

MySql笔记-innodb无特殊说明，基于5.7版本

[零 参考资料 3](#_Toc43156337)

[一 表的逻辑存储 3](#_Toc43156338)

[1 概述 3](#_Toc43156339)

[2 表空间 4](#_Toc43156340)

[2.1 系统表空间(System tablespace) 4](#_Toc43156341)

[2.2 file-per-table表空间 4](#_Toc43156342)

[2.3 一般表空间(General Tablespaces) 5](#_Toc43156343)

[2.4 Undo Tablespaces 6](#_Toc43156344)

[2.5 临时表空间(The Temporary Tablespace) 6](#_Toc43156345)

[3 段(Segment) 6](#_Toc43156346)

[4 区(Extent) 6](#_Toc43156347)

[5 页(page) 6](#_Toc43156348)

[5.1 概述 6](#_Toc43156349)

[5.2 File Header 7](#_Toc43156350)

[5.3 Page Header 7](#_Toc43156351)

[5.4 Infimun+Supremum Records 7](#_Toc43156352)

[5.5 User Records 8](#_Toc43156353)

[5.6 Free Space 8](#_Toc43156354)

[5.7 Page Directory 8](#_Toc43156355)

[5.8 File Trailer 8](#_Toc43156356)

[5.9 页文件分析 8](#_Toc43156357)

[6 行(row) 8](#_Toc43156358)

[6.1 概述 8](#_Toc43156359)

[二 索引 9](#_Toc43156360)

[1 概述 9](#_Toc43156361)

[2 B树 9](#_Toc43156362)

[3 B+树 9](#_Toc43156363)

[4 聚簇索引(Clustered index) 9](#_Toc43156364)

[5 次级索引(Secondary index) 9](#_Toc43156365)

[6 哈希索引 9](#_Toc43156366)

[7 覆盖索引(Covering Index) 9](#_Toc43156367)

[8 全文检索(Full-Text Search)和倒排索引(Inverted Index) 9](#_Toc43156368)

[9 空间索引()和R树(R-tree) 9](#_Toc43156369)

[三 主存存储结构(InnoDB In-Memory Structures) 9](#_Toc43156370)

[1 LUR(the least recently used) 算法 9](#_Toc43156371)

[2 缓冲池(Buffer Pool) 9](#_Toc43156372)

[2.1 概述 9](#_Toc43156373)

[2.2 缓冲池配置 10](#_Toc43156374)

[2.3 控缓冲池监控参数 10](#_Toc43156375)

[3 变更缓冲(Change Buffer) 10](#_Toc43156376)

[3.1 概述 10](#_Toc43156377)

[4 日志缓冲(Log Buffer) 11](#_Toc43156378)

[四 锁(Lock) 11](#_Toc43156379)

[五 日志(Log) 11](#_Toc43156380)

[六 事务(Transaction) 11](#_Toc43156381)

[七 其他 11](#_Toc43156382)

[1 、配置 11](#_Toc43156383)

[2 、优化 11](#_Toc43156384)

[3 、限制 11](#_Toc43156385)

# 零 参考资料

《Mysql技术内幕 InnoDB存储引擎》

《高性能Mysql 第三版》

《高可用Mysql》

<https://dev.mysql.com/doc/refman/5.7/en/>

<https://dev.mysql.com/doc/refman/5.7/en/innodb-on-disk-structures.html>



# 表的逻辑存储

## 概述

* + - 1. 索引组织表(index organized table)：表都是根据主键顺序存放
      2. 如果没有显式指定主键，则：
         1. 判断表中是否由唯一非空索引，有则指定其为主键，如果有多个，则根据定义索引的顺序，指定第一个为主键
         2. 如果b.1不符合，则自建一个6字节大小的指针键
      3. 单列主键情况下，可以用\_rowid查看主键情况。
      4. 所有数据都被逻辑地放在表空间中(tablespace),表空间又由段(segment),区(Extent),页(page,或者称为块(block))
      5. 逻辑存储结构



