

day01-基础

为什么需要数据库？

学习目标

- ☐ 能够理解数据库的概念
- ☐ 能够安装MySQL数据库
- ☐ 能够启动、关闭及登录MySQL
- ☐ 能够使用SQL语句操作数据库：创建、删除、修改、更新
- ☐ 能够使用SQL语句操作表结构
- ☐ 能够使用SQL语句进行数据的添加、修改、删除的操作
- ☐ 能够使用SQL语句添加约束

第1章 数据库介绍

1.1 数据库概述

什么是数据库？

数据库就是存储数据的仓库，其本质是一个文件系统，数据按照特定的格式将数据存储起来，用户可以对数据库中的数据进行增加，修改，删除及查询操作。

数据库分两大类：

- 关系型数据库
- 非关系型NoSQL

常用数据库排行榜：

Rank			DBMS	Database Model	Score		
Oct 2019	Sep 2019	Oct 2018			Oct 2019	Sep 2019	Oct 2018
1.	1.	1.	Oracle +	Relational, Multi-model i	1355.88	+9.22	+36.61
2.	2.	2.	MySQL +	Relational, Multi-model i	1283.06	+3.99	+104.94
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model i	1094.72	+9.66	+36.39
4.	4.	4.	PostgreSQL +	Relational, Multi-model i	483.91	+1.66	+64.52
5.	5.	5.	MongoDB +	Document, Multi-model i	412.09	+2.03	+48.90
6.	6.	6.	IBM Db2 +	Relational, Multi-model i	170.77	-0.79	-8.91
7.	7.	8.	Elasticsearch +	Search engine, Multi-model i	150.17	+0.90	+7.85
8.	8.	7.	Redis +	Key-value, Multi-model i	142.91	+1.01	-2.38
9.	9.	9.	Microsoft Access	Relational	131.18	-1.53	-5.62
10.	10.	10.	Cassandra +	Wide column	123.22	-0.18	-0.17

什么是关系型数据库

数据库中的【记录是有行有列的数据库】就是关系型数据库（RDBMS, Relational Database Management System）与之相反的就是 **NoSQL** 数据库了。

