Neuedu

MySQL数据库开发技术

—— 分区管理

V1.0

本章内容

节	知识点	掌握程度	难易程度
表分区概述	表为什么要分区	了解	
	什么是表分区	了解	
	表分区的优点	了解	
分区类型	range分区	掌握	
	list分区	掌握	
	column分区	掌握	
	hash分区	了解	难
	key分区	了解	难
分区管理	range和list分区管理	掌握	
	hash和key分区管理	掌握	
	分区管理	了解	
分区的局限性	约束的限制	了解	
	函数的限制	了解	

表为什么要分区

- 表为什么要分区
  - 当表中的数据量不断增大，查询数据的速度就会变慢，应用程序的性能就会下降，这时就应该考虑对表进行分区。
  - 表进行分区后，逻辑上表仍然是一张完整的表，只是将表中的数据在物理上存放到多个表空间(物理文件上)，这样查询数据时，不至于每次都扫描整张表。

什么是表分区

- 什么是表分区
  - 当表分区通俗来讲就是允许把一个数据表根据一定的规则，跨文件系统划分成多个可以设置为任意大小的部分。

- MySQL从5.1起开始支持表分区，MySQL安装后，默认是开启表分区支持的。

- 可以通过下面语句来查看你的MySQL是否支持分区：

```
show variables like ‘%partition%’;
```

表分区的优点

- 表分区的优点
  - 表分区可以存储更多的数据，与单个磁盘或文件系统分区相比。
  - 分区表的数据更容易维护，如：想批量删除大量数据可以使用清除整个分区的方式。另外，还可以对一个独立分区进行优化、检查、修复等操作。可以备份和恢复独立的分区，这在非常大的数据集的场景下效果非常好。
  - 分区表的数据可以分布在不同的物理设备上，从而高效地利用多个硬件设备。
  - 优化查询，在where字句中包含分区列时，可以只使用必要的分区来提高查询效率，同时在涉及sum()和count()这类聚合函数的查询时，可以在每个分区上面并行处理，最终只需要汇总所有分区得到的结果。

表分区的优点

- 表分区的优点
  - 性能的提升，在扫描操作中，如果MySQL的优化器知道哪个分区中才包含特定查询中需要的数据，它就能直接去扫描那些分区的数据，而不用浪费很多时间扫描不需要的地方了。
  - MySQL分区技术可以让DBA对数据的管理能力提升。通过优良的MySQL分区，DBA可以简化特定数据操作的执行方式。

分区类型

- 分区类型
  - RANGE分区
  - LIST分区
  - COLUMNS分区
  - HASH分区
  - KEY分区

分区类型

- RANGE分区
  - range 分区基于属于一个给定连续区间的列值进行分配

```
create table employees (
    id int not null,
    fname varchar(30),
    lname varchar(30),
    hired date not null default '1970-01-01',
    separated date not null default '9999-12-31',
    job_code int not null,
    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
);
```

分区类型

- LIST分区
  - list 分区类似range分区,它们的主要区别在于，list分区中每个分区的定义和选择是基于某列的值从属于一个集合，而range分区是从属于一个连续区间值的集合

```
create table employees (
    id int not null,
    fname varchar(30),
    lname varchar(30),
    hired date not null default '1970-01-01',
    separated date not null default '9999-12-31',
    job_code int,
    store_id int
)
partition by list(store_id)
    partition pnorth values in (3,5,6,9,17),
    partition peast values in (1,2,10,11,19,20),
    partition pwest values in (4,12,13,14,18),
    partition pcentral values in (7,8,15,16)
);
```

分区类型

- COLUMNS分区
  - columns分区是range分区或list分区的一种变体，支持非整形字段作为分区的键，也可以用多个字段组合起来作为分区的键
- columns分区可允许使用的分区键类型有：
  - 所有的整形：tinyint, smallint, mediumint, int , bigint （和range分区和list分区相同），不包括decimal和float这种数字类型的。
  - date 和 datetime
  - 字符型：chra, varchar, binary, varbinary，不包括text和blob型

