# 数据库简介

Mysql是最流行的关系型数据库管理系统，在WEB应用方面MySQL是最好的RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。

## 1什么是数据库？

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库，

每个数据库都有一个或多个不同的API用于创建，访问，管理，搜索和复制所保存的数据。

我们也可以将数据存储在文件中，但是在文件中读写数据速度相对较慢。

所以，现在我们使用关系型数据库管理系统（RDBMS）来存储和管理的大数据量。所谓的关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

RDBMS即关系数据库管理系统(Relational Database Management System)的特点：

(1)数据以表格的形式出现

(2)每行为各种记录名称

(3)每列为记录名称所对应的数据域

(4)许多的行和列组成一张表单

(5)若干的表单组成database

## 2 RDBMS术语

**数据库**: 数据库是一些关联表的集合。.

**数据表**: 表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。

**列（属性）**: 一列(数据元素) 包含了相同的数据, 例如邮政编码的数据。

**行（记录）**：一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。

**键：**属性的集合。所有属性组成的集合叫全键。

**超键（码）：**可以用来区分每一个记录的单个属性或者属性组合。一个表可以有多个超键。

**候选键（候选码）**：不含多余属性的超键（即其余属性对候选键都是完全函数依赖，不会对其子集依赖）。候选键包含的属性叫做主属性，除主属性之外的属性叫做非主属性。一个表可以有多个候选键。

**主键（主码）**：被选定的候选键。一个数据表中只能包含一个主键。选定主键后，主属性就是主键包含的属性，其余属性就是非主属性。

**外键（外码）**：外键用于关联两个表。一个表的外键是另一个表的主键。

**索引**：使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。

**冗余**：同一数据存储多次，冗余降低了性能，但提高了数据的安全性。

**函数依赖**：若在一张表中，在属性（或属性组）X的值确定的情况下，必定能确定属性Y的值，那么就可以说Y函数依赖于X，写作 X → Y。

**完全函数依赖**：在一张表中，若 X → Y，且对于 X 的任何一个真子集x’（假如属性组 X 包含超过一个属性的话），X’→ Y 不成立，那么我们称 Y 对于 X 完全函数依赖，记作 X Y。比如，学号姓名，（学号，课名） 分数，因为只知道学号或者课名都不能知道一个学生某门课的成绩。

**部分函数依赖**：假如 Y 函数依赖于 X，但同时 Y 并不完全函数依赖于 X，（即Y也函数依赖于X的子集）那么我们就称 Y 部分函数依赖于 X，记作 X Y。比如，（学号，课程）记作 X Y 姓名。

**传递函数依赖：**Y 不包含于 X，且 X 不函数依赖于 Y，假如 Z 函数依赖于 Y，且 Y 函数依赖于 X，那么我们就称 Z 传递函数依赖于 X。比如，学号→姓名→父亲，学号父亲。

**参照完整性**: 参照的完整性要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。

**数据库设计范式**：目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、第四范式(4NF）和第五范式（5NF，又称完美范式）。

**1NF**：每个属性都不可再分。

**2NF**：2NF在1NF基础上消除了非主属性对于某候选键（主键）的部分函数依赖。不满足第二范式会造成数据冗余过大，插入，删除，修改 异常。注意，主键为单属性，一定满足2NF。

**3NF**：3NF在2NF的基础之上消除了非主属性对于某候选键（主键）的传递函数依赖。注意，非主属性只有一个一定满足3NF。

**BCNF**：只有一个候选键的3NF。

**4NF**：消除多值依赖的

## 3 Mysql数据库

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB公司开发，目前属于Oracle公司。MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

Mysql是开源的，所以你不需要支付额外的费用。

Mysql支持大型的数据库。可以处理拥有上千万条记录的大型数据库。

MySQL使用标准的SQL数据语言形式。

Mysql可以允许于多个系统上，并且支持多种语言。这些编程语言包括C、C++、Python、Java、Perl、PHP、Eiffel、Ruby和Tcl等。

Mysql对PHP有很好的支持，PHP是目前最流行的Web开发语言。

MySQL支持大型数据库，支持5000万条记录的数据仓库，32位系统表文件最大可支持4GB，64位系统支持最大的表文件为8TB。

Mysql是可以定制的，采用了GPL协议，你可以修改源码来开发自己的Mysql系统。

# MySQL安装

## 1 ubuntu/linux

安装：

sudo apt install mysql-server //安装服务器

sudo apt install mysql-client //安装客户端

sudo apt install libmysqlclient-dev //安装c语言开发环境

sudo apt install mysql-workbench //安图形化界面

卸载：

sudo apt autoremove --purge mysql-server-5.7

sudo apt remove mysql-common

sudo rm -rf /etc/mysql/ /var/lib/mysql

dpkg -l |grep ^rc|awk '{print $2}' |sudo xargs dpkg -P

### 1.1 mysqladmin工具来获取服务器状态：

使用 mysqladmin 命令俩检查服务器的版本,在linux上该二进制文件位于 /usr/bin ，在window上该二进制文件位于C:\mysql\bin 。