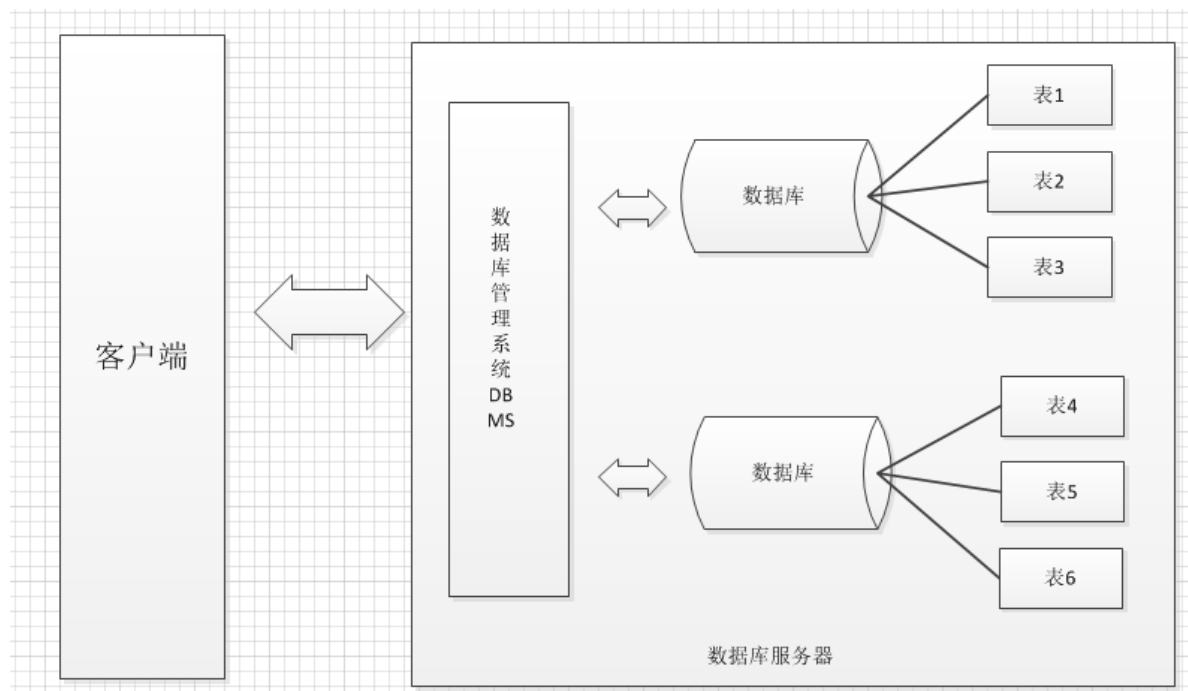
什么是数据库管理系统？

- 广义数据库：泛指数据库管理系统RDBMS
- 狭义数据库：真正存储数据的地方

数据库管理系统（DataBase Management System，DBMS）：指一种操作和管理数据库的大型软件，用于建立、使用和维护数据库，对数据库进行统一管理和控制，以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中表内的数据。

一个 DBMS 可以管理多个 数据库，我们建议每个项目系统，对应一个数据库，避免数据混乱。然后可以在数据库中，根据具体操作数据对象，对应创建多个表。比如，商城管理系统中，有商品表、订单表、用户表等等。

数据库与数据库管理系统的关系？



1.2 数据库表

数据库中以表为组织单位存储数据。表中的每个字段都有对应的数据类型。

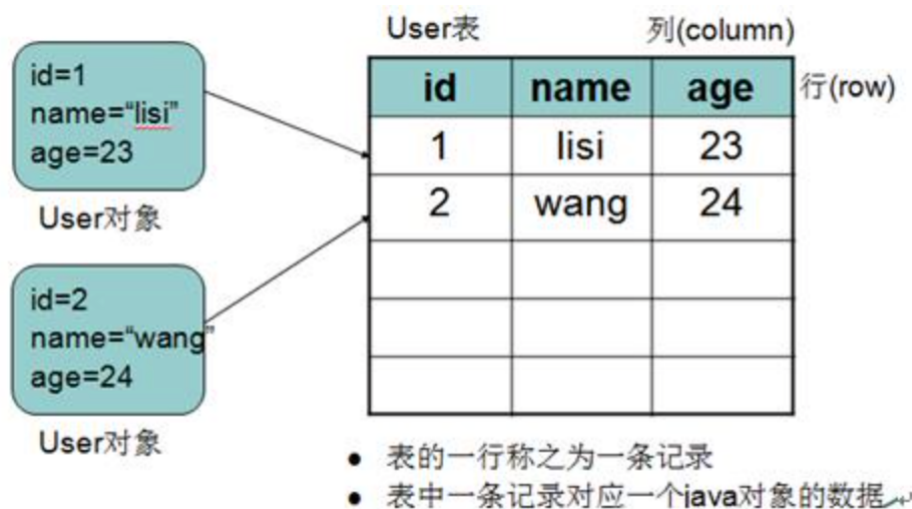
编号	名称	密码	年龄
u001	jack	1234	18
u002	rose	5678	21

字段（具有类型） 记录(行)

数据库表中，每一列数据存储的是固定的数据类型的数据；

1.3 表数据

表中的一行一行的信息我们称之为记录。根据表字段所规定的数据类型，向其中填入一条条数据。



1.4 常见数据库管理系统

1. **MYSQL**: 开源免费的数据库，小型的数据库。已经被Oracle收购了，MySQL6.x版本也开始收费。
2. **Oracle**: 收费的大型数据库，Oracle公司的产品。Oracle收购SUN公司，收购MYSQL。
3. **DB2**: IBM公司的数据库产品，收费的。常应用在银行系统中。
4. **SQLServer**: MicroSoft 公司收费的中型的数据库。C#、.net等语言常使用。
5. **SyBase**: 已经淡出历史舞台。提供了一个非常专业数据建模的工具PowerDesigner。
6. **SQLite**: 嵌入式的小型数据库，应用在手机端。

常用数据库: **MYSQL, Oracle**。

这里使用MySQL数据库。MySQL中可以有多个数据库，数据库中的表是真正存储数据的地方。

第2章 MySql数据库

2.1 什么是MySQL?

