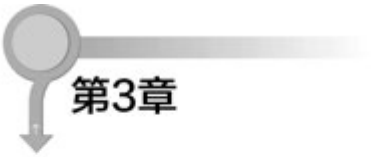

HiveQL操作

SQL是用于管理关系数据库的编程语言，本章将基于Hive介绍SQL语句，简称HiveQL。无论基于哪种技术，SQL语句都包含如下几种主要的语言类别。

3.1 Hive基本操作

数据定义语言(Data Definition Language，DDL)用于创建、修改和删除数据库结构中的对象，例如表、视图和索引。常用的DDL命令包括CREATE、ALTER、DROP、TRUNCATE。

数据管理语言(Data Manipulation Language，DML)用于在数据库中插入、更新、删除和查询数据。常用的DML命令包括INSERT、UPDATE、DELETE、SELECT。

数据控制语言(Data Control Language，DCL)用于管理数据库中的数据访问权限和安全权限。常用的DCL命令包括GRANT、REVOKE。

事务控制语言(Transaction Control Language，TCL)用于管理数据库中的事务，确保数据的一致性和完整性。常用的TCL命令包括COMMIT、ROLLBACK、SAVEPOINT。

数据查询语言(Data Query Language，DQL)虽然被视为DML的一部分，但DQL专门用于查询数据库中的数据。常用的DQL命令是SELECT。

结合Hive独特的SQL语句特性以及实际应用需求，本章重点介绍DDL、DML和DQL。

3.1.1 数据定义语言

在Hive中，数据定义语言主要用于定义和管理元数据，允许用户创建、修改和删除数据库、表、分区等。

1. 建表

在Hive中，建表操作是通过CREATE TABLE语句来实现的。用户需要指定表的名称、列名、数据类型以及其他可选的属性，如分区、存储格式等。

【例 3.1】 建表操作。

```
CREATE TABLE IF NOT EXISTS table_name (
    column1 data_type1,
    column2 data_type2,
    ...
)
COMMENT 'table comment'      -- 这是一个可选的子句,用于为表添加描述性的注释
PARTITIONED BY (partition_column data_type)
/* 这是一个可选的子句,用于指定表的分区列。分区可以显著提高查询性能,特别是对于非常大的表 */
STORED AS file_format;
/* 这是一个可选的子句,用于指定表数据的存储格式。不同的存储格式对性能和存储效率有不同的影响,常见的存储格式包括 TEXTFILE、SEQUENCEFILE、ORC 和 PARQUET。每种格式都有其特点和最佳使用场景 */
```

2. 修改表

修改表操作包括更改表名、增加/删除列、更改列的数据类型、增加/删除分区等。在 Hive 中,这些操作分别通过 ALTER TABLE 语句的不同子句来实现。

【例 3.2】 修改表。

```
-- 更改表名
ALTER TABLE old_table_name RENAME TO new_table_name;

-- 增加列
ALTER TABLE table_name ADD COLUMNS (new_column data_type);

-- 删除列(Hive 3.0.0 及以上版本支持)
ALTER TABLE table_name DROP COLUMNS (column_to_drop);

-- 更改列的数据类型(通过替换列的方式实现)
ALTER TABLE table_name CHANGE COLUMN old_column new_column new_data_type;

-- 增加分区
ALTER TABLE table_name ADD PARTITION (partition_column = 'value');

-- 删除分区
ALTER TABLE table_name DROP PARTITION (partition_column = 'value');
```

3. 删除表

删除表操作通过 DROP TABLE 语句来实现。该操作会删除表及其元数据,但不会删除外部表所指向的 HDFS 上的数据。

【例 3.3】 删除表。

```
DROP TABLE IF EXISTS table_name;
```

除此之外,可以使用 TRUNCATE 命令删除表中的所有数据,但保留表的结构。这与 DROP TABLE 命令不同,后者会删除整个表,包括其结构和数据。

【例 3.4】 使用 TRUNCATE 命令删除表。

```
TRUNCATE TABLE table_name;
```

需要注意的是：

（1）执行 TRUNCATE 命令后，无法恢复被删除的数据，因此在使用此命令前应确保确实要删除所有数据。

（2）如果表有分区，那么 TRUNCATE 命令会删除所有分区的数据，但不会删除分区本身。

（3）如果表是外部表，那么 TRUNCATE 命令将删除 Hive 中关于表数据的元数据，但实际存储在外部位置的数据文件不会被删除。

（4）在执行 TRUNCATE 命令之前，确保有足够的权限来修改表。

（5）TRUNCATE 操作可能会很耗时，尤其是在处理大型表时。

4. 分区

分区是 Hive 表的一种优化方式，通过将表的数据按照某个或某些列的值进行划分，可以提高查询效率。分区在建表时通过 PARTITIONED BY 子句指定。一般基于以下几种数据进行分区。