[root@host]# mysqladmin –version

linux上该命令将输出以下结果，该结果基于你的系统信息：

mysqladmin Ver 8.23 Distrib 5.0.9-0, for redhat-linux-gnu on i386

如果以上命令执行后未输入任何信息，说明你的Mysql未安装成

### 1.2用MySQL Client执行简单的SQL命令

你可以在 MySQL Client(Mysql客户端) 使用 mysql 命令连接到Mysql服务器上，默认情况下Mysql服务器的密码为空。如果设了密码则要使用-p选项。

命令如下：

[root@host]$ mysql -u 用户名 -p

以上命令执行后会输出 mysql>提示符，这说明你已经成功连接到Mysql服务器上，你可以在 mysql> 提示符执行SQL命令.

### 1.3 Msql安装后设置root密码

方法1 使用mysqladmin

Mysql安装成功后，默认的root用户密码为空，你可以使用以下命令来创建root用户的密码：

[root@host]$ mysqladmin -u root password "new\_password";

如果root已经设置过密码，采用如下方法

　　[root@host]$mysqladmin -u root password “oldpassword” "newpassword"

方法2：用SET PASSWORD命令

　　mysql -u root –p （如果未设置密码则不加-p）

　　mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('newpass');

方法3： 用UPDATE直接编辑user表

　　mysql -u root

　　mysql> use mysql;

　　mysql> UPDATE user SET Password = PASSWORD('newpass') WHERE user = 'root';

　　mysql> FLUSH PRIVILEGES;

现在你可以通过以下命令来连接到Mysql服务器：

[root@host]# mysql -u root -p

Enter password:\*\*\*\*\*\*\*

注意：在输入密码时，密码是不会显示了，你正确输入即可。如果已经设置root密码，则会出错。

## 2 Window上安装Mysql

Window上安装Mysql相对来说会较为简单，你只需要载 MySQL 下载中下载window版本的mysql安装包，并解压安装包。

双击 setup.exe 文件，接下来你只需要安装默认的配置点击"next"即可，默认情况下安装信息会在C:\mysql目录中。

接下来你可以通过"开始" =》在搜索框中输入 " cmd" 命令 =》 在命令提示符上切换到 C:\mysql\bin 目录，并输入一下命令：

mysqld.exe --console

如果安装成功以上命令将输出一些mysql启动及InnoDB信息。

# MySQL 管理

## 1开启、关闭、重启

### 1.1启动

使用 service 启动：service mysqld start

使用 mysqld 脚本启动：/etc/inint.d/mysqld start

使用 safe\_mysqld 启动：safe\_mysqld&

### 1.2停止

使用 service 启动：service mysqld stop

使用 mysqld 脚本启动：/etc/inint.d/mysqld stop

使用mysqladmin shutdown

### 1.3重启

使用 service 启动：service mysqld restart

使用 mysqld 脚本启动：/etc/inint.d/mysqld restart

### 1.4 查看状态

使用 service 启动：service mysqld status

使用 mysqld 脚本启动：/etc/inint.d/mysqld status

## 2 添加用户

mysql是通过mysql库的user表来管理用户的。

### 2.1用insert修改用户表

INSERT INTO mysql.user (host, user, authentication\_string,权限列表) VALUES ('主机’, '用户名', PASSWORD('密码'),’权限授予表’);

FLUSH PRIVILEGES

注意：在添加用户时，请注意使用MySQL提供的 PASSWORD() 函数来对密码进行加密。

注意：在 MySQL5.7 中 user 表的 password属性 已换成了authentication\_string。

注意：在注意需要执行 FLUSH PRIVILEGES 语句。 这个命令执行后会重新载入授权表。如果你不使用该命令，你就无法使用新创建的用户来连接mysql服务器，除非你重启mysql服务器。

注意：插入用户时如果出现ERROR 1364 (HY000): Field 'ssl\_cipher' doesn't have a default value错误。

