17 分区表是什么，又该怎么使用呢？

更新时间：2020-04-10 09:46:20

> 老骥伏枥，志在千里； 烈士暮年，壮心不已。 ——曹操

随着业务不断发展，用户量不断增加，MySQL 表中保存的数据也会越来越多，而数据量膨胀带来的最大问题是查询和删除数据的性能。为了解决这一问题，MySQL 提出了分区表的概念，即把数据按照某一种规则分开存放，这样无论是查询还是删除都可以仅仅针对数据的子集进行操作，极大的减轻了数据库的压力。这一节里，我们就来看一看分区表的概念，以及应该怎样使用分区表。

1 分区表概述

数据分区是一种物理（与它相对的概念叫做逻辑）数据库的设计技术，它的目的是为了在特定的 SQL 操作中减少数据读写的总量以降低响应延迟。下面，我们就一起来看看关于它的详细说明、它的优势以及分类。

1.1 分区表的说明

分区表并不是生成新的数据表，逻辑上仍然是一张表，但是物理数据却存储在多个文件中。在使用分区表之前，需要先确定你当前的 MySQL 是否支持分区表：

```
-- 如果存在 Name 是 partition，Status 是 ACTIVE 的记录，则说明当前的 MySQL 支持分区表
mysql> SHOW PLUGINS;
+----------------------------+----------+--------------------+----------------+---------+
| Name                       | Status   | Type               | Library        | License |
+----------------------------+----------+--------------------+----------------+---------+
.....
| partition                  | ACTIVE   | STORAGE ENGINE     | NULL           | GPL     |
.....
+----------------------------+----------+--------------------+----------------+---------+
```

MySQL 中的分区表只支持水平分区，即按行拆分，不支持垂直分区（按列拆分）。当然，拆分过程也不是随机的，MySQL 会根据用户定义的规则，将表中的数据行拆分到不同的分区中去。而这里所说的规则也就是分区表所支持的分类。

1.2 分区表的分类

目前，MySQL 支持四种分区类型，下面，我先简单地对它们进行概念性讲解：

- RANGE：范围分区，这种分类将数据划分为连续的范围
- LIST：枚举分区，这种分类通过预定义的枚举值对数据进行划分
- HASH：哈希分区，这种分类按照哈希函数将数据散列到不同的分区中
- KEY：键值分区，是 HASH 分区的一种延伸，哈希值是由 MySQL 系统来产生

在实际的应用中，RANGE 和 HASH 分区是最常被使用的，所以，对于接下来的内容，应该重点关注这两类分区。

1.3 分区表的优势

需要知道，只有当前或预期表的数据量很大（这里的很大确实很难定义，这里我建议不低于100万行）时，才有必要对表进行分区。相对来说，分区之后会带来以下优势：

- 分区表的数据可以分布在多个磁盘上，所以，它可以容纳更多的数据
- 数据管理方便，对于不需要的数据可以直接删除分区
- 查询数据不需要全表扫描，只需要查询对应的分区，大大提升检索效率
- 涉及聚合（函数）查询时，可以非常简单的做到数据合并

不过，分区表也只是在特定的场景下才会体现出这些优势。所以，不要盲目的使用分区表，确定需求的同时，也要理解分区表的意义。

2 RANGE 分区

RANGE 分区是最常见的分区类型，它的规则和思想非常简单，而且普适性强。这里我首先讲解 RANGE 分区的思想和特性，再以实例的形式讲解应该怎样做到 RANGE 分区。

2.1 RANGE 分区的特性

RANGE 分区基于一个给定连续区间的列值，最常见的是基于时间字段，但是 MySQL 建议分区列最好是整型。所以，如果分区列是时间，需要转换成整型。例如，日期可以使用 TO_DAYS 函数、timestamp 可以使用 UNIX_TIMESTAMP 函数等等。

