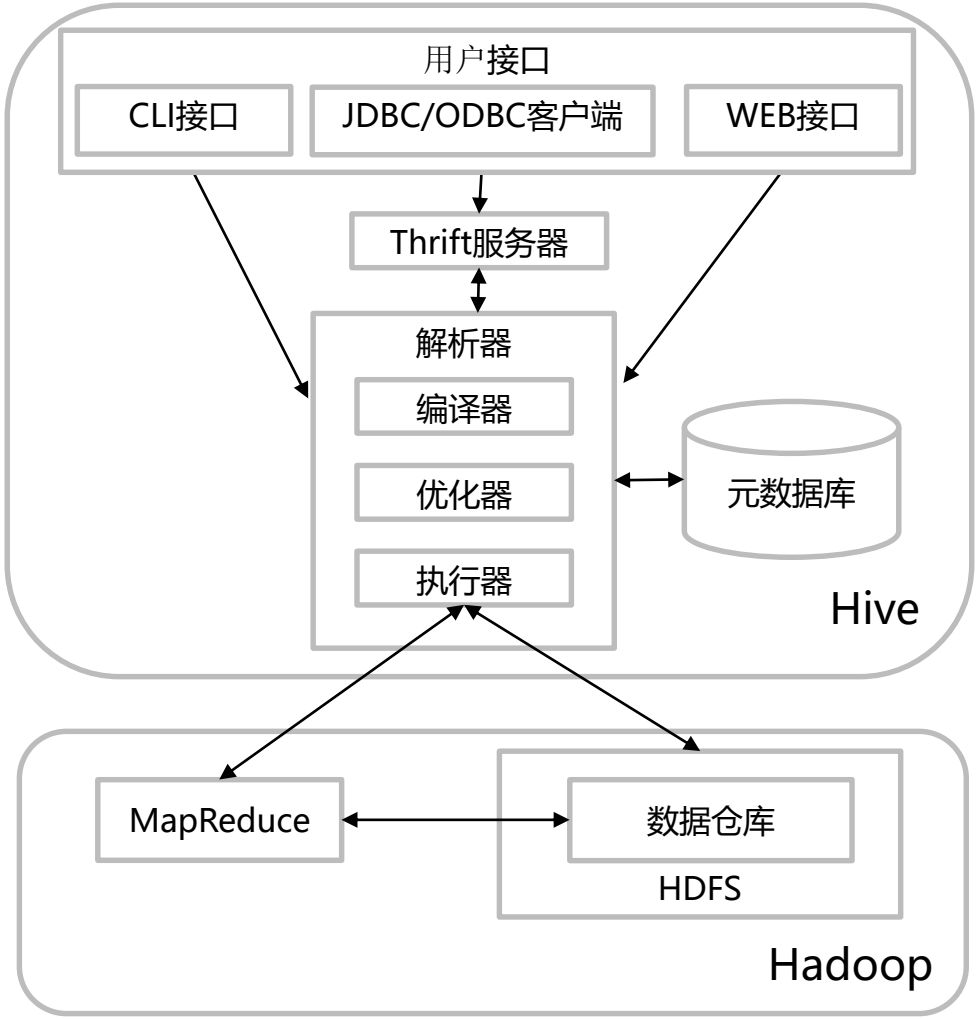
Hive体系结构

- 用户接口
  - CLI（命令行）：CLI 启动的时候，会同时启动一个 Hive 服务。
  - JDBC（Java数据库连接Java Database Connectivity）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（开放数据库连接Open Database Connectivity）客户端：ODBC驱动允许支持ODBC协议的应用程序连接到Hive。
  - Web UI 接口：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
  - 解释器、编译器、优化器完成 HQL 查询语句从词法分析、语法分析、编译、优化以及查询计划（plan）的生成。生成的查询计划存储在 HDFS 中，并在随后由 MapReduce 调用执行
  - 。
- 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、表的名字，表的列和分区及其属性，表的属性，表的数据所在目录等。
- Hadoop
  - Hive 的数据文件存储在 HDFS 中，大部分的查询由 MapReduce 完成。（对于包含 * 的查询，比如 select * from tbl 不会生成 MapRedcue 作业）



