分布式融合数据库 HTAP

分区表使用手册

天翼云科技有限公司

支持的分区类型

当前支持的类型包括 Range 分区、Range COLUMNS 分区、List 分区、List COLUMNS 分区、Hash 分区和 Key 分区。

- Range 分区、Range COLUMNS 分区、List 分区和 List COLUMNS 分区可以用于解决业务中大量删除带来的性能问题，支持快速删除分区。
- Hash 分区和 Key 分区可以用于大量写入场景下的数据打散。与 Hash 分区相比，Key 分区支持多列打散和非整数类型字段的打散。

Range 分区

Range 分区指的是在每个分区中，所有行的值根据分区表达式计算后都落在指定的范围内。该范围必须是连续的，且按从小到大递增，不能有重叠。可以使用 VALUES LESS THAN 语句来定义这些范围。

假设需要创建一个人事记录表，并按 store_id 列进行 Range 分区，示例如下：

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE DEFAULT '9999-12-31',
    job_code INT,
    store_id INT NOT NULL
)

PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

在此分区设置中，所有 store_id 为 1 到 5 的员工记录存储在分区 p0 中，而 store_id 为 6 到 10 的员工则存储在分区 p1 中。MAXVALUE 表示一个比所有整数都大的值，因此所有 store_id 大于等于 16 的记录将存储在 p3 分区中。

Range COLUMNS 分区

Range COLUMNS 分区是 Range 分区的一种变体。你可以使用一个或者多个列作为分区键，分区列的数据类型可以是整数 (integer)、字符串（CHAR/VARCHAR），DATE 和 DATETIME。不支持使用任何表达式。和 Range 分区一样，Range COLUMNS 分区同样需要分区的范围是严格递增的。

假设你想要按名字进行分区，并且能够轻松地删除旧的无效数据，那么你可以创建一个表格，如下所示：

```
CREATE TABLE t (
  valid_until datetime,
  name varchar(255) CHARACTER SET ascii,
  notes text
)
PARTITION BY RANGE COLUMNS(name, valid_until)
(PARTITION `p2022-g` VALUES LESS THAN ('G','2023-01-01 00:00:00'),
 PARTITION `p2023-g` VALUES LESS THAN ('G','2024-01-01 00:00:00'),
 PARTITION `p2022-m` VALUES LESS THAN ('M','2023-01-01 00:00:00'),
 PARTITION `p2023-m` VALUES LESS THAN ('M','2024-01-01 00:00:00'),
 PARTITION `p2022-s` VALUES LESS THAN ('S','2023-01-01 00:00:00'),
 PARTITION `p2023-s` VALUES LESS THAN ('S','2024-01-01 00:00:00'))
```

该语句将按名字和年份的范围 [('', ''), ('G', '2023-01-01 00:00:00')), [('G', '2023-01-01 00:00:00'), ('G', '2024-01-01 00:00:00')), [('G', '2024-01-01 00:00:00'), ('M', '2023-01-01 00:00:00')), [('M', '2023-01-01 00:00:00'), ('M', '2024-01-01 00:00:00')), [('M', '2024-01-01 00:00:00'), ('S', '2023-01-01 00:00:00')), [('S', '2023-01-01 00:00:00'), ('S', '2024-01-01 00:00:00')) 进行分区，删除无效数据，同时仍然可以在 name 和 valid_until 列上进行分区裁剪。其中，[,) 是一个左闭右开区间，比如 [('G', '2023-01-01 00:00:00'), ('G', '2024-01-01 00:00:00'))，表示 name 为 'G'，年份包含 2023-01-01 00:00:00 并大于 2023-01-01 00:00:00 但小于 2024-01-01 00:00:00 的数据，其中不包含 (G, 2024-01-01 00:00:00)。

List 分区

List 分区和 Range 分区有很多相似的地方。不同之处主要在于 List 分区中，对于表的每个分区中包含的所有行，按分区表达式计算的值属于给定的数据集合。每个分区定义的数据集合有任意个值，但不能有重复的值，可通过 PARTITION ... VALUES IN (...) 子句对值进行定义。

