基本使用指南 (Palo 2)

这里我们一个完整的流程对初次使用 Palo2.x 的用户展示 Palo 的基本使用方法。

1 Palo 接入

1.1 下载 mysql-client

从 Palo 2.0 版本开始，我们仅使用 mysql-client 作为 palo 的唯一操作入口。 理论上我们支持所有符合 mysql 标准协议的客户端。

如果你使用的机器 OS 环境是 Redhat4u3 环境，即百度当前服务器上部署最多的环境，就是那个 gcc 为3.4.5, glibc 为 2.3 的环境，你可以直接下载我们已经为你编译好的 Mysql Client 命令行工具来使用:

```
wget http://palo.baidu.com:8080/download/mysql-client
```

如果你的机器 OS 环境不是上面所说的，建议下载 mysql 的源代码，然后进行源码编译:

```
wget http://palo.baidu.com:8080/download/mysql-5.1.49.tar.gz
tar zxf mysql-5.1.49.tar.gz
cd mysql-5.1.49/
./configure --without-server --disable-shared
make
# make 成功后，mysql 程序就在 client/ 目录下生成了，拷走即可使用
```

Note

- 由于我们只需要构建 client，所以传入 –without-server 选项。
- 为了构建出来的 mysql 程序可以方便的被拷来拷去使用，构建时传入 –disabled-shared，以便使得 mysql 使用 libmysqlclient 静态库，而不是动态库。
- 构建后不需要执行 make install，直接从源码根目录下的 client 目录，直接把 mysql 拷走即可使用。

之后可以单独使用 ./mysql-5.1.49/client/mysql 二进制程序

1.2 权限申请

使用Palo系统前需要先申请权限，请联系 Palo 管理员创建用户及数据库。 管理员会给用户赋予一个或多个数据库的读写或只读权限，然后使用者就可以通过 mysql-client 登陆 Palo 进行后续操作。

1.3 登陆 Palo

Palo2.x 支持标准的 mysql 通讯协议，所以登陆 Palo 和登陆其他 mysql 库方法相同。 示例:

```
./mysql -h PALO_FE_HOST -P PALO_FE_PORT -uYOUR_USERNAME -pYOUR_PASSWORD
```

2 数据表的创建与数据导入

2.1 建立数据库

建立名为 example_db 的数据库:

```
CREATE DATABASE example_db;
```

Note

- 所有命令都可以使用 'HELP your_command' 查看到详细的中文帮助。
- 如果不清楚命令的全名，可以使用 'help 命令某一字段' 进行模糊查询。 如键入 'HELP CREATE'，可以匹配到 CREATE DATABASE, CREATE TABLE, CREATE USER 三个命令

2.2 建表

使用 CREATE TABLE 命令建立一个表（Table）。更多详细参数可以查看:

```
HELP CREATE TABLE;
```

首先切换数据库:

```
USE example_db;
```

从 Palo2.1 开始，我们支持单分区和复合分区两种建表方式。

在复合分区中：

- 第一级称为 Partition，即分区。用户可以指定某一维度列作为分区列（当前只支持整型和时间类型的列），并指定每个分区的取值范围。
- 第二级称为 Distribution，即分桶。用户可以指定某几个维度列（或不指定，即所有 KEY 列）以及桶数对数据进行 HASH 分布。

Note

- 以下场景推荐使用复合分区
    - 有时间维度或类似带有有序值的维度：可以以这类维度列作为分区列。分区粒度可以根据导入频次、分区数据量等进行评估。
    - 历史数据删除需求：如有删除历史数据的需求（比如仅保留最近 N 天的数据）。在 Palo 1 中，仅能使用 DELETE 命令进行删除，而这个命令会影响查询性能。使用复合分区，可以通过删除历史分区来达到目的。
    - 解决数据倾斜问题：每个分区可以单独指定分桶数量。如按天分区，当每天的数据量差异很大时，可以通过指定分区的分桶数，合理划分不同分区的数据。
- 用户也可以不使用复合分区，即使用单分区。则数据只做 HASH 分布（这种方式兼容 Palo 1.x 的建表方式中的 Hash 和 Random 方式）。但我们推荐使用复合分区以更方便的管理数据。
- 推荐阅读 高级使用指南 (Palo 2) 中的 合理的表模式 来完成表模式设计。

