1. InnoDB页
2. 页的概述

InnoDB是一个将表中的数据存储到磁盘上的存储引擎，真正处理数据的过程发生在内存中，需要把磁盘中的数据加载到内存中，如果处理写入或修改请求，还需要把内存中的内容刷新到磁盘上。InnoDB将数据划分为若干项，以页作为磁盘和内存之间交互的基本单位，页大小一般为16KB，即16384字节，可在初始化MySQL数据目录时指定，之后无法修改。

1. InnoDB行格式
2. 概述

以记录为单位向表中插入数据，这些记录在磁盘上的存放形式被称为行格式。InnoDB上有4中行格式，COMPACT, REDUNDANT, DYNAMIC, COMPRESSED。可在创建或修改表的语句中指定记录所使用的行格式：

CREATE TABLE 表名（列的信息） ROW\_FORMAT=行格式;

ALTER TABLE 表名 ROW\_FORMAT=行格式;

1. COMPACT行格式

一条完整的记录分为记录的额外信息和记录的真实数据两部分。

1. 记录的额外信息

这部分信息是为了更好地管理记录而额外添加的信息，分为三部分，变长字段长度列表，NULL值列表，记录头信息。

1. 变长字段长度列表

MySQL支持一些变长的数据类型，如VARCHAR, VARBINARY, TEXT等等，变长字段能够存储多少字节的数据是不固定的，因此在存储真实数据的同时还要存储这些数据占用的字节数。所有的变长字段占用的字节数按照列的顺序的逆序存放到变长字段长度列表中。如果变长字段所采用的字符集 最多需要W个字节来表示一个字符，而该变长字段在初始化时设定最多存储M个字符，那么该变长字段最多占用的字节数就是W x M，如果 W x M 《 255，则使用1个字节来表示真实数据占用的字节数；如果 W x M > 255，且真实字节数L <= 127，则使用1个字节，否则使用两个字节。

注意，变长字段长度列表中只存储值为非NULL的列的内容长度!

1. NULL值列表

为了节省NULL值的存储空间，将一条记录中值为NULL的列统一管理起来，存储到NULL值列表中。

首先统计表中允许存储NULL的列有哪些，主键列和使用NOT NULL修饰的列都是不可以存储NULL值的。如果表中没有允许存储NULL的列，那么NULL值列表就不会存在了，否则将每个允许存储NULL的列对应一个二进制位，二进制位按照列的顺序逆序排列。当某个二进制位为1时，表示该列的值为NULL，否则不为NULL。MySQL规定NULL值列表必须使用整数个字节的位表示，如果使用的二进制位个数不是整数个字节，则在字节的高位补齐0。

1. 记录头信息

记录头信息由固定的5个字节组成，用于描述记录的一些属性，具体的概念请到书中查看。

1. 记录的真实数据

记录的真实数据除了自己定义的列的数据外，还会为每个记录默认添加一些隐藏列。InnoDB表中优先使用用户自定义的主键作为主键，如果没有定义主键，则选取一个不允许存储NULL值的UNIQUE键作为主键，如果连这个键都没有，则InnoDB会默认添加一个row\_id的隐藏列作为主键。trx\_id为隐藏的事务ID列，roll\_pointer为隐藏的回滚指针列。

定长的列即时只存储很小的值，也会占用固定的字节数，NULL值直接存储在前面的NULL值列表处，在记录的真实数据处就不会冗余存储了，节省了存储空间。

1. CHAR(M)的存储格式

CHAR(M) 表示定长的字段，如果该字段采用的是定长编码字符集，如ascii，则它的长度不会存储到变长字段长度列表中；若该字段采用的是变长编码的字符集，如utf8, gbk，则它的长度也会存储到变长字段长度列表中。

采用变长编码字符集的CHAR(M) 类型的列要求至少占用M个字节，而VARCHAR(M)没有这个要求，比如utf8字符集，对于CHAR(10)，该列存储的数据占用的字节长度为10 – 30 字节，因为最多有10个字符，而一个字符占用1-3个字节。这种设计可以在更新记录时直接更新，而不会由于存储空间不够而重新分配空间，导致原有的记录空间称为碎片。

1. REDUNDANT行格式

比较古老的一种行格式。

1. 字段长度偏移列表

将该条记录中所有列（包括隐藏列）的长度信息都按照逆序存储到字段长度偏移列表中，且使用两个相邻偏移量的差值来计算各个列值的长度。

1. 偏移量的字节数

每个列对应的偏移量使用1个字节还是2个字节来存储，是通过记录的真实数据占用的字节数大于127还是小于127判断的。只要整条记录的真实数据占用的字节长度大于127，即时某个列的值占用的存储空间不大于127字节，也要使用两个字节来表示该列的偏移量。这是一种过时的方法，浪费空间。

1. NULL值的处理

REDUNDANT行格式中并没有NULL值列表，因此将列对应的偏移量值的第一个比特作为是否是NULL的依据，该比特位称为NULL比特位。

如果存储NULL值的字段是定长类型的，则NULL值将占用记录的真实数据部分，并把该字段对应的数据用0x00字节填充；如果是变长数据类型，则不占用记录的真实数据空间。

1. 溢出列

在COMPACT和REDUNDANT行格式中，记录都会被分配到某个页中存储，如果记录的大小超过了页的大小，则记录的真实数据处只会存储该列的一部分数据，而把剩余的数据分散在几个其他的页中，然后在记录的真实数据处使用20个字节存储指向这些页的地址。

如果某一列的数据非常多，则在该记录的真实数据处只会存储该列前768字节的数据以及一个指向其他页的地址，其他页称为溢出页，这个列称为溢出列。

1. 产生溢出页的临界点

一个列在存储了多少字节后才会变为溢出列呢？

MySQL规定一个页中至少存放两行记录，每个页中除了存放我们的信息，还有额外的信息，加起来需要132字节的空间，每个记录需要的额外信息时27字节，假设一个列的真实数据占用的字节数为n，如果该列不会发生溢出现象，则需要满足以下不等式：

132 + 2 x (27 + n) < 16384

即 n < 8099

1. DYNAMIC行格式

溢出时，不会存储该列的前768个字节，而是将所有字节存储到溢出页中。

1. COMPRESSED行格式

使用压缩算法岁页面进行压缩，节省空间。

只有REDUNDANT行格式是非紧凑的，是过时的，其他的都是较新的行格式，是紧凑的。