假设你要创建一张人事记录表，如果想把同一个地区商店员工的人事数据都存储在同一个分区中，你可以根据 store_id 来创建 List 分区：

```
CREATE TABLE employees (
    id INT NOT NULL,
    hired DATE NOT NULL DEFAULT '1970-01-01',
    store_id INT
)
PARTITION BY LIST (store_id) (
    PARTITION pNorth VALUES IN (1, 2, 3, 4, 5),
    PARTITION pEast VALUES IN (6, 7, 8, 9, 10),
    PARTITION pWest VALUES IN (11, 12, 13, 14, 15),
    PARTITION pCentral VALUES IN (16, 17, 18, 19, 20)
);
```

与 Range 分区的情况不同，List 分区没有类似的 MAXVALUE 分区来存储所有不属于其他 partition 的值。分区表达式的所有期望值都应包含在 PARTITION ... VALUES IN (...) 子句中。如果 INSERT 语句要插入的值不匹配分区的列值，该语句将执行失败并报错。

List COLUMNS 分区

List COLUMNS 分区是 List 分区的一种变体，可以将多个列用作分区键，并且可以将整数类型以外的数据类型的列用作分区列。你还可以使用字符串类型、DATE 和 DATETIME 类型的列。

与 List 分区不同的是，你不需要在 COLUMNS() 子句中使用表达式来将列值转换为整数，也可以使用 DATE 和 DATETIME 类型的列进行分区。另外，你也可以在 COLUMNS() 子句中添加多个列。例如：

```
CREATE TABLE t (
    id int,
    name varchar(10)
)
PARTITION BY LIST COLUMNS(id,name) (
    partition p0 values IN ((1,'a'),(2,'b')),
    partition p1 values IN ((3,'c'),(4,'d')),
    partition p3 values IN ((5,'e'),(null,null))
);
```

Hash 分区

Hash 分区主要用于保证数据均匀地分散到一定数量的分区里面。在 Range 分区中你必须为每个分区指定值的范围；在 Hash 分区中，你只需要指定分区的数量。

创建 Hash 分区表时，需要在 CREATE TABLE 后面添加 PARTITION BY HASH (expr)，其中 expr 是一个返回整数的表达式。当这一列的类型是整数类型时，它可以是一个列名。此外，你很可能还需要加上 PARTITIONS num，其中 num 是一个正整数，表示将表划分多少分区。如果不指定 PARTITIONS num，默认的分区数量为 1。

最高效的 Hash 函数是作用在单列上，并且函数的单调性是跟列的值是一样递增或者递减的。例如，date_col 是类型为 DATE 的列，表达式 TO_DAYS(date_col) 的值是直接随 date_col 的值变化的。YEAR(date_col) 跟 TO_DAYS(date_col) 就不太一样，因为不是每次 date_col 变化时 YEAR(date_col) 都会得到不同的值。作为对比，假设我们有一个类型是 INT 的 int_col 的列。考虑一下表达式 POW(5-int_col,3) + 6，这并不是一个比较好的 Hash 函数，因为随着 int_col 的值的变化，表达式的结果不会成比例地变化。改变 int_col 的值会使表达式的结果的值变化巨大。例如，int_col 从 5 变到 6 表达式的结果变化是 -1，但是从 6 变到 7 的时候表达式的值的变化是 -7。

总而言之，表达式越接近 y = cx 的形式，它越是适合作为 Hash 函数。因为表达式越是非线性的，在各个分区上面的数据的分布越是倾向于不均匀。

理论上，Hash 分区也是可以做分区裁剪的。而实际上对于多列的情况，实现很难并且计算很耗时。因此，不推荐 Hash 分区在表达式中涉及多列。

使用 PARTITIION BY HASH 的时候，通过表达式的结果做“取余”运算，决定数据落在哪个分区。换句话说，如果分区表达式是 expr，分区数是 num，则由 MOD(expr, num) 决定存储的分区。假设 t1 定义如下：

```
CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATE)
   PARTITION BY HASH( YEAR(col3) )
   PARTITIONS 4;
```