## 表空间

### 系统表空间(System tablespace)

<https://dev.mysql.com/doc/refman/5.7/en/innodb-system-tablespace.html>

* + - 1. 可以有多个表空间数据文件，默认情况下，为一个名为ibdata1的在数据目录下的初始大小为10M的单文件，可以通过innodb\_data\_file\_path参数配置存储了Innodb的数据字典(the InnoDB data dictionary)，双写缓冲(the doublewrite buffer)，改变缓冲(the change buffer)，undo日志(undo logs)以及一些创建在此空间的表的
      2. 当innodb\_file\_per\_table参数禁用且未指定表空间时，默认在此空间创建表
      3. 当操作引起表空间数据减少时，不会收缩表空间大小，只是将释放出来的空间标记为可用

### file-per-table表空间

https://dev.mysql.com/doc/refman/5.7/en/innodb-file-per-table-tablespaces.html

* + - 1. 当innodb\_file\_per\_table启用且未指定表空间时，会为每个创建表动作在数据目录下独立创建一个对应的文件名为${表名}.idb的表空间文件
      2. 优势
         1. 在truncating 或 dropping操作后，可以将存储空间还给系统
         2. 表复制ALTER TABLE操作，不会增加存储空间占用
         3. TRUNCATE TABLE操作性能相对更好
         4. 此表空间文件可以创建在不同的存储空间中
         5. 可以从其他Mysql直接迁移指定的表
         6. 可以使用Barracuda文件格式
         7. 当数据异常时可以更快速修复以及提高恢复概率
         8. 在不打断其他表使用的情况下，可以迅速恢复/备份表
         9. 可以通过监视表空间文件大小，监视文件系统上的表大小
         10. 当设置innodb\_flush\_method值为O\_DIRECT时，可以提高性能
         11. 每个表都是64TB大小极限(Tables in a shared tablespace are limited in size by the 64TB tablespace size limit. By comparison, each file-per-table tablespace has a 64TB size limit, which provides plenty of room for individual tables to grow in size.)
      3. 劣势
         1. 此表空间只能被指定表使用，导致资源浪费
         2. 会导致有很多fsync操作
         3. mysqld必须保存很多文件句柄，导致影响性能
         4. 每个表需要更多文件描述符
         5. 如果没管理好，会产生很多碎片，影响dropTable和表扫描操作。反之会提高性能
         6. 因为缓冲池扫描会用到一个内部锁，所以在此模式下，会延迟其他操作执行
         7. 配置innodb\_autoextend\_increment只应用于共享表空间文件，这个空间固定每次扩展4MB

### 一般表空间(General Tablespaces)

https://dev.mysql.com/doc/refman/5.7/en/general-tablespaces.html

* + - 1. 此空间是个通过CREATE TABLESPACE创建，可共享的表空间，文件后缀为.isl
      2. 存储表空间元数据消耗内存比file-per-table tablespaces少
      3. 表空间文件可以存在相对目录或独立的数据目录
      4. 支持Antelop和Barracuda文件格式，因此支持所有行格式和相关功能
      5. 参数innodb\_file\_format 和 innodb\_file\_per\_table不影响此表空间
      6. 在执行Create table时，可以指定tablespace
      7. 在执行Alter table时，可以通过Tablespace，对表做表空间迁移
      8. 由于物理页不同，压缩和非压缩的表不能在同一个表空间共存
      9. 表空间如果包含压缩表，FILE\_BLOCK\_SIZE必须指定具体值，同时innodb\_page\_size不能为32k或64k,而且KEY\_BLOCK\_SIZE必须等于FILE\_BLOCK\_SIZE/1024
      10. 当表空间中最后一张表被删除时，不会自动删除表空间文件，必须执行drop tablespace
      11. 此表空间不属于特定database，drop database操作能删掉此空间的表，但删不掉此空间
      12. 和系统空间一样，修改表数据不会改变空间大小，但会在表空间内部释放资源
      13. 限制
          1. 不能将其他已存在的表空间改成General tablespace
          2. 不支持创建临时General tablespace
          3. Generaltable不支持临时表
          4. 此空间中的表不支持DISCARD TABLESPACE 和IMPORT TABLESPACE操作
          5. MySQL 5.7.24以后会不支持此表空间中的表分区
          6. 不支持同主机主从复制
      14. 表空间创建语句

1. **CREATE** TABLESPACE tablespace\_name
2. **ADD** DATAFILE 'file\_name'
3. [FILE\_BLOCK\_SIZE = value]
4. [ENGINE [=] engine\_name]

