MySQL

## 1 MySQL分区和分表

#### 1.1 分区与分表介绍

**分表：**大表分解。建很多实体子表，将数据按规则存放。查询时直接定位查询哪一个子表。表之间是独的可以放在同一磁盘，也可是不同的机器上。

--优点：提高查询性能。缺点：查询结果集跨表查询复杂。

**分区：**区是将数据分段划分在多个位置存放，可以是同一块磁盘也可以在不同的机器。分区后，表面上还是一张表，但数据散列到多个位置了。app读写的时候操作的还是大表名字，db自动去组织分区的数据。

--优点：提高查询性能，查询感知是一个大表。缺点：由于分区数有限，不便于扩展。

**分表与分区异同：**

1. 都是为了解决mysql大表查询慢的问题，用于提高查询性能。高并发下性能优秀。
2. 分表和分区不矛盾，可以相互配合的，对于那些大访问量，并且表数据比较多的表，我们可以采取分表和分区结合的方式（如果merge这种分表方式，不能和分区配合的话，可以用其他的分表试），访问量不大，但是表数据很多的表，我们可以采取分区的方式等。
3. 分表技术是比较麻烦的，需要手动去创建子表，app服务端读写时候需要计算子表名。采用merge好一些，但也要创建子表和配置子表间的union关系。
4. 表分区相对于分表，操作方便，不需要创建子表。
5. 相对而言，分表理论上可以无限扩展（按日期建）。分区扩展有限，由于分区有限，你也不能在每个分区放太多数据导致性能问题。

#### 1.2 分区

<https://www.jianshu.com/p/2084216e81ce>

<https://www.cnblogs.com/javaIOException/p/7524945.html>

分区底层原理

1常量、随机或者依赖时区的表达式不能作为分区函数。如timestamp 不能用来分区。

2 分区字段。同时属于一个或以上的唯一性索引的键；否则就需要没有主键。

**选用场景适合主键不是必要的时候。**

##### 1.2.1 Range：把连续区间按范围划分

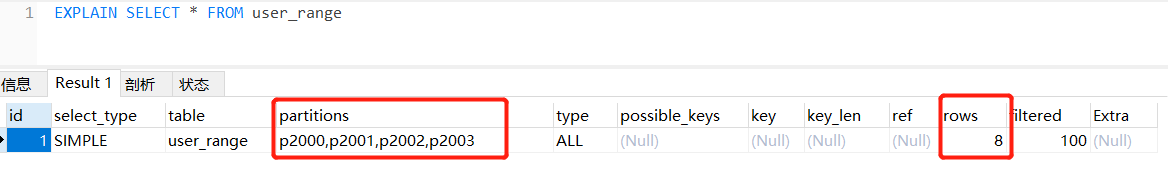
CREATE TABLE `USER\_RANGE` (  
 `id` bigint (19) NOT NULL COMMENT 'id',  
 `name` VARCHAR (100) NOT NULL COMMENT '姓名',  
 `sex` INT ( 1 ) NOT NULL DEFAULT '0' COMMENT '性别,0:男,1:女',  
 `remarks` VARCHAR ( 300 ) DEFAULT NULL COMMENT '描述',  
 `create\_user` INT ( 11 ) NOT NULL COMMENT '创建人id',  
 `create\_time` DATETIME NOT NULL COMMENT '创建时间',  
 `update\_user` INT ( 11 ) NOT NULL COMMENT '修改人id',  
 `update\_time` DATETIME NOT NULL COMMENT '修改时间'  
 ) PARTITION BY RANGE ( YEAR (create\_time) ) (  
 PARTITION p2000  
 VALUES  
 less than ( 2000 ),  
 PARTITION p2001  
 VALUES  
 less than ( 2001 ),  
 PARTITION p2002  
 VALUES  
 less than ( 2003 ),  
 PARTITION p2003  
 VALUES  
 less than MAXVALUE   
);

