

## 《数据库管理技术》实验报告

实验名称:		分区表和数据压缩			
班级	计算机 11706	学号	1704210630	姓名	董尧尧
<h3>一、实验目的</h3> <ol style="list-style-type: none"><li>1、创建学生 range 分区表 （入学年份 p1, p2, p3, p4）</li><li>2、插入数据</li><li>3、查看数据表的分区</li><li>4、查询 P2 中的数据</li><li>5、删除 P2</li><li>6、合并分区 p3 p4、limit</li><li>7、数据压缩测试（两种压缩比）</li></ol> <h3>二、实验内容和步骤</h3> <h4>1、创建学生 range 分区表</h4> <h5>(1) 创建学生表</h5> <pre>CREATE TABLE STUDENT(     ID INT(11) PRIMARY KEY,     NAME VARCHAR(20),     SEX INT(1) DEFAULT 1,     ENTER_DATE DATETIME );</pre> <pre>mysql&gt; CREATE TABLE STUDENT( -&gt; ID INT(11) PRIMARY KEY, -&gt; NAME VARCHAR(20), -&gt; SEX INT(1) DEFAULT 1, -&gt; ENTER_DATE DATETIME -&gt; ); Query OK, 0 rows affected (0.53 sec)</pre> <p>创建学生表</p> <h5>(2) 创建 range 分区表</h5> <pre>ALTER TABLE STUDENT PARTITION BY RANGE (YEAR(ENTER_DATE)) (     PARTITION P1 VALUES LESS THAN(1990),     PARTITION P2 VALUES LESS THAN(2000),     PARTITION P3 VALUES LESS THAN(2010),     PARTITION P4 VALUES LESS THAN MAXVALUE );</pre> <pre>mysql&gt; ALTER TABLE STUDENT PARTITION BY RANGE (YEAR(ENTER_DATE))( -&gt; PARTITION P1 VALUES LESS THAN(1990), -&gt; PARTITION P2 VALUES LESS THAN(2000), -&gt; PARTITION P3 VALUES LESS THAN(2010), -&gt; PARTITION P4 VALUES LESS THAN MAXVALUE -&gt; ); ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function</pre> <p>创建 range 分区</p> <h5>(3) 修改主键</h5> <pre>mysql&gt; ALTER TABLE STUDENT DROP PRIMARY KEY; Query OK, 0 rows affected (0.81 sec) Records: 0 Duplicates: 0 Warnings: 0  mysql&gt; ALTER TABLE STUDENT ADD PRIMARY KEY(ID,ENTER_DATE); Query OK, 0 rows affected (0.43 sec) Records: 0 Duplicates: 0 Warnings: 0</pre> <p>修改主键</p>					

#### (4) 创建 range 分区表

```
mysql> ALTER TABLE STUDENT PARTITION BY RANGE (YEAR(ENTER_DATE))(  
-> PARTITION P1 VALUES LESS THAN(1990),  
-> PARTITION P2 VALUES LESS THAN(2000),  
-> PARTITION P3 VALUES LESS THAN(2010),  
-> PARTITION P4 VALUES LESS THAN MAXVALUE  
-> );  
Query OK, 0 rows affected (3.44 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

执行查询

#### (5) 直接在创建表的时候创建 range 分区表

```
mysql> CREATE TABLE STUDENT(  
-> ID INT(11),  
-> NAME VARCHAR(20),  
-> SEX INT(1) DEFAULT 1,  
-> ENTER_DATE DATETIME  
-> )  
-> PARTITION BY RANGE (YEAR(ENTER_DATE))(  
-> PARTITION P1 VALUES LESS THAN(1990),  
-> PARTITION P2 VALUES LESS THAN(2000),  
-> PARTITION P3 VALUES LESS THAN(2010),  
-> PARTITION P4 VALUES LESS THAN(MAXVALUE)  
-> );  
Query OK, 0 rows affected (1.02 sec)
```

直接在创建表的时候创建 range 分区表

### 2、插入数据