### Undo Tablespaces

https://dev.mysql.com/doc/refman/5.7/en/innodb-undo-tablespaces.html

### 临时表空间(The Temporary Tablespace)

https://dev.mysql.com/doc/refman/5.7/en/innodb-temporary-tablespace.html

## 段(Segment)

* + - 1. 在Innodb中，段的管理由引擎自动完成
      2. 这里的段跟回滚段不同
      3. 如果表空间是目录，段就是目录中文件
      4. 段每次扩容都一次1个区，每次释放也是一次一个区
      5. file-per-table 表空间，数据都在一个段中，对应的索引在各自的段中
      6. system tablespace表空间则包含各种不同的段
      7. 在Innodb中每个索引分配两个段，一个用于Btree的非叶子节点，一个用于叶子节点
      8. 个别区会分配在多个段中

## 区(Extent)

* + - 1. 由连续页组成的空间，
         1. 当页大小为4KB、8KB、16KB时，区大小固定为1MB；
         2. 当页大小为32KB，对应的区大小为2MB，
         3. 页大小为64KB,对应的区大小为4MB
      2. 默认情况下，一个区包含64个大小为16KB的页

## 页(page)

### 概述

* + - 1. 结构：File Header，Page Header，Infimun和Supremum Records，User Records，Free Space，Page Directory，File Trailer



### File Header

### Page Header

### Infimun+Supremum Records



### User Records

### Free Space

### Page Directory

### File Trailer

### 页文件分析

## 行(row)

### 概述

* + - 1. 行记录数据格式有四种

| **格式** | **支持字符压缩** | **Enhanced Variable-Length Column Storage** | **Large Index Key Prefix Support** | **支持压缩** | **表空间类型** | **文件格式** |
| --- | --- | --- | --- | --- | --- | --- |
| REDUNDANT | No | No | No | No | system  file-per-table, general | Antelope or Barracuda |
| COMPACT | Yes | No | No | No | system  file-per-table  general | Antelope or Barracuda |
| DYNAMIC | Yes | Yes | Yes | No | system  file-per-table general | Barracuda |
| COMPRESSED | Yes | Yes | Yes | Yes | file-per-table general | Barracuda |

# 索引

## 概述

## B树

## B+树

## 聚簇索引(Clustered index)

## 次级索引(Secondary index)

## 哈希索引

## 覆盖索引(Covering Index)

## 全文检索(Full-Text Search)和倒排索引(Inverted Index)

## 空间索引()和R树(R-tree)

# 主存存储结构(InnoDB In-Memory Structures)

## LUR(the least recently used) 算法

## 缓冲池(Buffer Pool)

<https://dev.mysql.com/doc/refman/5.7/en/innodb-buffer-pool.html>

### 概述

* + - 1. 实现算法是LUR算法的一个变种，默认操作流程：
         1. 将缓冲池分为两部分：

年轻子链表(new/young sublist)：占5/8缓存池容量，最新访问的页

年老子链表(old sublist)：占3/8缓存池容量，非最近访问的页，淘汰页时候选页

* + - * 1. 设置一个边界midpoint：在年轻子链表的表尾链接年老子链的表头位置
        2. 当因为用户查询操作或作为innodb自动执行的预读操作的一部分，需要读取一个新页进入缓冲池时，将新页插入到midpoint位置
        3. 当缓冲池中的一个页被存取

如果是用户操作引起，在第一次存取时，立即将此页放入到年轻子链表的头部

如果是因为innodb自动执行的预读操作引起，在第一次存取时，不会将此页放入到年轻子链表的头部，可能直到此页被淘汰，都不会被放入到年轻子链表的头部

* + - * 1. 随着数据库操作，缓冲池中的页会被逐步移向缓冲池尾部(年老子链表尾部)

每个页随着其他页被移到年轻子链表的头部而逐渐老化

年老子链中的页，同时也因为新页插入到midpoint而加速老化

最后，缓冲池中始终未被使用的页因为到达了年老子链表的尾部而被淘汰



### 缓冲池配置

### 控缓冲池监控参数

* + - 1. 执行SHOW ENGINE INNODB STATUS 命令查看

## 变更缓冲(Change Buffer)

### 概述



## 日志缓冲(Log Buffer)

# 锁(Lock)

# 日志(Log)

# 事务(Transaction)

# 其他

## 、配置

## 、优化

## 、限制