**MySQL学习笔记**

**什么是数据库[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E4%BB%80%E4%B9%88%E6%98%AF%E6%95%B0%E6%8D%AE%E5%BA%93)**

严格来讲数据库是一个存放数据的容器（仓库），可以是文件或是内存空间等。当我们需要访问数据库中数据时，我们需要使用到数据库管理系统（DBMS）。我们平时说的数据库是指：数据库管理系统（DBMS）+数据库（DB）

数据库概念的正规定义：对大量数据进行管理的高效的解决方案。

常见的数据库有：

* Mysql
* SQLServer
* Oracle
* PGSQL
* DB2(IBM)

**什么是数据库管理系统[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E4%BB%80%E4%B9%88%E6%98%AF%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F)**

数据库管理系统（DBMS）是指访问操作数据库的软件。

**关系型数据库和非关系型数据库**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E5%92%8C%E9%9D%9E%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93)

关系型数据（RDBS，RELATIONAL DBS）是建立在关系模型上的数据库系统。关系模型就是指二维表格模型，因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据库系统 ，结构和实体关系。

非关系型数据库（NoSQL）: redis面向键值对, MongoDB面向文档，HBase面向列存储，NEO4J图形数据库。

**MySQL的体系架构[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "mysql%E7%9A%84%E4%BD%93%E7%B3%BB%E6%9E%B6%E6%9E%84)**

MySQL采用C/S架构。启动MySQL服务可以采用mysqld命令

*# 启动mysql服务*

mysqld start

mysqld stop

*# 链接服务器*

mysql -hlocalhost -P3306 -uroot -p

*# 退出mysql服务*

quit

上面的mysqld是启动MySQL服务的命令，MySQL还提供了很多其他的工具命令。

MySQL服务器端实用工具程序如下：  
● mysqld:SQL后台程序（即MySQL服务器进程）。该程序必须运行之后，客户端才能通过连接服务器来访问数据库。  
● mysqld\_safe：服务器启动脚本。在UNIX和NewWare中推荐使用mysqld\_safe来启动mysqld服务器。mysqld\_safe增加了一些安全性，例如，当出现错误时，重启服务器并向错误日志文件中写入运行时间信息。  
● mysql.server：服务器启动脚本。该脚本用于使用包含为特定级别的、运行启动服务器脚本的、运行目录的系统。它调用mysqld\_safe来启动MySQL服务器。  
● mysqld\_multi：服务器启动脚本，可以启动或停止系统上安装的多个服务器。  
● mysamchk：用来描述、检查、优化和维护MyISAM表的实用工具。  
● mysql.server：服务器启动脚本。在UNIX中的MySQL分发版包括mysql.server脚本。  
● mysqlbug:MySQL缺陷报告脚本。它可以用来向MySQL邮件系统发送缺陷报告。  
● mysql\_install\_db：该脚本用默认权限创建MySQL授予权表。通常只是在系统上首次安装MySQL时执行一次。

MySQL客户端实用工具程序如下：  
● myisampack：压缩MyISAM表以产生更小的只读表的一个工具。  
● mysql：交互式输入SQL语句或从文件经批处理模式执行它们的命令行工具。  
● mysqlacceess：检查访问主机名、用户名和数据库组合的权限的脚本。  
● mysqladmin：执行管理操作的客户程序，例如创建或删除数据库、重载授权表、将表刷新到硬盘上以及重新打开日志文件。Mysqladmin还可以用来检索版本、进程以及服务器的状态信息。  
● mysqlbinlog：从二进制日志读取语句的工具。在二进制日志文件中包含执行过的语句，可用来帮助系统从崩溃中恢复。  
● mysqlcheck：检查、修复、分析以及优化表的表维护客户程序。  
● mysqldump：将MySQL数据库转储到一个文件（例如SQL语句或Tab分隔符文本文件）的客户程序。  
● mysqlhotcopy：当服务器在运行时，快速备份MyISAM或ISAM表的工具。  
● mysql import：使用LOAD DATA INFILE将文本文件导入相应的客户程序。  
● mysqlshow：显示数据库、表、列以及索引相关信息的客户程序。  
● perror：显示系统或MySQL错误代码含义的工具。

**SQL语言简介[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "sql%E8%AF%AD%E8%A8%80%E7%AE%80%E4%BB%8B)**

有了数据库之后，我们需要一个语言来和数据进行交互。SQL（结构化查询语言）就是专门用来和数据库交互的的一种语言。

SQL有许多不同的类型，有3个主要的标准：ANSI（美国国家标准机构）SQL；对ANSI SQL修改后在1992年采纳的标准，称为SQL-92或SQL2；最近的SQL-99标准，从SQL2扩充而来并增加了对象关系特征和许多其他新功能。其次，各大数据库厂商提供不同版本的SQL（比如PLSQL），这些版本的SQL不但能包括原始的ANSI标准，而且在很大程度上支持SQL-92标准。

SQL语言大致可以分为以下几类类：

* 数据操作语言（DML，Data Manipulation Language ）（DQL+DML）：由select、insert、update和delete关键字完成，用来对数据库表内容的增删改查操作；
* 结构操作语言（数据定义语言，DDL，Data Definition Language ）：create、drop、truncate和alter完成，用来对数据库和数据库对象的增删改操作（drop会同时删除表结构和表数据，truncate会只是删除数据不删除表结构，其原理是先drop然后再创建表结构）；
* 数据库管理语言（数据库控制语言，DCL，DataBase Control Language）：grant、revoke完成；
* 事物控制语言：主要有commit、rollback、savepoint完成。

有上面的介绍可知，大多数与数据库的操作可以总结为对数据库对象（数据库、表、视图、存储过程等）的增删改查操作，所以下面会从各个数据库对象的增删改查来介绍。在MySQL中主要的数据库对象有：

[表格

描述已自动生成](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107155646127-172291988.png)

**MySQL基础知识[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "mysql%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86)**

在讲 MySQL 中数据表的增删改查之前，我们先需要创建表。创建表的过程中需要制定表字段的类型，字段的约束以及表的存储引擎等。所以非常有必要先介绍些MySQL的基础语法知识。

**字符集和校验规则**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%AD%97%E7%AC%A6%E9%9B%86%E5%92%8C%E6%A0%A1%E9%AA%8C%E8%A7%84%E5%88%99)

字符集：字符的集合，常见的字符集有ASCLL、GBK和Unicode等；  
校验规则：当前字符集内，字符之间大小的比较规则。

**数据类型[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B)**

MySQL中三大数据类型是：数值、日期和字符串。

[图示

中度可信度描述已自动生成](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107160408528-1555181622.png)

**数值型**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E5%80%BC%E5%9E%8B)

1. 数值类型

| 类型 | 字节 | 最小值 | 最大值 |
| --- | --- | --- | --- |
| TYNYINT | 1 | -128/0 | 127/255 |
| SMALLINT | 2 | -32768/0 | 32767/65535 |
| MEDIUMINT | 3 | -8388608/0 | 8388607/16777215 |
| INT | 4 | -2147483648/0 | 2147483647/4294967295 |
| BIGINT | 8 | 很大... | 很大... |