原因：配置文件my.cnf中有sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES，指定了严格模式，为了安全，严格模式禁止通过insert 这种形式直接修改mysql库中的user表进行添加新用户。 解决办法是将配置文件中的STRICT\_TRANS\_TABLES删掉。然后重启mysql即可，要是还不能解决，其原因是因为，mysql用户表的中某些字段不能为空，且没有默认值。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Host | char(60) | NO | PRI |  |  |
| User | char(32) | NO | PRI |  |  |
| Select\_priv | enum('N','Y') | NO |  | N |  |
| Insert\_priv | enum('N','Y') | NO |  | N |  |
| Update\_priv | enum('N','Y') | NO |  | N |  |
| Delete\_priv | enum('N','Y') | NO |  | N |  |
| Create\_priv | enum('N','Y') | NO |  | N |  |
| Drop\_priv | enum('N','Y') | NO |  | N |  |
| Reload\_priv | enum('N','Y') | NO |  | N |  |
| Shutdown\_priv | enum('N','Y') | NO |  | N |  |
| Process\_priv | enum('N','Y') | NO |  | N |  |
| File\_priv | enum('N','Y') | NO |  | N |  |
| Grant\_priv | enum('N','Y') | NO |  | N |  |
| References\_priv | enum('N','Y') | NO |  | N |  |
| Index\_priv | enum('N','Y') | NO |  | N |  |
| Alter\_priv | enum('N','Y') | NO |  | N |  |
| Show\_db\_priv | enum('N','Y') | NO |  | N |  |
| Super\_priv | enum('N','Y') | NO |  | N |  |
| Create\_tmp\_table\_priv | enum('N','Y') | NO |  | N |  |
| Lock\_tables\_priv | enum('N','Y') | NO |  | N |  |
| Execute\_priv | enum('N','Y') | NO |  | N |  |
| Repl\_slave\_priv | enum('N','Y') | NO |  | N |  |
| Repl\_client\_priv | enum('N','Y') | NO |  | N |  |
| Create\_view\_priv | enum('N','Y') | NO |  | N |  |
| Show\_view\_priv | enum('N','Y') | NO |  | N |  |
| Create\_routine\_priv | enum('N','Y') | NO |  | N |  |
| Alter\_routine\_priv | enum('N','Y') | NO |  | N |  |
| Create\_user\_priv | enum('N','Y') | NO |  | N |  |
| Event\_priv | enum('N','Y') | NO |  | N |  |
| Trigger\_priv | enum('N','Y') | NO |  | N |  |
| Create\_tablespace\_priv | enum('N','Y') | NO |  | N |  |
| ssl\_type | enum('','ANY','X509','SPECIFIED') | NO |  |  |  |
| ssl\_cipher | blob | NO |  | NULL |  |
| x509\_issuer | blob | NO |  | NULL |  |
| x509\_subject | blob | NO |  | NULL |  |
| max\_questions | int(11) unsigned | NO |  | 0 |  |
| max\_updates | int(11) unsigned | NO |  | 0 |  |
| max\_connections | int(11) unsigned | NO |  | 0 |  |
| max\_user\_connections | int(11) unsigned | NO |  | 0 |  |
| plugin | char(64) | NO |  | mysql\_native\_password |  |
| authentication\_string | text | YES |  | NULL |  |
| password\_expired | enum('N','Y') | NO |  | N |  |
| password\_last\_changed | timestamp | YES |  | NULL |  |
| password\_lifetime | smallint(5) unsigned | YES |  | NULL |  |
| account\_locked | enum('N','Y') | NO |  | N |  |

### 2.2 用grant授予权限

授权格式：

grant 权限列表 on 数据库.表名 to 用户名@登录主机 identified by "密码";

参数解释：

权限列表，是user表各属性名省略\_priv后缀，USAGE表示不授予任何权限只是创建用户，ALL PRIVILEGES 表示授予所有权限。

数据库，表示要为指定数据库创建指定权限的用户，可以为\*，表示所有数据库。

表名，表示要为指定表创建指定权限的用户，可以为\*，表示数据库的所有表。

用户名，要授权或添加的用户。

登录主机，可以为指定的IP地址，或者localhost或者%，要用引号引上。

密码，登录密码。

实例：

　　mysql>grant select,delete,update,create,drop on \*.\* to test@"%" identified by "1234";

### 2.3 使用create创建用户

CREATE USER 'username'@'host' IDENTIFIED BY 'password';

如果希望指定的用户只能从某台指定的域(domain)或主机访问，可以在创建用户时指定host，

### 2.3 删除用户

第一种取消一个账户

Drop USER username@登陆主机

第二种取消授权用户:

REVOKE privilege ON databasename.tablename FROM 'username'@'host';

补充，查看授权

SHOW GRANTS FOR 'username'@'host';

第三种修改用户表：

Delete from user where user = "user\_name" and host = "host\_name" ;

flush privileges;

### 2.4 修改指定用户密码

第一种直接修改user表的用户口令：

update mysql.user set password=password('新密码') where User="test" and Host="localhost";

flush privileges;

第二种使用SET PASSWORD语句修改密码：语法：

SET PASSWORD FOR 'username'@'host' = PASSWORD('newpassword');

如果是当前登陆用户用SET PASSWORD = PASSWORD("newpassword");

第三种使用mysqladmin语法，只用于mysql：

mysqladmin -u用户名 -p旧密码 password 新密码

## 3 /etc/my.cnf 文件配置

一般情况下，你不需要修改该配置文件，该文件默认配置如下：

[mysqld]

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock

[mysql.server]

user=mysql

basedir=/var/lib

[safe\_mysqld]

err-log=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid

在配置文件中，你可以指定不同的错误日志文件存放的目录，一般你不需要改动这些配置。

## 4 管理MySQL的命令

以下列出了使用Mysql数据库过程中常用的命令：

USE 数据库名 :