（1）时间维度：这是最常见的分区方式之一，通常按照天、小时、月或年来划分。例如，对于日志数据，可以按照日期进行分区，这样在查询特定时间段的数据时，Hive 只需要扫描相关的分区，而不是整个表。

（2）地域维度：对于具有地域特征的数据，比如销售记录，可以按照国家、省份、城市等地理信息进行分区。这样在查询特定区域的数据时可以快速定位。

（3）业务维度：根据业务需求，可以将数据按照不同的业务属性进行分区。例如，电商平台可以根据商品类别进行分区，广告数据可以根据广告类型分区。

（4）枚举值：当表中某些列的值是有限的枚举值时，可以根据这些列的值进行分区。比如，性别、状态（活跃/非活跃）等。

【例 3.5】 创建时间维度的分区。

```
CREATE TABLE log_table (
    log_id INT,
    user_id STRING,
    action STRING
)
PARTITIONED BY (date STRING);
```

3.1.2 数据管理语言

Hive 提供了一套数据管理语言，包括 SELECT、INSERT、UPDATE 和 DELETE 等命令，用于查询和操作数据。这些命令允许用户执行复杂的数据分析和处理任务。

1. 数据导入

数据导入操作通常通过 LOAD DATA 语句来实现，该操作可以将 HDFS 上的文件或目录加载到 Hive 表中。

【例 3.6】 数据导入操作。

```
LOAD DATA INPATH 'hdfs_path' INTO TABLE table_name;
-- 或者,对于分区表
LOAD DATA INPATH 'hdfs_path' INTO TABLE table_name PARTITION (partition_column = 'value');
```

【例 3.7】 将数据从一个表插入另一个表中。

```
INSERT INTO TABLE table_name
SELECT * FROM another_table;
-- 或者,对于分区表
INSERT INTO TABLE table_name PARTITION (partition_column = 'value')
SELECT column1, column2, ..., columnN
FROM another_table
WHERE another_table.partition_column = 'value';
```

2. 数据导出

数据导出操作可以通过 INSERT OVERWRITE 语句将查询结果导出到 HDFS 上的文件或目录中,或者使用 EXPORT TABLE 命令(仅适用于部分 Hive 版本和存储格式)。

【例 3.8】 数据导出操作。

```
INSERT OVERWRITE TABLE output_table
SELECT * FROM input_table;

-- 或者,将查询结果导出到 HDFS 文件(仅适用于部分版本和格式)
EXPORT TABLE table_name TO 'hdfs_path';
```

在实际应用中,更常见的是使用 Hadoop 生态系统中的其他工具(如 Sqoop、Pig 等)或编程语言(如 Java、Python 等)来实现数据的导入和导出。

【例 3.9】 将 HDFS 上指定目录中的数据导出到 Oracle 数据库的相应表中。

```
sqoop export \
-- connect JDBC URL:@IP 地址:端口号:服务名 \
-- username 数据库的用户名 \
-- password 数据库的密码 \
-- table $1
\* 指定要导出数据的目标数据库表名。这里使用 $1 表示该参数是一个位置参数,可以在命令行中传递 *\
-- export-dir 指定 HDFS 上的目录\
-m 1
\* 指定使用一个 map 任务来执行导出操作。这个数字可以根据数据量和集群资源进行调整 *\
-- input-fields-terminated-by '\t'\指定输入文件中字段之间的分隔符,这里是制表符(\t)
-- input-lines-terminated-by '\n'\指定输入文件中行结束的字符,这里是换行符(\n)
-- input-null-string '\\N'\指定输入文件中表示字符串类型 NULL 值的字符串,这里是 \\N
-- input-null-non-string '\\N'\指定输入文件中表示非字符串类型 NULL 值的字符串
```

Hive 在执行 DML 操作时,尤其是执行 UPDATE 和 DELETE 命令时,性能通常不如传统的 RDBMS,因为这些操作在 Hive 中不是原生的,需要进行额外的处理,因此建议大家在关系数据库中进行修改和删除。