INT(M) UNSIGNED ZEROFILL

可以这样定义数值类型。其中M，表示显示宽度，显示宽度不限制数值的范围。配合zerofill来使用，可以在小于显示宽度的位数前增加0。ZEROFILL 自动为unsigned。Unsigned 表示无符号，只表示正数。

2. 小数类型

单精度(不推荐使用)

FLOAT(M,D) UNSIGNED ZEROFILL

双精度

DOUBLE(M,D) UNSIGNED ZEROFILL

其中M表示总的位数（包括小数位数），D表示小数位数。M,D可以控制保存的范围。比如DOUBLE(10,2)可以表示 -99999999.99 到 99999999.99。

定点数（金额，科学数据推荐使用这个类型）

DECIMAL(M,D) UNSIGNED ZEROFILL

其中M表示总的位数，D表示小数位数。此M,D可以控制保存的范围。M，D省略，默认为10,0；

**日期型**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%97%A5%E6%9C%9F%E5%9E%8B)

可以考虑使用long类型来存储，也就是时间戳。

或者使用datetime（或timestamp）类型来存储。但是这两个类型有个问题：datetime不包含时区的概念，也就是你传'2019-09-08 12:56:34'，它存储的就是这个值。timestamp虽然包含时区的概念，但是它的取值范围比较小（1970-01-01 00:00:01 -- 2038-01-19 03:14:07），有些情况不够用。

所以会有上面的建议，使用long类型存时间戳。

**字符串型和二进制**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%AD%97%E7%AC%A6%E4%B8%B2%E5%9E%8B%E5%92%8C%E4%BA%8C%E8%BF%9B%E5%88%B6)

[表格

描述已自动生成](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107161651505-599264458.png)

**选择数据类型的建议[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E9%80%89%E6%8B%A9%E6%95%B0%E6%8D%AE%E7%B1%BB%E5%9E%8B%E7%9A%84%E5%BB%BA%E8%AE%AE)**

当选择字段类型时，应该遵循下面几个原则：

* 应该使用最精确的类型，占用的空间少，已经事先估计字段可能的长度；
* 还应该考虑到相关应用语言的处理，例如常常将时间日期保存成一个整型，便于计算；
* 考虑移植的兼容性。

**约束**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E7%BA%A6%E6%9D%9F)

MySQL中列约束一般用来保证数据库表数据的正确性和完整性。

[图示

描述已自动生成](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107161940426-1702335929.png)

**主键约束**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E4%B8%BB%E9%94%AE%E7%BA%A6%E6%9D%9F)

主键：可以唯一标识一条记录的一列或者多列的组合。主键具有以下特点：

* 每个表只能定义一个主键；
* 任意两行都不具有相同的主键值；
* 每一行都必须具有一个主键值（主键列不允许NULL值）；
* 主键列中的值不允许修改或更新；
* 主键值不能重用（如果某行从表中删除，它的主键不能赋给以后的新行）。

*-- 添加主键*

create table tbl\_user

(

user\_id int primary key,

user\_name varchar(20)

);

create table tbl\_user

(

user\_id int,

user\_name varchar(20),

primary key(user\_id)

);

alter table tbl\_user add primary key(user\_id,user\_name);

ALTER TABLE tbl\_test1 ADD CONSTRAINT pk\_tbl\_test1 PRIMARY KEY(id);

*-- 删除主键（假如主键列具有auto\_increment属性，需要先删除自增长属性，再删除主键）*

ALTER TABLE TABLE\_NAME DROP PRIMARY KEY;

**外键约束[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E5%A4%96%E9%94%AE%E7%BA%A6%E6%9D%9F)**

外键（FK，foreign key)：如果一个A表的字段指向另一个B表的主键，则此字段就为A表的外键。用于表示表之间的关系。存在外键的表，称之为从表（子表），外键指向的表，称之为主表（父表）。外键只被innodb存储引擎所支持。其他引擎是不支持的。一个表可以存在一个或者多个外键。

外键的默认作用有两点：

* 对子表(外键所在的表)的作用：子表在进行写操作的时候，如果外键字段在父表中找不到对应的匹配，操作就会失败；
* 对父表的作用：对父表的主键字段进行删和改时，如果对应的主键在子表中被引用，操作就会失败。

外键的定制有三种约束模式：

* district：严格模式(默认), 父表不能删除或更新一个被子表引用的记录的主键值；
* cascade：级联模式, 父表的主键值操作后，子表关联的数据也跟着一起操作；
* set null：置空模式,前提外键字段允许为NLL, 父表的主键值操作后，子表对应的字段被置空。

*-- 添加外键*

create table tbl\_user

(

user\_id int primary key,

user\_name varchar(20),

class\_id int,

CONSTRAINT fk\_tbl\_user foreign key(class\_id) references tbl\_class(class\_id)

)

alter table tbl\_user add CONSTRAINT fk\_tbl\_user foreign key(class\_id) references tbl\_class(class\_id);

*-- 删除外键*

alter table tbl\_user drop foreign key fk\_tbl\_user;

**唯一性约束[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E5%94%AF%E4%B8%80%E6%80%A7%E7%BA%A6%E6%9D%9F)**

唯一性约束可以确保一列或者多列不出现重复值。

Unique和Primary Key的区别：

* 一个表可以有多个字段声明为 Unique，但是只能有一个Primary Key生命；
* Primary Key的列不能为空值，但是Unique的列可以为空值。

*-- 创建唯一性约束*

CREATE TABLE tbl\_test2

(

id INT,

name1 VARCHAR(20),

nick\_name VARCHAR(20),

age INT,

CONSTRAINT unique\_tbl\_test2\_name UNIQUE(name1,nick\_name)

)

alter table tbl\_test2 add constraint unique\_tbl\_test2\_name UNIQUE(name1,nick\_name);

*-- 删除唯一性约束(从下面的语法我们可以看出唯一性约束其实是个索引)*

alter table tbl\_test2 drop index unique\_tbl\_test2\_name

**检查约束[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%A3%80%E6%9F%A5%E7%BA%A6%E6%9D%9F)**

检查约束用于检查数据库表中某些字段时候符合条件。

*-- 创建检查约束*

CREATE TABLE tbl\_test3

(

id INT,

name1 VARCHAR(20),

nick\_name VARCHAR(20),

age INT,

CONSTRAINT check\_tbl\_test3 CHECK(age>20 AND age<40)

)

alter table table\_test3 add constraint check\_tbl\_test3 check(age>20 and age<40);

*-- 删除检查约束*

alter table tbl\_test3 drop constraint check\_tbl\_test3

**默认值约束[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E9%BB%98%E8%AE%A4%E5%80%BC%E7%BA%A6%E6%9D%9F)**

插入一条记录时没有给这个字段赋值，如果建表时设置了默认值，系统就会给这个字段赋默认值。

*-- 添加默认值*

CREATE TABLE tbl\_test4

(

id INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(20) DEFAULT 'wuming',

age INT

)