分区类型

- HASH分区
  - hash分区基于用户定义的表达式的返回值来进行选择的分区，该表达式使用将要插入到表中的这些行的列值进行计算。这个函数可以包含MySQL 中有效的、产生非负整数值的任何表达式
- hash分区主要用来确保数据在预先确定数目的分区中平均分布。在range和list分区中，必须明确指定一个给定的列值或列值集合应该保存在哪个分区中；而在hash分区中，MySQL 自动完成这些工作，我们所要做的只是基于将要被哈希的列值指定一个列值或表达式，以及指定被分区的表将要被分割成的分区数量。

分区类型

- HASH分区
  - 要使用hash分区来分割一个表，要在create table 语句上添加一个“partition by hash (expr)”子句，其中“expr”是一个返回一个整数的表达式。它可以仅仅是字段类型为MySQL 整型的一列的名字。此外，需要在后面再添加一个“partitions num”子句，其中num 是一个非负的整数，它表示表将要被分割成分区的数量。
  - 如果没有包括一个PARTITIONS子句，那么分区的数量将默认为1
  - 如果在关键字“PARTITIONS”后面没有加上分区的数量，将会出现语法错误。

分区类型

- HASH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 not null default '9999-12-31',
    job_code int,
    store_id int
)
partition by hash(year(hired))
partitions 4 ;
```

分区类型

- KEY分区
  - key 分区按照key进行分区类似于按照hash分区，除了hash分区使用的用户定义的表达式，而key分区的哈希函数是由MySQL 服务器提供。

```
create table tk (
    col1 int not null,
    col2 char(5),
    col3 date
)
partition by linear key (col1)
partitions 3;
```

分区管理— range分区和list分区

- 删除分区
  - 从一个按照range或list分区的表中删除一个分区，可以使用带一个drop partition子句的alter table命令来实现:
- 例如删除分区：

```
alter table tr drop partition p2;
```

- 例如有如下分区表的定义：

```
create table tr(
id int,
name varchar(50),
purchased date
)
partition by range(year(purchased))
(
partition p0 values less than(1990),
partition p1 values less than(1995),
partition p2 values less than(2000),
partition p3 values less than(2005)
);
```

分区管理— range分区和list分区

- 添加分区
  - 要增加一个新的RANGE或LIST分区到一个前面已经分区了的表，使用“ALTER TABLE ... ADD PARTITION”语句。
- 例如有一个组织的全体成员数据的分区表，该表的定义如下：

```
create table members(
id int,
fname varchar(25),
lname varchar(25),
dob date
)
partition by range(year(dob))
(
partition p0 values less than (1970),
partition p1 values less than (1980),
partition p2 values less than (1990)
);
```

分区管理— range分区和list分区

- 添加分区
- 假设成员的最小年纪是16岁。随着日历接近2005年年底，要接纳1990年（以及以后年份）出生的成员。可以按照下面的方式，修改成员表来容纳出生在1990－1999年之间的成员：

```
alter table add partition (partition p3 values less
than (2000));
```

- 对于通过RANGE分区的表，只可以使用ADD PARTITION添加新的分区到分区列表的高端。设法通过这种方式在现有分区的前面或之间增加一个新的分区，将会导致下面的一个错误：

```
alter table add partition (partition p3 values less
than (1960));
```

分区管理— range分区和list分区

- 类似的可以增加新的分区到已经通过list分区的表。例如，假定有如下定义的一个表：

```
create table tt(
id int,
data int
)
partition by list(data)
(
partition p0 values in (5, 10, 15),
partition p1 values in (6, 12, 18)
);
```

分区管理— range分区和list分区

- 添加分区
- 可以通过下面的方法添加一个新的分区，用来保存拥有数据列值7，14和21的行：

```
alter table tt add partition (partition p2 values in (7, 14, 21));
```

- 注意：不能添加这样一个新的LIST分区，该分区包含有已经包含在现有分区值列表中的任意值。如果试图这样做，将会导致错误：

