《Hive 数据仓库应用教程》

第六章　Hive数据查询

—— 强义原稿（jiǎng yì yuán gǎo）

适用对象：专科院校大数据专业学生

语言：中文授课（含拼音标注）

幻灯片 1 — 项目六 Hive数据查询（封面）

同学们好！

欢迎来到我们Hive数据仓库课程的第六次课。

今天我们要学的内容是——Hive数据查询（shù jù chá xún）。

前面几节课我们学了怎么建表、怎么加载数据，那今天我们就要学最重要的一个技能——怎么从表里把数据查出来。

查询是我们用Hive做数据分析（fēn xī）最核心的操作，大家一定要认真听。

今天的课程分两大部分：

第一部分是Hive基本查询，学习SELECT的各种用法；

第二部分是连接语句（lián jiē yǔ jù），学习JOIN操作，也就是怎么把多张表的数据合在一起查。

幻灯片 2 — 任务 6-1 Hive基本查询

好，我们先进入第一个任务——Hive基本查询。

这个任务里面，我们会从最基础的SELECT语句开始，一步一步学习各种查询技巧（jì qiǎo）。

包括怎么选列、怎么过滤（guò lǜ）数据、怎么分组、怎么排序等等。

幻灯片 3 — 6.1.1 任务描述

这张幻灯片给大家介绍了我们这次任务的整体目标（zhěng tǐ mù biāo）。

简单来说，我们要深入学习Hive这个工具。Hive是什么呢？它是一个建立在Hadoop上面的数据仓库工具。

你可以把它想象成：Hadoop是一个很大很大的仓库，而Hive就是帮你管理这个仓库、帮你找东西的管理员。

通过今天的学习，大家要掌握（zhǎng wò）Hive的查询功能。

以后在工作中，遇到大规模（guī mó）数据需要分析的时候，你就可以用Hive来完成了。

幻灯片 4 — 6.1.1 Hive SQL-DQL-Select查询数据 — 语法树

好，现在我们来看SELECT查询的完整语法结构（jié gòu）。

大家看屏幕上的这个语法树，我来一个一个解释（jiě shì）。

语法树讲解：

首先，有几个重要的关键字：

SELECT — 就是"选择"，你要查什么列，就写在SELECT后面。

FROM — 就是"从哪里"查，后面跟表名。

WHERE — 就是"条件"，满足什么条件的数据才要。

GROUP BY — 就是"分组"，把数据按照某个字段分成几组。

ORDER BY — 就是"排序"，让结果按顺序排列。

LIMIT — 就是"限制"，只要前面几条数据。

> 💡 *记住两个要点：1）FROM后面可以是普通表、视图（shì tú）、子查询结果；2）表名和列名不区分大小写，也就是说写大写小写都一样。*

这个语法结构虽然看起来很多，但不用害怕，我们会一个一个来学。

幻灯片 5 — 案例：美国Covid-19新冠数据之select查询

为了让大家更好地练习SELECT语法，我们准备了一份真实的数据。

这份数据是美国Covid-19新冠疫情（yì qíng）的数据，文件名叫《us-covid19-counties.dat》。里面记录的是2021年1月28号，美国各个县（xiàn）的累计确诊（què zhěn）病例数和累计死亡病例数。

用真实数据来练习，大家会更有感觉，而且以后工作中处理的也都是真实数据。

幻灯片 6 — 数据环境准备(1) — 创建数据库和普通表

好，现在我们来做数据准备。大家跟着我一步一步操作。

第一步，我们需要创建一个数据库和一张普通表。

> **【实操练习】 创建数据库和表 t_usa_covid19**

```
-- 启动Hive
hive-bee

-- 创建数据库
create database unit6db;
show databases;
use unit6db;
show tables;
```

先启动Hive，然后创建一个叫unit6db的数据库，用use切换进去。

> **【实操练习】 创建表**

```
drop table if exists t_usa_covid19;
CREATE TABLE t_usa_covid19(
  count_date string,
  county string,
  state string,
  fips int,
  cases int,
  deaths int)
row format delimited
fields terminated by ',';
```

这张表有6个字段（zì duàn）：

count_date — 统计日期；

county — 县名；

state — 州名；

fips — 地区编码；

cases — 累计确诊数；

deaths — 累计死亡数。

字段之间用逗号（dòu hào）分隔，所以我们用 fields terminated by ','。

【实操练习】 加载数据并查看

```
-- 将数据文件加载到表中
load data local inpath '/home/hadoop/us-covid19-counties.dat'
into table t_usa_covid19;

-- 查看前10条数据
SELECT * FROM t_usa_covid19 LIMIT 10;
```

load data命令就是把本地的数据文件加载到Hive的表里。加载完之后，我们用SELECT * LIMIT 10查看前10条数据，确认（què rèn）数据是否加载成功。