我们这里对两种方式分别进行建表操作演示。

单分区

建立一个名字为 single_partition_tbl 的逻辑表。使用全 key （Random）分桶，桶数为 32。

这个表的 schema 如下：

- siteid：类型是 INT（4字节）, 默认值为10
- cidy_code：类型是 SMALLINT（2字节）
- username：类型是 VARCHAR, 最大长度为32, 默认值为空字符串
- pv：类型是 BIGINT（8字节）, 默认值是 100; 这是一个指标列, Palo 内部会对指标列做聚合操作, 这个列的聚合方法是求和（SUM）

建表语句如下:

```
CREATE TABLE single_partition_tbl
(
siteid INT DEFAULT '10',
citycode SMALLINT,
username VARCHAR(32) DEFAULT '',
pv BIGINT SUM DEFAULT '100'
)
DISTRIBUTED BY RANDOM BUCKETS 32;
```

复合分区

建立一个名字为 multi_partition_tbl 的逻辑表。

这个表的 schema 如下：

- event_day：类型是 DATE，无默认值
- siteid：类型是 INT（4字节）, 默认值为10
- cidy_code：类型是 SMALLINT（2字节）
- username：类型是 VARCHAR, 最大长度为32, 默认值为空字符串
- pv：类型是 BIGINT（8字节）, 默认值是 100; 这是一个指标列, Palo 内部会对指标列做聚合操作, 这个列的聚合方法是求和（SUM）

我们使用 event_day 列作为分区列，建立 3 个分区: p1, p2, p3:

```
* p1：范围为 [最小值,      2015-06-30)
* p2：范围为 [2015-06-30, 2015-07-31)
* p3：范围为 [2015-07-31, 2015-08-31)
```

每个分区使用 siteid 进行哈希分桶，桶数为 32

建表语句如下:

```
CREATE TABLE multi_partition_tbl
(
event_day DATE,
siteid INT DEFAULT '10',
citycode SMALLINT,
username VARCHAR(32) DEFAULT '',
pv BIGINT SUM DEFAULT '100'
)
PARTITION BY RANGE(event_day)
(
PARTITION p1 VALUES LESS THAN ('2015-06-30'),
PARTITION p2 VALUES LESS THAN ('2015-07-31'),
PARTITION p3 VALUES LESS THAN ('2015-08-31')
)
DISTRIBUTED BY HASH(siteid) BUCKETS 32;
```

Note

- 可以对复合分区表动态的增删分区。详见 'HELP ALTER TABLE' 中 PARTITION 相关部分。
- 数据导入可以导入指定的 partition。详见 'HELP LOAD'。
- 可以动态修改表的 Schema。详见 高级使用指南 (Palo 2) 中 修改 Schema
- 可以对 Table 增加上卷表（Rollup）以提高查询性能，关于更多 Rollup 的高级应用，请参阅 高级使用指南 (Palo 2) 中 创建 Rollup 以及 Table 与 上卷表（Rollup）的关系

2.3 导入数据

Palo2.x 支持两种数据导入方式：

- LOAD：使用 Hadoop 进行 ETL 的数据导入。详见 'HELP LOAD'
- BULK LOAD：针对小批量数据的导入。详见 'HELP BULK LOAD'

我们这里分别演示两种导入数据操作。

LOAD

示例：以 "ps_stats_20150717" 为 Label，使用 HDFS 上的文件 ps_stats_data 导入 example_tbl 表:

```
LOAD LABEL ps_stats_20150717
(
DATA INFILE("hdfs://your.namenode.host:54310/user/palo/data/input/ps_stats_data")
INTO TABLE example_tbl
);
```

