法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象学院
  - 新浪微博：小象AI学院

大纲

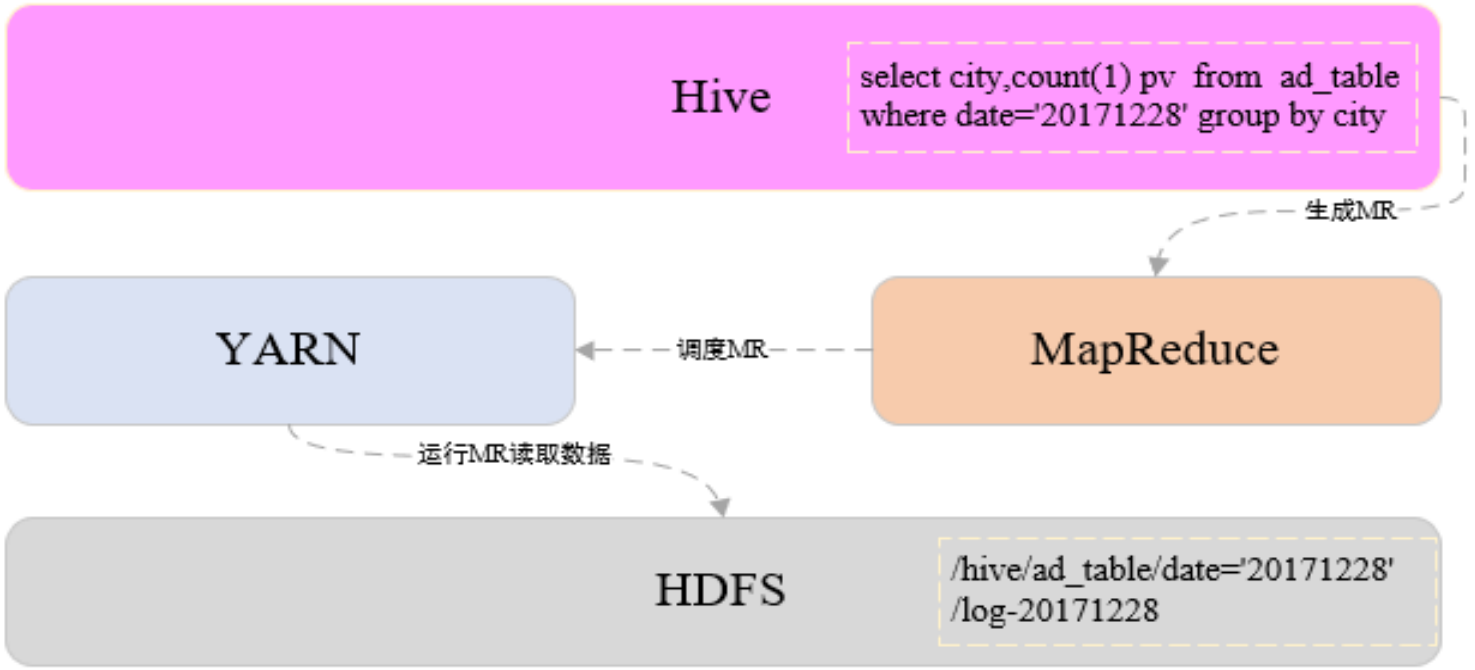
- Hive基本原理
- Hive配置安装
- Hive DDL和DML
- Hive函数
- Hive仓库的设计与实现