ALTER TABLE tbl\_test4 MODIFY COLUMN age INT DEFAULT 20;

*-- 删除默认值（删除age字段的默认值）*

ALTER TABLE tbl\_test4 CHANGE COLUMN age age INT DEFAULT NULL;

**非空约束[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E9%9D%9E%E7%A9%BA%E7%BA%A6%E6%9D%9F)**

*-- 添加非空约束*

CREATE TABLE tbl\_test5

(

id INT PRIMARY KEY AUTO\_INCREMENT,

`name` VARCHAR(20) NOT NULL,

age INT

)

ALTER TABLE tbl\_test5 MODIFY COLUMN age INT NOT NULL;

*-- 删除非空约束*

ALTER TABLE tbl\_test5 MODIFY COLUMN age INT NULL;

**自动增长约束[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%87%AA%E5%8A%A8%E5%A2%9E%E9%95%BF%E7%BA%A6%E6%9D%9F)**

该列上必须有索引，not null，只能存在一个自动增长的列。通常定义在主索引（主键）字段上。

通常 自动增长是从1开始递增，但是可以通过修改表属性，更改初始值。表属性 auto\_increment=x;（如果比已存在的小，则会从已有的最大值新记录）

**存储引擎**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%AD%98%E5%82%A8%E5%BC%95%E6%93%8E)

存储引擎可以理解为表的存储结构，每种存储引擎都支持不同的特性。MySQL支持插件式的存储引擎，可以为每张数据表指定不同的存储引擎。常用的存储引擎有以下几种：

[](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107162417632-1079222610.png)

我们也可以使用下面命令查看当前数据库支持哪些存储引擎：

*-- 查看支持的存储引擎*

show engines;

下面对最常用的三种存储引擎做下简单介绍：

* InnoDB：MySQL默认的存储引擎，支持事务、支持行级锁和表级锁、支持各类索引、支持外键，高版本的MySQL还支持全文索引，但是批量数据插入的效率较低；
* MyISAM：具有较高的数据插入效率和数据查询速度，支持全文索引，但是不支持数据库事务，不支持行级锁，只支持表级锁；
* MEMORY：使用这个存储引擎时，会将表中的数据加载到内存中，查询很快，但是对内存要求较高。

所以我们应该根据应用的具体需求选择合适的存储引擎，而不是不加思考的都选择默认存储引擎（INNODB）。

如果要提供提交、回滚和恢复的事务安全（ACID兼容）能力，并要求实现并发控制，InnoDB是一个很好的选择。如果数据表主要用来插入和查询记录，则MyISAM引擎提供较高的处理效率。如果只是临时存放数据，数据量不大，并且不需要较高的数据安全性，可以选择将数据保存在内存的MEMORY引擎中，MySQL中使用该引擎作为临时表，存放查询的中间结果。如果只有INSERT和SELECT操作，可以选择Archive引擎，Archive存储引擎支持高并发的插入操作，但是本身并不是事务安全的。Archive存储引擎非常适合存储归档数据，如记录日志信息可以使用Archive引擎。

**实体关系和表设计**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%AE%9E%E4%BD%93%E5%85%B3%E7%B3%BB%E5%92%8C%E8%A1%A8%E8%AE%BE%E8%AE%A1)

* 1对1，两个表保存的实体之间数据是对等的，比如一个学生有基本信息和详细信息；
* 1对N，A实体对应多个B实体 ，比如一个班级内有很多学生，而且一个学生只属于一个班级；
* M对N，A实体对应多个B实体，同时一个B实体也对应多个A实体，比如一个讲师可以给多个班级授课。同时一个班级可以由多个来授课；

表设计原则：

* 1对1：两个表具有相同的主键即可；
* 1对N：在多的实体端使用一个字段存储1端实体的主键信息；
* M对N：增加一个存储实体之间对应关系的表，保存A实体主键和B实体主键。每一个记录，对应一个对应关系。

**数据库范式**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E6%8D%AE%E5%BA%93%E8%8C%83%E5%BC%8F)

范式是指数据库设计规范。一共存在6个级别的范式，1NF，2NF，3NF，4NF，5NF，6NF。要求是从低到高逐渐递增。关系型数据库必须满足1NF，通常满足到3NF就可以了。

* 1NF：是指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。第一范式是对关系模式的最起码的要求。不满足第一范式的数据库模式不能称为关系数据库。比如地址这个字段不能存成：江苏省-南通市-启东县，而应该存成三个字段：省、市和县。
* 2NF：在满足1NF的基础上，要求表中的每条记录必须被唯一的区分。通常的做法是为每条记录添加主键。
* 3NF：满足第二范式的基础上，要求不能出现传递依赖，也就是不能出现属性依赖于非主属性的.

**MySQL数据库增删改查[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "mysql%E6%95%B0%E6%8D%AE%E5%BA%93%E5%A2%9E%E5%88%A0%E6%94%B9%E6%9F%A5)**

数据库是所有其他数据库对象的载体，在创建数据库表之前必有先创建一个数据库。

**数据库的创建**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E6%8D%AE%E5%BA%93%E7%9A%84%E5%88%9B%E5%BB%BA)

*-- 建数据库语句模板*

Create database [if not exists] 数据库名 [数据库选项]

*-- 建数据库例子,在建数据库时一般只要指定charset就可以了（有时你还可以设置校验规则 ）*

CREATE DATABASE if not exists my\_database charset utf8;

*-- 选择数据库*

use my\_database;

**数据库的删除[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%95%B0%E6%8D%AE%E5%BA%93%E7%9A%84%E5%88%A0%E9%99%A4)**

*-- 删除数据库语句模板*

drop database [if exists] 数据库名 [数据库选项]

*-- 删除数据库例子,在建数据库时一般只要指定charset就可以了*

drop DATABASE if exists my\_database;

**数据库的修改[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%95%B0%E6%8D%AE%E5%BA%93%E7%9A%84%E4%BF%AE%E6%94%B9)**

*-- 一般只要修改数据库的字符集就可以了，校验规则会默认选和字符集匹配的*

ALTER DATABASE my\_database CHARACTER SET gbk;

MySQL中一般不能直接对数据库重命名，如果想重命名数据库一般的做法有：

1. 创建新数据库，将原数据库内的表重命名到新数据库内，删除原数据库。
2. 将原数据库的数据和结构导出，然后在导入到新数据库内，删除原数据库。

**数据库的查询**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E6%8D%AE%E5%BA%93%E7%9A%84%E6%9F%A5%E8%AF%A2)

*-- 查询数据库名字以my打头的数据库*

SHOW DATABASES LIKE 'my%';

*-- 显示所有的数据库*

SHOW DATABASES;

*-- 显示数据库的创建结构*

SHOW CREATE DATABASE my\_database;

**MySQL数据表的增删改查[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "mysql%E6%95%B0%E6%8D%AE%E8%A1%A8%E7%9A%84%E5%A2%9E%E5%88%A0%E6%94%B9%E6%9F%A5)**