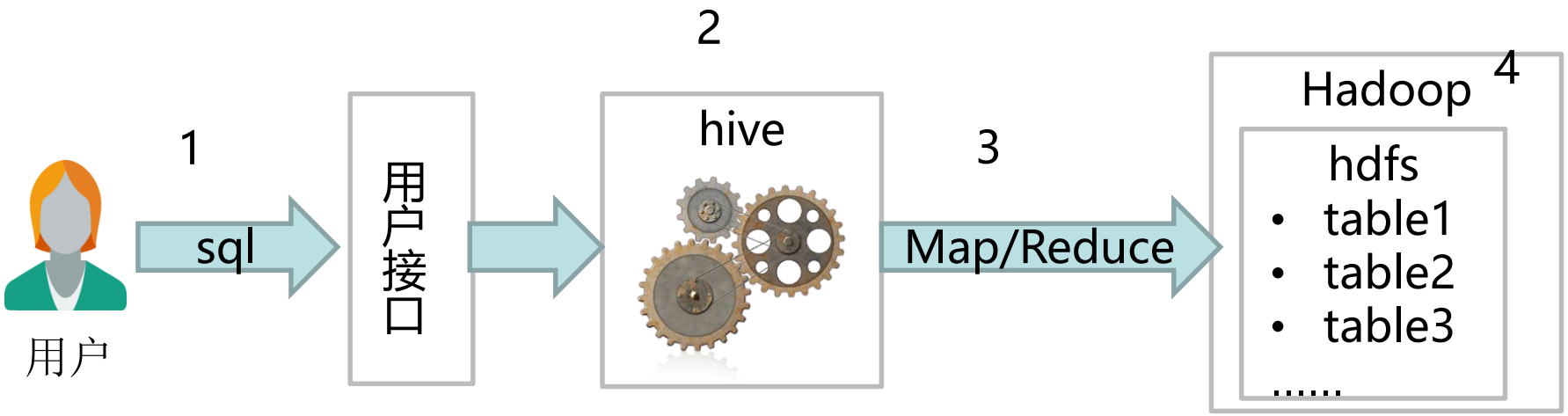
Hive的运行机制

① 用户通过用户接口连接Hive,发布Hive SQL

② Hive解析查询并制定查询计划

③ Hive将查询转换成MapReduce作业

④ Hive在Hadoop上执行MapReduce作业



Hive的优势

- 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的优势-上百行MR程序与一条HQL的对比

```
/**
 * 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
/**
 * 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 <i>maps</i>] [-r <i>reduces</i>] <i>in-dir</i> <i>out-dir</i>
 */
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
   * (<b>word</b>, <b>1</b>).
   */
  public static class MapClass extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }

                other_args.size() + " instead of 2.");
  return printUsage();
}
FileInputFormat.setInputPaths(conf, other_args.get(0));
```