大纲

Hive基本原理

Hive简介

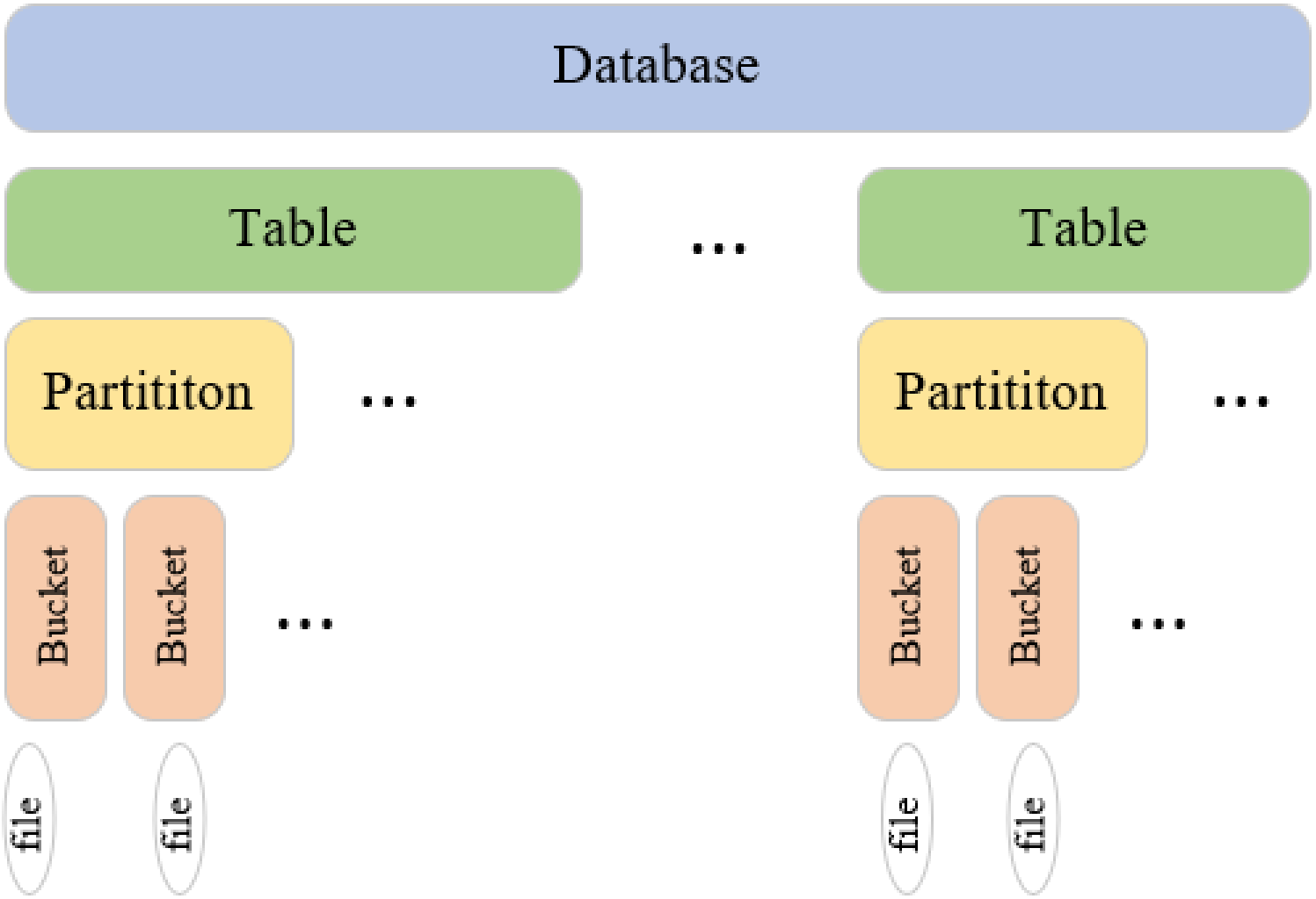
- Hive由Facebook开源，是一个构建在Hadoop之上的数据仓库
- 将结构化的数据映射成表
- 支持类SQL查询，Hive中称为HQL
- 无法实时更新，只支持向现有表中追加数据



使用Hive的原因

- 使用Hadoop构建数据仓库，数据分析所面临的问题
  - 人员培养、学习成本高，学习周期长
  - MapReduce实现复杂查询逻辑开发难度大，周期长
  - 开发速度无法快速满足业务发展
- 使用Hive的原因
  - 操作接口采用类SQL语法，提供快速开发的能力。
  - 避免了去写MapReduce，减少开发人员的学习成本，非开发人员也能够快速掌握
  - 使用灵活方便，扩展能力强

数据模型


Database
Table
...
Table
Partititon
...
Partititon
...
Bucket
Bucket
...
Bucket
Bucket
...
file
file
file
file

数据模型-分区

- 减少不必要的全表数据扫描
  - 对表使用分区，将表数据按照某个或某些字段划分
  - 分区在HDFS的表现为表路径下的不同文件目录
- 为了避免使用分区产生过多小文件，建议只对离散字段进行分区
  - 如日期、地域、类型等

数据模型-分桶

- 每一个表（table）或者分区， Hive可以进一步组织成桶，桶是更为细粒度的数据范围划分
  - hashcode(col_value) % nums_bucket
- 使用分桶的原因
  - 获得更高的查询处理效率
  - 使取样（sampling）更高效

常用文件格式

- TEXTFILE
  - 默认文件格式，建表时用户需要显示指定分隔符
  - 存储方式：行存储
- SequenceFile
  - 二进制键值对序列化文件格式
  - 存储方式：行存储
- 列式存储格式
  - RCFILE/ORC
  - 存储方式：列存储