Note

- 该方式导入 Palo 的源数据文件必须在 HDFS 上，并且其权限对 others 可读。也可以将 HDFS 用户 palo 加入你的用户组，并使得数据对用户组可读，以提高数据的安全性。
- 每一批导入数据都需要取一个 Label，Label 最好是一个和一批数据有关的字符串，方便阅读和管理。Palo 基于 Label 保证在一个 Database 内，同一批数据只可导入成功一次。

BULK LOAD

BULK LOAD 主要用于解决 ETL 操作依赖 Hadoop 的问题。使用 BULK LOAD，用户可以不依赖于 Hadoop 完成导入操作。

BULK LOAD 是 Palo 2 中唯一不使用 mysql-client 执行的命令。我们采用 http 协议完成通信。

示例：以 “ps_stats_20150717” 为 Label，使用本地文件 ps_stats_data 导入 example_tbl 表:

```
curl --location-trusted -u username:password -T ps_stats_data
http://fe.host:port/api/example_db/example_tbl/_load?label=ps_stats_20150717
```

Note

- 单批次导入的数据量限制为 1GB，用户如果要导入大量数据，需要自己手动拆分成多个小于 1GB 的文件，分多个批次导入。
- Label 的使用同 ‘LOAD’。
- 该方式可以支持用户同时向多个表进行导入，并且多表间原子生效。用法请参阅：’HELP MULTI LOAD’。

2.4 查询导入任务的状态

导入任务是异步执行的。执行导入命令后，需要通过 SHOW LOAD 命令查询导入任务的状态。 更多详细参数可以查看:

```
HELP SHOW LOAD;
```

导入任务的主要信息为:

- **State**：导入状态
  - *pending* 导入任务尚未被调度执行
  - *etl* 正在执行 ETL 计算, Palo 内部状态
  - *load* 正在进行加载, Palo 内部状态
  - *finished* 导入任务成功完成
  - *cancelled* 导入任务被取消或者失败
- Progress：导入进度
- **EtlInfo ETL**：阶段的作业信息
  - *dpp.abnorm.ALL* 输入数据中被过滤掉的非法数据条数
  - *dpp.norm.ALL* 输入数据中合法的数据条数
- TaskInfo：本次导入作业的参数信息
- ErrorMsg：导入任务失败原因
- CreateTime：任务创建时间
- EtlStartTime：ETL 开始时间
- EtlFinishTime：ETL 结束时间
- LoadStartTime：加载开始时间
- LoadFinishTime：加载结束时间
- **URL**
  - 在 LOAD 命令中，为 Hadoop ETL 任务链接
  - 在 BULK LOAD 命令中，为导入失败后的错误日志地址

示例1：显示当前数据库内最后20个导入任务的状态:

```
SHOW LOAD ORDER BY CreateTime DESC LIMIT 20;
```

示例2：显示当前数据库内以 “20120101” 为 Label 的所有任务的状态的详细信息:

```
SHOW LOAD WHERE LABEL = "20120101";
```

Note

如果任务失败，可以参考*常见问题*中的 导入任务失败原因。

2.5 取消导入任务

使用 CANCEL LOAD 命令取消一个正在执行的导入任务。 被取消的任务数据不会导入 Palo。 已经处于 cancelled 或 finished 状态的任务无法被取消。

示例：取消当前数据库中 Label 为 “20131028-fc” 的任务:

```
CANCEL LOAD WHERE LABEL = "20131028-fc";
```

3 数据的查询

3.1 查看数据库、表信息

查看你所拥有权限的数据库列表:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| star               |
| test               |
+--------------------+
2 rows in set (0.00 sec)
```

查看某个数据库中表的信息:

