# 概述

分区功能并不是存储引擎层完成的。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。

MySQL数据库支持的分区类型为水平分区，并不支持垂直分区。

# 类型

## RANGE分区

## LIST分区

## HASH分区

## KEY分区

# 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。