可以知道，RANGE 分区的规则是非常简单的，下面，我对此做出总结：

- 需要为每个分区指定分区范围

- 所有的分区范围必须保证是连续的，并且不能存在交集

2.2 RANGE 分区的实践

关于 RANGE 分区的实践操作，我这里会讲解三个主题：创建分区、删除分区以及新增分区。首先，对于创建分区来说，又分为两种情况：创建表时指定分区、对已经存在的表进行分区。

RANGE 分区最常见的应用是报表型数据存储，由于报表型数据通常按天存储（也可能按小时），那么，时间区间就是最好的分区键。首先，我们先去看创建表时指定分区的案例。如下所示：

```
CREATE TABLE `ad_unit_cost` (
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
  `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
  `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
  `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
  `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
PARTITION BY RANGE (TO_DAYS(date_)) (
 PARTITION p0 VALUES LESS THAN (TO_DAYS('2019-01-01')),
 PARTITION p1 VALUES LESS THAN (TO_DAYS('2019-02-01')),
 PARTITION p2 VALUES LESS THAN (TO_DAYS('2019-03-01')),
 PARTITION p3 VALUES LESS THAN (TO_DAYS('2019-04-01')),
 PARTITION p4 VALUES LESS THAN (TO_DAYS('2019-05-01')),
 PARTITION p5 VALUES LESS THAN (TO_DAYS('2019-06-01')),
 PARTITION p6 VALUES LESS THAN (MAXVALUE)
);
```

我这里创建了 ad_unit_cost 表，它基于 date_ 按照时间范围分区，除了 p6 之外，其他的分区都很好理解。MAXVALUE 是一个无穷大的值，p6 是一个默认分区，如果插入的数据日期大于等于 2019-06-01（p5 分区所指定的区间上限）时，将会进入到这个分区。但是，需要注意，如果在定义表时没有指定默认分区，插入数据时间在所有分区之外时，将会报错。

另外，可以看到，ad_unit_cost 表并没有定义主键，这并不是我忘记了，而是定义就会报错（你可以试一试给 ad_unit_cost 加上主键，再去创建分区表）。在 MySQL 中，创建分区表有一个限制条件：分区表中的唯一索引（包括主键），都必须包含分区表表达式中的所有列。

创建完了分区表，我们可以去查看分区表的基本信息，这些信息保存在系统库 information_schema 的 PARTITIONS 表中。如下所示：

```
mysql> SELECT
    ->     TABLE_NAME,
    ->     PARTITION_NAME,
    ->     PARTITION_METHOD,
    ->     PARTITION_EXPRESSION,
    ->     PARTITION_DESCRIPTION
    ->   FROM
    ->     information_schema.`PARTITIONS`
    ->   WHERE
    ->     table_name = 'ad_unit_cost';
+--------------+----------------+------------------+----------------------+------------------------+
| TABLE_NAME   | PARTITION_NAME | PARTITION_METHOD | PARTITION_EXPRESSION | PARTITION_DESCRIPTION |
+--------------+----------------+------------------+----------------------+------------------------+
| ad_unit_cost | p0             | RANGE            | TO_DAYS(date_)       | 737425                 |
| ad_unit_cost | p1             | RANGE            | TO_DAYS(date_)       | 737456                 |
| ad_unit_cost | p2             | RANGE            | TO_DAYS(date_)       | 737484                 |
| ad_unit_cost | p3             | RANGE            | TO_DAYS(date_)       | 737515                 |
| ad_unit_cost | p4             | RANGE            | TO_DAYS(date_)       | 737545                 |
| ad_unit_cost | p5             | RANGE            | TO_DAYS(date_)       | 737576                 |
| ad_unit_cost | p6             | RANGE            | TO_DAYS(date_)       | MAXVALUE               |
+--------------+----------------+------------------+----------------------+------------------------+
7 rows in set (0.02 sec)
```

搞明白了创建表时指定分区的情况，下面我们来看一看对已经存在的表进行分区。假设我们创建了 ad_unit_cost 表，执行如下语句：

```
CREATE TABLE `ad_unit_cost` (
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
  `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
  `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
  `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
  `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```


之后，随着数据量不断增长，想要对它按照时间范围进行分区，可以这样：