数据表是数据库中最基础的对象，平时大多数操作都是围绕着数据库表进行。在MySQL中，数据表是一个二维表，一般是一个实体的映射（当然也有很多表示关系的映射）。

**表的创建**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E8%A1%A8%E7%9A%84%E5%88%9B%E5%BB%BA)

MySQL中表的创建有下面几种模式：

Create table [if not exists] tbl\_name （列定义） [表选项]

*-- 只会创建表结构*

Create table [if not exists] tbl\_name like old\_tbl\_name;

*-- 创建表结构和内容*

Create table [if not exists] tbl\_name as select 语句; //创建表结构和内容

*-- 列定义部分*

列名 类型 [是否为空] [Default 默认值] [是否为自动增长] [是否为主索引或唯一索引] [comment 注释] [引用定义]

*-- 常见的表选项有ENGINE、字符集和注释等*

*-- 建表语句例子*

CREATE TABLE `sys\_user` (

`user\_id` bigint NOT NULL AUTO\_INCREMENT,

`username` varchar(50) NOT NULL COMMENT '用户名',

`password` varchar(100) COMMENT '密码',

`salt` varchar(20) COMMENT '盐',

`email` varchar(100) COMMENT '邮箱',

`mobile` varchar(100) COMMENT '手机号',

`status` tinyint COMMENT '状态 0：禁用 1：正常',

`create\_user\_id` bigint(20) COMMENT '创建者ID',

`create\_time` datetime COMMENT '创建时间',

PRIMARY KEY (`user\_id`),

UNIQUE INDEX (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='系统用户'

*-- 使用上面语句创建完表之后，可以使用下面语句查看表结构*

show create table sys\_user;

desc sys\_user;

**表的删除[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%A1%A8%E7%9A%84%E5%88%A0%E9%99%A4)**

Drop table [if exists] tbl\_name1，tbl\_name2;

*-- 删除内容，但是保留表结构*

TRUNCATE tbl\_name1;

一次性可以删除多了表。

**表的修改**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E8%A1%A8%E7%9A%84%E4%BF%AE%E6%94%B9)

*-- 修改表名字*

RENAME TABLE tbl\_name1 TO tbl\_name2;

*-- 跨数据库*

RENAME TABLE database1.tbl\_name1 TO database.tbl\_name2;

RENAME TABLE tb\_user TO tbl\_user;

*-- 表结构修改*

*-- 添加字段，First选项用于指定新加的字段是否在第一位，AFTER选项指定新加的字段在哪个字段后面*

ALTER TABLE <表名> ADD <新字段名> <数据类型> [约束条件] [FIRST|AFTER已存在的字段名]；

ALTER TABLE sys\_user ADD COLUMN `address` VARCHAR(50) NOT NULL COMMENT '用户地址' AFTER mobile;

ALTER TABLE sys\_user ADD COLUMN `job` VARCHAR(50) NOT NULL COMMENT '工作' AFTER address;

*-- 删除表字段*

ALTER TABLE sys\_user DROP COLUMN `address`;

ALTER TABLE sys\_user DROP COLUMN `job`;

*-- 修改表字段类型*

ALTER TABLE sys\_user MODIFY COLUMN job VARCHAR(100) NOT NULL DEFAULT '无业' COMMENT '工作';

*-- 修改表字段名称，将字段job改成job\_type*

ALTER TABLE sys\_user CHANGE COLUMN job job\_type VARCHAR(100) NOT NULL DEFAULT '无业' COMMENT '工作类型';

**表的查询[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%A1%A8%E7%9A%84%E6%9F%A5%E8%AF%A2)**

*-- 查看某个数据库下面的所有表*

show tables from database;

*-- 查看表结构*

show create tbl\_name;

desc tbl\_name;

**MySQL表数据的增删改查[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "mysql%E8%A1%A8%E6%95%B0%E6%8D%AE%E7%9A%84%E5%A2%9E%E5%88%A0%E6%94%B9%E6%9F%A5)**

**数据的增加[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%95%B0%E6%8D%AE%E7%9A%84%E5%A2%9E%E5%8A%A0)**

*--MySQL特殊语法，可以一次插入多列，Oracle不支持*

INSERT INTO `tb\_user` (`username`, `mobile`, `password`, `create\_time`) VALUES

('mark4', '13612345678', '8c6976e5b5410415bde908bd4d', '2017-03-23 22:37:41'),

('mark5', '13612345678', '8c6976e5b5410415bde908bd4d', '2017-03-23 22:37:41'),

('mark3', '13612345678', '8c6976e5b5410415bde908bd4d', '2017-03-23 22:37:41');

*--蠕虫复制*

INSERT INTO tbl\_user\_bak (password,create\_time)

SELECT password,create\_time FROM tbl\_user\_bak;

**数据的删除[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%95%B0%E6%8D%AE%E7%9A%84%E5%88%A0%E9%99%A4)**

*--删,支持删除指定条数的数据*

delete from tbl\_name where 条件

DELETE FROM tbl\_name [WHERE where\_definition] [ORDER BY ...] [LIMIT row\_count]

按照条件删除（delete可以指定一次删除几条记录，delete还支持多表删除，类似join语法）

**数据的更新**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E6%8D%AE%E7%9A%84%E6%9B%B4%E6%96%B0)

*--改,支持排序，支持只更新指定条数*

update tbl\_name set col1=value1,col2=value2 where 条件 [order by] [limit]

update支持多表联合更新，类似join语法。

**数据的查询**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E6%8D%AE%E7%9A%84%E6%9F%A5%E8%AF%A2)

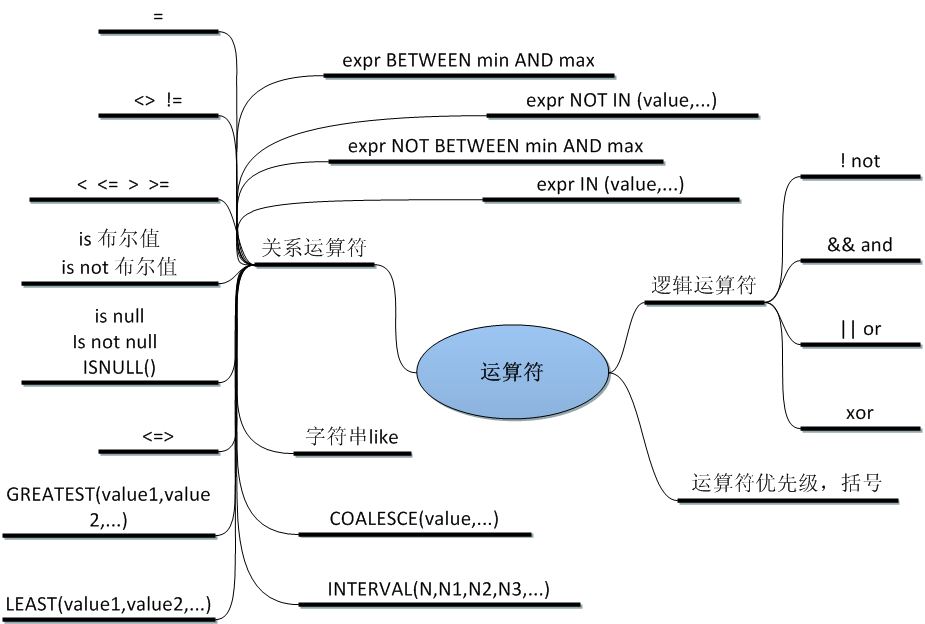
Select 主要用于数据。查询可以配合其5个字句获得，获得相应功能。

Select [distinct] select\_expr [from tbl\_name] [where] [group by] [having] [order by] [limit]