常规数据类型

- 整数类型
  - TINYINT、SMALLINT、INT、BIGINT
  - 空间占用分别是1字节、2字节、4字节、8字节
- 浮点类型
  - FLOAT、DOUBLE
  - 空间占用分别是32位和64位浮点数
- 布尔类型BOOLEAN
  - 用于存储true和false
- 字符串文本类型STRING
  - 存储变长字符串，对类型长度没有限制
- 时间戳类型TIMESTAMP
  - 存储精度为纳秒的时间戳

复杂数据类型

- ARRAY
  - 存储相同类型的数据，可以通过下标获取数据
  - 定义：ARRAY<STRING>
  - 查询：array[index]
- MAP
  - 存储键值对数据，键或者值的类型必须相同，通过键获取值。
  - 定义：MAP<STRING,INT>
  - 查询：map[‘key’]
- STRUCT
  - 可以存储多种不同的数据类型，一旦声明好结构，各字段的位置不能够改变。
  - 定义：STRUCT<city:STRING, address :STRING,door_num:STRING>
  - 查询：struct.fieldname

大纲

Hive配置安装

Hive配置安装

- 元数据存储选择
  - 默认使用derby数据库，不能够多个用户同时使用，多用于测试
  - 使用MySQL数据库存储元数据，多用于生产环境
- HDFS数据仓库目录
  - 创建数据仓库目录

```
hadoop fs -mkdir -p /user/hive/warehouse
```

  - 所有用户具有写权限

```
hadoop fs -chmod a+w /user/hive/warehouse
```

  - 为用户赋予HDFS temp临时目录写权限

```
hadoop fs -chmod a+w /temp
```

Hive配置安装

- hadoop用户将Hive安装包解压到/home/hadoop/apps安装目录
  - tar -zxvf apache-hive-1.2.2-bin.tar.gz -C /home/hadoop/apps
- root用户
  - 创建软链接

    ln -s /home/hadoop/apps/hive-1.2.2 /usr/local/hive
  - 修改属主

    chown -R hadoop:hadoop /usr/local/hive
  - 添加环境变量

    vim /etc/profile

    添加如下内容：

    export HIVE_HOME=/usr/local/hive

    export PATH=$PATH:${HIVE_HOME}/bin

    重新编译使环境变量生效

    source /etc/profile

Hive配置安装

- 修改hive-1.2.2/conf/hive-site.xml配置文件内容

  vim hive-site.xml

  添加mysql数据库配置信息

```
<property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://node01:3306/hive?createDatabaseIfNotExist=true</value>
        <description>JDBC connect string for a JDBC metastore</description>
</property>
<property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
        <description>Driver class name for a JDBC metastore</description>
</property>
<property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>hive</value>
        <description>username to use against metastore database</description>
</property>
<property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>hive123</value>
        <description>password to use against metastore database</description>
</property>
```

Hive配置安装

- 将mysql驱动jar文件拷贝到${HIVE_HOME}/lib目录下
- 安装完成，启动hive

  /usr/local/hive/bin/hive

Hive Server2(Thrift服务)

- hive提供的跨语言Thrift服务，使用不同的语言通过JDBC或者ODBC都可以连接到hive
- 启动bin/hiveserver2，默认端口10000
- 使用beeline连接Hive
  - bin/ beeline
  - !connect jdbc:hive2://localhost:10000

```
[hadoop@node01 hive]$ bin/beeline
Beeline version 1.2.2 by Apache Hive
beeline> !connect jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Enter username for jdbc:hive2://localhost:10000: hadoop
Enter password for jdbc:hive2://localhost:10000: ******
Connected to: Apache Hive (version 1.2.2)
Driver: Hive JDBC (version 1.2.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://localhost:10000> show databases;
+----------------+--+
| database_name  |
+----------------+--+
| default        |
| ods            |
+----------------+--+
2 rows selected (2.984 seconds)
0: jdbc:hive2://localhost:10000>
```

大纲

Hive DDL和DML

Hive DDL

- 创建数据库

  create database [if not exists] database_name
  [comment database_comment]
  [location hdfs_path]

  - 例：创建数据库ods

    create database ods;

- 删除数据库

  drop database [if exists] database_name

- Hive默认数据库default

  查看数据库：show databases;

Hive DDL—创建表

- 创建表

```
create [external] table [if not exists] table_name
  [like existed_table]
  [(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]
```