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 应用软件之一。

发展历程:

MySQL 的历史可以追溯到 1979 年，一个名为 Monty Widenius 的程序员在为TcX的小公司打工，并且用 BASIC 设计了一个报表工具，使其可以在 4MHz 主频和 16KB内存的计算机上运行。当时，这只是一个很底层的且仅面向报表的存储引擎，名叫Unireg。

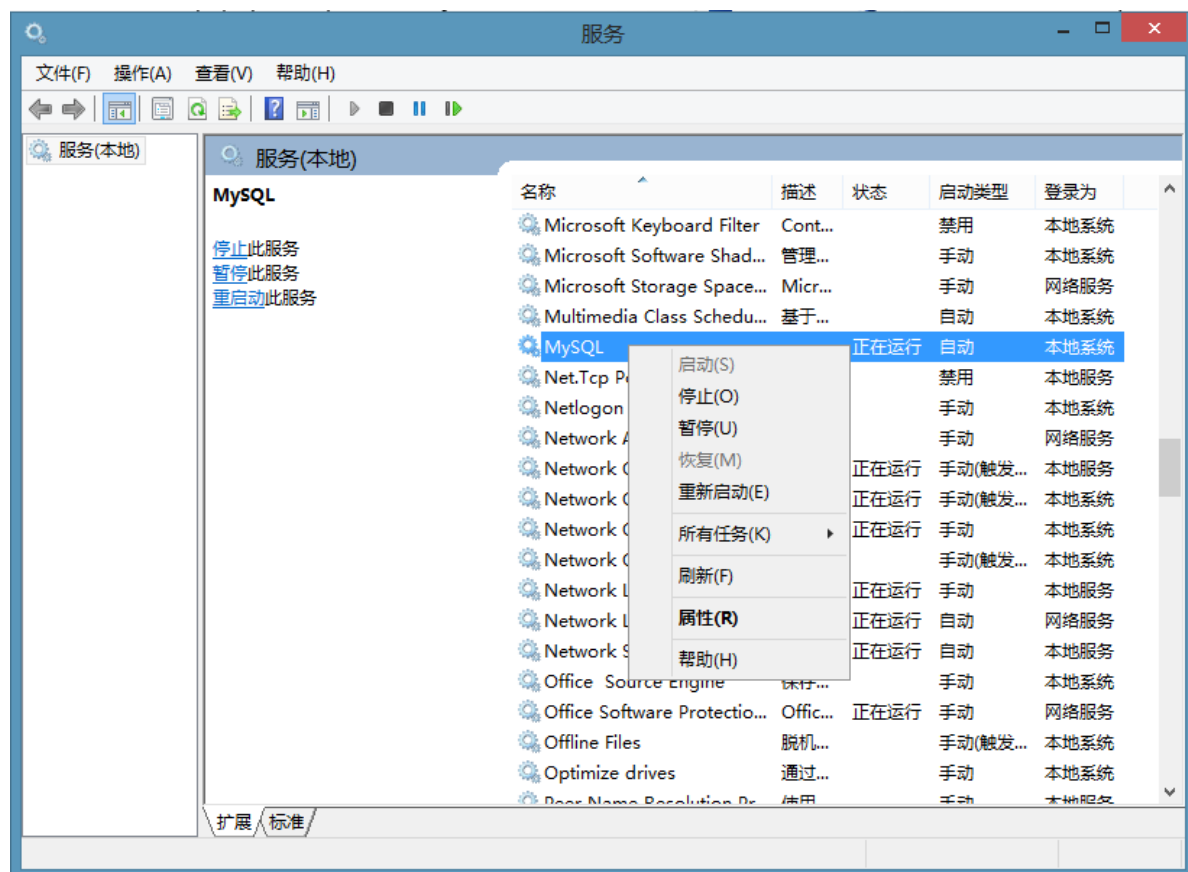
- 1990年: TcX 公司的客户中开始有人要求为他的 API 提供 SQL 支持。Monty 直接借助于 mSQL 的代码，将它集成到自己的存储引擎中。效果并不太令人满意，决心自己重写一个 SQL 支持。
- 1996年:
 - MySQL 1.0 发布，它只面向一小拨人，相当于内部发布。
 - 1996年10月，MySQL 3.11.1发布（MySQL 没有 2.x 版本），最开始只提供Solaris下的二进制版本。一个月后，Linux 版本出现了。在接下来的两年里，MySQL 被依次移植到各个平台。