幻灯片 7 — 数据环境准备(2) — 创建分区表

第二步，我们要创建一张分区表（fēn qū biǎo）。

什么是分区表？简单说，就是把数据按照某个条件分成不同的"文件夹"存放，查询的时候只需要看需要的文件夹就行了，不用看全部数据，这样查询速度会快很多。

我们按照count_date日期和state州来分区。

> **【实操练习】 创建分区表 t_usa_covid19_p**

```
CREATE TABLE if not exists t_usa_covid19_p(
  county string,
  fips int,
  cases int,
  deaths int)
partitioned by(count_date string, state string)
row format delimited fields terminated by ',';
```

注意看，分区表和普通表的区别（qū bié）：

普通表里count_date和state是普通的列；

分区表里count_date和state变成了分区字段，写在partitioned by后面。

创建完成后用show tables查看，应该能看到t_usa_covid19和t_usa_covid19_p两张表。

幻灯片 8 — 数据环境准备(3) — 动态分区插入数据

第三步，我们把普通表的数据导入到分区表里。

这里用到一个技巧（jì qiǎo）叫"动态分区插入"（dòng tài fēn qū chā rù）。
意思是：Hive会自动根据数据的值来决定放到哪个分区，不需要我们手动指定。

> **【实操练习】 动态分区插入**

```
-- 开启动态分区模式
set hive.exec.dynamic.partition.mode = nonstrict;

-- 从普通表插入数据到分区表
insert into table t_usa_covid19_p
partition (count_date, state)
select county, fips, cases, deaths, count_date, state
from t_usa_covid19;

-- 查看分区表数据
select * from t_usa_covid19_p limit 10;
```

注意：select后面的字段顺序很重要！前面4个是表的普通字段（county, fips, cases, deaths），后面2个是分区字段（count_date, state），要放在最后面。

这个操作需要一定时间，大概100多秒，因为它需要启动MapReduce任务来处理数据。

执行完之后，查看结果，应该能看到每条数据最后多了count_date和state两个字段。比如：Autauga县，5554个确诊，69个死亡，日期2021-01-28，州是Alabama。

幻灯片 9 — 6.1.2 Hive SQL-DQL-Select语法树 — 基础语法

现在数据准备好了，我们正式开始学习SELECT查询的各个部分。

大家再看一下这个语法树，我用黄色框框标出了最基础的两个部分：
第一个是最上面的 SELECT ... FROM ... WHERE ... GROUP BY 这一块，这是基础语法；
第二个是最下面的 LIMIT，用来限制返回行数。

接下来我们一个一个来学这些关键字的用法。

幻灯片 10 — 6.1.3 select_expr（选择表达式）

首先是select_expr，也就是SELECT后面跟的那些内容。

select_expr就是告诉Hive：你要返回哪些列。至少要写一个。

来看几个例子：

> **【实操练习】 select_expr 各种用法**

```
-- 查询所有字段（用星号*表示所有）
select * from t_usa_covid19_p;

-- 查询指定字段
select county, cases, deaths from t_usa_covid19_p;

-- 用正则表达式匹配列名（以c开头的所有列）
SET hive.support.quoted.identifiers = none;
select `^c.*` from t_usa_covid19_p;

-- 查询当前使用的数据库（不需要FROM）
select current_database();

-- 使用函数，比如count统计总行数
select count(county) from t_usa_covid19_p;
```

星号*就像点菜的时候说"全部都要"，而指定字段就像说"我只要这几个菜"。

在实际工作中，建议尽量不要用*，因为数据量很大的时候，查所有字段会非常慢（màn）。

幻灯片 11 — ALL 和 DISTINCT（去重）

接下来学习ALL和DISTINCT这两个关键字。

它们是用来处理重复（chóng fù）数据的。

ALL — 返回所有行，包括重复的。这是默认值，不写就等于写了ALL。

DISTINCT — 去掉重复的行，只保留不一样的。

举个生活中的例子：

你去超市买东西，收银台打印的小票上列出了所有商品，包括你买了3瓶水，小票上就会有3行"水"。这就是ALL。

但如果你只想看你买了哪些种类的商品，不管买了几个，那就是DISTINCT。

【实操练习】 ALL 和 DISTINCT 使用

```
-- 返回所有州名（包含重复值）
select state from t_usa_covid19_p;

-- 相当于使用ALL关键字（默认行为）
select all state from t_usa_covid19_p;

-- 去除重复，只返回不同的州名
select distinct state from t_usa_covid19_p;

-- 多个字段组合去重
select distinct county, state from t_usa_covid19_p;
```