Hive DDL—创建表

- 创建内部表
  - 创建学生信息表

```
create table student_info(
        student_id string comment '学号',
        name string comment '姓名',
        age int comment '年龄',
        origin string comment '地域'
)
comment '学生信息表'
row format delimited
fields terminated by '\t'
lines terminated by '\n'
stored as textfile;
```

Hive DDL—创建表

- 创建外部表
  - External关键词标识外部表
  - location关键字指定外部表HDFS数据路径

创建学生入学信息表，指定学生入学信息数据在HDFS的/user/hive/warehouse/data/student_school_info路径下：

```
create external table student_school_info(
        student_id string comment '学号',
        name string comment ‘姓名',
        institute_id string comment ‘学院ID',
        major_id string comment ‘专业ID',
        school_year string comment ‘入学年份',
)
row format delimited
fields terminated by '\t'
lines terminated by '\n'
stored as textfile
location '/user/hive/warehouse/data/student_school_info';
```

Hive DDL—创建表

- 创建分区表
  - 创建学生入学信息分区表
  - 字段信息：学号、姓名、学院ID、专业ID
  - 分区字段：入学年份

```
create table student_school_info_partition(
student_id string,
name string,
institute_id string,
major_id string
)
partitioned by(school_year string)
row format delimited
fields terminated by '\t'
lines terminated by '\n'
stored as textfile;
```

Hive DDL—创建表

- 创建分桶表
  - 创建学生入学信息分桶表
  - 字段信息：学号、姓名、学院ID、专业ID
  - 分桶字段：学号，4个桶，桶内按照学号升序排列

```
create table student_info_bucket(
student_id string,
name string,
age int,
origin string
)
clustered by (student_id) sorted by (student_id asc) into 4 buckets
row format delimited
fields terminated by '\t'
lines terminated by '\n'
stored as textfile;
```

Hive DDL—创建表

- 根据已存在的表创建新表
  - 关键字：LIKE

```
create table student_info2
like student_info;
```

- 创建新表的同时，将查询结果导入到表中
  - 关键字：AS

```
create table student_info3
as
select * from student_info;
```

Hive DDL—管理表

- 删除表

  drop table [if exists] tb_name

- 修改表
  - 表重命名

  alter table old_tb_name rename to new_tb_name
  - 为分区表添加分区

  可以同时添加多个分区，分区数据的hdfs存储路径为可选项

  alter table table_name add [if not exists] partition partition_spec [location 'location'][, partition partition_spec [location 'location'], ...]
  - 删除分区

  alter table table_name drop [if exists] partition partition_spec[, partition partition_spec, ...]

Hive DDL—管理表

- 修改表
  - 添加列
  - alter table table_name add columns (col_name data_type [comment col_comment], ...)

新添加的列字段会位于所有字段之后，在分区字段之前添加。

  - 删除或替换列

alter table table_name replace columns (col_name data_type [comment col_comment], ...)

Hive DML—数据导入

- 使用LOAD导入数据
  - 语法结构

load data [local] inpath 'filepath' [overwrite] into table tablename [partition (partcol1=val1, partcol2=val2 ...)]

  - 从本地加载数据文件到Hive表中

说明：本地文件会继续保留，复制一份数据文件到导入表的hdfs路径下

load data local inpath '/home/hadoop/apps/hive_test_data/data.txt' into table student_info

  - 从HDFS加载数据文件到Hive表中

说明：从HDFS加载数据到Hive表中会用原数据文件路径剪切到导入表的HDFS路径下

load data inpath '/data/hive_test_data/data.txt' into table student_info

Hive DML—数据导入

- 使用INSERT将查询结果插入Hive表
  - 语法结构

insert overwrite/into table tablename[partition (partcol1=val1, partcol2=val2 ...)]
select col1,col2, partcol1, partcol2 from from_statement

说明：

使用overwrite如果原分区中有数据会先删除原数据文件，然后将新的数据覆盖。

使用into不会删除原分区中的数据，会继续增加新的数据。

Hive DML—数据导入

- 将数据导入动态分区表
  - 不在执行语句中指定分区，通过查询结果自动创建分区

开启动态分区功能

```
set hive.exec.dynamic.partition=true;
```

表示允许所有分区都是动态的

```
set hive.exec.dynamic.partition.mode=nonstrict;
```

每个mapper或reducer可以动态创建的最大分区数，默认100

```
set hive.exec.max.dynamic.partitions.pernode=200;
```

一个动态分区创建语句可以创建的最大动态分区个数

```
set hive.exec.max.dynamic.partitions=1000;
insert overwrite table student_school_info_partition partition(school_year)
select t1.student_id,t1.name,t1.institute_id,t1.major_id,t1.school_year
from student_school_info t1;
```