- 1999年：【MySQL AB】公司在瑞典成立。Monty 雇了几个人与Sleepycat 合作，开发出了【Berkeley DB引擎】，由于 BDB 支持事务处理，因此MySQL 从此开始支持事务处理了。
- 2000年：MySQL 不仅公布自己的源代码，并采用 GPL（GNU General Public License）许可协议，正式进入开源世界。
 - 同年 4 月，MySQL 对旧的存储引擎 ISAM进行了整理，将其命名为 MyISAM。
- 2001年：集成 Heikki Tuuri 的存储引擎【InnoDB】，这个引擎不仅能【支持事务处理，并且支持行级锁】。后来该引擎被证明是最为成功的 MySQL 事务存储引擎。【MySQL与InnoDB的正式结合版本是4.0】
- 2003年：【MySQL 5.0】版本发布，提供了视图、存储过程等功能。
- 2008年：
 - 【MySQL AB 公司被 Sun 公司以 10 亿美金收购】，MySQL 数据库进入 Sun 时代。在 Sun 时代，Sun 公司对其进行了大量的推广、优化、Bug 修复等工作。
 - MySQL 5.1发布，它提供了分区、事件管理，以及基于行的复制和基于磁盘的 NDB 集群系统，同时修复了大量的 Bug。
- 2009年：Oracle 公司以74亿美元收购Sun公司，自此 MySQL 数据库进入Oracle 时代，而其第三方的存储引擎 InnoDB 早在 2005 年就被 Oracle 公司收购。
- 2010年：【MySQL 5.5发布】，其主要新特性包括半同步的复制及对 SIGNAL/ RESIGNAL 的异常处理功能的支持，【最重要的是 InnoDB 存储引擎终于变为当前MySQL的默认存储引擎】。
 - MySQL 5.5 不是时隔两年后的一次简单的版本更新，而是加强了 MySQL 各个方面在企业级的特性。
 - Oracle 公司同时也承诺 MySQL 5.5 和未来版本仍是采用 GPL 授权的开源产品。

