1、第一范式：1NF强调的是列的原子性，即列的属性不可再拆分。比如下表包含「姓名」、「性别」、「电话」字段。但是在实际应用场景中，「电话」是可以被拆分为「家庭电话」以及「工作电话」的。所以我们说，「电话」这一列不符合原子性原则，也就是说这个表不符合1NF。

正确的做法应该是将「电话」拆分为「家庭电话」和「工作电话」。

[](https://gss0.baidu.com/-Po3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/574e9258d109b3de0f39d726c1bf6c81810a4cfd.jpg)

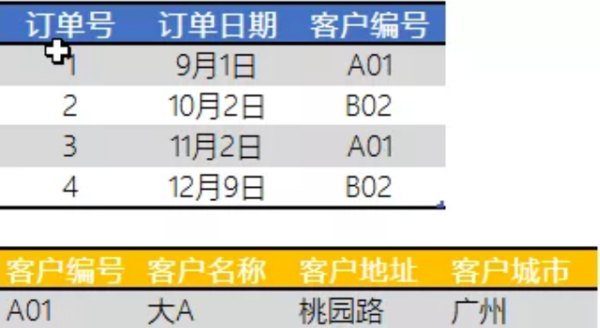
2、第二范式 2NF：2NF建立在1NF的基础上，即首先得满足1NF，然后再来谈2NF。2NF包含两部分内容：

⑴表必须有一个“主键”（主键可以是一个列，也可以是多个列的集合）；

⑵非主键的列必须完全依赖于主键，而不能仅仅是依赖主键的其中某个列。

[](https://gss0.baidu.com/-Po3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/77c6a7efce1b9d1686b9a92ffedeb48f8c546405.jpg)

3、第三范式 3NF：3NF建立在2NF的基础上，即首先得满足2NF，然后再来谈3NF。非主键列必须直接依赖主键，不能存在传递依赖。换句话说，就是假设A是主键，那么B，C都是直接依赖A，而不能是C依赖B，B依赖A这种传递依赖方式。

[](https://gss0.baidu.com/-fo3dSag_xI4khGko9WTAnF6hhy/zhidao/pic/item/3bf33a87e950352a206c7c4f5e43fbf2b2118b6c.jpg)

## 扩展资料

**区别：**

1、第二范式与第三范式的本质区别：在于有没有分出两张表。

2、第二范式是说一张表中包含了多种不同实体的属性，那么必须要分成多张表，第三范式是要求已经分好了多张表的话，一张表中只能有另一张标的ID，而不能有其他任何信息，（其他任何信息，一律用主键在另一张表中查询）。

3、必须先满足第一范式才能满足第二范式，必须同时满足第一第二范式才能满足第三范式。

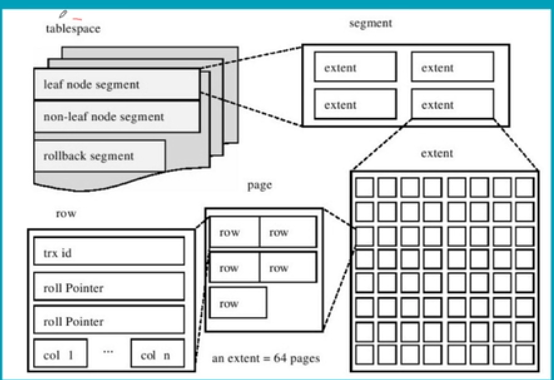
4.在满足第三范式的情况下，对于一个关系模式，每一个决定因素都包含码，则属于BCNF,所有的主属性和非主属性都是完全依赖的，