Hive DML—数据导入

- 从一个表查询数据将结果写入到多个表中

```
from table_source
insert overwrite table tablename1 [partition (partcol1=val1, partcol2=val2 ...)] select_statement1
[insert overwrite table tablename2 [partition ...] select_statement2] ...
```

Hive DML—数据导入

- 向分桶表中导入数据
  - 声明执行分桶操作

    set hive.enforce.bucketing = true
  - 需要指定reduce个数与分桶的数量相同

    set mapreduce.job.reduces=4
  - 需要在从其他表查询数据过程中将数据按照分区字段插入各个桶中，cluster by 默认按照分桶字段在桶内升序排列，如果需要在桶内降序排列，使用distribute by (col) sort by (col desc)组合实现
  - 样例

```
set hive.enforce.bucketing = true;
set mapreduce.job.reduces=4;
insert overwrite table student_info_bucket
select student_id,name,age,origin
from student_info
cluster by(student_id);

set hive.enforce.bucketing = true;
set mapreduce.job.reduces=4;
insert overwrite table student_info_bucket
select student_id,name,age,origin
from student_info
distribute by (student_id) sort by (student_id desc);
```

Hive DML—数据导出

- 从一个表导出到一个输出目录

  insert overwrite [local] directory directory1 select ... from ...

- 从一个表导出到多个输出目录

  from from_statement

  insert overwrite [local] directory directory1 select_statement1

  [insert overwrite [local] directory directory2 select_statement2] ...

- 样例
  - 导出文件到本地，指定分隔符为’\t’

  insert overwrite local directory '/tmp/student_info/’

  row format delimited fields terminated by '\t’ select * from student_info
  - 导出数据到本地的常用方法

  hive -e"select * from rel.student_info"> ./student_info_data.txt

Hive DML—Select查询

- 语法结构

```
select [all | distinct] select_expr, select_expr, ...
from table_reference
[where where_condition]
[group by col_list [having condition]]
[sort by| order by col_list]
[limit number]
```

- distinct对重复记录去重
- where 目前支持 and,or,between…and…,in,not in等，不支持exist ,not exist
- order by 全局排序，只有一个reducer，当输入规模较大时，需要较长的计算时间
- sort by不是全局排序，其在数据进入reducer前完成排序。因此，如果用sort by进行排序，并且reduce task个数大于1，则sort by只保证每个reducer的输出有序，不保证全局有序

Hive JOIN连接

- join

  select * from a join b on a.id=b.id;

  两个表通过id关联，只把id值相等的数据查询出来。join的查询结果与inner join的查询结果相同。

- inner join

  select * from a inner join b on a.id=b.id;

- full join

  select * from a两个表通过id关联，把两个表的数据全部查询出来

  full join b on a.id=b.id;

.

Hive JOIN连接

- left join

  select * from a left join b on a.id=b.id;

  左连接时，左表中出现的join字段都保留，右表没有连接上的都为空

- right join

  select * from a right join b on a.id=b.id;

  右连接，右表中出现的join字段都保留，左表没有连接上的都是空

Hive JOIN连接

- left semi join

  左半连接实现了类似IN/EXISTS的查询语义，hive不支持in …exists这种关系型数据库中的子查询结构，hive暂时不支持右半连接。例如：

  select a.id, a.name from a where a.id in (select b.id from b);

  使用Hive对应于如下语句：

  select a.id, a.name from a left semi join b on (a.id = b.id)

Hive JOIN连接

- map side join

  使用分布式缓存将小表数据加载都各个map任务中，在map端完成join，map任务输出后，不需要将数据拷贝到reducer阶段再进行join，降低的数据在网络节点之间传输的开销。多表关联数据倾斜优化的一种手段。多表连接，如果只有一个表比较大，其他表都很小，则join操作会转换成一个只包含map的Job，例如：

  select /*+ mapjoin(b) */ a.id, a.name from a join b on a.id = b.id

大纲

Hive函数

常用运算符

- 关系运算符
  - 支持大多数关系运算符，在Hive中不等于表示方法A <> B
- 算数运算符
  - 常用的算数运算符：加（+）减（-）乘（*）除（/）取余（%）
- 逻辑运算符
  - 与AND、或OR、非NOT

常用函数

- 数值函数
  - 指定精度取整函数: round

语法：round(double a, int d)

返回值： double

说明：返回指定精度d的double类型（遵循四舍五入）

例如：

hive> select round(3.14,1);

3.1

hive> select round(3.16,1);