```
INSERT INTO STUDENT VALUES ('1','张三',1,'1989-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('2','John',1,'1991-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('3','Tom',1,'1992-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('4','Tack',1,'1993-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('5','Smile',1,'1994-01-25  
00:00:00');  
INSERT INTO STUDENT VALUES ('6','Dse',1,'1997-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('7','李四',1,'2001-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('8','王五',1,'2011-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('9','赵六',1,'2020-01-25 00:00:00');  
INSERT INTO STUDENT VALUES ('10','田七',1,'2019-01-25 00:00:00');
```

```
mysql> SELECT * FROM STUDENT;  
+-----+-----+-----+-----+  
| ID | NAME | SEX | ENTER_DATE |  
+-----+-----+-----+-----+  
| 1 | 张三 | 1 | 1989-01-25 00:00:00 |  
| 2 | John | 1 | 1991-01-25 00:00:00 |  
| 3 | Tom | 1 | 1992-01-25 00:00:00 |  
| 4 | Tack | 1 | 1993-01-25 00:00:00 |  
| 5 | Smile | 1 | 1994-01-25 00:00:00 |  
| 6 | Dse | 1 | 1997-01-25 00:00:00 |  
| 7 | 李四 | 1 | 2001-01-25 00:00:00 |  
| 8 | 王五 | 1 | 2011-01-25 00:00:00 |  
| 9 | 赵六 | 1 | 2020-01-25 00:00:00 |  
| 10 | 田七 | 1 | 2019-01-25 00:00:00 |  
+-----+-----+-----+-----+  
10 rows in set (0.00 sec)
```

### 3、查看数据表的分区

SHOW CREATE TABLE STUDENT; 或者 SHOW CREATE TABLE STUDENT \G;

```
mysql> SHOW CREATE TABLE STUDENT \G;  
***** 1. row *****  
Table: STUDENT  
Create Table: CREATE TABLE `student` (  
  `ID` int(11) DEFAULT NULL,  
  `NAME` varchar(20) DEFAULT NULL,  
  `SEX` int(1) DEFAULT '1',  
  `ENTER_DATE` datetime DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
/*!50100 PARTITION BY RANGE (YEAR(ENTER_DATE))  
(PARTITION P1 VALUES LESS THAN (1990) ENGINE = InnoDB,  
PARTITION P2 VALUES LESS THAN (2000) ENGINE = InnoDB,  
PARTITION P3 VALUES LESS THAN (2010) ENGINE = InnoDB,  
PARTITION P4 VALUES LESS THAN MAXVALUE ENGINE = InnoDB) */  
1 row in set (0.00 sec)
```

查看数据表的分区

#### 4、查询 P2 中的数据

##### ①使用分区查询

```
SELECT * FROM STUDENT PARTITION(P2);
```

```
mysql> SELECT * FROM STUDENT WHERE ENTER_DATE BETWEEN '1990-0-0 00:00:00' AND '1999-12-31 00:00:00';
```

ID	NAME	SEX	ENTER_DATE
2	John	1	1991-01-25 00:00:00
3	Tom	1	1992-01-25 00:00:00
4	Tack	1	1993-01-25 00:00:00
5	Smile	1	1994-01-25 00:00:00
6	Dse	1	1997-01-25 00:00:00

5 rows in set, 3 warnings (0.00 sec)

##### 执行计划

```
mysql> EXPLAIN SELECT * FROM STUDENT WHERE ENTER_DATE BETWEEN '1990-0-0 00:00:00' AND '1999-12-31 00:00:00';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	STUDENT	P2	ALL	NULL	NULL	NULL	NULL	5	20.00	Using where

1 row in set, 3 warnings (0.00 sec)

##### ②使用 WHERE 查询

```
SELECT * FROM STUDENT WHERE ENTER_DATE BETWEEN '1990-0-0 00:00:00' AND '1999-12-31 00:00:00';
```

```
mysql> SELECT * FROM STUDENT PARTITION(P2);
```