💡 *多字段使用DISTINCT时，是整体组合去重，也就是说county和state的组合不重复就行。*

幻灯片 12-13 — WHERE子句（条件过滤）

WHERE是用来过滤数据的，就像筛子（shāi zi）一样，只留下满足条件的数据。

WHERE后面跟的是一个布尔表达式（bù ěr biǎo dá shì），也就是结果是true或false的表达式。只有结果为true的行才会被返回。

重要规则：

在WHERE中可以使用各种函数和运算符（yùn suàn fú）。

但是！不能使用聚合函数（jù hé hán shù），比如count()、sum()这些。

为什么不能用聚合函数呢？

打个比方：WHERE就像是"海选"阶段，还在一个一个挑人呢，你让它统计"一共有多少人"，它还没挑完，怎么统计？

聚合函数需要在数据已经确定了之后才能用。

【实操练习】 WHERE子句使用

```
-- 简单条件：1 > 2 返回false，所以没有结果
select * from t_usa_covid19_p where 1 > 2;

-- 简单条件：1 = 1 返回true，所以返回所有数据
select * from t_usa_covid19_p where 1 = 1;

-- 在where中使用函数：找出州名长度超过10的记录
SELECT * from t_usa_covid19_p where length(state) > 10;

-- where子句支持子查询
SELECT * FROM A WHERE A.a IN (SELECT foo FROM B);

-- 错误示范：where中不能用聚合函数！
-- 下面这个会报错：SemanticException
select state, count(deaths)
from t_usa_covid19_p where count(deaths) > 100
group by state;

-- 正确做法：用HAVING（后面会学）
select state, count(deaths)
from t_usa_covid19_p
group by state
HAVING count(deaths) > 100;
```

幻灯片 14 — 分区查询、分区裁剪（cái jiǎn）

前面我们建了分区表，那分区表查询有什么特别的地方呢？

答案就是——分区裁剪（fēn qū cái jiǎn）。

什么意思呢？举个例子：

假如一个图书馆有100个书架，你要找一本计算机的书。如果没有分区，你要从第1个书架找到第100个书架。但如果图书馆按类别（lèi bié）分了区，计算机的书都在第5个书架上，你只需要看第5个书架就行了。这就是分区裁剪——直接跳过不需要的分区。

在Hive中，当你在WHERE或JOIN的ON条件中用到了分区字段（比如我们的state或count_date），Hive就会自动进行分区裁剪，只读取相关分区的数据。

【实操练习】 分区裁剪示例

```
-- 找出加州California，累计死亡超过1000的县
-- state是分区字段，会自动进行分区裁剪，避免全表扫描
select * from t_usa_covid19_p
where state = "California" and deaths > 1000;

-- 多分区裁剪：同时指定日期和州
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California" and deaths > 1000;
```

💡 *使用分区裁剪可以大大提高查询速度，是实际工作中非常重要的优化（yōu huà）技巧！*

幻灯片 15-16 — GROUP BY（分组）

GROUP BY是用来分组的，通常和聚合函数一起使用。

什么是分组？举个例子：

班里有30个同学，你想知道男生的平均分和女生的平均分。这时候你就需要先按性别分组，然后再分别计算平均分。这就是GROUP BY + 聚合函数。

大家看幻灯片上的这张图：

左边是原始数据，有day、category、value三列。

按照category进行Group By之后，相同颜色的数据被分到了同一组：

— Blue（蓝色）的放在一起；

— Green（绿色）的放在一起；

— Red（红色）的放在一起。

GROUP BY的语法限制（很重要！）：

SELECT里面出现的字段，只能是两种之一：

第一种：GROUP BY后面的分组字段本身；

第二种：被聚合函数（如count、sum、avg等）包裹（bāo guǒ）的字段。

为什么有这个限制？

因为分组之后，同一组里的分组字段值都是一样的，返回没有问题。但如果你选了一个不是分组字段、也没有用聚合函数的字段，那这一组里这个字段可能有好几个不同的值，Hive不知道该返回哪个，就会报错（bào cuò）。