```
ALTER TABLE ad_unit_cost PARTITION BY RANGE (TO_DAYS(date_)) (
 PARTITION p0 VALUES LESS THAN (TO_DAYS('2019-01-01')),
 PARTITION p1 VALUES LESS THAN (TO_DAYS('2019-02-01')),
 PARTITION p2 VALUES LESS THAN (TO_DAYS('2019-03-01')),
 PARTITION p3 VALUES LESS THAN (TO_DAYS('2019-04-01')),
 PARTITION p4 VALUES LESS THAN (TO_DAYS('2019-05-01')),
 PARTITION p5 VALUES LESS THAN (TO_DAYS('2019-06-01')),
 PARTITION p6 VALUES LESS THAN (MAXVALUE)
);
```

我们使用 ALTER TABLE 的方式实现了同样的效果（查看下 information_schema.PARTITIONS 表，验证下是否和我说的一致），MySQL 会自动按照规则将表中的数据分配到对应的分区中去。

如果创建了多余的分区，实际上用不到，我们也可以手动把它删除。但是，需要注意，删除一个分区将会把它所有的数据都删除，即物理删除。我们可以尝试删除 ad_unit_cost 的 p3 分区，如下所示：

```
ALTER TABLE ad_unit_cost DROP PARTITION p3;
```

非常简单，直接 ALTER TABLE DROP PARTITION 就可以了。那么，想要增加分区又应该怎么办呢？MySQL 允许我们使用 ALTER TABLE ADD PARTITION 的语法增加分区，执行如下语句：

```
mysql> ALTER TABLE ad_unit_cost ADD PARTITION(PARTITION p3 VALUES LESS THAN (TO_DAYS('2019-04-01')));
ERROR 1481 (HY000): MAXVALUE can only be used in last partition definition
```

很遗憾，执行报错了。这是因为对于 RANGE 分区的表，只能添加新分区到分区列表的高端，且当前的分区不能带有 MAXVALUE 分区，否则无法增加分区。所以，增加分区的工作就交给你去完成（创建分区表时不要指定 MAXVALUE 分区即可）。

3 LIST 分区

LIST 分区与 RANGE 分区在语法方面是高度相似的，区别在于 LIST 是枚举值列表的集合，RANGE 是连续的区间值集合。也正是因为 LIST 分区限制枚举值，它在实际的企业级开发中使用的并不多。

3.1 LIST 分区的特性

关于 LIST 分区的特性，你需要知道：创建 LIST 分区时，如果有主键，分区时主键必须在其中，不然就会报错；如果表中没有定义主键，分区则不会有限制。但是，如果定义的是联合主键索引（多个列构成主键），那么，分区键包含某一个列就可以，不要求整体的主键。

最后，建议 LIST 分区键是 NOT NULL 的列，否则插入 NULL 值记录时，如果枚举值列表中没有定义 NULL 则会插入失败。这和之前所说的 RANGE 分区不同，RANGE 分区遇到这种情况，会把记录作为最小分区存储。

3.2 LIST 分区的实践

首先，我们先来使用如下的建表语句创建 LIST 分区表：

```
mysql> CREATE TABLE `ad_unit_cost` (
    ->   `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键 id',
    ->   `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
    ->   `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
    ->   `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
    ->   `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
    ->   `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd',
    ->   PRIMARY KEY (`id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    -> PARTITION BY LIST (user_id) (
    -> PARTITION p0 VALUES IN (1, 3, 5),
    -> PARTITION p1 VALUES IN (2, 4, 6)
    -> );
-- ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function
```


遗憾的事情又发生了，创建分区表失败了。这也就是我之前所说的：主键必须在分区键中，否则会报错。所以，我们换一种方式继续创建 LIST 分区表。如下所示：

```
mysql> CREATE TABLE `ad_unit_cost` (
    ->   `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT 'id',
    ->   `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
    ->   `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
    ->   `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
    ->   `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
    ->   `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd',
    ->   PRIMARY KEY (`id`, `user_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    -> PARTITION BY LIST (user_id) (
    -> PARTITION p0 VALUES IN (1, 3, 5),
    -> PARTITION p1 VALUES IN (2, 4, 6)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