ID	NAME	SEX	ENTER_DATE
2	John	1	1991-01-25 00:00:00
3	Tom	1	1992-01-25 00:00:00
4	Tack	1	1993-01-25 00:00:00
5	Smile	1	1994-01-25 00:00:00
6	Dse	1	1997-01-25 00:00:00

5 rows in set (0.00 sec)

##### 执行计划

```
mysql> EXPLAIN SELECT * FROM STUDENT PARTITION(P2);
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	STUDENT	P2	ALL	NULL	NULL	NULL	NULL	5	100.00	NULL

1 row in set, 1 warning (0.02 sec)

#### 5、删除 P2，查看分区信息

##### ①删除 P2 分区前所有数据

```
mysql> SELECT * FROM STUDENT;
```

ID	NAME	SEX	ENTER_DATE
1	张三	1	1989-01-25 00:00:00
2	John	1	1991-01-25 00:00:00
3	Tom	1	1992-01-25 00:00:00
4	Tack	1	1993-01-25 00:00:00
5	Smile	1	1994-01-25 00:00:00
6	Dse	1	1997-01-25 00:00:00
7	李四	1	2001-01-25 00:00:00
8	王五	1	2011-01-25 00:00:00
9	赵六	1	2020-01-25 00:00:00
10	田七	1	2019-01-25 00:00:00

10 rows in set (0.00 sec)

##### ②执行删除，并查看

```
mysql> ALTER TABLE STUDENT DROP PARTITION P2;
Query OK, 0 rows affected (0.81 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> SHOW CREATE TABLE STUDENT;
+-----+
| Table | Create Table
+-----+
| STUDENT | CREATE TABLE `student` (
  ID int(11) DEFAULT NULL,
  NAME varchar(20) DEFAULT NULL,
  SEX int(1) DEFAULT '1',
  ENTER_DATE datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
/*150100 PARTITION BY RANGE (YEAR(ENTER_DATE))
(PARTITION P1 VALUES LESS THAN (1990) ENGINE = InnoDB,
PARTITION P3 VALUES LESS THAN (2010) ENGINE = InnoDB,
PARTITION P4 VALUES LESS THAN MAXVALUE ENGINE = InnoDB) */ |
+-----+
1 row in set (0.00 sec)
```

### ③查看删除后数据

```
mysql> SELECT * FROM STUDENT;
+----+-----+-----+-----+
| ID | NAME | SEX | ENTER_DATE |
+----+-----+-----+-----+
| 1  | 张三 | 1   | 1989-01-25 00:00:00 |
| 7  | 李四 | 1   | 2001-01-25 00:00:00 |
| 8  | 王五 | 1   | 2011-01-25 00:00:00 |
| 9  | 赵六 | 1   | 2020-01-25 00:00:00 |
| 10 | 田七 | 1   | 2019-01-25 00:00:00 |
+----+-----+-----+-----+
5 rows in set (0.00 sec)
```

## 6、合并分区 p3、p4

### (1) 合并前查看 P2、P3、P4 分区

```
SELECT * FROM STUDENT PARTITION (P2);
SELECT * FROM STUDENT PARTITION (P3);
SELECT * FROM STUDENT PARTITION (P4);
```

```
mysql> SELECT * FROM STUDENT PARTITION(P2);
ERROR 1735 (HY000): Unknown partition 'P2' in table 'STUDENT'
mysql> SELECT * FROM STUDENT PARTITION(P3);
+----+-----+-----+-----+
| ID | NAME | SEX | ENTER_DATE |
+----+-----+-----+-----+
| 7  | 李四 | 1   | 2001-01-25 00:00:00 |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM STUDENT PARTITION(P4);
+----+-----+-----+-----+
| ID | NAME | SEX | ENTER_DATE |
+----+-----+-----+-----+
| 8  | 王五 | 1   | 2011-01-25 00:00:00 |
| 9  | 赵六 | 1   | 2020-01-25 00:00:00 |
| 10 | 田七 | 1   | 2019-01-25 00:00:00 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