选择要操作的Mysql数据库，使用该命令后所有Mysql命令都只针对该数据库。

SHOW DATABASES；

列出 MySQL 数据库管理系统的数据库列表。

SHOW TABLES:

显示指定数据库的所有表，使用该命令前需要使用 use 命令来选择要操作的数据库。

SHOW COLUMNS FROM 数据表 //与describe 数据表等效:

显示数据表的属性，属性类型，主键信息 ，是否为 NULL，默认值等其他信息。

SHOW INDEX FROM 数据表:

显示数据表的详细索引信息，包括PRIMARY KEY（主键）。

SHOW TABLE STATUS LIKE [FROM db\_name] [LIKE 'pattern'] \G:

该命令将输出Mysql数据库管理系统的性能及统计信息。\G表示按列打印。

# MySQL 连接

## 1 使用mysql二进制方式连接

您可以使用MySQL二进制方式进入到mysql命令提示符下来连接MySQL数据库。

实例

以下是从命令行中连接mysql服务器的简单实例：

[root@host]# mysql -u 用户名 -p

Enter password:\*\*\*\*\*\*

在登录成功后会出现 mysql> 命令提示窗口，你可以在上面执行任何 SQL 语句。

以上命令执行后，登录成功输出结果如下:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 2854760 to server version: 5.0.9

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

在以上实例中，我们使用了root用户登录到mysql服务器，当然你也可以使用其他mysql用户登录。

如果用户权限足够，任何用户都可以在mysql的命令提示窗口中进行SQL操作。

退出 mysql> 命令提示窗口可以使用 exit 命令，如下所示：

mysql> exit

Bye

## 2 在C/C++中连接mysql

### 2.1 C的连接方法

第一种 使用C API，要求本机安装了mysql-server mysql-client libmysqlclient-dev

第二种 使用C connect 不需安装mysql-server

# MYSQL 默认数据库

## 1 information\_schema数据库

|  |  |
| --- | --- |
| CHARACTER\_SETS | INNODB\_LOCKS |
| COLLATIONS | INNODB\_TRX |
| COLLATION\_CHARACTER\_SET\_APPLICABILITY | INNODB\_SYS\_DATAFILES |
| COLUMNS | INNODB\_FT\_CONFIG |
| COLUMN\_PRIVILEGES | INNODB\_SYS\_VIRTUAL |
| ENGINES | INNODB\_CMP |
| EVENTS | INNODB\_FT\_BEING\_DELETED |
| FILES | INNODB\_CMP\_RESET |
| GLOBAL\_STATUS | INNODB\_CMP\_PER\_INDEX |
| GLOBAL\_VARIABLES | INNODB\_CMPMEM\_RESET |
| KEY\_COLUMN\_USAGE | INNODB\_FT\_DELETED |
| OPTIMIZER\_TRACE | INNODB\_BUFFER\_PAGE\_LRU |
| PARAMETERS | INNODB\_LOCK\_WAITS |
| PARTITIONS | INNODB\_TEMP\_TABLE\_INFO |
| PLUGINS | INNODB\_SYS\_INDEXES |
| PROCESSLIST | INNODB\_SYS\_TABLES |
| PROFILING | INNODB\_SYS\_FIELDS |
| REFERENTIAL\_CONSTRAINTS | INNODB\_CMP\_PER\_INDEX\_RESET |
| ROUTINES | INNODB\_BUFFER\_PAGE |
| SCHEMATA | INNODB\_FT\_DEFAULT\_STOPWORD |
| SCHEMA\_PRIVILEGES | INNODB\_FT\_INDEX\_TABLE |
| SESSION\_STATUS | INNODB\_FT\_INDEX\_CACHE |
| SESSION\_VARIABLES | INNODB\_SYS\_TABLESPACES |
| STATISTICS | INNODB\_METRICS |
| TABLES | INNODB\_SYS\_FOREIGN\_COLS |
| TABLESPACES | INNODB\_CMPMEM |
| TABLE\_CONSTRAINTS | INNODB\_BUFFER\_POOL\_STATS |
| TABLE\_PRIVILEGES | INNODB\_SYS\_COLUMNS |
| TRIGGERS | INNODB\_SYS\_FOREIGN |
| USER\_PRIVILEGES | INNODB\_SYS\_TABLESTATS |
| VIEWS |  |

SCHEMATA表：提供了当前mysql实例中所有数据库的信息。是show databases的结果取之此表。

TABLES表：提供了关于数据库中的表的信息（包括视图）。详细表述了某个表属于哪个schema，表类型，表引擎，创建时间等信息。是show tables from schemaname的结果取之此表。

COLUMNS表：提供了表中的列信息。详细表述了某张表的所有列以及每个列的信息。是show columns from schemaname.tablename的结果取之此表。