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

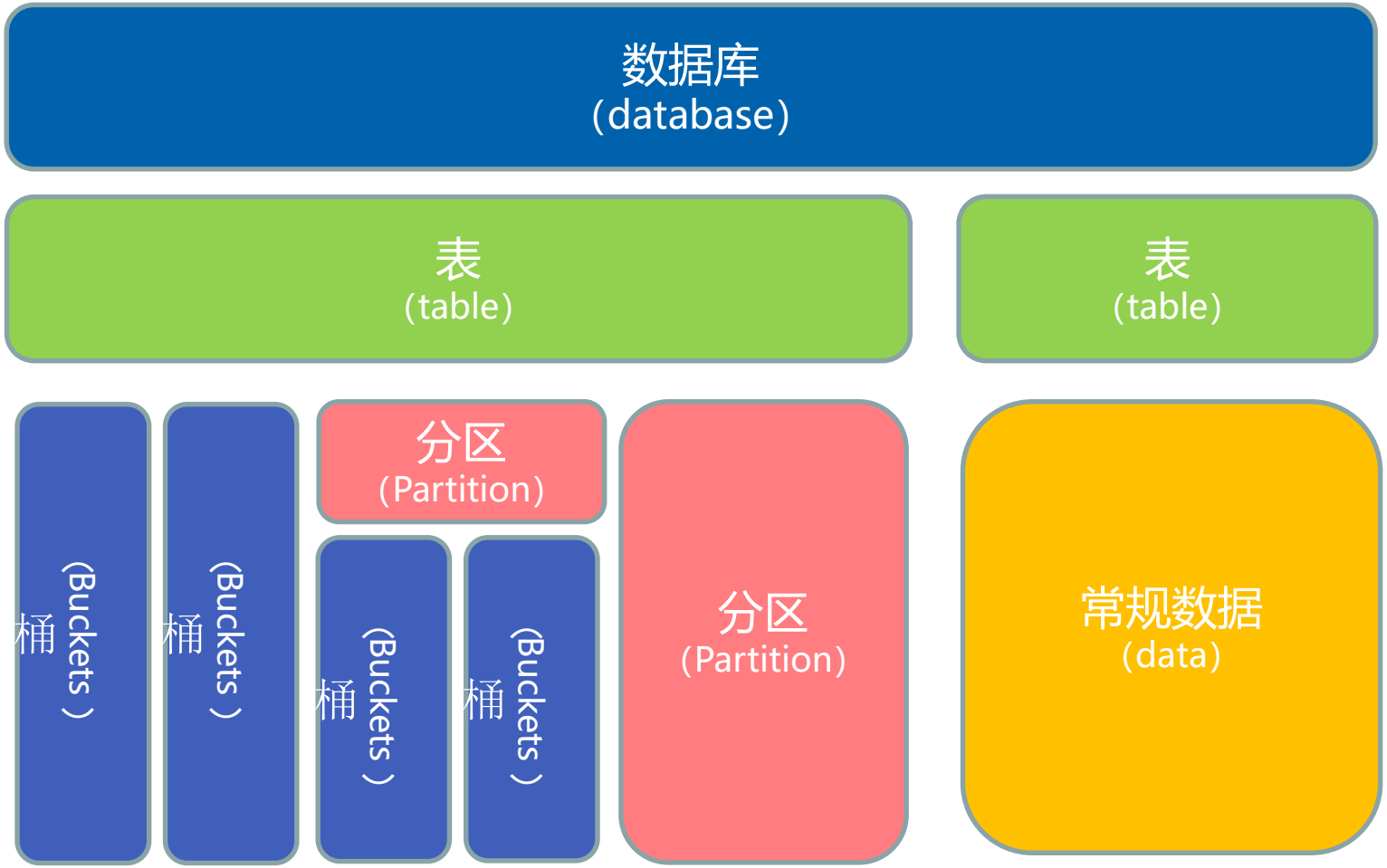
Hive的缺点

- 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。

Hive的应用场景

- **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

Hive的数据存储模型



Hive的数据模型

内部表

内部表与关系数据库中的Table在概念上类似。每一个Table在Hive中都有一个相应的目录存储数据。所有的Table数据（不包括External Table）都保存在这个目录中。删除表时，元数据与数据都会被删除。

Hive的数据模型

外部表

外部表指向已经在HDFS中存在的数据。它和内部表在元数据的组织上是相同的，而实际数据的存储则有较大的差异。内部表的创建过程和数据加载过程这两个过程可以分别独立完成，也可以在同一个语句中完成，在加载数据的过程中，实际数据会被移动到数据仓库目录中；之后对数据访问将会直接在数据仓库目录中完成。删除表时，表中的数据和元数据将会被同时删除。而外部表只有一个过程，加载数据和创建表同时完成（CREATE EXTERNAL TABLE ……LOCATION），实际数据是存储在LOCATION后面指定的 HDFS 路径中，并不会移动到数据仓库目录中。当删除一个External Table时，仅删除该链接。

Hive的数据模型

如何选择使用内部表或外部表?

- 如果所有处理都由Hive来完成，则使用内部表
- 如果需要用Hive和外部其他工具处理同一组数据集，则使用外部表。

Hive的数据模型

分区

Partition就是一种对表进行粗略划分的机制,可以实现加快查询速度的组织形式.在Hive中，表中的一个Partition对应于表下的一个目录，所有的Partition的数据都存储在对应的目录中。例如pvs表中包含ds和city两个Partition，则

对应于ds = 20090801, city= jinan 的HDFS子目录为：/wh/pvs/ds=20090801/city=jinan ；

对应于 ds = 20090801, city= qingdao 的HDFS子目录为：/wh/pvs/ds=20090801/city=qingdao 。

Hive的数据模型

桶

Buckets是将表的列通过Hash算法进一步分解成不同的文件存储。它对指定列计算Hash，根据Hash值切分数据，目的是为了并行，每一个Bucket对应一个文件。分区是粗粒度的划分，桶是细粒度的划分，这样做为了可以让查询发生在小范围的数据上以提高效率。适合进行表连接查询、适合用于采样分析。