逐层进行投影分解

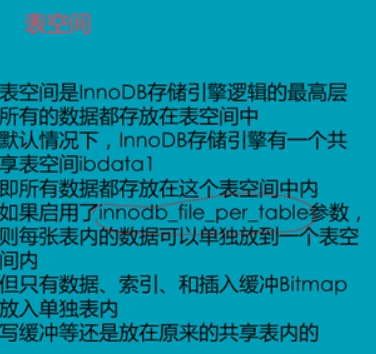
**数据库索引：**



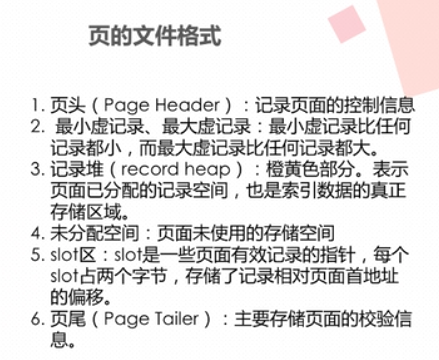
InnoDB存储引擎是不一样的，他是将数据存储在**表空间中**的，就是说我所有的表，所有的数据库都会放到一个数据库文件里面，如果不更改他的设置的话，他的表空间文件会越来越大，最终可能会出现操作系统的单文件最大限制，如果我们想让一个表对应一个文件的话，需要设置innodb\_file\_per\_table参数，它的意思就是把每张表放到一个表空间内，这样当表越来越大，防止出现一些问题，

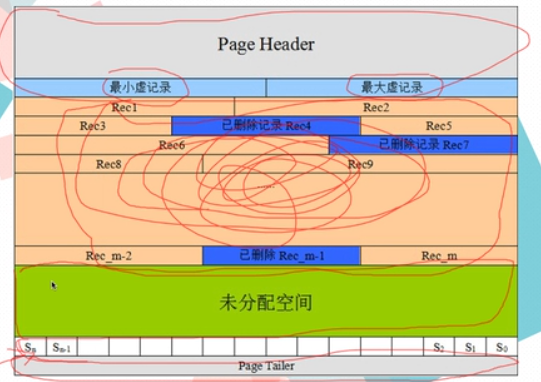


表空间-》段(这些段有的用于做缓冲，有的用于回撤等用途）-》区（这些区中有相应的职能，事务区，隔离区等，）-》页（一个区里面有64页，每一页默认是16k的大小）-》行（默认占2k）.

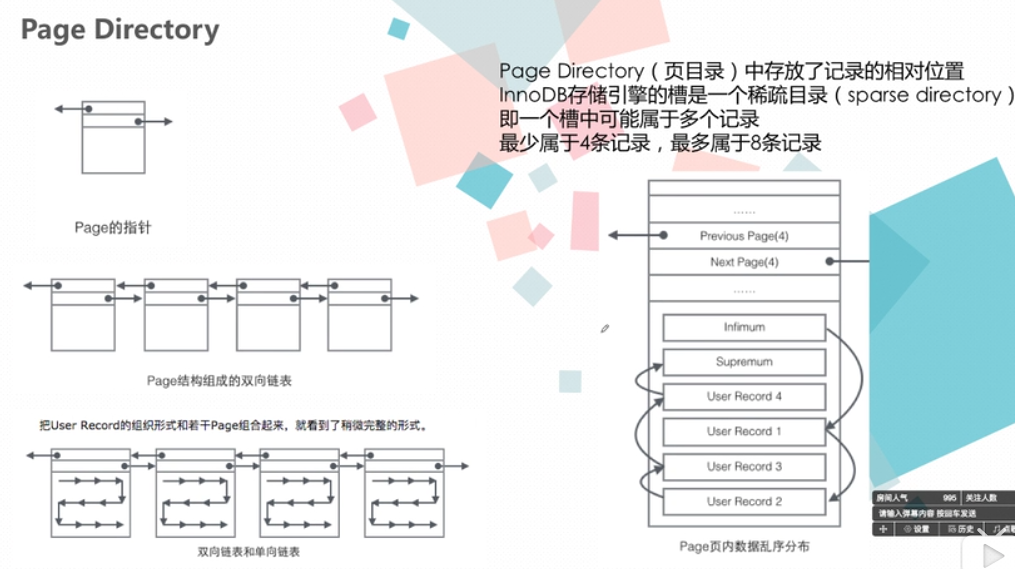


**页（因为磁盘读取数据是非常慢的，它需要寻道时间和磁头移动的时间，可以利用数据预读原理（如果一个地址的数据被用到了，那么它附近的数据很有可能很快也会被用到，因此每次是把一块区域的数据读取到内存中，且这一块就是16K的大小）InnoDB存储的一页也是16K就是为了对于一块的大小）**





㡟：



一页中有一部分数用来存储数据的