合并前查看 P2、P3、P4 分区

### (2) 合并 P3、P4 分区、查看表分区

```
SELECT * FROM STUDENT PARTITION (P2);
SELECT * FROM STUDENT PARTITION (P3);
SELECT * FROM STUDENT PARTITION (P4);
```

```
mysql> ALTER TABLE STUDENT REORGANIZE PARTITION P3,P4 INTO(PARTITION P2 VALUES LESS THAN (MAXVALUE));
Query OK, 0 rows affected (1.65 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> SHOW CREATE TABLE STUDENT;
+-----+
| Table | Create Table
+-----+
| STUDENT | CREATE TABLE `student` (
  `ID` int(11) DEFAULT NULL,
  `NAME` varchar(20) DEFAULT NULL,
  `SEX` int(1) DEFAULT '1',
  `ENTER_DATE` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
/*!50100 PARTITION BY RANGE (YEAR(ENTER_DATE))
(PARTITION P1 VALUES LESS THAN (1990) ENGINE = InnoDB,
PARTITION P2 VALUES LESS THAN MAXVALUE ENGINE = InnoDB) */ |
+-----+
1 row in set (0.00 sec)
```

合并 P3、P4 分区、查看表分区

### (3) 合并后查看 P3、P2 分区、验证分区后结果







```
mysql> SELECT * FROM STUDENT PARTITION(P3);
ERROR 1735 (HY000): Unknown partition 'P3' in table 'STUDENT'
mysql> SELECT * FROM STUDENT PARTITION(P2);
+----+-----+-----+-----+
| ID | NAME | SEX | ENTER_DATE |
+----+-----+-----+-----+
| 7  | 李四 | 1   | 2001-01-25 00:00:00 |
| 8  | 王五 | 1   | 2011-01-25 00:00:00 |
| 9  | 赵六 | 1   | 2020-01-25 00:00:00 |
| 10 | 田七 | 1   | 2019-01-25 00:00:00 |
+----+-----+-----+-----+
4 rows in set (0.00 sec)
```

合并后查看 P3、P2 分区、验证分区后结果

## 7、数据压缩测试（两种压缩比）

### (1) 数据表准备

先将表 ORDERST8（数据量 8 万左右）复制 5 份。分别为 ORDERST8\_1, ORDERST8\_2, ORDERST8\_4, ORDERST8\_8, ORDERST8\_16; 查看其大小

	orderst8.ibd	2020/3/20 22:28	IBD 文件	15,360 KB
	orderst8_1.ibd	2020/4/18 22:40	IBD 文件	15,360 KB
	orderst8_2.ibd	2020/4/18 22:40	IBD 文件	15,360 KB
	orderst8_4.ibd	2020/4/18 22:40	IBD 文件	15,360 KB
	orderst8_8.ibd	2020/4/18 22:40	IBD 文件	15,360 KB
	orderst8_16.ibd	2020/4/18 22:40	IBD 文件	15,360 KB

压缩前 5 份数据表大小信息

### (2) 数据表压缩

①将 ORDERST8\_1 按照 KEY\_BLOCK\_SIZE = 1;

```
ALTER TABLE ORDERST8_1 KEY_BLOCK_SIZE = 1;
```

```
mysql> ALTER TABLE ORDERST8_1 KEY_BLOCK_SIZE = 1;
Query OK, 0 rows affected (27.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

压缩耗时 27.02 秒

②将 ORDERST8\_2 按照 KEY\_BLOCK\_SIZE = 2;

```
ALTER TABLE ORDERST8_2 KEY_BLOCK_SIZE = 2;
```

```
mysql> ALTER TABLE ORDERST8_2 KEY_BLOCK_SIZE = 2;
Query OK, 0 rows affected (12.46 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