STATISTICS表：提供了关于表索引的信息。是show index from schemaname.tablename的结果取之此表。

USER\_PRIVILEGES（用户权限）表：给出了关于全程权限的信息。该信息源自mysql.user授权表。是非标准表。

SCHEMA\_PRIVILEGES（方案权限）表：给出了关于方案（数据库）权限的信息。该信息来自mysql.db授权表。是非标准表。

TABLE\_PRIVILEGES（表权限）表：给出了关于表权限的信息。该信息源自mysql.tables\_priv授权表。是非标准表。

COLUMN\_PRIVILEGES（列权限）表：给出了关于列权限的信息。该信息源自mysql.columns\_priv授权表。是非标准表。

CHARACTER\_SETS（字符集）表：提供了mysql实例可用字符集的信息。是SHOW CHARACTER SET结果集取之此表。

COLLATIONS表：提供了关于各字符集的对照信息。

COLLATION\_CHARACTER\_SET\_APPLICABILITY表：指明了可用于校对的字符集。这些列等效于SHOW COLLATION的前两个显示字段。

TABLE\_CONSTRAINTS表：描述了存在约束的表。以及表的约束类型。

KEY\_COLUMN\_USAGE表：描述了具有约束的键列。

ROUTINES表：提供了关于存储子程序（存储程序和函数）的信息。此时，ROUTINES表不包含自定义函数（UDF）。名为“mysql.proc name”的列指明了对应于INFORMATION\_SCHEMA.ROUTINES表的mysql.proc表列。

VIEWS表：给出了关于数据库中的视图的信息。需要有show views权限，否则无法查看视图信息。

TRIGGERS表：提供了关于触发程序的信息。必须有super权限才能查看该表

## 2 mysql

| columns\_priv |

| db |

| engine\_cost |

| event |

| func |

| general\_log |

| gtid\_executed |

| help\_category |

| help\_keyword |

| help\_relation |

| help\_topic |

| innodb\_index\_stats |

| innodb\_table\_stats |

| ndb\_binlog\_index |

| plugin |

| proc |

| procs\_priv |

| proxies\_priv |

| server\_cost |

| servers |

| slave\_master\_info |

| slave\_relay\_log\_info |

| slave\_worker\_info |

| slow\_log |

| tables\_priv |

| time\_zone |

| time\_zone\_leap\_second |

| time\_zone\_name |

| time\_zone\_transition |

| time\_zone\_transition\_type |

| user |

### 3 performance\_schema

## 4 sys

# SQL DDL

## 1库操作

### 1.1 创建数据库

语法：

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db\_name [create\_specification ,...]

参数create\_specification:

[DEFAULT] CHARACTER SET charset\_name //指定字符集,一般为utf8,用show CHARACTER SET 查看；

[DEFAULT] COLLATE collation\_name//指定数据库的校对集，用show collation查看；

参数SCHEMA是数据库中的命名空间，内部可以放表。

显示数据库语句：

SHOW DATABASES

显示数据库创建时的语句：

SHOW CREATE DATABASE db\_nam

针对mysql还可以使用mysqladmin工具

[root@host]# mysqladmin -u root -p create RUNOOB

Enter password:\*\*\*\*\*\*

以上命令执行成功后会创建 MySQL 数据库 RUNOOB。

### 1.2 删除数据库

语法：

DROP DATABASE [IF EXISTS] db\_name

针对mysql使用 mysqladmin 删除数据库

以下实例删除数据库RUNOOB(该数据库在前一章节已创建)：

[root@host]# mysqladmin -u root -p drop RUNOOB

Enter password:\*\*\*\*\*\*

### 1.3 修改数据库

ALTER DATABASE [IF NOT EXISTS] *db\_name*    [*alter\_specification* [, *alter\_specification*] ...]

*alter\_specification*:

[DEFAULT] CHARACTER SET *charset\_name*

[DEFAULT] COLLATE *collation\_name*

### 1.4 备份数据库

BACKUP DATABASE db\_name TO DISK='D:/Test.bak'

针对mysql，数据库备份： cmd命令下

mysqldump -u root -p test>test.sql

### 1.5 还原数据库

RESTORE DATABASE db\_name FROM DISK='D:/Test.bak'

针对mysql，数据库恢复，

创建数据库并选择该数据库，在cmd命令下：

mysql -u root -p test<test.sql

或者，在mysql >命令行下执行

SOURCE 数据库文件

## 2 表操作

### 2.1 创建表

语法：

CREATE TABLE [IF NOT EXISTS] table\_name (field1 datatype [NOT NULL][AUTO\_INCREMENT], field2 datatype, PRIMARY KEY(filedn),…) [character set 字符集 collate 校对规则];

field：指定列名　datatype：指定列的数据类型

注意：创建表前，要先使用use db语句使用库

注意：创建表时，要根据需保存的数据创建相应列，并根据数据的类型定义0.