上面的select\_expr可以是表达式、函数表达式、纯列名或者是用as指定别名。表名可以是一个表，可以是多个表，也可以用as给表指定别名。另外表名可以是虚拟表dual。（纯粹是为了满足select \* from dual这种形式）

**Where子句[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "where%E5%AD%90%E5%8F%A5)**

在select查询语句中where条件运算用于初步删选条件，满足这个条件的记录将被进一步删选。常用的运算符有下面这些。除了这些，正则运算符 也是常用的。Mysql本身也提供了很多内置函数帮我们解决问题，我们要善于利用这些函数解决问题。

[](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107164415411-1575090299.png)

where子句表达式的判定结果有3种情况，分别是：true、flase和unknown。

在比较表达式exp1=exp2中有一个表达式的值是null时，整个表达式返回null，where对于null的处理就是false。对于安全等与运算符<=>需要强调下和=号的区别。<=>对于null值不会返回unknown。

对于上面的coalesce函数说明下，coalesce（exp1，exp2 ...，expn，'defaultValue'），这个函数返回第一个不是null的值，如果都是null就返回默认值。

*-- 返回第一个参数，在有序数组中的位置*

mysql>SELECT INTERVAL(6,1,2,3,4,5,6,7,8,9,10);

+*---------------------------------------------------------+*

| INTERVAL(6,1,2,3,4,5,6,7,8,9,10) |

+*---------------------------------------------------------+*

| 6 |

+*---------------------------------------------------------+*

1 row in set (0.00 sec)

*---------------------*

和like关键字类似，Mysql中还提供了基于正则表达式的查询，基本语法如下：

*-- 正则表达式查询*

select exp1，exp2 from tbl\_name where col1 REGEXP 'reg\_exp\_str';

在like子句中使用通配符

* %：匹配：0个或多个字符
* \_:匹配1个字符
* []:匹配一个字符集

**Group By子句[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "group-by%E5%AD%90%E5%8F%A5)**

Group by 根据一个或多个列对结果集进行分组，语法为：

[GROUP BY {col\_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]]

分组后，每组内显示一条记录。原则上，分组后，查询字段应该只有分组字段，但是也可以有其他字段。此时其他字段就只是组内第一条记录的信息，不能准确表示组内所有数据信息，因此通常不查询其他字段。分组字段必须出现在查询字段中，不然会报错。

注意：group by后面可以跟多列，此时会根据多列的组合来分组。比如说几个列的组合不同，mysql就认为这是一个不同的分组，会将它显示出来。null会被单独作为一个分组，若该列中存在多个null值，他们会被分为一组。

group by经常和聚合函数一起使用，用来统计分组后组内的信息。但是可以单独使用，相当于将所有行看成一组。通常合计函数是不统计null值的。

[图示

中度可信度描述已自动生成](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107164958602-1024091384.png)

**Having子句[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "having%E5%AD%90%E5%8F%A5)**

having子句用来对group by之后的结果进行进一步删选。举个栗子：

select teacher, sum(days) as sum\_days from lesson where 1 group by teache having sum\_days > 44;

* <条件>：指定过滤条件。  
  HAVING子句和WHERE子句非常相似，HAVING子句支持WHERE子句中所有的操作符和语法，但是两者存在几点差异：
* WHERE子句主要用于过滤数据行，而HAVING子句主要用于过滤分组，即HAVING子句基于分组的聚合值而不是特定行的值来过滤数据，主要用来过滤分组。
* WHERE子句不可以包含聚合函数，HAVING子句中的条件可以包含聚合函数。
* HAVING子句是在数据分组后进行过滤，WHERE子句会在数据分组前进行过滤。WHERE子句排除的行不包含在分组中，可能会影响HAVING子句基于这些值过滤掉的分组。

**Order By子句[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "order-by%E5%AD%90%E5%8F%A5)**

基本语法如下：

*-- 下面的位置是指select后面字段的位置*

ORDER BY {<列名>|<表达式>|<位置>} [AESC|DESC]

Order by，可以使用一列或者多个列对结果进行排序。如果存在多个排序字段，在前一个不能比较出结果后，后边的才起作用。可以分别指明是升序还是降序：asc（ascending） desc（descending）。Order by会把null值当做最小值来处理。如果不使用Order By排序的话，默认使用插入顺序来显示。 order by的字段可以不出现在查询的字段中，比如：

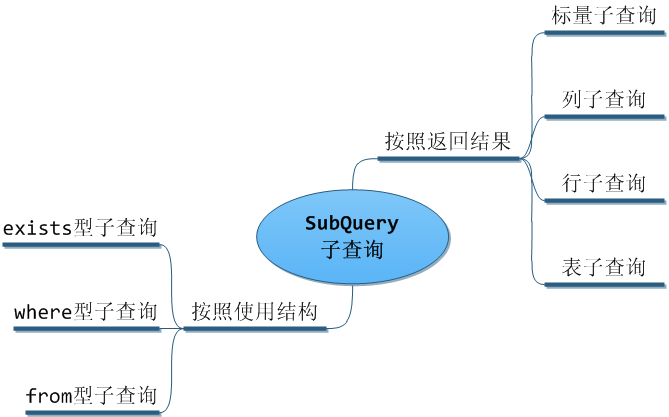
SELECT vend\_id, prod\_name FROM products ORDER BY prod\_price;

**limit子句**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#limit%E5%AD%90%E5%8F%A5)

用于限制select语句最后选中的记录的个数。limit offset，rowcount，如果省略offset则默认从0开始（第一条数据的偏移量是0）。

**子查询**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%AD%90%E6%9F%A5%E8%AF%A2)

子查询是指嵌套在其他查询语句内部的查询。

[](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107165245387-384675222.png)

1. 根据子查询返回的结果

一个子查询会返回一个标量（单一值）、一个行、一个列或一个表（一行或多行及一列或多列）。这些子查询被称为标量、列、行和表子查询。

2. 根据子查询的位置  
根据子查询select 出现的的位置不同分为Where型子查询，from型子查询和exists子查询

* Where型子查询出现在where子句，通过比较运算符进行操作的。
* Exists(subquery)如果子查询返回记录，则exists返回1；
* From (subquery) as tmp\_table; [AS] name子句是强制性的，因为FROM子句中的每个表必须有一个名称。