比如按category分组后，如果你在SELECT里写了day字段但没有用聚合函数，那同一组里有好几个不同的day值，Hive不知道返回哪个day，所以会报错。

幻灯片 17 — GROUP BY 使用示例

来看具体的例子：

【实操练习】 GROUP BY 使用

```
-- 错误示范！deaths不是分组字段，也没用聚合函数
-- 会报错：SemanticException: Expression not in GROUP BY key 'deaths'
select state, deaths
from t_usa_covid19_p where count_date = "2021-01-28"
group by state;

-- 正确写法：deaths用聚合函数count()包裹
select state, count(deaths)
from t_usa_covid19_p where count_date = "2021-01-28"
group by state;
```

第一个查询会报错，因为deaths既不是分组字段，也没有被聚合函数包裹。

第二个查询就对了，count(deaths)是一个聚合函数，意思是"统计每个州有多少条死亡记录"。

幻灯片 18 — HAVING（分组后过滤）

刚才我们说了，WHERE里面不能用聚合函数。那如果我分组之后，想要过滤掉某些组怎么办呢？

答案就是用HAVING！

HAVING是在分组之后对数据进行过滤的，并且可以使用聚合函数。

因为到了HAVING这一步，WHERE和GROUP BY都已经执行完了，数据结果集已经确定了，所以可以用聚合函数。

> **【实操练习】 HAVING 使用**

```
-- 错误示范：在where中使用sum聚合函数会报错
select state, sum(deaths)
from t_usa_covid19_p
where count_date = "2021-01-28" and sum(deaths) > 10000
group by state;

-- 正确写法：先where过滤，再group by分组，最后having过滤
select state, sum(deaths)
from t_usa_covid19_p
where count_date = "2021-01-28"
group by state
having sum(deaths) > 10000;

-- 更好的写法：用别名（bié míng），避免重复计算
select state, sum(deaths) as cnts
from t_usa_covid19_p
where count_date = "2021-01-28"
group by state
having cnts > 10000;
```

最后一种写法用了as关键字给sum(deaths)起了一个别名叫cnts，然后在having里直接用cnts > 10000，这样更简洁（jiǎn jié）。

幻灯片 19 — HAVING与WHERE的区别

来做一个简单的总结对比（duì bǐ）：

HAVING vs WHERE 区别：

WHERE — 在分组前过滤数据（先过滤再分组）

HAVING — 在分组后过滤数据（先分组再过滤）

WHERE — 后面不可以使用聚合函数

HAVING — 后面可以使用聚合函数

用一句话记住：WHERE管"个体"，HAVING管"群体"。

就像选拔运动员：WHERE是海选阶段，先把不合格的个人筛掉；HAVING是分组赛阶段，把成绩不好的组淘汰。

幻灯片 20-21 — LIMIT（限制返回行数）

LIMIT是用来限制查询返回多少行数据的。

LIMIT可以接受一个或两个参数（cān shù）：

一个参数：表示返回的最大行数，偏移量默认为0。比如LIMIT 5就是返回前5行。

两个参数：第一个是偏移量（piān yí liàng），第二个是最大行数。比如LIMIT 2, 3表示从第2行开始（注意从0开始计数），返回3行。

> **【实操练习】 LIMIT 使用**

```
-- 没有限制，返回2021.1.28加州的所有记录
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California";

-- 返回结果集的前5条
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
limit 5;

-- 从第2行开始（偏移量从0开始），返回3行
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
limit 2, 3;
```

> 💡 *LIMIT在处理大量数据时特别有用。比如你只想看看数据长什么样，不需要把几百万条数据都查出来，用LIMIT 10就够了。*

幻灯片 22 — SQL执行顺序（很重要！）

这个知识点非常重要，大家要记住！

SQL的执行顺序：

from → where → group（含聚合）→ having → order → select

注意！这个顺序和我们写SQL语句的顺序是不一样的。

我们写的时候是先写SELECT，再写FROM，再写WHERE...

