# MySQL重点笔记

**初级：**

1. 特点

* 关系型数据库
* 跨平台
* 支持多种编程语言
* 基于磁盘存储，数据以文件形式存放在数据库目录/var/lib/mysql下

1. 启动及连接

服务端启动sudo service mysql start|stop|restart|status

客户端连接 mysql -hIP地址 -u用户名 -p密码

1. 基本命令

* 库管理

查已有库 show databases;

创建加库 create database 库名 charset utf8 | character set utf8;

查所在库 select database();

使用库 use 库名

查看库中所有表 show tables;

删除库 drop database 库名;

* 表管理

创建表 create table 表名(字段名 字段类型 xxx,…)charset=utf8;

查看表创建语句 show create table 表名；

查看表结构 desc 表名；

删除表 drop table 表名；

* 表记录管理

增 insert into 表名(字段名) values(字段对应值),(…);

删 delete from 表名 where 条件;

查 select 字段名 from 表名 where 条件；

改 update 字段名 set 字段名=值 where 条件;

* 表记录管理

增 alter table 表名 add 字段名 字段类型 first|after 字段名;

删 alter table 表名 drop 字段名;

改 alter table 表名 modify 字段名 字段类型;

表重命名 alter table 表名 rename 新表名;

1. 数据类型

* 整型

int—4 bytes smallint—2 bytes bigint—8 bytes tiny—1 byte

* 字符型

char()--定长,长度不足空格填充 varchar(n)—变长，实际长度相对储字节数多相应字节用于保存当前字段实际长度

* 枚举类型

Enum 枚举 set 集合

* 日期时间类型

date datetime timestamp time year

1. 日期时间函数

* NOW( )：返回当前日期时间 CURDATE( )：=NOW( )
* YEAR(时间):返回日期时间表达式的年份 DATE(时间)：返回日期部分 TIME(时间)：返回时间部分

1. 运算符

* 数值比较: >、<、=、>=、<=、!=
* 逻辑比较: and 、or
* 范围内比较: between 值1 and 值2 、in( ) 、not in ( )
* 模糊比较: Where 字段名 like 表达式(%\_)----%代替多个字符 \_代替一个字符
* NULL判断: is NULL 、is not NULL

1. 查询

* 排序： order by 字段名 ASC/DESC----ASC升序 DESC降序
* 分页: limit m,n----limit(m-1)\*n,n 从第m页显示n条记录 m当前页 n每页记录条数

**高级：**

1. 普通查询:

* 优先级

3、select … 聚合函数 from 表名

1、where …

2、group by … 分组

4、having … 对分组后的数据进行筛选

5、order by …

6、limit…

* 聚合函数

avg(字段名) 该字段的平均值

max(字段名) 该字段的最大值

min(字段名 该字段的最小值

sum(字段名 该字段所有记录的和

count(字段名 统计该字段记录的个数

* 排重

distinct

* 查询表记录时做的数学运算

运算符：+ 、- 、\* 、/ 、% 、\*\*

1. 索引

* MySQL数据结构

Btree 每个节点存储多个索引和数据

B+tree 根节点只存储索引，叶子结点存储数据，且数据有序相连，范围查询效果Good！

* 索引优缺点

优点： 加快查询速度

缺点： 占用物理存储空间(/var/lib/mysql)、当对表中数据更新时，索引需动态维护，降低数据维护速度

* 运行时间检测

show variables like ‘%pro%’; set profiling=1; 1开启 0关闭

* 查看执行时间

show profiles；

* 索引分类

普通索引(MUL) 唯一索引(UNI)

* 使用规则

可以设置多个字段

普通索引： 字段值无约束，KEY标志位MUL

唯一索引： 字段值不允许重复，但可以为NULL，KEY标志为UNI

使用字段： 经常查询字段、where条件判断字段、order by排序字段

* 创建索引：

1. 建表时：create table 表名(

字段名 数据类型,

字段名 数据类型，

index(字段名),

unique(字段名)

);

1. 已由表中创建

create [unique] index 索引名 on 表名 (字段名）；

* 查看索引

desc 表名；🡪KEY标志为：MUL、UNI

show index from 表名\G；

* 删除索引

drop index 索引名 on 表名；

* 主键（PRI）and自增长（auto\_increment）

1. 使用规则

一个表只能有一个主键字段

所带约束：不能重复，且不能为NULL

KEY标志（primary）：PRI

通常设置记录编号字段id，能唯一锁定一条记录

1. 创建

创建表添加主键 ---- 主键字段后接 primary key

已有表添加主键 ---- alter table 表名 add primary key (id);