相应的列类型。至少有一列。

### 2.2 数据类型

|  |  |  |
| --- | --- | --- |
| 分类 | 数据类型 | 说明 |
| 数值类型 | BIT(M)  BOOL，BOOLEAN  TINYINT [UNSIGNED] [ZEROFILL]  SMALLINT [UNSIGNED] [ZEROFILL]  INT [UNSIGNED] [ZEROFILL]  BIGINT [UNSIGNED] [ZEROFILL]  FLOAT[(*M*,*D*)] [UNSIGNED] [ZEROFILL]  DOUBLE[(*M*,*D*)] [UNSIGNED] [ZEROFILL] | 位类型。M指定位数，默认值1，范围1-64  使用0或1表示假或真  带符号的范围是-128到127。无符号0到255。  2的16次方  2的32次方  2的64次方  M指定显示长度，d指定小数位数 （5, 2）  表示比float精度更大的小数 |
| 文本、二进制类型 | CHAR(size)  VARCHAR(size)  BLOB LONGBLOB  TEXT(clob) LONGTEXT(longclob) | 固定长度字符串（size表示 有多少个字符）  可变长度字符串，varchar（5）表示最大可以存放的长度是5  二进制数据  大文本 |
| 时间日期 | DATE  DATETIME  TimeStamp | 日期类型(YYYY-MM-DD)  YYYY-MM-DD HH:MM:SS)，  TimeStamp表示时间戳，它可用于自动记录操作时间 |

注意：VARCHAR、BLOB和TEXT类是变长类型。每个类型的存储需求取决于列值的实际长度。size不可省略。

### 2.3 修改表

#### 2.3.1 添加列

ALTER TABLE *table\_name* ADD (*columnn datatype* [DEFAULT *expr*] [, *column datatype*]...);

#### 2.3.2 修改列

ALTER TABLE *table\_name* CHANGE *columnold columnnew datatype* [DEFAULT *expr*] [, *column datatype*]...;

#### 2.3.3 删除列

ALTER TABLE *table\_name* DROP (*column*);

#### 2.3.4 其他修改

修改表的名称：

Rename table 表名 to 新表名

修改表的字符集：

ALTER TABLE tablename CHARACTER SET utf8;

### 2.4 删除表

#### 2.5 其他表操作

显示指定数据库的所有表，使用该命令前需要使用 use 命令来选择要操作的数据库。

SHOW TABLES;

显示数据表的属性，属性类型，主键信息 ，是否为 NULL，默认值等其他信息。

SHOW COLUMNS FROM 数据表 //与describe 数据表等效:

显示数据表的详细索引信息，包括PRIMARY KEY（主键）。

SHOW INDEX FROM 数据表;

该命令将输出Mysql数据库管理系统的性能及统计信息。

SHOW TABLE STATUS LIKE [FROM db\_name] [LIKE 'pattern'] \G:

# SQL DML

## 1 插入记录

语法：

INSERT INTO tablename [(column [, column...])] VALUES (value [, value...]);

注意：

（1）插入的数据应与字段的数据类型相同。

（2）数据的大小应在列的规定范围内，例如：不能将一个长度为80的字符串加入到长度为40的列中。

（3）在values中列出的数据 。

（4）字符和日期型数据应包含在单引号中。

（5）插入空值，不指定或insert into table value(null)

注意：中文乱码

mysql有六处使用了字符集，分别为：client 、connection、database、results、server 、system。

client是客户端使用的字符集。

connection是连接数据库的字符集设置类型，如果程序没有指明连接数据库使用的字符集类型就按照服务器端默认的字符集设置。

database是数据库服务器中某个库使用的字符集设定，如果建库时没有指明，将使用服务器安装时指定的字符集设置。

results是数据库给客户端返回时使用的字符集设定，如果没有指明，使用服务器默认的字符集。

server是服务器安装时指定的默认字符集设定。

system是数据库系统使用的字符集设定。

获取以上六处字符集信息：

show variables like 'character%';

修改客户端工具所用的编码

set character\_set\_client=gbk;

客户端显示结果的编码

set character\_set\_results=gbk;

## 2. 更新记录

语法:

UPDATE table\_name  SET col\_name1=expr1 [, col\_name2=expr2 ...]  [WHERE where\_definition]

注意：

（1）UPDATE语法可以用新值更新原有表行中的各列。

（2）SET子句指示要修改哪些列和要给予哪些值。

（3）WHERE子句指定应更新哪些行。如没有WHERE子句，则更新所有的行。

## 3. 删除记录

语法:

delete from table\_name  [WHERE where\_definition] ;

注意：

（1）如果不使用where子句，将删除表中所有数据。

（2）Delete语句不能删除某一列的值（可使用update）

（3）使用delete语句仅删除记录，不删除表本身。如要删除表，使用drop table语句。

（4）同insert和update一样，从一个表中删除记录将引起其它表的参照完整性问题，在修改数据库数据时，头脑中应该始终不要忘记这个潜在的问题。

（5）删除表中数据也可使用TRUNCATE TABLE 语句，它和delete有所不同

# SQL DQL

## 1 基本select语句

语法：

SELECT [DISTINCT] [\* ] |{column1, column2,..} FROM table [WHERE where\_definition];

