# 概述

用来存储元数据信息的表。比如表的描述、字段、对象、对象之间的关系。InnoDB中大多以系统表的形式保存。主要包括sys\_tables，sys\_columns，sys\_indexs，sys\_fields：

sys\_table存储表的信息，包括表面，ID，表空间号

sys\_columns存储表中列的信息，包括列名，列ID，列的序号，列的类型，长度等信息

sys\_index存储表的索引信息，包括索引名，索引对应的表空间，表ID，索引ID,索引类型

sys\_fields存储索引中定义的索引列

# 原理

## 字典表的加载

InnoDB启动的时候，如果是新建数据库，则需要初始化库，索引需要创建字典管理的B+树信息。因为InnoDB中的系统表的结构，个数等都是固定的，所以在初始化库的时候只需要创建这几个表的存储B+树即可。同时把将这几个B+树的根页号存储在一个固定位置，就不需要将这几个表自身的信息存储在系统表中了。对于一个B树，只要找到其根页面，就可以找检索其数据了。

对于数据字典表根页面位置的存储方式，InnoDB用了一个专门的页面（0号表空间0号文件的7号页面）来管理数据字典信息。这个页面用来存储4个系统表的五个根页面号(有5个索引)。

普通用户表的加载过程，当用户访问一个表时，系统首先会从表对象缓冲池中查找这个表SHARE对象，如果找到，则直接从其实例化表对象空间链表中拿一个空闲的实例化的表对象使用，如果没有一个可用的实例化对象，则需要重新打开(实例化这个表)，在实例化这个表的时候，需要找到这个表的字典信息，包括这个表本身，列信息及索引信息等，这些信息很多都是从SHARE对象处获得。如果没有SHARE对象，则需要从系统表中构造SHARE对象。

## Rowid管理

在InnoDB中，用户表中的记录不一定都会有一个Rowid列，Rowid只有在一个表没有定义主键时，才会分配。而Rowid的管理分配，并不是一个表独享一个ID空间，而是全局的，使用表都共享这个ID号。

Rowid的分配并不会直接修改页面，只要这个值为256的倍数的时候才会写入一次。那么如果插入200次，这些值还没有被写入，这是系统重新启动，ID号岂不是重复使用，因为数据库启动的时候会调用函数做一个工作，就是将上次写入的Rowid值向上对齐256后在加上256，这样就不会有问题了。

# 恢复方法