[《mysql》入门指南 - 掘金 (juejin.cn)](https://juejin.cn/post/7085523252189069343#heading-5)



### 数据库表

* 创建数据库

create database 数据库名;

* 查看所有数据库

*# 注意末尾有分号*

show databases;

* 删除数据库

drop database 数据库名;

* 选择数据库

use 数据库名;

* 查看指定数据库的所有表

*# 前提是先使用use先选择数据库*

show tables;

* 创建数据库表

# 如果`user`表不存在则创建创建user表

create table if not exists `user`( `uid` int unsigned auto\_increment, `username` varchar(128) not null, primary key ( `uid` ) )engine=innodb default charset=utf8;

* 删除数据库表

drop table 表名;

* 查看创建的数据表结构

desc 表名;

### 插入数据

* 往表中插入数据

insert into 表名 (field1, field2, ...fieldn) values (value1, value2,...valuen);

### 查询数据

* 查询表所有列数据

*# 查询user表的所有记录*

select \* from user;

* 查询表中指定列的所有数据

*# 查询user表的username列的所有数据*

select username from user;

*# 查询user表的多个列的所有数据*

select username, othercolumn1, othercolumn2 from user;

* where关键字可以用于指定查询条件

*# 查询user表中username为make的这条记录所有字段信息*

select \* from user where username='make';

* and和or可以在where后面使用多个表达式，表示与和或的逻辑

*# 选取books表中type为666且author为施耐庵的记录数据集合*

select \* from books

where type=666 and author='施耐庵';

*# 选取books表中type为666或者author为make的记录数据集合*

select \* from books where type=666 or author='make';

* 表达式操作符可以为=、!=、>、<、>=、<=。

*# 选取books表中id大于2的数据记录集合*

select \* from books where id > 2;

* limit指定查询数据，offset指定查询位置

*# 从books表中下标位置为2开始，查询2条记录的数据*

select \* from books limit 2 offset 2;

* like关键词模糊查找

*# like语法*

select field1, field2,...fieldn

from table\_name

where field1 like condition1 [and [or]] filed2 = 'somevalue'

*查询books表中所有bookname为西游开头的记录:*

select \* from books where username like '西游%';

* union连接多个select表达式

*# 查询level表中level字段小于2和大于5的记录*

select \* from level where level < 2 union select \* from level where level > 5;

*默认情况下，union得到的交集会去除重复的数据记录，也可以显示的指定是否去除*

*# distinct字段表示去除重复记录，也是默认值*

select \* from level where level < 2 union distinct select \* from level where level > 5;

*# all字段表示不去除重复记录*

select \* from level where level < 2 union all select \* from level where level > 5;

### 查询结果排序

* 升序，order by默认是升序，默认值是asc

*# 查询level表中所有记录，按field1升序排序*

select \* from level order by field1;

* 降序，order by desc指定降序排序

*# 查询level表中所有记录，按field1降序排序*

select \* from level order by field1 desc;

### 分组

group by可以将数据表按指定字段分组，然后可以对分组的数据进一步操作。

例如下面将数据表按照name分组并统计每个name有多少条数据

select name, count(\*) from level group by name;

### 多表联查

* join可以用于在多张表中查询数据，获取两个表中字段匹配关系的记录。类似于交集。

*# 查询存在于user表中且在level表中有相同level值的记录*

*# 查询返回的fields为user.uid、user.author、user.level、level.name、level.description*

select a.uid, a.author, a.level, b.name as levelname, b.description as leveldesc

from user a join level b

on a.level = b.level;

* left join 获取左表符合条件的所有记录，即使右表没有对应匹配的记录

*# 对于在b表中没有查到的记录的数据，则以null填充*

select a.uid, a.author, a.level, b.name as levelname

from user a left join level b

on a.level = b.level;

* right join 获取右表符合条件的所有记录，即使左表没有对应匹配的记录

select a.uid, a.author, a.level, b.name as levelname from user a right join level b on a.level = b.level;

### 条件表达式使用正则

除了使用like关键词进行模糊匹配，where的条件表达式也可以使用正则表达式进行查询，关键词是regexp:

*# 查询books表中bookname是西游开头的所有记录*

select \* from books where bookname regexp '^西游';

**更新数据**

* 将指定列的所有数据进行更新

update 表名 set field1=value1, field2=value2;

* where根据筛选条件进行更新

update 表名 set field1=value1, field2=value2 where 表达式;

### 删除数据

* 删除表中符合条件的记录数据

*# 从books表中删除id为6的记录*

delete from books where id=6;

* 删除表中全部数据

*# 删除user表中全部记录*

delete from user;

### 事务

事务的主要作用是保证一组数据库操作都成功执行，不然如果有一步数据操作出错了会导致不完整性。举个例子，当你删除一个用户时，同时需要用户的文章信息、登录数据等等时，要确保删除用户时的其他删除操作也必须成功。事务具有如下几个重要特征：

* 在 mysql 中只有使用了 innodb 数据库引擎的数据库或表才支持事务。
* 事务处理可以用来维护数据库的完整性，保证成批的 sql 语句要么全部执行，要么全部不执行。
* 事务用来管理 insert，update，delete 语句

使用用法主要有2步，一是使用begin;开始一个事务；二是使用rollback;可以回滚事物或者使用commit;进行事务确认。下面我们看具体的使用示例：

*# 开始事务*

begin;

*# 一些数据库操作*

*# 此时所有数据操作并未被真正写入数据库*

insert into books (bookname, type, author) values ('深入浅出vue.js', 123, '刘博文');

insert into books (bookname, type, author) values ('new book', 123, 'make');

insert into books (bookname, type, author) values ('new book2', 123, 'make');

*# 事物确认，此时所有数据才被全部写入*

commit;

### alter更新数据表

如果我们数据表创建完成之后，在开发过程中突然发现我们缺少了一些字段，或者一些数据类型不对，我们需要修改表那又该怎么办呢？

* 向数据表增加列

*# 向user表中新增newfield1列，类似为int*

alter table user add newfield1 int;

* 修改数据表字段的名称和类型

modify作用是修改字段类型，change可以同时修改名称和类型。

*# 将user表的newfield1字段的类型修改为varchar(64)*

alter table user modify newfield1 varchar(64);

*# 将user表的newfield1字段名称修改为new\_field1，类型修改为varchar(128)*

alter table user change newfield1 new\_field1 varchar(128);

alter修改类型时覆盖操作，不是增量覆盖，因此修改类型时要加上原先所有的

*# 举个例子，原先的uid字段如下*

`uid` bigint(20) not null auto\_increment comment '用户id'

*# 在使用alter修改类型时，*

*# 哪怕仅仅改一个数据类型为int，也需要把后面的not null等类型携带上*

alter table user modify uid int not null auto\_increment comment '用户id'

* drop移除字段

*# 移除user表的new\_field1字段*

alter table user drop new\_field1;

* 设置/删除字段的默认值

alter操作时如果没有指定默认值，则默认值是null，可以通过下面的例子删除/设置字段的默认值

*# 设置user表的new\_field字段的默认值为100*

*# 注意，alter是用了两次*

alter table user alter new\_field set default 100;

*# 移除user表的new\_field字段的默认值*

alter table user alter new\_field drop default;

* 修改数据表的名称

*# 我们先创建一个数据表*

create table new\_table(uid int);

*# 修改new\_table数据表的名称为new\_table2*

alter table new\_table rename to new\_table2;