压缩耗时 12.46 秒

③将 ORDERST8\_4 按照 KEY\_BLOCK\_SIZE = 4;

```
ALTER TABLE ORDERST8_4 KEY_BLOCK_SIZE = 4;
```

```
mysql> ALTER TABLE ORDERST8_4 KEY_BLOCK_SIZE = 4;
Query OK, 0 rows affected (6.83 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

压缩耗时 6.83 秒

④将 ORDERST8\_8 按照 KEY\_BLOCK\_SIZE = 8;

```
ALTER TABLE ORDERST8_8 KEY_BLOCK_SIZE = 8;
```

```
mysql> ALTER TABLE ORDERST8_8 KEY_BLOCK_SIZE = 8;
Query OK, 0 rows affected (4.62 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

压缩耗时 4.62 秒






⑤将 ORDERST8\_16 按照 KEY\_BLOCK\_SIZE = 16;

```
ALTER TABLE ORDERST8_16 KEY_BLOCK_SIZE = 16;
```

```
mysql> ALTER TABLE ORDERST8_16 KEY_BLOCK_SIZE = 16;
Query OK, 0 rows affected (6.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

压缩耗时 6.00 秒

(3) 压缩后数据文件大小

	orderst8_1.ibd	2020/4/18 22:50	IBD 文件	11,264 KB
	orderst8_2.ibd	2020/4/18 22:50	IBD 文件	8,192 KB
	orderst8_4.ibd	2020/4/18 22:51	IBD 文件	6,144 KB
	orderst8_8.ibd	2020/4/18 22:51	IBD 文件	7,168 KB
	orderst8_16.ibd	2020/4/18 22:51	IBD 文件	14,336 KB

压缩后 5 份数据表大小信息

通过计算，不同 KEY\_BLOCK\_SIZE 的压缩率如下：

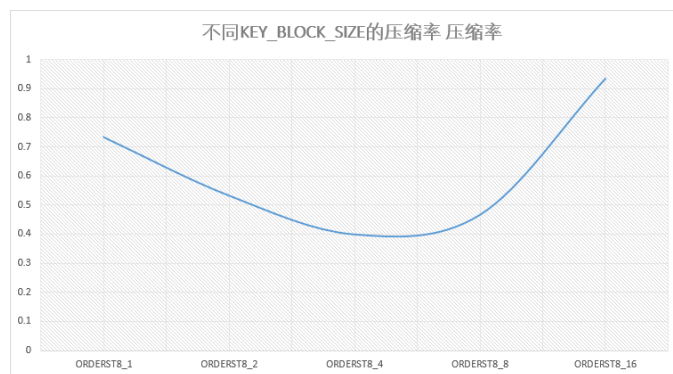
ORDERST8\_1  $11264 \div 15360 \approx 0.733 = 73.3\%$

ORDERST8\_2  $8192 \div 15360 \approx 0.533 = 53.3\%$

ORDERST8\_4  $6144 \div 15360 = 0.4 = 40.0\%$

ORDERST8\_8  $7168 \div 15360 \approx 0.467 = 46.7\%$

ORDERST8\_16  $14336 \div 15360 \approx 0.933 = 93.3\%$



不同 KEY\_BLOCK\_SIZE 的压缩率曲线图

### 三、实验总结（介绍分工、每个同学的工作）

通过分区实验，可以看出，我们进行分区时候分区字段必须包含在主键内，否则会提示 `A PRIMARY KEY must include all columns in the table's partitioning function`,也就是说分区字段必须在主键里边，其次分区之后我们将一个分区删除之后，所在分区的数据也会从所有数据中删除，实验 5、删除 P2，查看分区信息，分区合并中，合并后的分区所有数据都在一个分区里边。压缩数据实验中，通过设置不同 KEY\_BLOCK\_SIZE 的值大小，其压缩时间和效率不成正比，不是说值越大压缩越好，也不是值越小压缩越小，通过不同值得压缩率计算，可以看出当值为 4 的时候压缩率最低。压缩数据的好处在于数据压缩能够让数据库变得更小，从而减少磁盘的 I/O, 还能以很小的成本（耗费较多的 CPU 资源）提高系统吞吐量。