3.2

常用函数

- 日期函数
  - 时间戳转日期函数:from_unixtime

语法: from_unixtime(bigint unixtime, string format)

返回值: string

演示:

hive> select from_unixtime(1512813319,'yyyyMMdd');

20171209

常用函数

- 条件函数
  - 语法1：CASE a WHEN b THEN c [WHEN d THEN e]* [ELSE f] END

说明：如果a等于b，那么返回c；如果a等于d，那么返回e；否则返回f

例如：

hive> select case 1 when 2 then 'two' when 1 then 'one' else 'zero' end;

one

  - 语法2：CASE WHEN a THEN b [WHEN c THEN d]* [ELSE e] END

说明：如果a为TRUE，则返回b；如果c为TRUE，则返回d；否则返回e

例如：

hive> select case when 1=2 then 'two' when 1=1 then 'one' else 'zero' end;

one

常用函数

- 字符串函数
  - 字符串截取函数：substr

语法: substr(string str, int start,[len])

返回值: string

说明：返回字符串str从start位置到结尾的字符串,len是可选项，表示截取的字符串长度

演示：

```
hive> select substr('abcde',2);
bcde
hive> select substr('abcde',2,2);
bc
```

常用函数

➢ 集合统计函数

- 个数统计函数: count

语法: count(*), count(expr), count(DISTINCT expr[, expr_.])

返回值: int

说明: count(*)统计检索出的行的个数，包括NULL值的行；count(expr)返回指定字段的非空值的个数；count(DISTINCTexpr[, expr_.])返回指定字段的去重的非空值的个数

- 求和函数: sum

语法: sum(col), sum(DISTINCT col)

返回值: double

说明: sum(col)统计结果集中col的相加的结果；sum(DISTINCT col)统计结果中col不同值相加的结果

自定义UDF函数

当Hive提供的内置函数无法满足你的业务处理需要时，此时就可以考虑使用用户自定义函数（UDF：user-defined function）。UDF 作用于单个数据行，产生一个数据行作为输出。

步骤：

1. 先开发一个java类，继承UDF，并重载evaluate方法
2. 打成jar包上传到服务器
3. 在使用的时候将jar包添加到hive的classpath

```
hive>add jar /home/hadoop/HiveUdfPro-1.0-SNAPSHOT.jar;
```

4. 创建临时函数与开发好的java class关联

```
hive>create temporary function age_partition as
'cn.chinahadoop.udf.AgePartitionFunction’;
```

5. 即可在hql中使用自定义的函数

大纲

Hive仓库的设计与实现

数据仓库概念

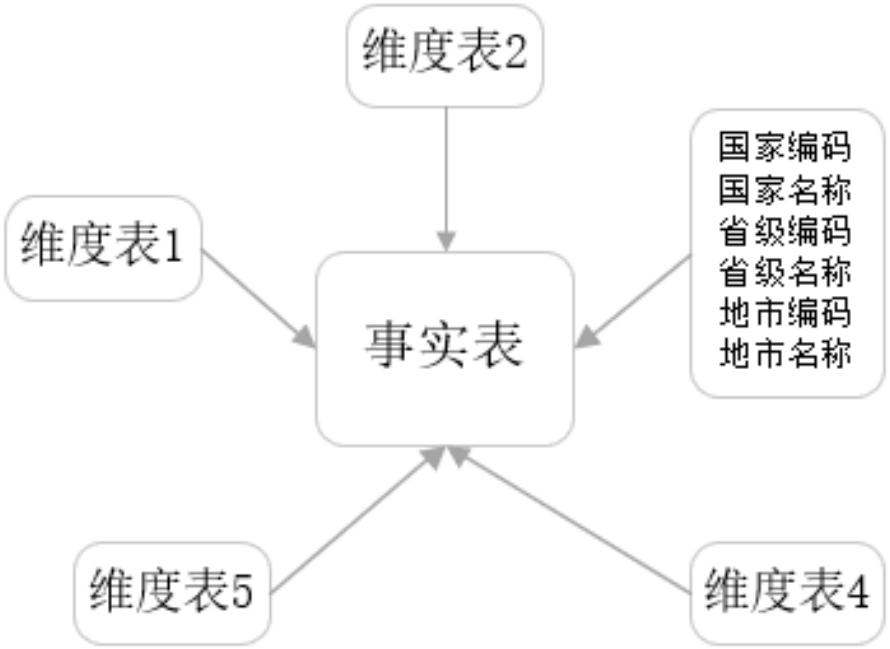
数据仓库，英文名称为Data Warehouse，可简写为DW或DWH。数据仓库，是为企业所有级别的决策制定过程，提供所有类型数据支持的战略集合。它是单个数据存储，出于分析性报告和决策支持目的而创建。为需要业务智能的企业，提供指导业务流程改进、监视时间、成本、质量以及控制。（源自百度百科数据仓库定义）