但Hive执行的时候，是先执行FROM确定（què dìng）查哪张表，然后WHERE过滤，然后GROUP BY分组和聚合计算，然后HAVING过滤组，然后ORDER BY排序，最后SELECT选择要返回的列。

这也解释了为什么WHERE里不能用聚合函数——因为WHERE执行的时候，GROUP BY还没执行呢，聚合结果还没有出来。

用下面这个SQL来感受一下执行顺序：

```
select state, sum(deaths) as cnts
from t_usa_covid19_p            -- ① 先确定表
where count_date = "2021-01-28" -- ② 再过滤
group by state                   -- ③ 再分组+聚合
having cnts > 10000;             -- ④ 再过滤组
```

幻灯片 23 — 6.1.4 高阶语法概览（gài lǎn）

好，基础语法我们已经学完了。现在进入高阶（gāo jiē）语法部分。

高阶语法主要包括以下几个内容：

ORDER BY — 全局排序

CLUSTER BY — 分组加排序

DISTRIBUTE BY + SORT BY — 更灵活的分组和排序

WITH（CTE） — 公用表表达式

另外还有UNION联合查询和子查询（zǐ chá xún）。让我们一个一个来学。

幻灯片 24-25 — ORDER BY（全局排序）

ORDER BY用来对查询结果进行全局排序（pái xù）。

重要特点：

默认是升序（shēng xù）ASC，也就是从小到大。可以指定为DESC降序（jiàng xù），从大到小。

因为是全局排序，所以在MapReduce执行时只有一个reduce任务，数据量大的时候会非常慢。

对于NULL值：ASC时NULL排在最前面（NULLS FIRST），DESC时NULL排在最后面（NULLS LAST）。

【实操练习】 ORDER BY 使用

```
-- 按deaths升序排列（默认ASC，NULLS FIRST）
select * from t_usa_covid19_p
where count_date = "2021-01-28" and state = "California"
order by deaths;

-- 按deaths降序排列（DESC，NULLS LAST）
select * from t_usa_covid19_p
where count_date = "2021-01-28" and state = "California"
order by deaths desc;

-- 强烈建议ORDER BY与LIMIT一起使用！
-- 避免数据集行数过大导致排序太慢
select * from t_usa_covid19_p
where count_date = "2021-01-28" and state = "California"
order by deaths desc
limit 3;
```

💡 *实际工作中，ORDER BY一定要配合LIMIT使用！如果设置了hive.mapred.mode=strict严格模式，不带LIMIT的ORDER BY会直接报错。*

幻灯片 26-29 — CLUSTER BY（分组+排序）

CLUSTER BY的功能是：根据指定字段将数据分组（fēn zǔ），并且每组内按该字段正序排序（只能正序）。

简单说就是：用同一个字段又分又排。

分组的规则是hash散列（sǎn liè），公式是：Hash_Func(字段值) % reduce task的个数。
分成几组取决于reduce task有几个。

大家看幻灯片上的这张图：
左边是原始数据（DATA），经过Hash函数处理后，按照f(X)%3的余数（yú shù）分到3个桶（BUCKET）里：
— 余数为0的进BUCKET 0
— 余数为1的进BUCKET 1
— 余数为2的进BUCKET 2
并且每个桶内的数据是正序排列的。

> **【实操练习】 CLUSTER BY 使用**

```
-- 创建student表
create table student(
  num int,
  name string,
  sex string,
  age int,
  dept string)
row format delimited
fields terminated by ',';

-- 加载数据
LOAD DATA LOCAL inpath '/usr/datas/student.dat'
OVERWRITE INTO TABLE student;

-- 查看数据
SELECT * FROM student;

-- 不指定reduce task个数，默认自动决定
select * from student cluster by num;

-- 手动设置reduce task为2个
set mapreduce.job.reduces = 2;
select * from student cluster by num;
```

查看结果可以看到，数据按照num字段被分到了不同的组，每个组内按num从小到大排列。

幻灯片 30 — CLUSTER BY的局限性（jú xiàn xìng）

CLUSTER BY有一个很大的限制：分组和排序只能用同一个字段，而且只能正序。

比如我们有一个需求（xū qiú）：根据sex性别分成两组，每组内再按age年龄倒序排列。

CLUSTER BY做不到！因为分组用的是sex，排序想用age，两个字段不一样。

ORDER BY也做不到！因为ORDER BY是全局排序，只有一个输出，没法分组。

而且，CLUSTER BY和ORDER BY不能同时使用，CLUSTER BY和SORT BY也不能同时使用，会报错。

那怎么解决这个问题呢？答案就是用DISTRIBUTE BY + SORT BY。

幻灯片 31-32 — DISTRIBUTE BY + SORT BY

DISTRIBUTE BY + SORT BY就是把CLUSTER BY的功能拆成了两部分：

DISTRIBUTE BY — 负责分组，决定数据去哪个reduce task。

SORT BY — 负责每个组内的排序规则，可以正序也可以倒序。

而且最关键的是：分组和排序可以用不同的字段！

> **【实操练习】 DISTRIBUTE BY + SORT BY 使用**

```
-- 按性别分组，每组内按年龄倒序排列
select * from student distribute by sex sort by age desc;

-- 当DISTRIBUTE BY和SORT BY的字段一样时
-- 等价于CLUSTER BY
select * from student distribute by num sort by num limit 5;
-- 上面这个等价于：
select * from student cluster by num limit 5;
```