Select 指定查询哪些列的数据。

column指定列名。

\*号代表查询所有列。

From指定查询哪张表。

DISTINCT可选，指显示结果时，是否剔除重复数据

WHERE 指定查询哪些行，如果没有则查询所有行。

## 2 select 对查询结果处理

语法：

SELECT \*|{column1|expression，..} FROM 表名 [WHERE where\_definition];

## 3. select将查询结果另存为一个表

语法：

SELECT \*|{column1|expression，..} as 别名 from 表名 [WHERE where\_definition];

## 4. select 将查询结果排序

语法：

SELECT \*|{column1, column2. column3..} FROM 表名order by column asc|desc

注意：

（1）Order by 指定排序的列，排序的列即可是表中的列名，也可以是select 语句后指定的列名。

（2）Asc 升序、Desc 降序

（3）ORDER BY 子句应位于SELECT语句的结尾。

## 5.where 定义过滤条件

|  |  |  |
| --- | --- | --- |
| 比较运算符 | > < <= >= = <> | 大于、小于、大于(小于)等于、不等于 |
| **BETWEEN ...AND...** | 显示在某一区间的值(含头含尾) |
| **IN(set)** | 显示在in列表中的值，例：in(100,200) |
| **LIKE ‘张pattern’** | 模糊查询，Like语句中，% 代表零个或多个任意字符，\_ 代表一个字符; |
| **IS NULL** | 判断是否为空 |
| 逻辑运算符 | and | 多个条件同时成立 |
| or | 多个条件任一成立 |
| not | 不成立，例：where not(salary>100); |

注意：NULL需要占用空间。

## 6 查询分类

（1）连接查询：（2）嵌套查询（3）联合查询（4）报表查询

### 6.1 连接查询

连接查询的from子句的连接语法格式为：

Select \* from TABLE1 join\_type TABLE2 [on (join\_condition)] [where (query\_condition)]

参数说明：

TABLE1和TABLE2表示参与连接操作的表，TABLE1为左表，TABLE2为右表。

on子句设定连接条件，

where子句设定查询条件，

join\_type表示连接类型。cross join、inner join、left out join、right outer join。

#### 6.1.1 交叉连接查询CUSTOMER表和ORDERS表

交叉连接（cross join）:不带on子句，返回连接表中所有数据行的笛卡儿积。

SELECT \* FROM customer CROSS JOIN orders;

SELECT \* FROM customer,orders;

#### 6.1.2 内连接查询

内连接（inner join）：返回连接表中符合连接条件及查询条件的数据行。分为

显式内连接：使用inner join关键字，在on子句中设定连接条件

SELECT \* FROM customer as c INNER JOIN orders as o ON c.id=o.customer\_id;

隐式内连接：不包含inner join关键字和on关键字，在where子句中设定连接条件

SELECT \* FROM customer c,orders o WHERE c.id=o.customer\_id;

内连接查询操作列出与连接条件匹配的数据行，它使用比较运算符比较被连接列的列值。内连接分三种：

（1）等值连接：在连接条件中使用等于号(=)运算符比较被连接列的列值，其查询结果中列出被连接表中的所有列，包括其中的重复列。例，下面使用等值连接列出authors和publishers表中位于同一城市的作者和出版社：

SELECT \* FROM authors AS a INNER JOIN publishers AS p ON a.city=p.city

（2）不等连接： 在连接条件使用除等于运算符以外的其它比较运算符比较被连接的列的列值。这些运算符包括>、>=、<=、<、!>、!<和<>。

（3）自然连接：在连接条件中使用等于(=)运算符比较被连接列的列值，但它使用选择列表指出查询结果集合中所包括的列，并删除连接表中的重复列。

#### 6.1.3 左外连接查询

使用left outer join关键字，在on子句中设定连接条件

SELECT \* FROM customer c LEFT OUTER JOIN orders o ON c.id=o.customer\_id;

不仅包含符合c.id=o.customer\_id连接条件的数据行，还包含左表中的其他数据行

带查询条件的左外连接查询，在where子句中设定查询条件

SELECT \* FROM customer c LEFT OUTER JOIN orders o ON c.id=o.customer\_id WHERE o.price>250;

#### 6.1.4 右外连接查询

使用right outer join关键字，在on子句中设定连接条件

SELECT \* FROM customer c RIGHT OUTER JOIN orders o ON c.id=o.customer\_id;

不仅包含符合c.id=o.customer\_id连接条件的数据行，还包含orders右表中的其他数据行

带查询条件的右外连接查询，在where子句中设定查询条件

SELECT \* FROM customer c RIGHT OUTER JOIN orders o ON c.id=o.customer\_id WHERE o.price>250;