在子查询中会经常用到ANY（SOME）、ALL、IN和EXISTS这些操作符。

**连接查询[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%BF%9E%E6%8E%A5%E6%9F%A5%E8%AF%A2)**

[图示

描述已自动生成](https://img2020.cnblogs.com/blog/1775037/202011/1775037-20201107165646618-542303758.png)

1. 内连接

内连接是把两个表先做笛卡尔积，然后再根据连接条件删选出相应的记录。

SELECT A.\*, B.\* FROM

student\_info A inner join student\_score B

ON A.student\_id = B.student\_id

内连接中还存在一个cross join，一般用来做笛卡尔积，就是没有条件的inner join。但是需要注意的是在mysql数据库中corss join的行为和inner join一致，也是可以添加on条件的。建议平时使用inner join。

2. 外连接

mysql中外连接分为左外连接和右外连接（不支持全外连接）。如果是左外连接的话，除了满足连接条件的记录，左表中不满足连接条件记录也会被返回，右表中不满足连接条件的字段以null填充。右连接则相反。

Where和on可以使用条件表达式。Using (公共字段)，需要使用公共字段。

内联可以使用where on 和 条件，外联 只可以使用using和on作为条件。

通常 where还有筛选 含义，而on和using只有连接条件的意思。因此通常连接条件使用on或using。而筛选条件使用where加以区分。推荐使用on关键字来表示连接条件。

3. 自连接

将同一张表自己和自己进行Inner join。暂时没看到有太大的用处。

4. 各种连接查询的区别

* 内连接和外连接区别
* 左外连接和右外连接的区别

**联合查询**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E8%81%94%E5%90%88%E6%9F%A5%E8%AF%A2)

联合查询，使用union关键字，用于把来自许多SELECT语句的结果组合到一个结果集合中。相当于将两个多个select语句的结果垂直的合并在一起。使用语法如下：

select column1，column2 from tbl\_name1

union

select column1，column2 from tbl\_name2;

在第一个SELECT语句中被使用的列名称被用于结果的列名称 .默认情况下，会去除重复的数据行，可以使用union all保留重复的数据行。如要对union的结果作为整体进行排序或limit，（最好对单个地SELECT语句加圆括号），在后边增加order by 或limit。order by引用的数据列来自第一个select。分别排序，再联合，则需要用括号将各select语句括起来，且order by只能在limit 出现时才有效。

**函数**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%87%BD%E6%95%B0)

**数值型函数[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%95%B0%E5%80%BC%E5%9E%8B%E5%87%BD%E6%95%B0)**

* ABS函数：绝对值
* MOD函数：取余函数
* SQRT函数：平方根函数
* SIGN函数：判断参数是整数、负数或者是0；
* CEIL和CEILING函数：返回CEIL(x)，返回不小于x的最小整数；

**字符串型函数**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E5%AD%97%E7%AC%A6%E4%B8%B2%E5%9E%8B%E5%87%BD%E6%95%B0)

* LENGTH函数：返回字符串的长度
* UPPER和LOWER函数
* LEFT（s,n）：返回字符串最左边的n个字符
* CONCAT（s1,s2,s3,...）：字符串拼接函数，有一个参数是null就返回null
* TRIM函数
* SUBSTRING函数
* REVERSE函数：字符串逆序
* REPLACE：替换函数

**日期函数**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%97%A5%E6%9C%9F%E5%87%BD%E6%95%B0)

* CURDATE和CURRENT\_DATE：返回当前日期；
* NOW和SYSDATE；
* CURTIME和CURRENT\_TIME；
* DAYOFWEEK和WEEKDAY：返回某天是一个星期的第几天；
* DAYOFMONTH；
* DAYOFYEAR；
* MONTH；
* MONTHNAME；
* DATEDIFF；
* ADDDATE；
* DATE\_FORMAT。

**聚合函数**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E8%81%9A%E5%90%88%E5%87%BD%E6%95%B0)

* MAX：max不仅可以计算数值，还可以计算字符串
* MIN
* COUNT：统计记录数，count（col\_name）会忽略null值，count（\*）统计所有行；
* SUM：对某一列求和，忽略值为null的行；
* AVG：忽略null行

**自定义函数（见存储过程）[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%87%AA%E5%AE%9A%E4%B9%89%E5%87%BD%E6%95%B0%E8%A7%81%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B)**

**视图[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%A7%86%E5%9B%BE)**

在数据库中，视图是一张虚拟的表，视图和普通的表没有明显的区别。视图的结构和内容来自定义视图的查询语句。使用视图有如下优势：

1. 视图可以简化查询，可以把复杂的查询结果放到视图中，当我们再需要进行相同的查询时，可以直接从视图进行查询，简化查询。
2. 增加数据的安全性：表的有些敏感数据可能不许让用户查询，这时可以将允许查询的数据放入视图中，让用户直接查询视图即可。

关于在查询视图时使用Order By做下说明：

ORDER BY子句可以用在视图中，但若该视图检索数据的SELECT语句中也含有ORDER BY子句，则该视图中的ORDER BY子句将被覆盖。

视图管理

*--创建视图：*

CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] VIEW view\_name [(column\_list)] AS select\_statement

视图名必须唯一，同时不能与表重名。视图可以使用select语句查询到的列名，也可以自己指定相应的列名。可以指定视图执行的算法，通过algorithm指定。

*--查看结构*

SHOW CREATE VIEW view\_name

*--删除视图*

DROP VIEW [IF EXISTS] view\_name [, view\_name];

*--修改视图结构*

ALTER VIEW view\_name [(column\_list)] AS select\_statement

**存储过程（自定义函数）[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B%E8%87%AA%E5%AE%9A%E4%B9%89%E5%87%BD%E6%95%B0)**

自定义函数是一种与存储过程十分相似的过程式数据库对象。它与存储过程一样，都是由SQL语句和过程式语句组成的代码片段，并且可以被应用程序和其他SQL语句调用。但是，自定义函数与存储过程之间仍存在几点区别：

* 自定义函数不能拥有输出参数，这是因为自定义函数自身就是输出参数；而存储过程可以拥有输出参数。
* 自定义函数中必须包含一条RETURN语句，而这条特殊的SQL语句不允许包含于存储过程中。
* 可以直接对自定义函数进行调用而不需要使用CALL语句，而对存储过程的调用需要使用CALL语句。

**自定义函数**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E8%87%AA%E5%AE%9A%E4%B9%89%E5%87%BD%E6%95%B0)

CREATE FUNCTION <函数名> ( [ <参数1> <类型1> [ , <参数2> <类型2>] ] … )

RETURNS <类型>

<函数主体>

在RETURN VALUE语句中包含SELECT语句时，SELECT语句的返回结果只能是一行且只能有一列值。

自定义函数的管理

*-- 删除函数*

drop function 函数名;

*-- 修改函数*

alter function ...;

*-- 查看函数*

show create function name;

**存储过程[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B)**

存储过程是一组为了完成特定功能的SQL语句表，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数（如果该存储过程带有参数）来调用执行。具有如下优点：