2.2 安装

安装：参考MySQL安装图解.pdf

安装后，MySQL会以windows服务的方式为我们提供数据存储功能。开启和关闭服务的操作：右键点击我的电脑→管理→服务→可以找到MySQL服务开启或停止。



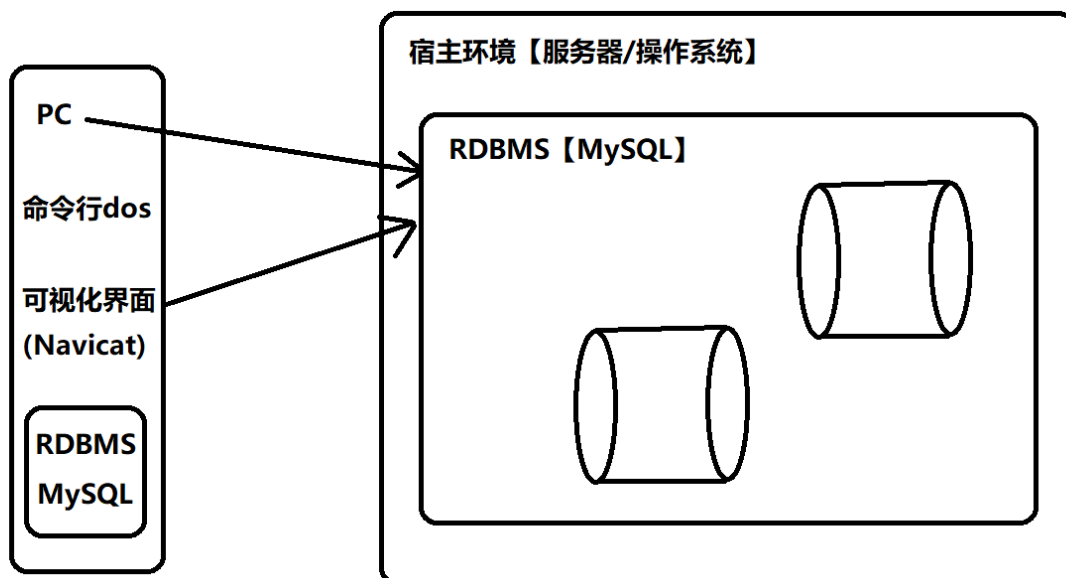
也可以在DOS窗口，通过命令完成MySQL服务的启动和停止（必须以管理员身份运行cmd命令窗口）

```
管理员: C:\Windows\System32\cmd.exe
C:\Windows\system32>net start mysql
MySQL 服务正在启动 ..
MySQL 服务已经启动成功。

C:\Windows\system32>net stop mysql
MySQL 服务正在停止..
MySQL 服务已成功停止。
```

- 1 开启mysql服务:net start mysql
- 2 关闭mysql服务:net stop mysql

2.3 登录



MySQL是一个需要账户名密码登录的数据库，登陆后使用，它提供了一个默认的root账号，使用安装时设置的密码即可登录。

- 1 格式1: cmd> mysql -u用户名 -p密码
- 2 例如: mysql -uroot -proot
- 3
- 4 格式2: mysql -u用户名 -p
- 5 请输入密码: root

- 1 格式2: cmd> mysql --host=ip地址 --user=用户名 --password=密码
- 2 例如: mysql --host=127.0.0.1 --user=root --password=root

第3章 SQL语句

3.1 SQL概述

SQL语句介绍

结构化查询语言(Structured Query Language)简称SQL，是关系型数据库管理系统都需要遵循的规范。不同的数据库生产厂商都支持SQL语句，但都有特有内容。

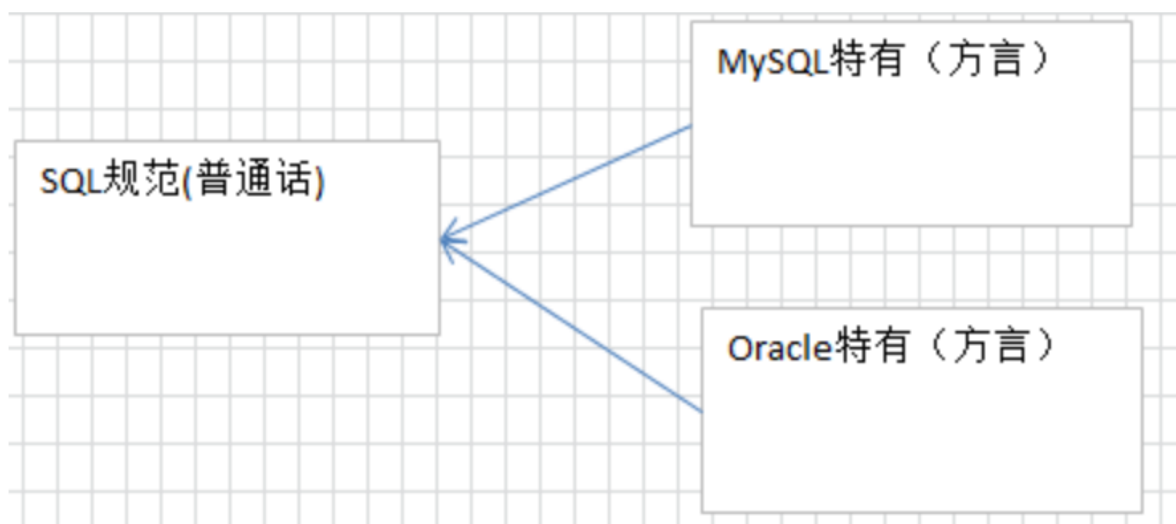
扩展：SQL作为一种访问【关系型数据库的标准语言】，SQL自问世以来得到了广泛的应用，不仅是著名的大型商用数据库产品 Oracle、DB2、Sybase、SQL Server 支持它，很多开源的数据库产品如 PostgreSQL、MySQL也支持它，甚至一些小型的产品如 Access 也支持 SQL。近些年蓬勃发展的 NoSQL 系统最初是宣称不再需要 SQL 的，后来也不得不修正为 Not Only SQL，来拥抱 SQL。

