# 概述

分区功能并不是存储引擎层完成的。因此不只有InnoDB存储引擎支持分区，常见的存储引擎MyISAM、NDB等都支持分区。但是并不是所有的存储引擎都支持，如CSV、FEDORATED、MERGE等就不支持分区。分区的过程是将一个表或者索引分解为多个更小、更可管理的部分。就访问数据库的应用而言，从逻辑上讲，只有一个表或者索引，但是在物理上这个表或者索引可能由十个物理分区组成。

分区，是利用MySQL的一种特性。数据分区是一种物理数据库的设计技术，**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缩减响应时间**。

分区就是把一张表的数据分成N多个区块，**这些区块可以在同一个磁盘上，也可以在不同的磁盘上**。

**分区并不是生成新的数据表，而是将表的数据均衡分摊到不同的硬盘，系统或是不同服务器存储介质中，实际上还是一张表（对用户来说，逻辑上，分区后的表仍然是一个逻辑表）**。另外，分区可以做到将表的数据均衡到不同的地方，提高数据检索的效率，降低数据库的频繁I/O压力值。

## 特点

**优点：**

1. 相对于单个文件系统或是硬盘，分区可以存储更多的数据；
2. 数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可；
3. 精确定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率；
4. 可跨多个分区磁盘查询，来提高查询的吞吐量；
5. 在涉及聚合函数查询时，可以很容易进行数据的合并。

**缺点：**

分区利用不好会存在性能问题。

## 子分区

子分区（subpartitioning）是在分区的基础上再进行分区，有时也称这种分区为**复合分区（composite partitioning）**。

# 分类

MySQL数据库支持的分区类型为水平分区，并不支持垂直分区。

**水平分区**

这种形式分区是对表的行进行分区，通过这样的方式不同分组里面的物理列分割的数据集得以组合，从而进行个体分割（单分区）或集体分割（1个或多个分区）。所有在表中定义的列在每个数据集中都能得到，所以表的特性依然得以保持。

**垂直分区**

这种分区方式一般来说是通过对表的垂直划分来减少目标表的宽度，使某些特定的列被划分到特定的分区，每个分区都包含了其中的列所对应的行。

注：MySQL是不支持这种垂直分区的。

# 类型

## RANGE分区

## LIST分区

## HASH分区

## KEY分区

# 操作

1、分区，使用MySQL自身的语法。分区的SQL语句如下：

CREATE TABLE sales (order\_date DATETIME NOT NULL)

ENGINE=InnoDB

PARTION BY RANGE(YEAR(order\_date))

(

PARTION p\_2010 VALUES LESS THAN (2010),

PARTION p\_2011 VALUES LESS THAN (2011),

PARTION p\_2012 VALUES LESS THAN (2012),

PARTION p\_catchall VALUES LESS THAN MAXVALUE

);

执行之后，在mysql/data/test目录下发现两个文件sales.frm（对象结构定义文件）和用于存储表对象的结构sales.par（应该是分区语句创造的，和分区有关）。

再执行普通的建表SQL语句：

CREATE TABLE sales2 (order\_date DATETIME NOT NULL)

ENGINE=InnoDB;

执行之后，在mysql/data/test目录下生成了文件sales2.frm

2、在information\_schema表中执行SQL语句

SELECT \* FROM PARTITIONS WHERE PARTITION\_NAME IS NOT NULL\G

看到了与上面的建分区表SQL对应的分区信息。

3、往分区表中插入数据

INSERT INTO sales (order\_date) VALUES (2007-01-01);

INSERT INTO sales (order\_date) VALUES (2008-01-01);

INSERT INTO sales (order\_date) VALUES (2009-01-01);

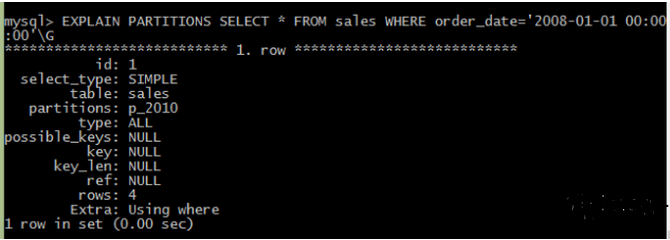
INSERT INTO sales (order\_date) VALUES (2011-01-01);

INSERT INTO sales (order\_date) VALUES (2011-01-01);

4、检测数据是否按分区存储了，执行下面的SQL语句

EXPLAIN PARTITIONS SELECT \* FROM sales WHERE order\_date = '2008-01-01 00:00:00'\G

查询结果如图：



# 性能

## 分区中NULL值

MySQL数据库允许对NULL值做分区，但是处理的方法与其他数据库可能完全不同。MySQL数据库分区总是视NULL值小于任何的一个非NULL值。

对于RANGE分区，如果向分区列插入NULL值，则MySQL数据库会将该值插入最左边的分区。

在LIST分区下要使用NULL值，则必须显式地指出哪个分区中放入NULL值，否则会报错。

HASH和KEY分区对于NULL的处理方式和RANGE分区、LIST分区不一样，任何分区函数都会将含有NULL值的记录返回为0.

## 分区和性能

数据库的应用分为两类：一类是OLTP（在线事务处理），如Blog、电子商务、网络游戏等；另一类是OLAP（在线分析处理），如数据仓库、数据集市。

对于OLAP的应用，分区的确是可以很好地提高查询的性能，因为OLAP应用大多数查询需要频繁地扫描一张很大的表。如果按照时间戳进行分区，则只需要扫描相应的分区即可。

**然而对于OLTP的应用，分区应该非常小心。在这种应用中，通常不可能获取一张大表中10%的数据，大部分都是通过索引返回几条记录即可。而根据B+树索引的原理，对于一张大表，一般的B+树需要2~3次的磁盘I/O。因此B+树可以很快地完成操作，不需要分区的帮助，并且设计不好的分区会带来严重的性能问题**。

## 表和分区间交换数据

要使用ALTER TABLE…EXCHANGE PARTITION语句，必须满足下面的条件：

1. 要交换的表需和分区表有着相同的表结构，但是表不能含有分区；
2. 在非分区表中的数据必须在交换的分区定义内；
3. 被交换的表中不能含有外键，或者其他的表含有对该表的外键引用；
4. 除了需要ALTER、INSERT和CREATE权限外，还需要DROP的权限外，

两个小的细节需要注意：

1. 使用该语句时，不会触发交换表和被交换表上的触发器；
2. AUTO\_INCREMENT列将被重置。