这一次我将主键设置为 id 和 user_id 的联合索引，符合 LIST 分区的限制，创建成功。同样，与 RANGE 分区类似，对于不再需要的分区，也可以直接删除掉（注意，删除分区也会删除该分区中所有的数据）。如下所示：

```
ALTER TABLE ad_unit_cost DROP PARTITION p1;
```

LIST 分区表增加分区的限制会比 RANGE 分区表弱一些，它在增加分区时，仅要求不包含现在分区值列表中的任意值。下面，我们验证这个说法并给 ad_unit_cost 表增加新的分区：

```
-- 由于原来的 p0 分区中枚举值存在5，所以，重复定义报错了
mysql> ALTER TABLE ad_unit_cost ADD PARTITION(PARTITION p1 VALUES IN (5, 7, 9));
ERROR 1495 (HY000): Multiple definition of same constant in list partitioning

-- 与原来的分区枚举值不冲突，增加分区成功
mysql> ALTER TABLE ad_unit_cost ADD PARTITION(PARTITION p1 VALUES IN (7, 9, 11));
Query OK, 0 rows affected (0.22 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4 HASH 分区

日常工作中，当我们提到哈希时，你第一时间会想到什么呢？我会想到的是负载均衡，把流量、数据等等按照哈希值打散，以减轻系统的整体压力。HASH 分区的应用理念也是类似的，接下来，我们就一起来看看。

4.1 HASH 分区的特性

HASH 分区与之前介绍的 RANGE 和 LIST 有很大不同，它是基于给定的分区个数，将数据分配到不同的分区中。另外，HASH 分区只能针对整数进行 HASH，对于非整型的字段只能通过表达式（可以是 MySQL 中任意有效的函数或表达式）将其转换为整数。但是，对于非整型的哈希过程会多出一步表达式的计算操作，会影响数据库性能。

HASH 分区的优点在于数据分布的比较均匀（但是，这也要看你实际存储的数据），且可以在分区之后重新定义分区的大小，MySQL 会将数据重新再分配（但是这会严重的影响数据库性能，应该谨慎使用）。

4.2 HASH 分区的实践

想要创建 HASH 分区表，需要在 CREATE TABLE 语句之后添加 PARTITION BY HASH(expr) 子句，其中，expr 是一个返回一个整数的表达式。下面，我给出一个 HASH 分区表的示例：

```
CREATE TABLE `ad_unit_cost` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键 id',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
  `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
  `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
  `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
  `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd',
  PRIMARY KEY (`id`, `user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
PARTITION BY HASH(user_id)
PARTITIONS 4;
```

“PARTITIONS 4” 标识创建4个分区，如果不显示的指定 PARTITIONS 子句，则默认分区数是1。但是，MySQL 不允许只写 PARTITIONS，而不指定分区数。执行创建语句完成创建之后，我们可以验证下结果是否符合预期：

```
mysql> SELECT
    ->     TABLE_NAME,
    ->     PARTITION_NAME,
    ->     PARTITION_METHOD,
    ->     PARTITION_EXPRESSION,
    ->     PARTITION_DESCRIPTION
    ->   FROM
    ->     information_schema.`PARTITIONS`
    ->   WHERE
    ->     table_name = 'ad_unit_cost';
+--------------+----------------+------------------+----------------------+------------------------+
| TABLE_NAME   | PARTITION_NAME | PARTITION_METHOD | PARTITION_EXPRESSION | PARTITION_DESCRIPTION |
+--------------+----------------+------------------+----------------------+------------------------+
| ad_unit_cost | p0             | HASH             | user_id              | NULL                   |
| ad_unit_cost | p1             | HASH             | user_id              | NULL                   |
| ad_unit_cost | p2             | HASH             | user_id              | NULL                   |
| ad_unit_cost | p3             | HASH             | user_id              | NULL                   |
+--------------+----------------+------------------+----------------------+------------------------+
4 rows in set (0.00 sec)
```

正如我之前所说，可以给 HASH 分区表重新指定分区大小。可以执行以下语句，将 ad_unit_cost 表的分区个数增加到10个（自行验证结果）：