### 6.2 嵌套查询

嵌套查询，是指在where子句或from子句中又嵌入select查询语句（一般写在where字句）

查询“郭靖”的所有订单信息

SELECT \* FROM orders WHERE customer\_id=(SELECT id FROM customer WHERE name LIKE‘%郭靖%');

### 6.3 联合查询

联合查询能够合并两条查询语句的查询结果，去掉其中的重复数据行，然后返回没有重复数据行的查询结果。联合查询使用union关键字

SELECT \* FROM orders WHERE price>100 UNION SELECT \* FROM orders WHERE customer\_id=1;

注意：联合查询的各子查询使用的表结构应该相同，同时两个子查询返回的列也应相同。

### 6.4 报表查询

报表查询对数据行进行分组统计，其语法格式为：

select … from … [where…] [ group by … [having… ]] [ order by … ]

其中group by 子句指定按照哪些字段分组，having子句设定分组查询条件。

在报表查询中可以使用SQL函数。

前几行 select \* from pats limit 0,3;前三行

#### 6.4.1 使用group by对列进行分组

SELECT column1, column2. column3.. FROM table group by column

#### 6.4.2 使用having 子句过滤

SELECT column1, column2. column3.. FROM table group by column having

Having和where均可实现过滤，但having可以使用合计函数,having常跟在group by后，它作用于组。

## 7 常用统计函数

### 7.1 count

Count(列名)返回某一列，行的总数（有多少条记录）

Select count(\*) | count(列名) from tablename [WHERE where\_definition]

### 7.2 SUM

Sum函数返回满足where条件的行的和

Select sum(列名)｛,sum(列名)…｝ from tablename [WHERE where\_definition]

 注意：sum仅对数值起作用，否则会报错。

注意：对多列求和，“，”号不能少。

### 7.3 AVG

AVG函数返回满足where条件的一列的平均值

Select sum(列名)｛,sum(列名)…｝ from tablename [WHERE where\_definition]

### 7.4 Max/min

Max/min函数返回满足where条件的一列的最大/最小值

Select max(列名)　from tablename [WHERE where\_definition]

# 数据完整性

数据完整性是为了保证插入到数据中的数据是正确的，它防止了用户可能的输入错误。

## 1 数据完整性分类

数据完整性主要分为以下三类：

（1）实体完整性：

规定表的一行（即每一条记录）在表中是唯一的实体。实体完整性通过表的主键来实现。

（2）域完整性：

指数据库表的列（即字段）必须符合某种特定的数据类型或约束。比如NOT NULL。

（3）参照完整性：

保证一个表的外键和另一个表的主键对应。

## 2 主键、外键、索引的区别

|  |  |  |  |
| --- | --- | --- | --- |
|  | 主键 | 外键 | 索引 |
| 定义 | 唯一标识一条记录，不能有重复的，不允许为空 | 表的外键是另一表的主键, 外键可以有重复的, 可以是空值 | 该字段没有重复值，但可以有一个空值 |
| 作用 | 用来保证数据完整性 | 用来和其他表建立联系用的 | 是提高查询排序的速度 |
| 个数 | 主键只能有一个 | 一个表可以有多个外键 | 一个表可以有多个惟一索引 |

数据库主键主键：表中经常有一个列或列的组合，其值能唯一地标识表中的每一行。这样的一列或多列称为表的主键，通过它可强制表的实体完整性。当创建或更改表时可通过定义 PRIMARY KEY 约束来创建主键。一个表只能有一个 PRIMARY KEY 约束，而且 PRIMARY KEY 约束中的列不能接受空值。由于 PRIMARY KEY 约束确保唯一数据，所以经常用来定义标识列。 作用 :

1）保证实体的完整性;

2）加快数据库的操作速度

3）在表中添加新记录时，会自动检查新记录的主键值，不允许该值与其他记录的主键值重复。

4) 自动按主键值的顺序显示表中的记录。如果没有定义主键，则按输入记录的顺序显示表中的记录。

## 3 定义表的约束

定义约束

ALTER TABLE table\_name ADD CONSTRAINT constraint\_name constraint\_type constraint\_definition

参数解释

（1）table\_name 要修改的表的名称

（2）constraint\_name 约束的名称

（3）constraint\_type 约束的类型，有以下四种：

PRIMARY KEY 主键约束

FOREIGN KEY 外键约束

CHECK 检查约束

UNIQUE 唯一约束

（4）constraint\_defintion 约束的定义，不同的约束种类约束的定义是不一样的，每种约束的定义如下：

PRIMARY KEY (field\_list)// 定义主键约束，不允许为空，不允许重复

FOREIGN KEY (source\_field\_name) REFERENCES target\_table(target\_field\_name) //外键约束

CHECK (condition) //检查约束

UNIQUE (field\_name)//唯一约束

定义主键自动增长

auto\_increment

定义非空约束

not null

删除主键

alter table tablename drop primary key ;

# 多表设计

一对多

多对多

一对一

避免数据的冗余