数据仓库特点

- 数据仓库的数据是面向主题的
- 数据仓库的数据是集成的
- 数据仓库的数据是不可更新的
- 数据仓库的数据是随时间不断变化的

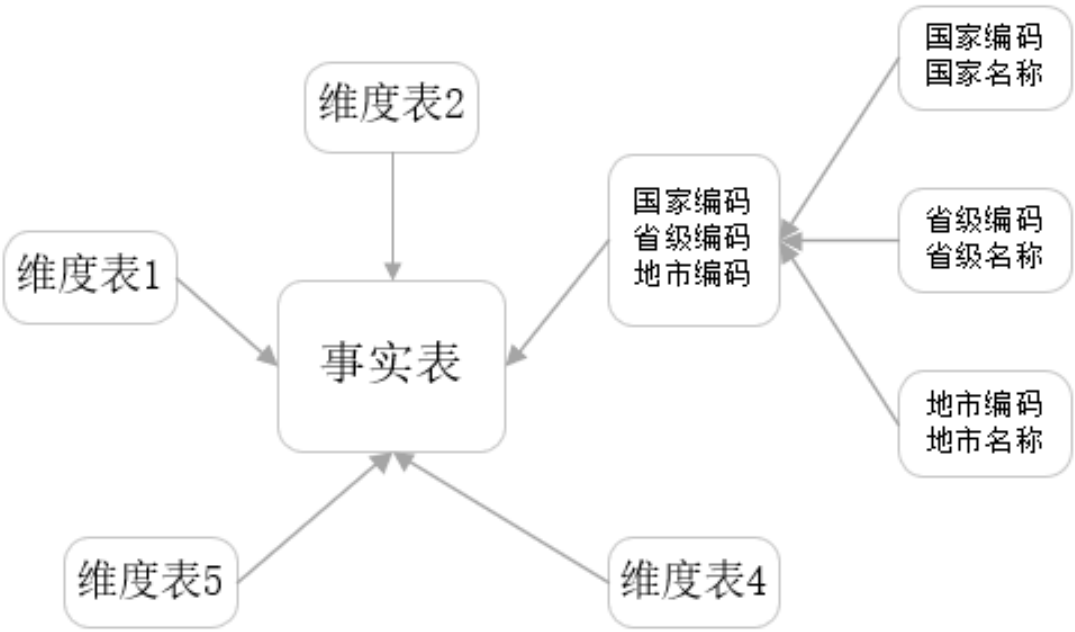
数据仓库模型-星型模型

- 当所有维表都直接连接到“ 事实表”上，就像星星一样，将这种模型称为星型模型
- 非正规化结构
- 不存在渐变维度
- 数据有一定的冗余



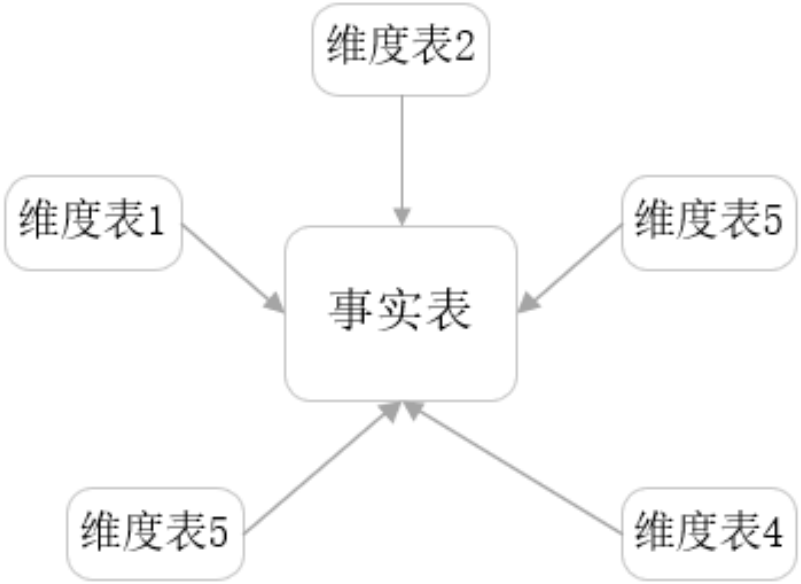
数据仓库模型-雪花模型

- 当有一个或多个维表没有直接连接到事实表上，而是通过其他维表连接到事实表上时，就像多个雪花连接在一起，将这种模型称为雪花模型。