蓝色巨人 IBM 对关系数据库以及 SQL 语言的形成和规范化产生了重大的影响，第一个版本的 SQL 标准 SQL86 就是基于 System R 的手册而来的。

Oracle 在 1979 年率先推出了支持 SQL 的商用产品。随着数据库技术和应用的发展，为不同 RDBMS提供一致的语言成了一种现实需要。

对 SQL 标准影响最大的机构自然是那些著名的数据库产商，而具体的制订者则是一些非营利机构，例如【国际标准化组织 ISO、美国国家标准委员会 ANSI】等。

各国通常会按照 ISO 标准和 ANSI 标准（这两个机构的很多标准是差不多等同的）制定自己的国家标准。



SQL作用

- 在数据库中检索信息。
- 对数据库的信息进行更新。
- 改变数据库的结构。
- 更改系统的安全设置。
- 增加或回收用户对数据库、表的许可权限。

SQL语句分类

- **数据定义语言：简称DDL(Data Definition Language)**
 - 作用：用来定义数据库对象：数据库，表，列等。
 - 关键字：create, alter, drop等
- **数据操作语言：简称DML(Data Manipulation Language),**
 - 作用：用来对数据库中表的记录进行更新。
 - 关键字：insert, delete, update等

- **数据查询语言：简称DQL(Data Query Language),**
 - 作用：用来查询数据库中表的记录。
 - 关键字：select, from, where等
- **数据控制语言：简称DCL(Data Control Language),**
 - 作用：用来定义数据库的访问权限和安全级别，及创建用户。

SQL通用语法

```
1  -- 创建用户表
2  create table user (
3      uid int primary key auto_increment,    -- 用户 id
4      uname varchar(20),                    -- 用户名
5  );
```

- SQL语句可以单行或多行书写，以分号结尾
- 可使用空格和缩进来增强语句的可读性
- MySQL数据库的SQL语句不区分大小写，关键字建议使用大写
 - 例如：SELECT * FROM user。
- 常用注释形式

```
1  -- 单行注释内容
2  /* 多行注释 */
3  # 单行注释内容
```

- MySQL中的我们常使用的数据类型如下

类型名称	说明
int (integer)	整数类型
double	小数类型
decimal (m,d)	指定整数位与小数位长度的小数类型 decimal(10,2)
date	日期类型，格式为yyyy-MM-dd，包含年月日，不包含时分秒 2019-05-06
datetime	日期类型，格式为 YYYY-MM-DD HH:MM:SS，包含年月日时分秒 2019-05-06 09:49:30
timestamp	日期类型，时间戳
varchar (M)	文本类型， M为0~65535之间的整数

3.2 DDL之数据库操作：database

创建、查看、删除、修改...