```
mysql> ALTER TABLE ad_unit_cost ADD PARTITION PARTITIONS 6;
Query OK, 0 rows affected (0.80 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

那么，如果想要缩减分区的个数呢 ？假如我只想让 ad_unit_cost 表保留2个分区，则可以使用 COALESCE 关键字。如下所示（同样，自行去验证结果）：


```
mysql> ALTER TABLE ad_unit_cost COALESCE PARTITION 2;
Query OK, 0 rows affected (0.55 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

5 KEY 分区

相对来说，在实际的应用中，KEY 分区的出场率是最低频的。KEY 分区的语法和思想与 HASH 分区是极为相似的，只是在分区算法的实现上不同。接下来，一起简单的看一看吧。

5.1 KEY 分区的特性

由于 KEY 分区与 HASH 分区是类似的，所以，对于它的特性讲解主要集中在与 HASH 分区特性的对比。

- KEY 分区对象必须为数据列，不能是基于列的表达式
- 如果表中存在主键或唯一键，KEY 分区的分区键可以不指定；如果没有，必须指定
- HASH 分区使用 MOD 算法实现分区，而 KEY 分区则使用的是 MD5

5.2 KEY 分区的实践

如果我们想要创建的 KEY 分区表定义了主键，则分区键可以不指定，默认即为主键。如下所示：

```
CREATE TABLE `ad_unit_cost` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键 id',
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
  `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
  `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
  `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
  `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
PARTITION BY KEY()
PARTITIONS 4;
```

但是，如果既没有主键，也没有唯一键，不指定的情况下就会报错。可以看到如下建表示例：

```
mysql> CREATE TABLE `ad_unit_cost` (
    ->   `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键 id',
    ->   `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
    ->   `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
    ->   `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
    ->   `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
    ->   `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd'
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    -> PARTITION BY KEY()
    -> PARTITIONS 4;
ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it must be defined as a key
```

这种情况下，想要正确的创建 KEY 分区表，就必须显示的指定分区键。创建表的语句可以修改为（可以自行完成表的创建以及验证结果是否符合预期）：


```
CREATE TABLE `ad_unit_cost` (
  `user_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联所属用户',
  `plan_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广计划 id',
  `unit_id` bigint(20) NOT NULL DEFAULT '0' COMMENT '关联推广单元 id',
  `cost` bigint(20) NOT NULL DEFAULT '0' COMMENT '推广单元花费金额',
  `date_` date NOT NULL COMMENT '数据日期，精确到天，yyyy-MM-dd'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
PARTITION BY KEY(user_id)
PARTITIONS 4;
```

由于 HASH 分区与 KEY 分区的思想是一样的，所以，对于分区数量的管理（增加或减少）它们也保持一致。我们可以使用 `COALESCE` 减少分区的数量，也可以使用 `ALTER TABLE ADD PARTITION` 增加分区的数量。

6 总结

分区表的使用频率确实是不高的，它通常仅仅应用于业务系统的报表型（OLAP）应用。但是，它却是 MySQL 对大数据量存储、管理的一种解决方案，也是需要你去理解和掌握的。至少要知道这个概念、MySQL 提供了这样一种技术，等遇到此类问题时，才不至于措手不及，一筹莫展。

7 问题

你能给 RANGE 分区表增加新的分区吗？你是怎么做的呢？

如果我只是想清空分区，但是不删除分区，你知道应该怎么做吗？

你觉得对于分区表来说，查询会带来什么好处呢？可以使用 EXPLAIN 语句验证下你的想法。

你在工作中使用过分区表吗？是怎么使用的呢？或者谈一谈你理解的分区表的应用场景？

8 参考资料

《高性能 MySQL（第三版）》

MySQL 官方文档：Partitioning

}

16 视图应该怎样去应用和管理呢？

18 外键是一个非常特殊的存在