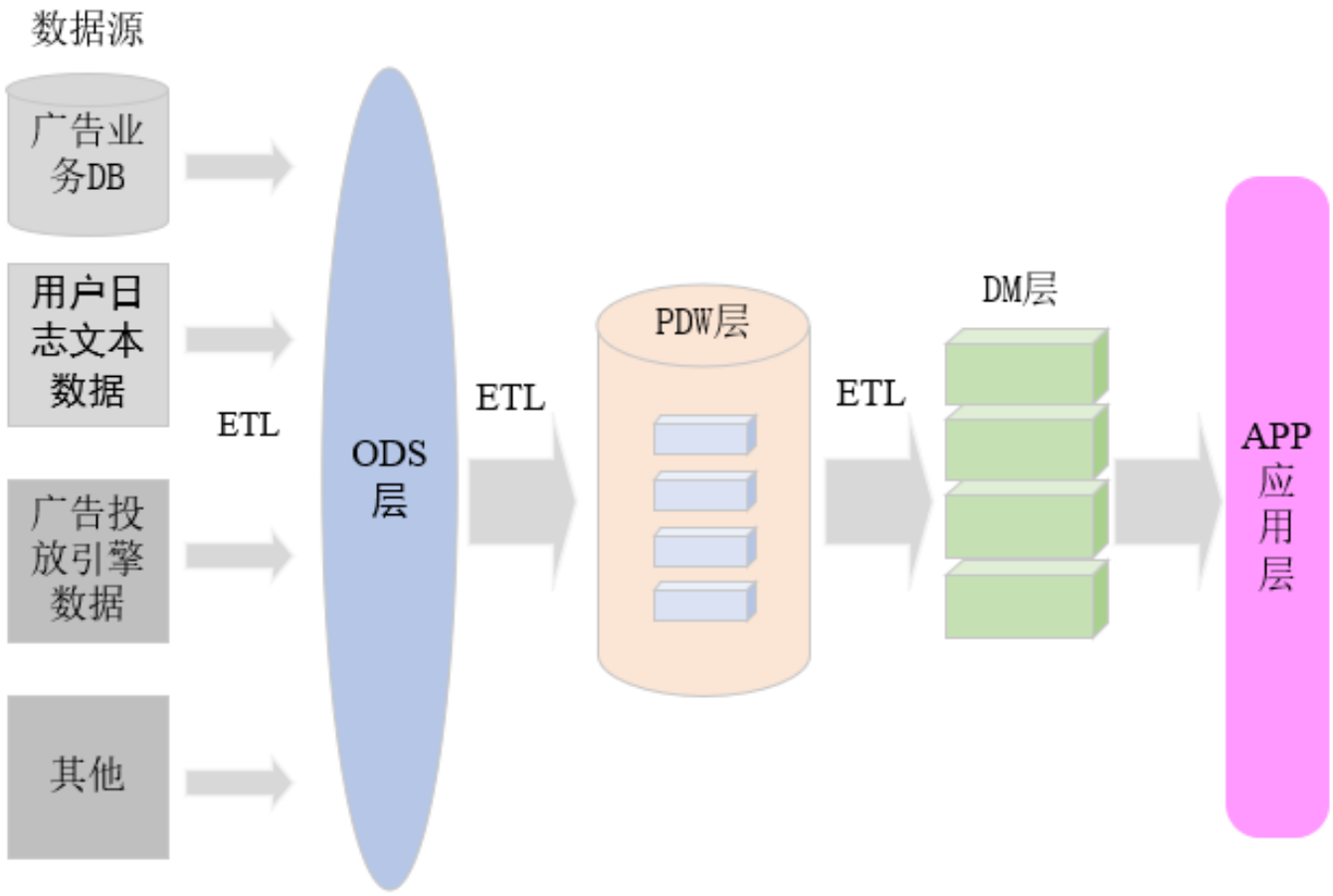


维度建模法

- 按照事实表，维表来构建数据仓库
- 使用星型模型
- 针对各个维度进行预处理，提升数据仓库处理能力
- 高效、直观



数据仓库架构


数据源
广告业务DB
用户日志文本数据
广告投放引擎数据
其他
ETL
ODS层
ETL
PDW层
ETL
DM层
APP应用层

疑问

- 小象问答官网
  - http://wenda.chinahadoop.cn

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象学院
- 新浪微博：小象AI学院