1) 创建数据库

```
1 create database 数据库名;  
2 create database 数据库名 character set 字符集;  
3  
4 -- 【案例】  
5 -- 1. 创建一个叫 hello 的数据库  
6 create database hello;  
7 -- 2. 如果不存在则创建  
8 create database hello if not exists hello;  
9 -- 3. 创建数据库并指定字符集  
10 create database hello default character set gbk;
```

2) 查看数据库

```
1 查看数据库服务器中的所有的数据库: show databases;  
2 查看某个数据库的定义的信息: show create database 数据库名;  
3  
4 -- 【案例】  
5 -- 查看数据库服务器中的所有的数据库  
6 show databases;  
7 -- 查看某个数据库的定义信息  
8 show create database hello;
```

3) 删除数据库 (慎用)

```
1 drop database 数据库名称;  
2  
3 -- 【案例】  
4 drop database hello;
```

4) 修改数据库

```
1 修改数据库默认的字符集:  
2 alter database 数据库名 default character set 字符集  
3  
4 -- 【案例】  
5 alter database hello3 character set utf8;
```

5) 其他数据库操作命令

```
1 切换数据库: use 数据库名;  
2 查看正在使用的数据库: select database();
```

3.3 DDL之表操作: table

- 创建数据库表
- 查看表
- 快速创建: 克隆
- 删除表

- 修改表结构：如果表内有数据，修改需要谨慎
 - 字段更改
 - 字段删除
 - 修改表名称

1) 创建表

建立数据表，就是建立表结构，指定数据表中的一共有多少列，每一列的数据类型

```

1  -- create 指的是【创建】，table 指的是【数据表】。
2  -- 一张表中可以指定多个字段，用逗号隔开，最后的字段不需要逗号
3  create table 表名(
4      -- 可以定义多个列
5      字段名 类型(长度) 约束,
6      字段名 类型(长度) 约束
7  );
8
9  -- 【案例】
10 -- 创建用户表
11 create table t_user (
12     uid int primary key auto_increment,      -- 用户 id
13     uname varchar(20),                       -- 用户名
14 );

```

字段类型

常用的类型有：

- ① 数字型：int、integer、bigint、mediumint、smallint、tinyint
- ② 浮点型：double、float、decimal (精确小数类型)
- ③ 字符型：char (定长字符串)、varchar (可变长字符串)
- ④ 日期类型：date (只有年月日)、time (只有时分秒)、datetime (年月日, 时分秒)、year (年)
- ⑤ 二进制字符串类型：binary (定长, 以二进制形式保存字符串)、varbinary (可边长)

单表约束

```

1  主键约束: primary key
2  唯一约束: unique
3  非空约束: not null

```

注意

```

1  主键约束 = 唯一约束 + 非空约束

```

2) 查看表

```

1  查看数据库中的所有表: show tables;
2  查看表结构: desc 表名;
3  查看创建表的 SQL 语句: show create table 表名;
4
5  -- 【案例】
6  use hello;          -- 使用 hello 数据库
7  show tables;        -- 查看所有表
8  desc user;          -- 查看 user 表的结构
9  show create table user; -- 查看 user 表的创建语句

```

3) 快速创建一个表结构相同的表

```

1  create table 新的表名 like 旧的表名;
2
3  -- 【案例】
4  create table tb_user like user;
5  desc tb_user;

```

4) 删除表

```

1  drop table 表名;
2  drop table if exists 表名;
3
4  -- 【案例】
5  -- 删除用户表
6  drop table user;

```

5) 修改表:

```

1  -- 1. 修改表添加列
2  alter table 表名 add 列名 类型(长度) 约束;
3  -- 2. 修改表修改列的类型长度及约束
4  alter table 表名 modify 列名 类型(长度) 约束;
5  -- 3. 修改表修改列名
6  alter table 表名 change 旧列名 新列名 类型(长度) 约束;
7  -- 4. 修改表删除列
8  alter table 表名 drop 列名;
9  -- 5. 修改表名
10 rename table 表名 to 新表名;
11
12 -- 【案例】
13 -- 修改表添加列
14 alter table user add address varchar(50);
15 -- 修改表修改列的类型长度及约束
16 alter table user modify address int(30);
17 -- 修改表修改列名
18 alter table user change address addr varchar(50);
19 -- 修改表删除列
20 alter table user drop addr;
21 -- 修改表名
22 rename table user to tb_user;

