幻灯片 38-39 — 任务 6-2 连接语句（Join）

好，现在我们进入第二个任务——连接语句，也就是Join操作。

这个任务包含三个内容：

1. Hive Join的语法规则
2. Hive 6种Join方式详解（xiáng jiě）
3. Hive Join使用注意事项（zhù yì shì xiàng）

Join是数据查询中非常重要的操作，因为在实际工作中，数据通常分布在不同的表里，我们需要把多张表的数据组合在一起来分析。

幻灯片 40-41 — Join概念回顾

为什么需要Join呢?

因为根据数据库设计的规范（guī fàn），我们通常不会把所有数据都放在一张表里。而是不同类型的数据放在不同的表中。

大家看幻灯片上的例子:
上面是一张"订单信息表"，里面有订单编号、订单项目、负责人、业务员、订单数量和客户编号。
下面是一张"客户信息表"，里面有客户编号、客户名称、所属公司和联系方式。

这两张表通过"客户编号"这个字段关联（guān lián）起来。订单表里只存客户编号，不存客户的详细信息（像姓名、公司等）。

如果现在有个需求：查询每个订单对应的客户名称和联系方式。这时候就需要把两张表Join起来查询了。

Join就是根据两个或多个表中列之间的关系，从这些表中共同组合查询数据。

幻灯片 42-44 — Join语法规则

在Hive 3.1.2版本中，总共支持6种Join语法：

1. inner join — 内连接：只返回两张表都匹配的数据
2. left join — 左连接：以左表为主，返回左表所有数据
3. right join — 右连接：以右表为主，返回右表所有数据
4. full outer join — 全外连接：返回两张表的所有数据
5. left semi join — 左半开连接：只返回左表中匹配的数据
6. cross join — 交叉连接（笛卡尔乘积）：返回所有组合

语法格式：

```
-- 内连接
table_reference [INNER] JOIN table_factor [join_condition]
-- 外连接（左/右/全）
table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference join_condition
-- 左半开连接
table_reference LEFT SEMI JOIN table_reference join_condition
-- 交叉连接
table_reference CROSS JOIN table_reference [join_condition]

-- join_condition:
ON expression
```

从Hive 0.13.0开始，还支持隐式（yǐn shì）连接写法，就是在FROM后面用逗号分隔表名，在WHERE中写连接条件。

从Hive 2.2.0开始，支持不等连接（bù děng lián jiē），比如 ON a.id <> b.id。

幻灯片 46-47 — Join查询数据环境准备

为了练习Join，我们需要创建3张表并加载数据。

【实操练习】 创建3张表

```
-- 表1：员工表
CREATE TABLE employee(
  id int,
  name string,
  deg string,
  salary int,
  dept string
) row format delimited
fields terminated by ',';

-- 表2：员工家庭住址信息表
CREATE TABLE employee_address (
  id int,
  hno string,
  street string,
  city string
) row format delimited
fields terminated by ',';

-- 表3：员工联系方式信息表
CREATE TABLE employee_connection (
  id int,
  phno string,
  email string
) row format delimited
fields terminated by ',';
```

【实操练习】 加载数据

```
load data local inpath '/usr/datas/employee.txt'
into table employee;
select * from employee;

load data local inpath '/usr/datas/employee_address.txt'
into table employee_address;
select * from employee_address;

load data local inpath '/usr/datas/employee_connection.txt'
into table employee_connection;
select * from employee_connection;
```

注意看这3张表的数据：

employee表有5个员工：id分别是1201, 1202, 1203, 1204, 1206。

employee_address表有5条地址：id分别是1201, 1202, 1204, 1206, 1207。

employee_connection表有5条联系方式：id分别是1201, 1203, 1204, 1205, 1206。

注意它们的id不完全一样，这样我们就能清楚地看到不同Join类型的效果。

幻灯片 48-49 — inner join 内连接

内连接是最常见的连接方式。inner关键字可以省略，直接写JOIN就是内连接。

规则很简单：只有两张表中都有匹配的数据才会被返回。

大家看维恩图（Venn图）：两个圆代表两张表，内连接返回的是中间交叉（jiāo chā）的部分——也就是两张表都有的数据。

> **【实操练习】 inner join 示例**

```
select e.id, e.name, e_a.city, e_a.street
from employee e inner join employee_address e_a
on e.id = e_a.id;
```