```
alter table tt add partition (partition np values in (4, 8, 12));
```

分区管理— range分区和list分区

- 修改分区
- 使用“reorganize partition”拆分或合并分区，没有数据丢失。在执行上面的语句中，MySQL 把保存在分区s0和s1中的所有数据都移到分区p0中。

- “reorganize partition”的基本语法是：

```
alter table tbl_name reorganize partition
partition_list into (partition_definitions);
```

分区管理— range分区和list分区

- 修改分区
- alter table members reorganize partition p0 into(
  partition s0 values less than (1960),
  partition s1 values less than (1970)
  );

- alter table members reorganize partition s0,s1 into(
  partition p0 values less than (1970)
  );

- alter table members reorganize partition p0,p1,p2,p3
  into (
  partition m0 values less than (1980),
  partition m1 values less than (2000)
  );

分区管理— range分区和list分区

- 修改分区
- alter table tt reorganize partition p1,np into(
  partition p1 values in (6, 18),
  partition np values in (4, 8, 12)
  );

分区管理— range分区和list分区

- 修改分区
- 当使用“ALTER TABLE ... REORGANIZE PARTITION”来对已经按照RANGE和LIST分区表进行重新分区时，需要注意：
  1. 用来确定新分区模式的PARTITION子句使用与用在CREATE TABLE中确定分区模式的PARTITION子句相同的规则

  2. 新分区模式不能有任何重叠的区间（适用于按照RANGE分区的表）或值集合（适用于重新组织按照LIST分区的表）。

分区管理— range分区和list分区

3. partition_definitions 列表中分区的合集应该与在 partition_list 中命名分区的合集占有相同的区间或值集合。

例如，分区p1和p2总共覆盖了1980到1999的这些年。因此，对这两个分区的重新组织都应该覆盖相同范围的年份。

4. 对于按照range分区的表，只能重新组织相邻的分区，不能跳过range分区。

5. 不能使用REORGANIZE PARTITION来改变表的分区类型；也就是说，例如，不能把RANGE分区变为HASH分区，反之亦然。也不能使用该命令来改变分区表达式或列。

分区管理—hash分区和key分区

- hash分区和key分区的管理：

```
create table clients (
id int,
fname varchar(30),
lname varchar(30),
signed date
)
partition by hash(month(signed)) partitions 12;
```

分区管理—hash分区和key分区

- 不能使用与从按照range或list分区的表中删除分区相同的方式来从hash或key分区的表中删除分区。但是，可以使用“alter table ... coalesce partition”命令来合并hash或key分区。

- coalesce不能用来增加分区的数量，要增加顾客表的分区数量从12到18，使用“alter table ... add partition”，具体如下：

  ```
  alter table clients add partition partitions 18;
  ```

- 要减少分区的数量从12到6，执行下面的ALTER TABLE命令：

  ```
  alter table clients coalesce partition 6;
  ```

分区管理

- 要减少分区的数量从12到6，执行下面的ALTER TABLE命令：
- alter table clients coalesce partition 6;

- 对于按照HASH，KEY，LINEAR HASH，或LINEAR KEY分区的表，COALESCE能起到同样的作用。下面是一个类似于前面例子的另外一个例子，它们的区别只是在于表是按照LINEAR KEY 进行分区：

```
create table clients_lk (
id int,
fname varchar(30),
lname varchar(30),
signed date
)
partition by linear key(signed) partitions 12;

alter table clients_lk coalesce partition 6;
```

分区管理

- 改变分区的方式

```
alter table tablename  partition by hash(id) partitions 2;
```

- 重建分区

```
alter table t1 rebuild partition p0, p1;
```

- 优化分区:

```
alter table t1 optimize partition p0, p1;
```

- 分析分区

```
alter table t1 analyze partition p3;
```

- 修护分区:

```
alter table t1 repair partition p0,p1;
```

- 检查分区:

```
alter table t1 check partition p1;
```

分区的局限性

- 关于Partitioning Keys, Primary Keys, and Unique Keys的限制
  - 在5.1中分区表对唯一约束有明确的规定，每一个唯一约束必须包含在分区表的分区键（也包括主键约束）。

分区的局限性

- 例如：

```
create table t1(
id int not null,
uid int not null,
primary key (id),
unique key (uid)
)
partition by range (id)
(
partition p0 values less than(5),
partition p1 values less than(10)
);
```

- ERROR 1503 (HY000): A UNIQUE INDEX must include all columns in the table's partitioning function

分区的局限性

- 关于函数的限制
- 在建立分区表的语句中，只能包含例如ABS()　、CEILING()　、FLOOR()　、DAY()　、HOUR()　、MINUTE()　、MOD()　、MONTH()　、QUARTER()　、SECOND()　、TO_DAYS()　、YEAR()等返回整型的函数。

- 分区键必须是INT类型，或者通过表达式返回INT类型，可以为NULL。唯一的例外是当分区类型为KEY分区的时候，可以使用其他类型的列作为分区键（ BLOB or TEXT 列除外）。

```
create table tkc (c1 char)
partition by key(c1)
partitions 4;

query ok, 0 rows affected (0.00 sec)
```

分区的局限性

- 运算限制
  - 支持加减乘等运算出现在分区表达式，但是运算后的结果必须是一个INT或者NULL。 |, &, ^, <<, >>, , ~ 等不允许出现在分区表达式。
- 最多支持1024个分区，包括子分区。
  - 当你建立分区表包含很多分区但没有超过1024限制的时候，如果报错 Got error 24 from storage engine，那意味着你需要增大open_files_limit参数。

应用案例

- 假定有20个音像店，员工表信息按照如下分区进行存储：

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    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 (21)
);
```

应用案例

- 按照这种分区方案，在商店1到5工作的雇员相对应的所有行被保存在分区P0中，商店6到10的雇员保存在P1中，依次类推。注意，每个分区都是按顺序进行定义，从最低到最高。
- 对于一条新数据(72, ‘Michael’, ‘Widenius’, ‘1998-06-25’, NULL, 13)， 可以很容易地确定它将插入到p2分区中
- 如果删除商店1到5工作的雇员，可以直接找到p0分区，进行删除，效率会比不分区高

应用案例

- 假定这20个音像店，分布在4个有经销权的地区，如下表所示：

地区	商店ID 号
北区	3，5，6，9，17
东区	1，2，10，11，19，20
西区	4，12，13，14，18
中心区	7，8，15，16

- 要按照属于同一个地区商店的行保存在同一个分区中的方式来分割表

应用案例

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)   PARTITION BY LIST(store_id)
    PARTITION pNorth VALUES IN (3,5,6,9,17),
    PARTITION pEast VALUES IN (1,2,10,11,19,20),
    PARTITION pWest VALUES IN (4,12,13,14,18),
    PARTITION pCentral VALUES IN (7,8,15,16)
);
```

应用案例

- 分区使得在表中增加或删除指定地区的雇员记录变得容易起来。
- 例如，假定西区的所有音像店都卖给了其他公司。那么与在西区音像店工作雇员相关的所有记录（行）可以使用语句“ALTER TABLE employees DROP PARTITION pWest；”来进行删除，它与具有同样作用的DELETE （删除）查询“DELETE FROM employees WHERE store_id IN (4,12,13,14,18)；”比起来，要有效得多。

小结

- 掌握常用的类型的分区
- 了解修改分区和添加分区
- 了解分区的局限性

课后作业

- 1.创建员工表，结构如下：

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

- 2.按照入职的年份进行分区，在1991年前入职的所有雇员的记录保存在分区p0中，1991年到1995年期间入职的所有雇员的记录保存在分区p1中， 1996年到2000年期间入职的所有雇员的记录保存在分区p2中，2000年后入职的所有工人的信息保存在p3中。

Neuedu