向 t1 插入一行数据，其中 col3 列的值是 '2005-09-15'，这条数据会被插入到分区 1 中：

```
MOD(YEAR('2005-09-01'),4)
=  MOD(2005,4)
=  1
```

Key 分区

Key 分区与 Hash 分区都可以保证将数据均匀地分散到一定数量的分区里面，区别是 Hash 分区只能根据一个指定的整数表达式或字段进行分区，而 Key 分区可以根据字段列表进行分区，且 Key 分区的分区字段不局限于整数类型。Key 分区表的 Hash 算法与 MySQL 不一样，因此表的数据分布也不一样。

创建 Key 分区表时，你需要在 CREATE TABLE 后面添加 PARTITION BY KEY (columList)，其中 columnList 是字段列表，可以包含一个或多个字段。每个字段的类型可以是除 BLOB、JSON、GEOMETRY 之外的任意类型。此外，你很可能还需要加上 PARTITIONS num，其中 num 是一个正整数，表示将表划分多少个分区；或者加上分区名的定义，例如，加上 (PARTITION p0, PARTITION p1) 代表将表划分为两个分区，分区名为 p0 和 p1。

下面的语句将创建一个 Key 分区表，按 store_id 分成 4 个分区：

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)

PARTITION BY KEY(store_id)
PARTITIONS 4;
```

如果不指定 PARTITIONS num，默认的分区数量为 1。

你也可以根据 VARCHAR 等非整数字段创建 Key 分区表。下面的语句按 fname 将表分成 4 个分区：

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE DEFAULT '9999-12-31',
```

```
    job_code INT,
    store_id INT
)

PARTITION BY KEY(fname)
PARTITIONS 4;
```

你还可以根据多列字段创建 Key 分区表。下面的语句按 fname、store_id 将表分成 4 个分区：

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)

PARTITION BY KEY(fname, store_id)
PARTITIONS 4;
```

分区对 NULL 值的处理

允许计算结果为 NULL 的分区表达式。注意，NULL 不是一个整数类型，NULL 小于所有的整数类型值，正如 ORDER BY 的规则一样。

如果插入一行到 Range 分区表，它的分区列的计算结果是 NULL，那么这一行会被插入到最小的那个分区。

在 Hash 分区、Key 分区中，如果分区表达式的计算结果为 NULL，它会被当作 0 值处理。

分区管理

对于 RANGE、RANGE COLUMNS、LIST、LIST COLUMNS 分区表，你可以进行以下分区管理操作：

- 使用 ALTER TABLE <表名> ADD PARTITION (<分区说明>) 语句添加分区。
- 使用 ALTER TABLE <表名> DROP PARTITION <分区列表> 删除分区。

- 使用 ALTER TABLE <表名> TRUNCATE PARTITION <分区列表> 语句清空分区里的数据。
- 使用 ALTER TABLE <表名> REORGANIZE PARTITION <分区列表> INTO (<新的分区说明>) 语句对分区进行合并、拆分、或者其他修改。

对于 HASH 和 KEY 分区表，你可以进行以下分区管理操作:

- 使用 ALTER TABLE <table name> COALESCE PARTITION <要减少的分区数量> 语句减少分区数量。此操作会重组分区，将所有数据按照新的分区个数复制到对应的分区。
- 使用 ALTER TABLE <table name> ADD PARTITION <要增加的分区数量 | (新的分区说明)> 语句增加分区的数量。此操作会重组分区，将所有数据按照新的分区个数复制到对应的分区。
- 使用 ALTER TABLE <table name> TRUNCATE PARTITION <分区列表> 语句清空分区里的数据。

管理 List 分区、List COLUMNS 分区、Range 分区、Range COLUMNS 分区

本小节将以如下 SQL 语句创建的分区表为例，介绍如何管理 Range 分区和 List 分区。

```
CREATE TABLE members (
    id int,
    fname varchar(255),
    lname varchar(255),
    dob date,
    data json
)
PARTITION BY RANGE (YEAR(dob)) (
 PARTITION pBefore1950 VALUES LESS THAN (1950),
 PARTITION p1950 VALUES LESS THAN (1960),
 PARTITION p1960 VALUES LESS THAN (1970),
 PARTITION p1970 VALUES LESS THAN (1980),
 PARTITION p1980 VALUES LESS THAN (1990),
 PARTITION p1990 VALUES LESS THAN (2000));