结果分析：

employee表有id: 1201, 1202, 1203, 1204, 1206

employee_address表有id: 1201, 1202, 1204, 1206, 1207

两张表都有的id是：1201, 1202, 1204, 1206（4条记录）

所以内连接的结果是4行：gopal/jubilee, manisha/ny, prasanth/hyd, kranthi/la。

1203（khalil）只在employee表中有，不在address表中，所以不在结果中。

1207只在address表中有，不在employee表中，所以也不在结果中。

幻灯片 50-51 — left join 左连接

左连接也叫左外连接（Left Outer Join），outer可以省略。

核心思想：以左表（LEFT）为准。左表的数据全部返回，右表能匹配上的就返回，匹配不上的显示NULL。

大家看维恩图：左连接返回的是整个左边的圆，包括和右边重叠（chóng dié）的部分。

> **【实操练习】 left join 示例**

```
select e.id, e.name, e_conn.phno, e_conn.email
from employee e left join employee_connection e_conn
on e.id = e_conn.id;
```

结果分析：

左表employee有5个人：1201, 1202, 1203, 1204, 1206

右表employee_connection有：1201, 1203, 1204, 1205, 1206

1202（manisha）在左表有但右表没有，所以phno和email显示NULL。

1205在右表有但左表没有，所以不在结果中（因为是以左表为准）。

结果是5行，左表的每个人都出现了。

幻灯片 52-53 — right join 右连接

右连接和左连接相反：以右表为准。

右表数据全部返回，左表能匹配上的就返回，匹配不上的显示NULL。

其实right join和left join就是方向不同。把表的位置交换一下，right join就变成了left join。

在实际工作中，大多数人习惯用left join，很少用right join。你只要学会left join，right join就自然会了。

幻灯片 54-55 — full outer join 全外连接

全外连接也叫全连接（full join）。

核心思想：两张表的数据都全部返回。匹配上的正常显示，匹配不上的用NULL填充。

大家看维恩图：全外连接返回的是两个圆的全部区域。

> **【实操练习】 full outer join 示例**

```
select e.id, e.name, e_a.city, e_a.street
from employee e full outer join employee_address e_a
on e.id = e_a.id;
```

结果分析：

employee有：1201, 1202, 1203, 1204, 1206

employee_address有：1201, 1202, 1204, 1206, 1207

结果是6行：

— 1201, 1202, 1204, 1206正常匹配显示

— 1203（khalil）只在左表，右表的city和street显示NULL

— 1207只在右表，左表的id和name显示NULL

幻灯片 56-57 — left semi join 左半开连接

左半开连接是一种特殊的连接方式。

它只返回左表中能在右表找到匹配的记录，但不返回右表的任何列！

从效果上看，它有点像先做inner join，然后只返回左表的列。
也类似于SQL中的EXISTS子查询：只关心右表中"是否存在"匹配的数据，不关心右表的具体内容。

> **【实操练习】 left semi join 示例**

```
select *
from employee e left semi join employee_address e_addr
on e.id = e_addr.id;
```

结果分析：
返回的只有employee表的列（id, name, deg, salary, dept）。
在两张表中都能匹配到的id有：1201, 1202, 1204, 1206
所以结果是4行，只包含左表employee的数据。
注意：不会出现employee_address表的任何列！

幻灯片 58 — cross join 交叉连接（笛卡尔乘积）

最后一种是交叉连接（jiāo chā lián jiē），也叫笛卡尔乘积（dí kǎ ěr chéng jī）。

什么是笛卡尔乘积？就是左表的每一行和右表的每一行都组合（zǔ hé）一次。
如果左表有5行，右表有5行，那结果就是5×5=25行！

对于大表来说，千万要慎用（shèn yòng）cross join！因为行数会爆炸式增长。

在SQL标准中，cross join就是无条件的inner join，不需要指定关联键。
但在HiveSQL中，cross join后面可以跟where或on条件进行过滤。如果加了条件，效果就和inner join一样了。

大家看幻灯片上的结果表：
employee表的每一行都和employee_address表的每一行进行了组合。
比如gopal(1201)和每个地址都组合了一次，manisha(1202)也和每个地址都组合了一次。

幻灯片 59 — 课程总结

好的，同学们，今天的课程到这里就结束了。

今天我们学习了以下内容：

第一部分 Hive基本查询：

— SELECT查询的完整语法树

— select_expr选择表达式

— ALL和DISTINCT去重

— WHERE条件过滤

— 分区查询和分区裁剪

— GROUP BY分组和HAVING过滤

— LIMIT限制行数

— SQL执行顺序

— ORDER BY全局排序

— CLUSTER BY、DISTRIBUTE BY + SORT BY

— UNION联合查询、子查询和CTE

第二部分 连接语句：

— 6种Join类型：inner join, left join, right join, full outer join, left semi join, cross join

这些内容都是以后工作中每天都会用到的，所以大家课后一定要多练习！

有什么不懂的随时可以问我。谢谢大家！