例如将user列分散至32个bucket，首先对user列的值计算hash，则

对应hash值为0的HDFS目录为：/wh/pvs/ds=20090801/ctry=US/part-00000；

对应hash值为20的HDFS目录为：/wh/pvs/ds=20090801/ctry=US/part-00020。

Hive的数据模型

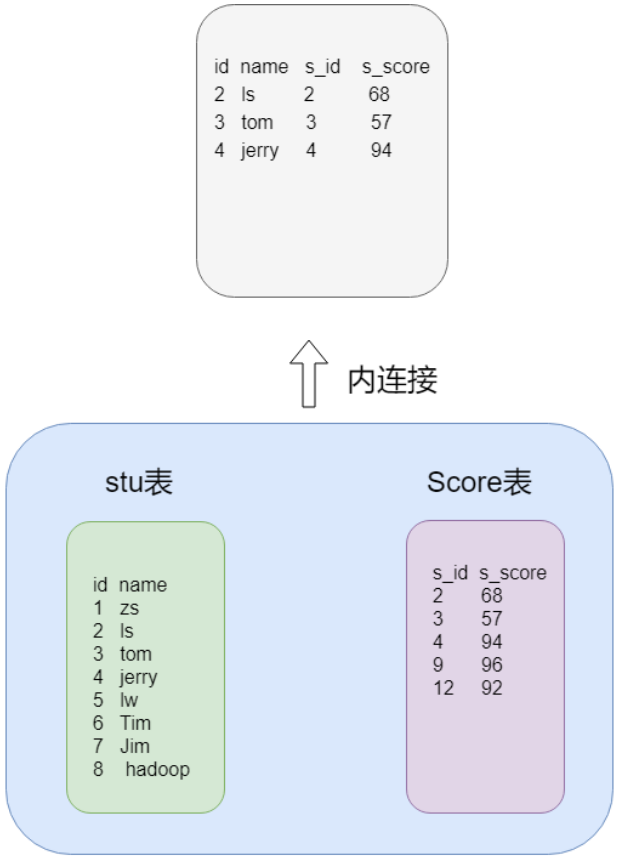
视图

视图与传统数据库的视图类似。视图是由从数据库的基本表中选取出来的数据组成的逻辑窗口，与基本表不同，它是一个虚表。在数据库中，存放的只是视图的定义，而不存放视图包含的数据项，这些项目仍然存放在原来的基本表结构中。

