## 一、MySQL代码目录结构

* BUILD: 内含在各个平台、各种编译器下进行编译的脚本。如compile-pentium-debug表示在pentium架构上进行调试编译的脚本。
* client: 客户端工具，如mysql,mysqladmin之类。
* cmd-line-utils: readline,libedit工具。
* config: 给aclocal使用的配置文件。
* dbug: 提供一些调试用的宏定义。
* Docs: MySQL在不同平台下的参考手册
* extra: 提供innochecksum,resolveip等额外的小工具。
* include: 包含的头文件
* libmysql: 库文件，生产libmysqlclient.so。
* libmysql\_r: 线程安全的库文件，生成libmysqlclient\_r.so。
* libmysqld: 嵌入式MySQL Server库.
* libservices: 5.5.0中新加的目录，实现了打印功能。
* man: 适合man命令查看的帮助文件。
* mysql-test: mysqld的测试工具套件。
* mysys: 为实现跨平台，MySQL自己实现了一套常用的数据结构和算法，如string, hash等。还包含一些底层函数的跨平台封装,一般以my\_开头。
* netware: 在netware平台上进行编译时需要的工具和库。
* plugin: MySQL 5.1开始支持一个插件式API接口,不需要重启mysqld即可动态载入插件,FullText就是一个例子。
* pstack: GNU异步栈追踪工具。
* regex: 正则表达式实现(来自多伦多大学Henry Spencer大牛的源码)。
* scripts: 提供脚本工具，如mysql\_install\_db/mysqld\_safe等。
* server-tools:包含instance\_manager子目录,负责实例的本地和远程管理。
* sql: MySQL Server主要代码，将会生成mysqld文件。
* sql-bench: 一些基准测试代码代码,主要是Perl程序(虽然后缀是sh)。
* sql-common: 存放部分服务器端和客户端都会用到的代码,有些地方的同名文件是这里lin过去的。
* storage: 存储引擎所在目录。
* strings: string库,包含很多字符串处理的函数。
* support-files: my.cnf示例配置文件及编译所需的一些工具。
* tests: 测试文件所在目录。
* unittest: 单元测试文件。
* vio: 虚拟io系统，是对network io的封装,把不同的协议封装成统一的IO函数。
* win: 在windows平台编译所需的文件和一些说明。
* zlib: zlib算法库(GNU)

## 二、innodb目录结构

* btr: B+树的实现
* buf: 缓冲池的实现,包括LRU算法,Flush刷新算法等
* dict: InnoDB内存数据字典的实现
* dyn: InnoDB动态数组的实现
* fil: InnoDB文件数据结构以及对于文件的一些操作
* fsp: 对InnoDB物理文件的管理,如页/区/段等(即File Space)
* ha: 哈希算法的实现
* handler: 继承与MySQL的handler,实现handler API与Server交互
* ibuf: 插入缓冲(Insert Buffer)的实现
* include: InnoDB所有头文件都放在这个目录,是查找结构定义的最佳地点
* lock: InnoDB的锁实现及三种锁算法实现
* log:日志缓冲(Log Buffer)和重做日志组(Redo Log)的实现
* mem: 辅助缓冲池(Additional Memory Pool)的实现,用来申请一些内部数据结构的内存
* mtr: 事务的底层实现(日志,缓冲)
* os: 封装一些对于操作系统的操作
* page: 页的实现,研究InnoDB文件结构,这个目录至关重要
* pars: 重载部分MySQL的SQL Parser(有待商榷)
* que: Query graph,基本上没啥用
* read: 读取游标的实现
* rem: 行管理操作(比较操作,打印等)
* row: 对于各种类型行数据操作的实现
* srv: InnoDB后台线程,启动服务,Master Thread,SQL队列等
* sync: InnoDB互斥变量(Mutex)的实现,基本同步机制
* thr: InnoDB封装的可移植线程库
* trx: 事务的实现
* usr: Session管理
* ut: 各种通用小工具

## 三、核心类库

* THD: 线程类
* Item: Item类(查询条目,函数,WHERE,ORDER,GROUP,ON子句等)
* TABLE: 表描述符
* TABEL\_LIST: JOIN操作描述符
* Field: 列数据类型及属性定义
* LEX: 语法树
* Protocol: 通讯协议
* NET: 网络描述符
* handler: 存储引擎接口

## 四、核心函数库

* 内存操作:
* init\_alloc\_root: 内存池初始化,生成内存池根(MEM\_ROOT)
* alloc\_root: 申请内存池内存,从mem\_root制定的内存池申请内存块
* free\_root: 释放内存池,通过MyFlags指定哪种内存可以被释放
* 文件操作:
* my\_open: 打开一个文件
* my\_close: 关闭一个文件
* my\_b\_flush\_io\_cache: 讲数据从内存缓冲写到物理磁盘
* end\_io\_cache: 释放一个IO\_CACHE对象
* 哈希操作:
* \_hash\_init: 初始化HASH描述符
* hash\_search: 搜索哈希表,调用hash\_first
* hash\_first: 返回哈希表中找到的第一个行指针,否则返回0
* 字符串操作:
* strappend:填充字符串
* strmov: 移动字符串到新地址

## 五、核心算法

* Bitmaps
* bitmap\_init/bimap\_free: 创建与释放一个位图 (8\*n个位为单位)
* bitmap\_set\_bit/bitmap\_fast\_test\_and\_set: 设置位图的一个位
* bitmap\_clear\_all/bitmap\_set\_all: 清空或全部设置一个位图
* bitmap\_cmp: 对两个位图的特定位比较
* Join Buffer
* 如果存在条件过滤,则第一次过滤完的记录将放入Join Buffer,避免第二次再判断
* Sort Buffer
* 算法一: 将排序字段和主键放入Sort Buffer排序,按照结果用主键取出数据返回
* 算法二: 将整行数据放入Sort Buffer,排序完成后直接从Sort Buffer返回数据