创建表添加自增 ---- 自增字段后接auto\_increment

已有表添加自增 ---- alter table 表名 modify id int auto\_increment；

已有表重新指定起始值 ---- alter table 表名 auto\_increment=20000；

1. 删除

删除自增长属性（modify）---- alter table 表名 modify id int；

删除主键索引 ---- alter table 表名 drop primary key；

1. 外键

* 定义

让当前表字段的值在另一个表的范围内选择

* 语法

foreign key（参考字段）references 主表名（被参考字段名） on delete 级联动作 on update 级联动作

* 使用规则

主表、从表字段数据类型要一致

主表被参考字段：KEY的一种，一般为主键

* 删除外键

alter table 表名 drop foreign key 外建名；

外建名：show create table 表名；

* 级联动作

cascade 数据级联删除、更新(参考字段)

restrict 从表有相关联记录，不允许主表操作有关联数据

set null 主表删除、更新，从表相关联记录字段值为null

* 已有表添加外键

alter table 表名 add foreign key（参考字段）references 主表名（被参考字段名） on delete 级联动作 on update 级联动作；

1. 嵌套查询

select … from 表名 where 条件 (select …);

1. 多表查询

* 笛卡尔积

select 字段列表 from 表名列表

* 多表查询

select 字段列表 from 表名列表 where 条件;

* 连接查询

1. 内连接（结果同多表查询，显示匹配到的记录）

select 字段名 from 表名 inner join 表2 on 条件 inner join 表3 on 条件;

1. 左外连接（以左表为主显示查询结果）

select 字段名 from 表名 left join 表2 on 条件 left join 表3 on 条件;

1. 右外连接（以右表为主显示查询结果）

select 字段名 from 表名 right join 表2 on 条件 right join 表3 on 条件;

1. 数据导入导出

* 导入

1. 导入sql格式文件: source 文件名.sql----文件中存放的是sql语句
2. 把普通文件导入数据库: load data infile ‘文件名’

into table 表名

fields terminated by ‘分隔符’

lInes terminated by ‘\n’;

* 导出

1. 数据库文件导出至文件中: select … from 表名

into outfile ‘文件名’

fields terminated by ‘分隔符’

lines terminated by ‘\n’;

1. 导出的内容由SQL查询语句决定

执行导出命令时路径必须指定在对应的数据库目录下

1. 表的复制

* 注意

表能根据实际需求复制数据

复制表时不会把KEY属性复制过来

* 语法

create table 表名 select查询命令

1. 锁（自动加锁和释放锁）

* 目的

解决客户端并发访问的冲突问题

* 锁类型分类

读锁（共享锁） select 加读锁后别人不能更改表记录，但可以进行查询

写锁（互斥锁，排它锁） 加锁后别人不能查，不能改

* 锁力度分类

表级锁 myisam

行级锁 innodb

注：myisam和innodb都是引擎

1. 存储引擎

* 定义

处理表的处理器

* 基本操作

1. 查看所有存储引擎 show engines;
2. 查看已有表的存储引擎 show create table 表名;
3. 创建表指定存储引擎 create table 表名(…)engine=引擎;
4. 已有表指定存储引擎 alter table 表名 engines=引擎;

* 常用存储引擎的特点

1. innoDB

支持行级锁

支持外键、事务、事务回滚

表字段和索引同存储在一个文件中： 表名.frm—表结构

表名.ibd—表记录及索引文件

1. MyISAM

支持表级锁

表字段和索引分开存储： 表名.frm—表结构

表名.MYI—索引文件（my\_index）

表名.MYD—表记录（my\_data）

1. MEMORY

表记录存储在内存中，效率高

服务或主机重启，表记录清楚

* 如何选择存储引擎

MyISAM:

查询快，如下原因：

1. 压缩前缀索引 - 加快了查询效率
2. 查询缓存中 由于压缩前缀索引技术，可容纳更多的查询结果

其他：

1. count(id) ；提供了一个计数器

innodb:

更新快，如下原因:

1. MVCC多版本控制 导致 查询有耗损
2. 由于本身为行级锁，插入时效率较高（行级锁减少了插入、更新时锁的交换时间【按需执行锁的交换】；类比myisam为表锁，表锁插入、更新时均需执行锁的交换）

其他：

事务

执行查操作多的表用 MyISAM

执行写操作多的表用 InnoDB

临时表 ： MEMORY

最终选型： 依据压测结果【请按照实际表结构及索引结构进行有效压测】

1. MySQL用户账户管理

* 开启MySQL远程连接

更改配置文件，重启服务！  
1、sudo su  
2、cd /etc/mysql/mysql.conf.d  
3、cp mysqld.cnf mysqld.cnf.bak  
4、vi mysqld.cnf #找到44行左右,加 # 注释bind-address = 127.0.0.1  
5、保存退出 vi使用 : 按a ->编辑文件 ->ESC ->shift+: ->wq  
6、service mysql restart

* 添加授权用户

1、用root用户登录mysql

mysql -uroot -p123456

2、授权

grant 权限列表 on 库.表 to "用户名"@"%" identified by "密码" with grant option;

3、刷新权限

flush privileges;

* 权限列表

all privileges 、select 、insert ... ...