第一个查询就完美解决了刚才的需求：按sex分组，每组内按age倒序。

查看结果可以看到：男生和女生被分开了，男生组内按年龄从大到小排列，女生组也是。

> 💡 *当DISTRIBUTE BY和SORT BY的字段相同时，CLUSTER BY = DISTRIBUTE BY + SORT BY*

幻灯片 33 — 四种排序方式总结对比

现在来总结一下这四种排序方式：

1. ORDER BY

全局排序，只有一个reducer，结果输出在一个文件中。数据量大时非常慢。

2. DISTRIBUTE BY

根据指定字段将数据分组（hash散列），不负责排序。

3. SORT BY

在分组之后，每个组内局部排序。

4. CLUSTER BY

既分组又排序，但分组和排序只能是同一个字段，而且只能正序。

当DISTRIBUTE BY + SORT BY的字段一样时，等于CLUSTER BY。

大家看图：

左边是ORDER BY，所有数据送到一个Reducer里做全局排序：0,1,2,3,4,5,6。

右边是CLUSTER BY，数据被DISTRIBUTE BY分到3个Reducer里，每个Reducer内部SORT BY排序：

Reducer 1: 0, 3, 6

Reducer 2: 1, 4

Reducer 3: 2, 5

幻灯片 34 — Union联合查询

UNION是用来把多个SELECT语句的结果合并（hé bìng）成一个结果集的。

就像你有两份名单（míng dān），你想合并成一份完整的名单。

UNION的两种模式：

UNION DISTINCT（或者只写UNION）：合并后去掉重复的行。

UNION ALL：合并后保留所有行，包括重复的。

使用要求：

每个SELECT返回的列的数量和名称必须相同！不然会报错。

语法格式：

```
select_statement
UNION [ALL | DISTINCT]
select_statement
UNION [ALL | DISTINCT]
select_statement ...;
```

幻灯片 35 — From子句中的子查询（Subqueries）

子查询就是在一个查询里面嵌套（qiàn tào）另一个查询。

From子句中的子查询：就是把一个SELECT的结果当成一张"临时表"来用。

重要规则：

子查询必须给一个别名（bié míng），因为FROM后面的每张表都得有名字。

子查询返回的列名必须唯一。

Hive支持任意层级（céng jí）的嵌套子查询。

> **【实操练习】 From子查询示例**

```
-- 简单子查询
SELECT num
FROM (
    select num, name from student_local
) tmp;

-- 包含UNION ALL的子查询
SELECT t3.name
FROM (
    select num, name from student_local
    UNION distinct
    select num, name from student_hdfs
) t3;
```

幻灯片 36 — Where子句中的子查询

WHERE子句中也可以使用子查询，从Hive 0.13版本开始支持。

两种类型：

1. 不相关子查询（bù xiāng guān zǐ chá xún）：子查询不引用父查询的列，可以理解为一个独立的查询。常用于IN和NOT IN。

2. 相关子查询（xiāng guān zǐ chá xún）：子查询引用了父查询中的列，常用于EXISTS和NOT EXISTS。

> **【实操练习】 Where子查询示例**

```
-- 不相关子查询：用IN过滤
SELECT *
FROM student_hdfs
WHERE student_hdfs.num IN (select num from student_local limit 2);

-- 相关子查询：用EXISTS
SELECT A
FROM T1
WHERE EXISTS (SELECT B FROM T2 WHERE T1.X = T2.Y);
```

幻灯片 37 — 6.1.6 Common Table Expressions (CTE) — 公用表表达式

CTE是一个非常实用（shí yòng）的功能。

CTE全名叫Common Table Expression，公用表表达式。它可以创建一个临时结果集（lín shí jié guǒ jí），用WITH关键字定义。

你可以把CTE想象成一个"临时的表"，只在当前这条SQL语句里有效，执行完就没了。

CTE的好处是什么？当你的SQL很复杂，有很多子查询嵌套的时候，用CTE可以让SQL变得更清楚、更容易读。

CTE可以在SELECT、INSERT、CREATE TABLE AS SELECT、CREATE VIEW AS SELECT语句中使用。

语法格式就是在SELECT前面加上WITH子句来定义临时结果集。