内连接 inner join

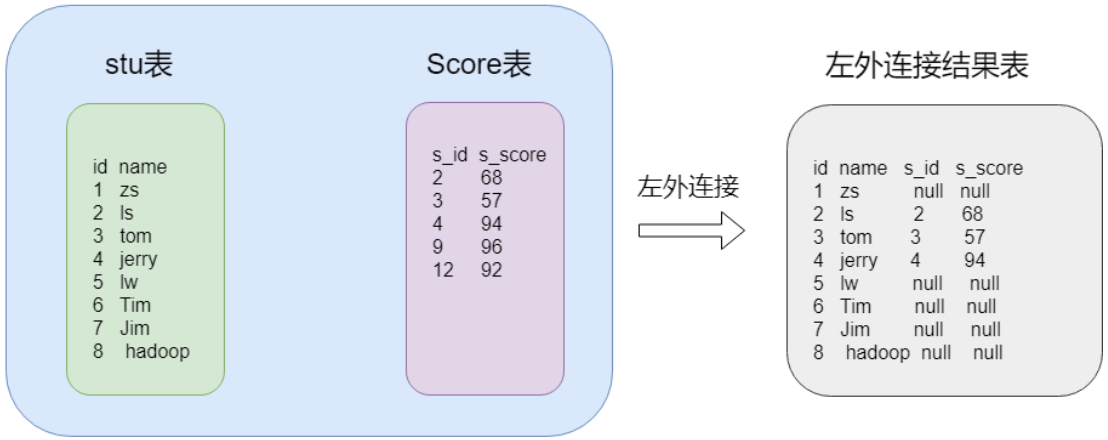
内连接：只有进行连接的两个表中都存在与连接条件相匹配的数据才会被保留下来。

join默认是inner join



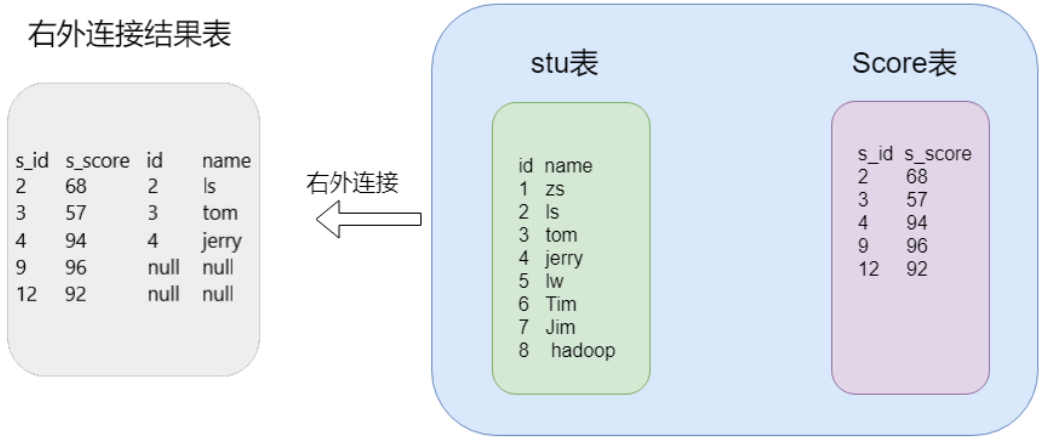
左外连接 left outer join

左外连接：左外连接是显示左边的表的所有数据，如果有右边表与之对应，则显示；否则显示null。



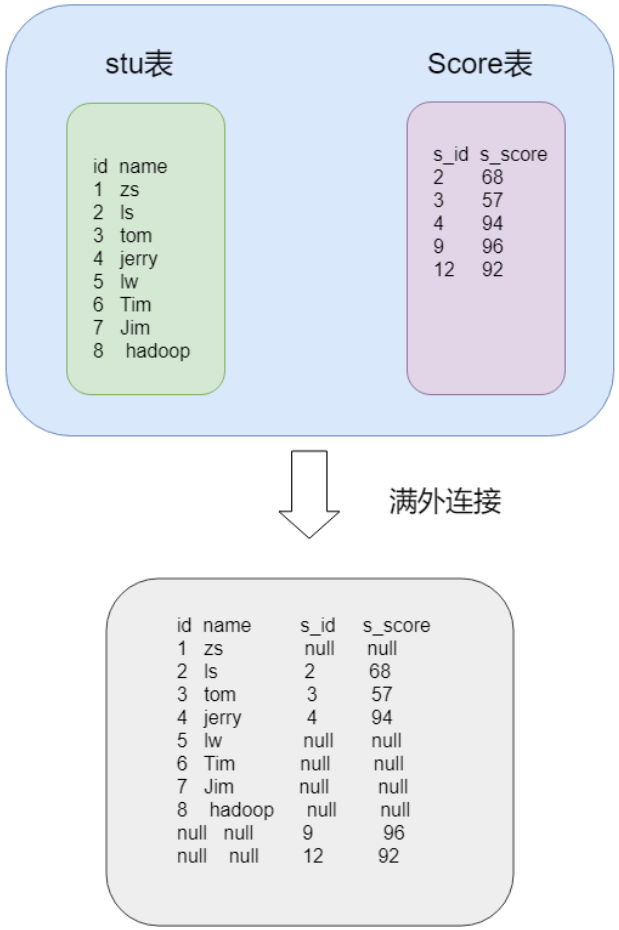
右外连接 right outer join

右外连接：右外连接是显示右边的表的所有数据，如果有左边表与之对应，则显示；否则显示null。



满外连接 full outer join

满外连接：两个表的数据都显示，如果没有对应的数据，则显示null.



Hive QL

Hive QL 是Hive支持的类SQL的查询语言。Hive QL大体可以分为DDL，DML，UDF三种类型。

DDL(Data Definition Language) 可以创建数据库，数据表，进行数据库和表的删除；

DML(Data Manipulation Language) 可以进行数据的添加、查询；UDF(User Defined Function) 支持用户自定义查询函数。

Hive创建数据表命令

- **CREATE TABLE** 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXIST 选项来忽略这个异常。
- **EXTERNAL** 关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION），
- 有分区的表可以在创建的时候使用 **PARTITIONED BY** 语句。一个表可以拥有一个或者多个分区，每一个分区单独存在一个目录下。
- 表和分区都可以对某个列进行 **CLUSTERED BY** 操作，将若干个列放入一个桶（bucket）中。
- 可以利用**SORTED BY** 对数据进行排序。这样可以为特定应用提高性能。
- 默认的字段分隔符为ascii码的控制符\001(^A)　tab分隔符为 \t。只支持单个字符的分隔符。
- 如果文件数据是纯文本，可以使用 **STORED AS TEXTFILE**。如果数据需要压缩，使用 **STORED AS SEQUENCE** 。

Hive创建数据表命令

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...)
  [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
  [ROW FORMAT row_format]
  [STORED AS file_format]
  [LOCATION hdfs_path]
```