库.表 ： \*.\* 代表所有库的所有表

1. 事务和事务回滚

* 事务定义

一件事从开始发生到结束的过程

* 作用

确保数据的一致性、准确性、有效性

* 事务操作

1. 开启事务

begin；| start transaction；

1. 开始执行事务中的1条或n条SQL命令
2. 终止事务

commit； 事务中SQL命令都执行成功，提交到数据库，结束！

rollback; 有SQL命令执行失败，回滚到初始状态，结束！

* 事务四大特性

1. 原子性

一个失误必须视为一个不可分割的最小工作单元，整个事务中所有操作要么全部提交成功，要么全部失败回滚，对于一个事务来说，不可能只执行其中的一部分操作

1. 一致性

数据库总是从一个一致性的状态转换到另一个一致性的状态—中间态对于其他事物是不可见的

1. 隔离性

一个事务所做的修改在最终提交前，对于其实事务是不可见的

1. 持久性

一旦事务提交，则其所做的修改就会永久保存到数据库中，此时即使系统崩溃，修改的数据也不会丢失

* 注意

1、事务只针对于表记录（增删改）有效，对于库和表的操作无效

2、事务一旦提交结束，对数据库中的数据更改是永久性的

1. E-R模型（Entry-Relationship）

* 定义

E-R模型即 实体-关系 数据模型，用于数据库的设计

用简单的图反映了显示时节中存在的实物或数据以及他们之间的关系

* 实体、属性、关系

1. 实体
2. 描述客观事务的概念
3. 表示方法：矩形框
4. 实例：一个人、一本书、一杯咖啡、一个学生…
5. 属性
6. 实体具有的某种特性

2、表示方法 ：椭圆形

3、示例

学生属性 ：学号、姓名、年龄、性别、专业 ...

感受属性 ：悲伤、喜悦、刺激、愤怒 ...

1. 关系

1、实体之间的联系  
2、一对一关联(1:1) ：老公对老婆  
   A中的一个实体,B中只能有一个实体与其发生关联  
   B中的一个实体,A中只能有一个实体与其发生关联  
3、一对多关联(1:n) ：父亲对孩子  
   A中的一个实体,B中有多个实体与其发生关联  
   B中的一个实体,A中只能有一个与其发生关联  
4、多对多关联(m:n) ：兄弟姐妹对兄弟姐妹、学生对课程  
   A中的一个实体,B中有多个实体与其发生关联  
   B中的一个实体,A中有多个实体与其发生关联

* ER图的绘制

矩形框代表实体，菱形框代表关系，椭圆形代表属性 m-n多对多 1-n一对多 1-1一对一

* 关系映射实现（重要）

1:1实现 --> 主外键关联,外键字段添加唯一索引

表t1 : id int primary key,

表t2 : t2\_id int unique,

foreign key(t2\_id) references t1(id)

1:n实现 --> 主外键关联

表t1 : id int primary key,

表t2 : t2\_id int,

foreign key(t2\_id) references t1(id)

m:n实现(借助中间表):

t1 : t1\_id

t2 : t2\_id

1. MySQL调优

* 存储引擎优化

1、读操作多：MyISAM

2、写操作多：InnoDB

* 索引优化

在 select、where、order by 常涉及到的字段建立索引

* SQL语句优化

1、单条查询最后添加 LIMIT 1，停止全表扫描

2、where子句中不使用 != ,否则放弃索引全表扫描

3、尽量避免 NULL 值判断,否则放弃索引全表扫描

优化前：select number from t1 where number is null;

优化后：select number from t1 where number=0;

# 在number列上设置默认值0,确保number列无NULL值

4、尽量避免 or 连接条件,否则放弃索引全表扫描

优化前：select id from t1 where id=10 or id=20;

优化后：select id from t1 where id=10 union all select id from t1 where id=20;--union友联

5、模糊查询尽量避免使用前置 % ,否则全表扫描

select name from t1 where name like "c%";

6、尽量避免使用 in 和 not in,否则全表扫描

优化前：select id from t1 where id in(1,2,3,4);

优化后：select id from t1 where id between 1 and 4;

7、尽量避免使用 select \* ...;用具体字段代替 \* ,不要返回用不到的任何字段