CREATE TABLE member_level (
 id int,
 level int,
```

```
 achievements json
)
PARTITION BY LIST (level) (
 PARTITION l1 VALUES IN (1),
 PARTITION l2 VALUES IN (2),
 PARTITION l3 VALUES IN (3),
 PARTITION l4 VALUES IN (4),
 PARTITION l5 VALUES IN (5));
```

删除分区

```
ALTER TABLE members DROP PARTITION p1990;

ALTER TABLE member_level DROP PARTITION l5;
```

清空分区

```
ALTER TABLE members TRUNCATE PARTITION p1980;

ALTER TABLE member_level TRUNCATE PARTITION l4;
```

添加分区

```
ALTER TABLE members ADD PARTITION (PARTITION `p1990to2010` VALUES LESS THAN (20
10));

ALTER TABLE member_level ADD PARTITION (PARTITION l5_6 VALUES IN (5,6));
```

重组分区

拆分分区：

```
ALTER TABLE members REORGANIZE PARTITION `p1990to2010` INTO
(PARTITION p1990 VALUES LESS THAN (2000),
 PARTITION p2000 VALUES LESS THAN (2010),
 PARTITION p2010 VALUES LESS THAN (2020),
 PARTITION p2020 VALUES LESS THAN (2030),
 PARTITION pMax VALUES LESS THAN (MAXVALUE));
```

```
ALTER TABLE member_level REORGANIZE PARTITION l5_6 INTO
(PARTITION l5 VALUES IN (5),
 PARTITION l6 VALUES IN (6));
```

合并分区：

```
ALTER TABLE members REORGANIZE PARTITION pBefore1950,p1950 INTO (PARTITION pBefore1960 VALUES LESS THAN (1960));
```

```
ALTER TABLE member_level REORGANIZE PARTITION l1,l2 INTO (PARTITION l1_2 VALUES IN (1,2));
```

修改分区表定义：

```
ALTER TABLE members REORGANIZE PARTITION pBefore1960,p1960,p1970,p1980,p1990,p2000,p2010,p2020,pMax INTO
(PARTITION p1800 VALUES LESS THAN (1900),
 PARTITION p1900 VALUES LESS THAN (2000),
 PARTITION p2000 VALUES LESS THAN (2100));
```

```
ALTER TABLE member_level REORGANIZE PARTITION l1_2,l3,l4,l5,l6 INTO
(PARTITION lOdd VALUES IN (1,3,5),
 PARTITION lEven VALUES IN (2,4,6));
```

在重组分区时，需要注意以下关键点：

- 重组分区（包括合并或拆分分区）只能修改分区定义，无法修改分区表类型。例如，无法将 List 类型修改为 Range 类型，或将 Range COLUMNS 类型修改为 Range 类型。
- 对于 Range 分区表，你只能对表中相邻的分区进行重组：

  ```
  ALTER TABLE members REORGANIZE PARTITION p1800,p2000 INTO (PARTITION p2000 VALUES LESS THAN (2100));
  ```

  ```
  ERROR 8200 (HY000): Unsupported REORGANIZE PARTITION of RANGE; not adjacent partitions
  ```

- 对于 Range 分区表，如需修改 Range 定义中的最大值，必须保证 VALUES LESS THAN 中新定义的值大于现有分区中的所有值。否则将报错，提示现有的行值对应不到分区。

  ```
  INSERT INTO members VALUES (313, "John", "Doe", "2022-11-22", NULL);
  ALTER TABLE members REORGANIZE PARTITION p2000 INTO (PARTITION p2000 VALUES LESS THAN (2050)); -- 执行成功，因为 2050 包含了现有的所有行
  ```

```
ALTER TABLE members REORGANIZE PARTITION p2000 INTO (PARTITION p2000 VALUES LESS THAN (2020)); -- 执行失败，因为 2022 将对应不到分区
```