-- 测试数据

INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (2, 'cs', 0, 'cs', 1, '1999-06-17 22:29:58', 1, '1999-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (3, 'cs', 0, 'cs', 1, '2000-01-01 00:00:00', 1, '2000-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (4, 'cs', 0, 'cs', 1, '2000-06-17 22:29:58', 1, '2000-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (5, 'cs', 0, 'cs', 1, '2001-06-17 22:29:58', 1, '0000-00-00 00:00:00');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (6, 'cs', 0, 'cs', 1, '2002-06-17 22:29:58', 1, '2002-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (7, 'cs', 0, 'cs', 1, '2020-06-17 22:29:58', 1, '2020-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (8, 'cs', 0, 'cs', 1, '2003-06-17 22:29:58', 1, '2013-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (9, 'cs', 0, 'cs', 1, '2014-06-17 22:29:58', 1, '2014-06-17 22:30:08');  
 INSERT INTO `blog`.`user\_range`(`id`, `name`, `sex`, `remarks`, `create\_user`, `create\_time`, `update\_user`, `update\_time`) VALUES (10, 'cs', 0, 'cs', 1, '2005-06-17 22:29:58', 1, '2005-06-17 22:30:08');  
  
}

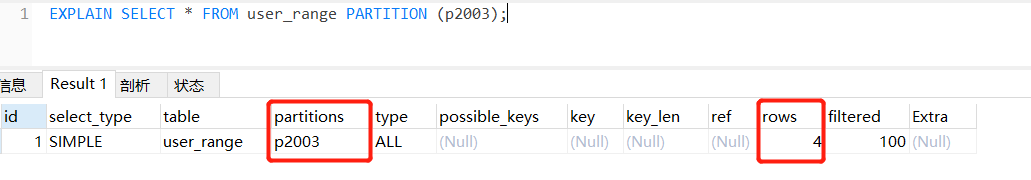
p2000 小于2000;p2001[2000，2001)；p2002[2002，2003)；p2003大于等于2003

(1)查询所有，有数据八条

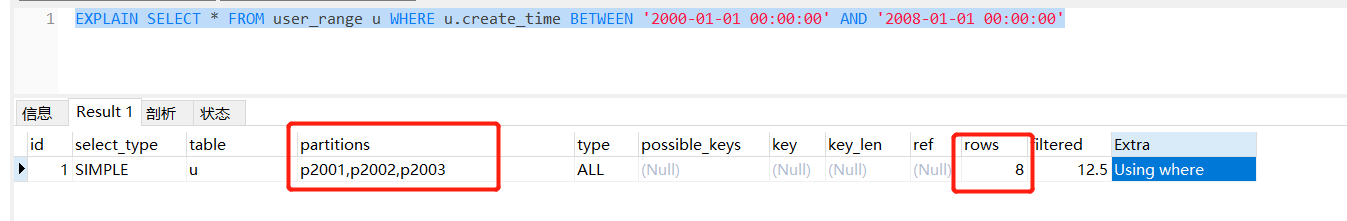


1. 查询p2003所有，有数据4条。指定区。不做全表扫描。

查询具体p2003区的所有数据：SELECT \* FROM RANGEPARTITION (p2003);



1. 不指定区，还是扫描全表。



##### 1.2.2 List：把离散值分成集合

按集合划分，适合有固定取值列的表.  
例：

create table user\_list(

a int(11),

b int(11))

partition by list(b)(

partition p0 values in (1,3,5,7,9),

partition p1 values in (2,4,6,8,0));

##### 1.2.3 Hash：随机分配，分区数固定 例：

create table user(

a int(11),

b datetime )

partition by hash(YEAR(b))

partitions 4;

##### 1.2.4 Key：类似Hash

区别是只支持1列或多列,且mysql提供自身的Hash函数  
例：

create table user(

a int(11),

b datetime )

partition by key(b)

partitions 4;

#### 1.3 分表

**水平分表**：表记录多。每张表结构一致。只是数据按日期或者取模存入对应表。

**垂直分表**：表的记录并不多，但是字段却很长，表占用空间很大，检索表的时候需要执行大量的IO，严重降低了性能。这时需要把大的字段拆分到另一个表，并且该表与原表是一对一的关系。