3.1.3 数据查询语言

Hive 数据库的查询操作与关系数据库相似。在实际应用中,Hive 的查询语句可能更

加复杂，包括嵌套查询、子查询、窗口函数等高级特性。用户需要根据具体的数据和业务需求来编写合适的查询语句。

【例 3.10】 查询操作。

```
SELECT
[ALL | DISTINCT]          -- ALL 表示选择所有匹配的行，而 DISTINCT 表示只选择唯一的行
    select_expr, ...      -- 指定要选择的列。可以使用 * 来选择所有列
FROM
    table_reference       -- 指定查询的数据源表
[WHERE where_condition]    -- 过滤条件，只选择满足条件的行
[GROUP BY col_list]        -- 按指定列进行分组
[HAVING having_condition]-- 对分组后的结果进行过滤
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]
  -- CLUSTER BY col_list: 同时指定分组的列和排序的列
  -- DISTRIBUTE BY col_list: 指定数据分发到 reducer 的依据
  -- SORT BY col_list: 指定每个 reducer 内部的数据排序方式
[LIMIT number];            -- 限制返回的行数
```

在实际中，Hive 的查询语句更多用于窗口函数的应用中。Hive 提供的窗口函数可以对数据集中的相关行集（即“窗口”）执行计算，而无须对数据进行分组。窗口函数常用于执行运行总计、移动平均、排名等操作。Hive 支持的窗口函数包括排名函数、分析函数、聚合函数等。

常见的排名函数有如下几种。

（1）ROW_NUMBER()：为窗口内的每一行返回一个唯一的序号，从 1 开始。

（2）RANK()：返回窗口内相同值的行的排名，排名之间会跳号。

（3）DENSE_RANK()：类似于 RANK()，但排名之间不会跳号。

（4）NTILE(n)：将窗口内的行分配到指定数量的组(n)中，并返回组号。

常见的分析函数有如下几种。

（1）LEAD(expression[,offset])：返回窗口中当前行之后的第 offset 行的值。

（2）LAG(expression[,offset])：返回窗口中当前行之前的第 offset 行的值。

（3）FIRST_VALUE(expression)：返回窗口中第一个表达式的值。

（4）LAST_VALUE(expression)：返回窗口中最后一个表达式的值。

常见的聚合函数有如下几种。

（1）SUM() OVER()：计算窗口内值的总和。

（2）AVG() OVER()：计算窗口内值的平均值。

（3）MIN() OVER()：计算窗口内的最小值。

（4）MAX() OVER()：计算窗口内的最大值。

【例 3.11】 定义窗口。

```
SELECT
  column_name(s),
  window_function() OVER (
    [PARTITION BY column_name(s)]      -- 将数据集分区，窗口函数将在每个分区内单独计算
    [ORDER BY column_name(s)]          -- 指定窗口内行的排序方式
    [ROWS BETWEEN frame_start AND frame_end]     -- 定义窗口的框架，即窗口应包含哪些行
  ) as window_function_alias
```

【例 3.12】 假设有一个销售数据表 sales，其中包含列 date、employee_id、amount。使用 ROW_NUMBER()窗口函数为每个员工的销售记录分配一个唯一的序号。

```
SELECT
  date,
  employee_id,
  amount,
  ROW_NUMBER() OVER (PARTITION BY employee_id ORDER BY date) as row_num
FROM sales;
```

【例 3.13】 查询每个部门中当前员工薪水之前上一个员工的薪水。

```
SELECT
  name,
  department,
  salary,
  LAG(salary) OVER (PARTITION BY department ORDER BY salary DESC) as prev_salary
FROM employees;
```

【例 3.14】 使用对应的聚合函数作为窗口函数进行查询，计算每个部门的平均薪水。

```
SELECT
  name,
  department,
  salary,
  AVG(salary) OVER (PARTITION BY department) as avg_salary
FROM employees;
```

3.2 HiveQL 实例

大数据技术应用广泛，特别是在金融领域。下面通过一个银行的例子介绍 HiveQL 的操作方法。例如，作为一名银行后台的数据开发工程师，现在需要在 Hive 仓库中创建一个贷款相关的表，名字为 RPT_CL_LOAN。这个表中不仅需要存放借据信息的所有字段，也有一些新的字段，新字段如表 3.1 所示。

表 3.1 表中的新字段

字段名	含 义
date_id	日期标识，按照需求传入参数
bank_fr_flag	判断是否首次借款，首次设置为 1
channel_fr_flag	每个客户在特定渠道上的首次借款标识符
num	用于统计，全设置为 1
ovd_type	逾期类型，还款逾期天数大于或等于 30 天或者小于 90 天设置为 01，大于或等于 90 天设置为 02，否则设置为 03