* 不必要每次都编译SQL语句；
* 可以增强SQL的灵活性，可以使用流程控制语句，完成复杂的判断和较复杂的运算；
* 可以减少网络流量，因为存储过程是保存在数据库端的，不需要客户端传复杂的SQL语句到数据进行执行。

流程控制相关语法

DELIMITER $$

CREATE FUNCTION getProdName(X INT) RETURNS VARCHAR(50)

BEGIN

*-- 定义一个局部变量i，默认值是1*

*-- 局部变量只能在Begin和end之间定义，作用域也只在begin和end之间，且必须在开头定义*

*-- 全局变量使用@定义*

DECLARE i INT DEFAULT 1;

DECLARE res VARCHAR(50);

*-- set用来赋值*

SET i = 2;

IF(X=i) THEN

*-- 使用select into 语句对一个变量赋值*

SELECT prod\_name FROM products WHERE prod\_INTO res;

RETURN res;

END IF;

SELECT prod\_name FROM products WHERE prod\_INTO res;

RETURN res;

END

$$

DELIMITER ;

条件控制语句

if(条件判断) then

语句1；

else if(判断)

语句2；

else

语句3;

end if

case

when(判断1) then

语句1；

when（判断2）

语句2；

default

语句3；

end case

循环控制语句

*-- 除非使用leave语句离开循环，不然的话循环会一直执行*

<标签>loop

语句；

end loop [标签]

<标签> while <条件判断> do

语句；

end while <标签>

<标签> repeat

语句；

until<条件>

end repeat <标签>

存储过程的详细定义请参考这篇[教程](https://www.runoob.com/w3cnote/mysql-stored-procedure.html" \t "_blank)

**触发器[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E8%A7%A6%E5%8F%91%E5%99%A8)**

触发器是关联到某张表的一个数据库对象。当表的一个特定事件（增删改事件）发生时将会触发这个触发器。

MySQL中有Insert、Update和delete事件，每种事件又分为前后两种事件，所以一共有6种触发器类型。

需要使用时可以学习相关知识（主要注意的是触发器不能控制在事务中）

**索引**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E7%B4%A2%E5%BC%95)

索引就是根据表中的一列或若干列按照一定顺序建立的列值与记录行之间的对应关系表，实质上是一张描述索引列的列值与原表中记录行之间一一对应关系的有序表。

**索引分类**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E7%B4%A2%E5%BC%95%E5%88%86%E7%B1%BB)

根据索引的底层实现方式分，MySQL中的索引可以分为：

* B+树索引
* Hash索引：只支持等值比较，比如“=” “IN（）”等
* R树索引

按使用逻辑分，可以分为：

* 普通索引是最基本的索引类型，唯一任务是加快对数据的访问速度，没有任何限制。创建普通索引时，通常使用的关键字是INDEX或KEY。
* 唯一性索引是不允许索引列具有相同索引值的索引。如果能确定某个数据列只包含彼此各不相同的值，在为这个数据列创建索引的时候就应该用关键字UNIQUE把它定义为一个唯一性索引。创建唯一性索引的目的往往不是为了提高访问速度，而是为了避免数据出现重复。
* 主键索引是一种唯一性索引，即不允许值重复或者值为空，并且每个表只能有一个主键。主键可以在创建表的时候指定，也可以通过修改表的方式添加，必须指定关键字PRIMARY KEY。
* 空间索引主要用于地理空间数据类型GEOMETRY。
* 全文索引只能在VARCHAR或TEXT类型的列上创建，并且只能在MyISAM存储引擎上创建。

索引也可以被分成单列索引和组合索引。

* 单列索引就是索引只包含原表的一个列。
* 组合索引也称为复合索引或多列索引，相对于单列索引来说，组合索引是将原表的多个列共同组成一个索引。

一个表可以有多个单列索引，但这些索引不是组合索引。一个组合索引实质上为表的查询提供了多个索引，以此来加快查询速度。比如，在一个表中创建了一个组合索引(c1, c2, c3)，在实际查询中，系统用来实际加速的索引有三个：单个索引(c1)、双列索引(c1, c2)和多列索引(c1, c2, c3)。

**使用原则和注意事项**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E4%BD%BF%E7%94%A8%E5%8E%9F%E5%88%99%E5%92%8C%E6%B3%A8%E6%84%8F%E4%BA%8B%E9%A1%B9)

虽然索引可以加快查询速度，提高MySQL的处理性能，但是过多地使用索引也会造成以下弊端。

● 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。  
● 除了数据表占数据空间之外，每一个索引还要占一定的物理空间。如果要建立聚簇索引，那么需要的空间就会更大。  
● 当对表中的数据进行增加、删除和修改的时候，索引也要动态地维护，  
创建索引可以遵循下列原则：  
● 在经常需要搜索的列上建立索引，可以加快搜索的速度。  
● 在作为主键的列上创建索引，强制该列的唯一性，并组织表中数据的排列结构。  
● 在经常使用表连接的列上创建索引，这些列主要是一些外键，可以加快表连接的速度。  
● 在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，所以其指定的范围是连续的。  
● 在经常需要排序的列上创建索引，因为索引已经排序，所以查询时可以利用索引的排序，加快排序查询。  
● 在经常使用WHERE子句的列上创建索引，加快条件的判断速度。

对于那些很少使用的列，切忌建立索引；对于那些只有很少几个值的列（比如性别）也不要建立索引；对于那些text IMG和Bit类型的字段不要建立索引。

**索引新增**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E7%B4%A2%E5%BC%95%E6%96%B0%E5%A2%9E)

创建索引的三种方式

1. create index语法

create <index\_name> on <tbl\_name> ( <列名> [<长度>] )

比如

create index index\_cust\_address on test.customers(cust\_address);

语法说明如下。

* <索引名>：指定索引名。一个表可以创建多个索引，但每个索引在该表中的名称是唯一的。
* <表名>：指定要创建索引的表名。
* <列名>：指定要创建索引的列名，包含下列三个要素。
* <列名>：指定要创建索引的列名。通常可以考虑将查询语句中在JOIN子句和WHERE子句里经常出现的列作为索引列。
* <长度>：可选项。指定使用列前的length个字符来创建索引。使用列的一部分创建索引有利于减小索引文件的大小，节省索引列所占的空间。在某些情况下，只能对列的前缀进行索引。索引列的长度有一个最大上限255个字节（MyISAM和InnoDB表的最大上限为1000个字节），如果索引列的长度超过了这个上限，就只能用列的前缀进行索引。另外，BLOB或TEXT类型的列也必须使用前缀索引。
* ASC | DESC：可选项。ASC指定索引按照升序来排列，DESC指定索引按照降序来排列，默认为ASC。