```
mysql> use star;
Database changed

mysql> show tables;
+----------------+
| Tables_in_star |
+----------------+
| customer       |
| dates          |
| lineorder      |
| part           |
| supplier       |
+----------------+
5 rows in set (0.01 sec)

mysql> desc customer;
+--------------+-------------+------+------+---------+-------+
| Field        | Type        | Null | Key  | Default | Extra |
+--------------+-------------+------+------+---------+-------+
| c_custkey    | int(11)     | NO   |      | NULL    |       |
| c_name       | varchar(20) | NO   |      | NULL    |       |
| c_address    | varchar(20) | NO   |      | NULL    |       |
| c_city       | varchar(20) | NO   |      | NULL    |       |
| c_nation     | varchar(20) | NO   |      | NULL    |       |
| c_region     | varchar(20) | NO   |      | NULL    |       |
| c_phone      | varchar(20) | NO   |      | NULL    |       |
| c_mktsegment | varchar(20) | NO   |      | NULL    |       |
+--------------+-------------+------+------+---------+-------+
8 rows in set (0.01 sec)
```

3.2 简单查询

示例:

```
mysql> select * from customer limit 5;
+-----------+--------------------+--------------------+-----------+----------+----------+-----
-----------+--------------+
| c_custkey | c_name             | c_address          | c_city    | c_nation | c_region |
c_phone         | c_mktsegment |
+-----------+--------------------+--------------------+-----------+----------+----------+-----
-----------+--------------+
|   1048577 | Customer#001048577 | EOqOMa             | CHINA    9 | CHINA    | ASIA     | 28-
341-253-8578 | MACHINERY    |
|   1048578 | Customer#001048578 | ow3ZStMHa9u9KOoury | RUSSIA   3 | RUSSIA   | EUROPE   | 32-
252-250-2399 | BUILDING     |
|   1048579 | Customer#001048579 | t734I8Ire2s        | BRAZIL   1 | BRAZIL   | AMERICA  | 12-
165-291-3850 | AUTOMOBILE   |
|   1048580 | Customer#001048580 | UV1NQo0kmAgGKv6    | ROMANIA  9 | ROMANIA  | EUROPE   | 29-
581-381-6413 | MACHINERY    |
|   1048581 | Customer#001048581 | SIjDLiFMhEEMn8WB2M | ARGENTINA7 | ARGENTINA | AMERICA  | 11-
812-761-5023 | HOUSEHOLD    |
+-----------+--------------------+--------------------+-----------+----------+----------+-----
-----------+--------------+
5 rows in set (0.04 sec)
```

Note

- 进行数据查看时，请尽量使用limit进行限制，防止返回过多数据。

3.3 order by查询

示例:

```
mysql> select c_nation, count(*) from customer group by c_nation order by c_nation limit 5;
+-----------+----------+
| c_nation  | COUNT(*) |
+-----------+----------+
| ALGERIA   |   114996 |
| ARGENTINA |   114965 |
| BRAZIL    |   115243 |
| CANADA    |   115176 |
| CHINA     |   115123 |
+-----------+----------+
5 rows in set (1.92 sec)
```

Note

鉴于 order by 的特殊性，order by 后面建议一定要加入 limit，如果未加 limit，系统当前默认会自动为你添加 limit 65535。

3.4 带有join的查询

示例:

```
mysql> select sum(lo_ordtotalprice) from lineorder, customer where lo_custkey = c_custkey and
c_name = "Customer#001048579";
+-----------------------+
| SUM(lo_ordtotalprice) |
+-----------------------+
|            8996548708 |
+-----------------------+
1 row in set (6.68 sec)
```

3.5 带有子查询的查询

示例:

```
mysql> select sum(lo_ordtotalprice) from lineorder, (select c_custkey from customer where c_name
= "Customer#001048579") as cc where lo_custkey = cc.c_custkey;
+-----------------------+
| SUM(lo_ordtotalprice) |
+-----------------------+
|            8996548708 |
+-----------------------+
1 row in set (6.48 sec)
```

Note

当前只支持在 from 后面支持子查询。where 子查询正在开发中...

Previous Next