当然，并不需要去创建每个表中的字段，通过对基础表 cl_loan 的字段加以处理，就可以得到想要的数据表了，表 3.2 给出的是基础表的部分关键字段。

表 3.2 基础表的部分关键字段

字 段 名	含 义
channel_num	渠道号
LOAN_NO	借据号
CLIENT_NO	客户号
cust_name	客户姓名
loan_status	贷款状态 1：放款中 2：已放款 3：已冲正 4：已撤销 5：已还款 6：已结清 7：已代偿 8：已核销 9：其他
DD_DATE	贷款起息日期
EXPIRE_DATE	贷款到期日期
clear_date	结清日期，整笔贷款实际结清日期
LOAN_AMT	贷款金额，该笔借据的余额 贷款余额＝发放金额－已还本金
START_DATE	记录开始日期
END_DATE	记录结束日期

具体代码如下所示：

```
DROP TABLE RPT.RPT_CL_LOAN;
CREATE TABLE RPT.RPT_CL_LOAN AS
SELECT a.*,'${hiveconf:date}' as date_id,
ROW_NUMBER() OVER(PARTITION BY client_no ORDER BY dd_date asc) as bank_fr_flag,
 --判断首次借款,dd_date为1即为首次借款
ROW_NUMBER() OVER(PARTITION BY channel_num,client_no ORDER BY dd_date asc) as channel_fr_
flag,
1 as num,
case when greatest(cur_ovd_days,cur_int_ovd_days)>=30 and
greatest(cur_ovd_days,cur_int_ovd_days)<90 then "01"
when greatest(cur_ovd_days,cur_int_ovd_days)>=90 then "02" else "03" end as ovd_type
FROM ODS.cl_loan A
WHERE A.END_DATE > '${hiveconf:date}'
AND A.START_DATE <= '${hiveconf:date}'
and a.loan_status in ('2','5','6','8')
and a.channel_num in ('DAD','DAS',);
```

按照生产开发流程，在创建新表之前需要将数据库可能存在的同名表删除，因此在代码中，首先使用 DROP TABLE RPT.RPT_CL_LOAN 删除已存在的同名表。这里我们使用别名，将 cl_loan 表命名为 A，因为 HiveQL 中对大小写并不做区分，因此 A 和 a 是相同的意思。这里，我们使用变量形式 ${hiveconf：date}动态地实现控制，比如每天都要创建一个包含最新日期数据的借款信息，这时可以在调度器中将变量 ${hiveconf：date}设置为当前

最新日期，那么数据表中每天都会得到最新的数据。另外，代码中使用了两个开窗函数，分别对渠道号和客户号两个进行分组，并对贷款起息日期进行升序排序，获取最早的一条数据，这是用来判断客户是否为首次借款以及在某借款渠道上是否为首次交易。这里也使用了 case when 语句进行条件判断，对不断逾期的客户进行识别，并在字段中赋予不同的标识值。下方的 where 条件限制我们只统计两个渠道 DAD 和 DAS 的数据，并且只记录贷款状态为已放款、已还款、已结清、已核销的记录。完成创建之后，数据开发人员可以使用 Sqoop 工具将数据仓库的数据导入业务数据库（通常使用 Oracle 或 MySQL 数据库）中，并在各类应用系统中通过 JDBC 或者 ODBC 进行数据调用。

课后习题

1. 描述 Hive 中的 DDL（数据定义语言）、DML（数据管理语言）和 DQL（数据查询语言）操作的主要内容。

2. 举例说明如何在 Hive 中进行创建表、修改表、删除表、分区以及数据导入和导出的操作。

3. 简述 Hive 中托管表和外部表的区别，以及它们在实际中的适用场景。

4. 使用语句创建 sales 表，语句如下：

```
CREATE TABLE sales (
    sale_id INT,                -- 销售记录的唯一标识符
    employee_id INT,            -- 执行销售的员工唯一标识符
    amount DECIMAL(10,2),       -- 销售金额
    sale_date DATE              -- 销售发生的日期
);
```

（1）编写一个 HiveQL 查询，为每条销售记录分配一个行号，行号按照 sale_date 升序排列。

（2）使用 sales 表，编写一个查询，显示每个员工的总销售额，并按照销售额降序排列，同时为每个员工分配一个排名（即每个员工销售额的排名）。

（3）使用 sales 表，编写一个查询，计算每个员工的累积销售额，即到当前记录为止的销售额总和。

（4）使用 sales 表，编写一个查询，计算每个员工过去 3 天的平均销售额。

（5）假设 sales 表中每个员工可能有多条销售记录，编写一个查询，找出每个员工的第一条销售记录。

（6）使用 sales 表，编写一个查询，显示每条销售记录及其前一条销售记录之间的金额差异。