**查看索引**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%9F%A5%E7%9C%8B%E7%B4%A2%E5%BC%95)

show index from tbl\_name

返回字段说明如下。

* Table：表的名称。
* Non\_unique：用于显示该索引是否是唯一索引。若不是唯一索引，则该列的值显示为1；若是唯一索引，则该列的值显示为0。
* Key\_name：索引的名称。
* Seq\_in\_index：索引中的列序列号，从1开始计数。
* Column\_name：列名称。
* Collation：显示列以何种顺序存储在索引中。在MySQL中，升序显示值“A”（升序），若显示为NULL，则表示无分类。
* Cardinality：显示索引中唯一值数目的估计值。基数根据被存储为整数的统计数据计数，所以即使对于小型表，该值也没有必要是精确的。基数越大，当进行联合时，MySQL使用该索引的机会就越大。
* Sub\_part：若列只是被部分编入索引，则为被编入索引的字符的数目。若整列被编入索引，则为NULL。
* Packed：指示关键字如何被压缩。若没有被压缩，则为NULL。
* Null：用于显示索引列中是否包含NULL。若列含有NULL，则显示为YES。若没有，则该列显示为NO。
* Index\_type：显示索引使用的类型和方法（BTREE、FULLTEXT、HASH、RTREE）。
* Comment：显示评注。

**索引删除**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E7%B4%A2%E5%BC%95%E5%88%A0%E9%99%A4)

drop index index\_name on tbl\_name

**用户和权限[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E7%94%A8%E6%88%B7%E5%92%8C%E6%9D%83%E9%99%90)**

MySQL中用户信息都存储在mysql这个库的user表中。root用户具有超级权限，应尽量少用这个用户，避免误操作。

**用户创建**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E7%94%A8%E6%88%B7%E5%88%9B%E5%BB%BA)

*-- 创建一个用户chen，密码也是chen，只能在本地登录*

create user 'chen'@'localhost' identified by 'chen'

*-- 创建一个用户zhao，密码也是zhao,不指定host的话在哪个网断都能登陆*

create user 'zhao' identified by 'zhao'；

*-- 修改用户名，但是密码不变*

*-- 查看原来用户信息*

select \* from mysql.user;

RENAME USER 'chen'@'localhost' TO 'chen\_new'@'localhost'；

*-- 修改密码*

SET PASSWORD FOR 'chen\_new'@'localhost' = PASSWORD('chen\_new');

*-- 使用mysqladmin将root的密码改成root*

mysqladmin -u root password ‘root’

*-- 删除用户*

drop user 'chen\_new'@'localhost';

**权限管理[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E6%9D%83%E9%99%90%E7%AE%A1%E7%90%86)**

*-- 查看权限*

*-- 如果查出来是USAGE ON \*.\* 表示这个用户没有任何权限*

SHOW GRANTS FOR 'root'@'localhost'

*-- 用户授权*

*-- 取消用户权限*

**事务和数据备份**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E4%BA%8B%E5%8A%A1%E5%92%8C%E6%95%B0%E6%8D%AE%E5%A4%87%E4%BB%BD)

**事务[#](https://www.cnblogs.com/54chensongxia/p/13941662.html" \l "%E4%BA%8B%E5%8A%A1)**

事务是数据库中的一个操作单元，这些操作要么全部成功要么全部失败，是一个不可分割的单位。

事务具有4个特性：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持续性（Durability）。这4个特性简称为ACID特性。

* 原子性：事务必须是原子工作单元，事务中的操作要么全部执行，要么全都不执行，不能只完成部分操作。原子性在数据库系统中，由恢复机制来实现。
* 一致性：事务开始之前，数据库处于一致性的状态；事务结束后，数据库必须仍处于一致性状态。数据库一致性的定义是由用户负责的。例如，在银行转账中，用户可以定义转账前后两个账户金额之和保持不变。
* 隔离性：系统必须保证事务不受其他并发执行事务的影响，即当多个事务同时运行时，各事务之间相互隔离，不可互相干扰。事务查看数据时所处的状态，要么是另一个并发事务修改它之前的状态，要么是另一个并发事务修改它之后的状态，事务不会查看中间状态的数据。隔离性通过系统的并发控制机制实现。
* 持久性：一个已完成的事务对数据所做的任何变动在系统中是永久有效的，即使该事务产生的修改不正确，错误也将一直保持。持久性通过恢复机制实现，发生故障时，可以通过日志等手段恢复数据库信息。

**数据备份恢复**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E6%95%B0%E6%8D%AE%E5%A4%87%E4%BB%BD%E6%81%A2%E5%A4%8D)

1. 备份

*-- 这种方式只能备份表的内容，但是不能备份表结构*

select into file

*-- 列子*

select \* from tbl\_name into outfile 'D://file.txt';

1. 数据恢复

*-- 导入数据前需要先创建表结构*

load data infile into tbl\_name;

**一些优化建议**[**#**](https://www.cnblogs.com/54chensongxia/p/13941662.html#%E4%B8%80%E4%BA%9B%E4%BC%98%E5%8C%96%E5%BB%BA%E8%AE%AE)

* MySQL是用一系列的默认设置预先配置的，这些设置开始通常是很好的。但过一段时间后你可能需要调整内存分配、缓冲区大小等。（为查看当前设置，可使用SHOW VARIABLES；和SHOW STATUS）
* MySQL是一个多用户多线程的DBMS，换言之，它经常同时执行多个任务。如果这些任务中的某一个执行缓慢，则所有请求都会执行缓慢。如果你遇到显著的性能不良，可使用SHOW PROCESSLIST显示所有活动进程（以及它们的线程ID和执行时间）。你还可以用KILL命令终结某个特定的进程（使用这个命令需要作为管理员登录）。
* 总是有不止一种方法编写同一条SELECT语句。应该试验联结、并、子查询等，找出最佳的方法。
* 使用EXPLAIN语句让MySQL解释它将如何执行一条SELECT语句。
* 一般来说，存储过程执行得比一条一条地执行其中的各条MySQL语句快。
* 应该总是使用正确的数据类型。
* 决不要检索比需求还要多的数据。换言之，不要用SELECT 所有列（除非你真正需要每个列）。
* 有的操作（包括INSERT）支持一个可选的DELAYED关键字，如果使用它，将把控制立即返回给调用程序，并且一旦有可能就实际执行该操作。
* 在导入数据时，应该关闭自动提交。你可能还想删除索引（包括FULLTEXT索引），然后在导入完成后再重建它们。
* 必须索引数据库表以改善数据检索的性能。确定索引什么不是一件微不足道的任务，需要分析使用的SELECT语句以找出重复的WHERE和ORDER BY子句。如果一个简单的WHERE子句返回结果所花的时间太长，则可以断定其中使用的列（或几个列）就是需要索引的对象。
* 你的SELECT语句中有一系列复杂的OR条件吗？通过使用多条SELECT语句和连接它们的UNION语句，你能看到极大的性能改进。
* 索引改善数据检索的性能，但损害数据插入、删除和更新的性能。如果你有一些表，它们收集数据且不经常被搜索，则在有必要之前不要索引它们。（索引可根据需要添加和删除。）
* LIKE很慢。一般来说，最好是使用FULLTEXT而不是LIKE。
* 数据库是不断变化的实体。一组优化良好的表一会儿后可能就面目全非了。由于表的使用和内容的更改，理想的优化和配置也会改变。
* 最重要的规则就是，每条规则在某些条件下都会被打破。