# InnoDB 数据字典

**innodb数据字典**

**概念**

      数据字典，它是用来存储元数据信息的表，属于系统表。从数据字典中可以得知目前数据库有哪些对象，对象之间的关系等。

**系统表结构**

      Innodb有四个最基本的系统表结构，用来存储用户定义的表、列、索引及索引列等信息，这些表分别为SYS\_TABLES、SYS\_COLUMNS、SYS\_INDEXES、SYS\_FIELDS。下面分别对每个表做简单介绍：

**SYS\_TABLES**

用来存储所有以InnoDB为存储引擎的表，每条记录对应已经定义的一个表。

NAME：表示一个表的表名。

ID：表示这个表的ID号（8个字节）

N\_COLS：表示这个表的列的个数，建表指定的列数（4个字节）。

TYPE：表示这个表的存储类型，包括记录的格式、压缩等信息（4个字节）

SPACE：表示这个表所在的表空间ID号（4字节）。这个表对应的主键列为NAME，同时还有一个在ID号上的唯一索引。

**SYS\_COLUMNS**

用来存储InnoDB中定义的所有表中所有列的信息，每一列对应这个表中的一条记录。此表的主键列是（TABLE\_ID，POS）

TABLE\_ID：表示这个列所属的表的ID号（8字节）。

POS：表示这个列在表中是第几列（4字节）。

NAME：表示这个列的列名。

MTYPE：表示这个列的主数据类型（4字节）。

PRTYPE：表示这个列的一些精确数据类型，它是一个组合值，包括NULL标志、是否有符号数的标志、是否是二进制字符串的标志及表示这个列是真的VARCHAR（数据存储用两个字节）（4字节）。

LEN：表示这个列的数据长度，但不包括VARCHAR类型，因为这个类型在记录里面存储了数据长度（4字节）。

**SYS\_INDEXES**

用来存储InnoDB中所有表的索引信息，每条记录对应一个索引。这个表的主键列为（TABLE\_ID，ID）。

TABLE\_ID：表示这个索引所属的表的ID号（8字节）。

ID：表示这个索引的索引ID号（8字节）。

NAME：表示这个索引的索引名。

N\_FILEDS：表过这个索引包含的列个数（4字节）。

TYPE：表示这个索引的类型，包括聚簇、唯一索引、DICT\_UNIVERSAL、DICT\_IBUF（插入缓冲区B+树）（4字节）。

SPACE：表示这个 索引数据所在的表空间ID号（4字节）。

PAGE\_NO：表示这个索引对应的B+树的根页面（4字节）。

**SYS\_FIELDS**

用来存储所有索引中定义的索引列，每一条记录对应一个索引列。这个表的主键是（INDEX\_ID，POS）

INDEX\_ID：这个列所在的索引（8字节）。

POS：这个列在某个索引中是第几个索引列（4字节）。

COL\_NAME：这个索引列的列名。

**字典表加载**

在InnoDB启动的时候，需要初使化库，所以首先需要做的就是创建字典管理的B+树等信息。因为InnoDB中系统表的结构、个数等都是固定的，不会有人修改，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，同时将这几个表的根叶号存储在一个固定的位置，就不需要将这几个表自身的信息存储在系统表中了。关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）来管理字典信息，这个页面用来存储上面提到的到这四个表的五个根页面号，以及下个表ID值（全局变量，创建新表时的ID号从这里分配，每分配一个，这个ID号要加1） ，下一个索引ID值，下一个表空间ID值，Rowid这4个值。

**普通用户表加载**

      当用户访问一个表时，系统首先会从表对象缓存池中查找这个表的 SHARE对象，如果找到了，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出来使用，如果没有一个可用的实例化对象，则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。具体查找过程如下：

首先找到SYS\_TABLES，找到之后，构造一个查询键值，因为是通过名字查询的，同时SYS\_TABLES有一个按照名字排序的索引，所以可以直接按照名字构造查询键值。

然后从B+树中找到对应记录，取出ID，N\_COLS，TYPE，SPACE这些基本信息。

最后根据这些信息创建一个表的内存对象。

到这里这个表的自身对象就加载完成了，接着开始加载所有列信息。加载列与加载表原理类似，在此不详述，但有一点需要注意，在InnoDB中一个表的列包括两部分，一部分是用户创建表时指定的列，另一部分是系统列，包括Rowid、TRXID及ROLLPTR，这三个列的含义分别是：

        Rowid：表示记录的行号；

        TRXID：表示这条记录最后一次被修改的事务号，主要用于事务的多版本（MVCC）管理；

        ROLLPTR：也是用来实现多版本的，如果一条记录被某一个用户修改之后，另一个用户在这条记录不可见时，查询这条记录要找到其原来的值，那么ROLLPTR指定的就是其原来值的位置，这个位置其实就是在修改时写下的回滚记录位置。

接下来是加载这个表的索引信息，加载索引是从SYS\_INDEXES中查询的，原理与上面的SYS\_COLUMNS一样。这个表的关键字是TABLE\_ID及索引ID，所以具有相同TABLE\_ID的所有索引记录都是按照索引ID排序的。每一条记录对应一个索引，需要加载ID、名字、N\_FIELDS、TYPE、PAGENO、SPACE等基本信息。

对于索引加载还需要加载它对应的关键字信息，这些信息存储在SYS\_FIELDS系统表中，这个表的关键字是INDEX\_ID、POS，所以一个索引的所有关键字列都是按照POS从小到大排序进行加载的。PS：

        如果是非唯一索引，则需要将表中已经加载的Rowid列以这个索引第一个关键字列的身份加载到这个索引中；如果是唯一索引则只需要加载自定义的列即可。

        对于聚簇索引，因为这个索引存储了表中所有的列，所以后面还需要加载除关键字之外的所有列，这些列都是按照建表时的顺序加载的，而对于二级索引，这些列是不需要加载的。