```
ERROR 1526 (HY000): Table has no partition for value 2022
```

- 对于 List 分区表，如需修改分区定义中的数据集合，必须保证新的数据集合能覆盖到该分区中现有的所有值，否则将报错。

```
INSERT INTO member_level (id, level) values (313, 6);
ALTER TABLE member_level REORGANIZE PARTITION lEven INTO (PARTITION lEven VALUES IN (2,4));
```

```
ERROR 1526 (HY000): Table has no partition for value 6
```

管理 Hash 分区和 Key 分区

本小节将以如下 SQL 语句创建的分区表为例，介绍如何管理 Hash 分区。对于 Key 分区，你也可以使用与 Hash 分区相同的分区管理语句。

```
CREATE TABLE example (
  id INT PRIMARY KEY,
  data VARCHAR(1024)
)
PARTITION BY HASH(id)
PARTITIONS 2;
```

增加分区数量

将 example 表的分区个数增加 1 个（从 2 增加到 3）：

```
ALTER TABLE example ADD PARTITION PARTITIONS 1;
```

你也可以通过添加分区定义来指定分区选项。例如，你可以通过以下语句将分区数量从 3 增加到 5，并指定新增的分区名为 pExample4 和 pExample5：

```
ALTER TABLE example ADD PARTITION
(PARTITION pExample4 COMMENT = 'not p3, but pExample4 instead',
 PARTITION pExample5 COMMENT = 'not p4, but pExample5 instead');
```

减少分区数量

与 Range 和 List 分区不同，Hash 和 Key 分区不支持 DROP PARTITION，但可以使用 COALESCE PARTITION 来减少分区数量，或使用 TRUNCATE PARTITION 清空指定分区的所有数据。

将 example 表的分区个数减少 1 个（从 5 减少到 4）：

```
ALTER TABLE example COALESCE PARTITION 1;
```

清空分区

清空指定分区的所有数据:

```
ALTER TABLE example TRUNCATE PARTITION p0;
```

分区裁剪

分区裁剪是只有当目标表为分区表时，才可以进行的一种优化方式。分区裁剪通过分析查询语句中的过滤条件，只选择可能满足条件的分区，不扫描匹配不上的分区，进而显著地减少计算的数据量。

分区选择

SELECT 语句中支持分区选择。实现通过使用一个 PARTITION 选项实现。

```
SET @@sql_mode = '';

CREATE TABLE employees (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    fname VARCHAR(25) NOT NULL,
    lname VARCHAR(25) NOT NULL,
    store_id INT NOT NULL,
    department_id INT NOT NULL
)

PARTITION BY RANGE(id) (
    PARTITION p0 VALUES LESS THAN (5),
    PARTITION p1 VALUES LESS THAN (10),
    PARTITION p2 VALUES LESS THAN (15),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);

INSERT INTO employees VALUES
```

```
    ('', 'Bob', 'Taylor', 3, 2), ('', 'Frank', 'Williams', 1, 2),
    ('', 'Ellen', 'Johnson', 3, 4), ('', 'Jim', 'Smith', 2, 4),
    ('', 'Mary', 'Jones', 1, 1), ('', 'Linda', 'Black', 2, 3),
    ('', 'Ed', 'Jones', 2, 1), ('', 'June', 'Wilson', 3, 1),
    ('', 'Andy', 'Smith', 1, 3), ('', 'Lou', 'Waters', 2, 4),
    ('', 'Jill', 'Stone', 1, 4), ('', 'Roger', 'White', 3, 2),
    ('', 'Howard', 'Andrews', 1, 2), ('', 'Fred', 'Goldberg', 3, 3),
    ('', 'Barbara', 'Brown', 2, 3), ('', 'Alice', 'Rogers', 2, 2),
    ('', 'Mark', 'Morgan', 3, 3), ('', 'Karen', 'Cole', 3, 2);
```

你可以查看存储在分区 p1 中的行:

```
SELECT * FROM employees PARTITION (p1);
+----|-------|--------|----------|---------------+
| id | fname | lname  | store_id | department_id |
+----|-------|--------|----------|---------------+
|  5 | Mary  | Jones  |        1 |             1 |
|  6 | Linda | Black  |        2 |             3 |
|  7 | Ed    | Jones  |        2 |             1 |
|  8 | June  | Wilson |        3 |             1 |
|  9 | Andy  | Smith  |        1 |             3 |
+----|-------|--------|----------|---------------+
5 rows in set (0.00 sec)
```

如果希望获得多个分区中的行，可以提供分区名的列表，用逗号隔开。例如，SELECT * FROM employees PARTITION (p1, p2) 返回分区 p1 和 p2 的所有行。

使用分区选择时，仍然可以使用 where 条件，以及 ORDER BY 和 LIMIT 等选项。使用 HAVING 和 GROUP BY 等聚合选项也是支持的。

```
SELECT * FROM employees PARTITION (p0, p2)
    WHERE lname LIKE 'S%';
+----|-------|-------|----------|---------------+
| id | fname | lname | store_id | department_id |
+----|-------|-------|----------|---------------+
|  4 | Jim   | Smith |        2 |             4 |
| 11 | Jill  | Stone |        1 |             4 |
+----|-------|-------|----------|---------------+
2 rows in set (0.00 sec)
```

```
SELECT id, CONCAT(fname, ' ', lname) AS name
    FROM employees PARTITION (p0) ORDER BY lname;
+----|----------------+
| id | name           |
+----|----------------+
|  3 | Ellen Johnson  |
|  4 | Jim Smith      |
|  1 | Bob Taylor     |
|  2 | Frank Williams |
+----|----------------+
4 rows in set (0.06 sec)
SELECT store_id, COUNT(department_id) AS c
    FROM employees PARTITION (p1,p2,p3)
    GROUP BY store_id HAVING c > 4;
+---|----------+
| c | store_id |
+---|----------+
| 5 |        2 |
| 5 |        3 |
+---|----------+
2 rows in set (0.00 sec)
```

分支选择支持所有类型的分区表，无论是 Range 分区或是 Hash 分区等。对于 Hash 分区，如果没有指定分区名，会自动使用 p0、p1、p2、……、或 pN-1 作为分区名。

在 INSERT ... SELECT 的 SELECT 中也是可以使用分区选择的。

分区的约束和限制

- 不支持使用 ALTER TABLE ... CHANGE COLUMN 语句更改分区表的列类型。
- 不支持使用 ALTER TABLE ... CACHE 语句将分区表设为缓存表。
- 不支持在分区表上创建外键。

分区键，主键和唯一键

分区表的每个唯一键，必须包含分区表达式中用到的所有列

这里所指的唯一也包含了主键，因为根据主键的定义，主键必须是唯一的。

如果既没有主键，也没有唯一键，则不存在这个限制。

关于函数的分区限制

只有以下函数可以用于分区表达式：

ABS()
CEILING()
DATEDIFF()
DAY()
DAYOFMONTH()
DAYOFWEEK()
DAYOFYEAR()
EXTRACT() (see EXTRACT() function with WEEK specifier)
FLOOR()
HOUR()
MICROSECOND()
MINUTE()
MOD()
MONTH()
QUARTER()
SECOND()
TIME_TO_SEC()
TO_DAYS()
TO_SECONDS()
UNIX_TIMESTAMP() (with TIMESTAMP columns)
WEEKDAY()
YEAR()
YEARWEEK()

兼容性

目前支持 Range 分区、Range Columns 分区、List 分区、List COLUMNS 分区、Hash 分区和 Key 分区，其它的 MySQL 分区类型尚不支持。

对于 Key 分区，目前不支持分区字段为空的情况。

在分区管理方面，任何可能涉及数据移动的操作目前都不支持。这包括但不限于调整 Hash 分区

表的分区数量、修改 Range 分区表的范围以及合并分区等。

对于暂不支持的分区类型，在建表时会忽略分区信息，以普通表的形式创建，并且会报 Warning。

Load Data 暂时不支持分区选择。

对于分区表，select * from t 的返回结果是分区之间无序的。这跟 MySQL 不同，MySQL 的返回结果是分区之间有序，分区内部无序。