```

3.4 DML数据操作语言

1) 插入记录: insert

语法:

```
1  -- 1.向表中插入某些列
2  insert into 表 (列名1,列名2,列名3..) values (值1,值2,值3..);
3  -- 2.向表中插入所有列
4  insert into 表 values (值1,值2,值3..);
5
6  -- 3.从另外一张表查某些列的结果插入当前表
7  insert into 表 (列名1, 列名2, 列名3..) values select (列名1,列名2,列名3..) from
   表
8  -- 4.从另外一张表查所有列的结果插入当前表
9  insert into 表 values select * from 表
10
11 -- 【案例】
12 -- 向表中插入某些列, 必须写列名
13 insert into user (uid, uname) values (001, 'cuihua');
14 -- 向表中插入所有列
15 insert into user values (002, 'aqiang');
```

5个注意事项:

- ① 列名数与 values 后面的值的个数相等
- ② 列的顺序与插入的值得顺序一致
- ③ 列名的类型与插入的值要一致.
- ④ 插入值得时候不能超过最大长度.
- ⑤ 值如果是字符串或者日期需要加引号” (一般是单引号)

2) 更新记录: update

语法格式: update 更新、set 修改的列值、where 指定条件。

```
1  -- 1.不指定条件, 会修改表中当前列所有数据
2  update 表名 set 字段名=值, 字段名=值;
3  -- 2.指定条件, 符合条件的才会修改
4  update 表名 set 字段名=值, 字段名=值 where 条件;
5
6  -- 【案例】
7  -- 更新所有字段的值
8  update user set uname='xiaodong';
9  -- 根据指定的条件来更新
10 update user set uname='hashiqi' where uid = 2;
```

注意:

- ① 列名的类型与修改的值要一致
- ② 修改值得时候不能超过最大长度
- ③ 值如果是字符串或者日期需要加'' 引号

3) 删除记录: delete & truncate

语法格式:

```
1 delete from 表名 [where 条件];
2
3 -- 【案例】
4 -- 删除表中所有数据
5 delete from user;
6 -- 删除 uid 为 1 的用户
7 delete from user where uid = 1;
```

```
1 truncate table 表名;
```

注意

删除表中所有记录使用【delete from 表名】, 还是用【truncate table 表名】?

删除方式的区别:

```
1 delete : 一条一条删除, 不清空 auto_increment 记录数。
2 truncate : 直接将表删除, 重新建表, auto_increment 将置为零, 从新开始
```

第4章 SQL约束

约束类型:

- 主键约束 primary key
- 唯一性约束 unique
- 非空约束 not null
- 外键约束 foreign key

4.1 主键约束

PRIMARY KEY 约束唯一标识数据库表中的每条记录。

特点:

- 主键必须包含唯一的值。
- 主键列不能包含 NULL 值。
- 每个表都应该有一个主键, 并且每个表只能有一个主键。

添加主键约束

- 方式一: 创建表时, 在字段描述处, 声明指定字段为主键:

```
1 CREATE TABLE persons(
2     id_p int PRIMARY KEY,
3     lastname varchar(255),
4     firstname varchar(255),
5     address varchar(255),
6     city varchar(255)
7 );
```

- 方式二: 创建表时, 在constraint约束区域, 声明指定字段为主键:

- 格式: `[constraint 名称] primary key (字段列表)`
- 关键字constraint可以省略, 如果需要为主键命名, constraint不能省略, 主键名称一般没用。
- 字段列表需要使用小括号括住, 如果有多字段需要使用逗号分隔。声明两个以上字段为主键, 我们称为联合主键。

```
1 CREATE TABLE persons_cons(
2     firstname varchar(255),
3     lastname varchar(255),
4     address varchar(255),
5     city varchar(255),
6     CONSTRAINT pk_personID PRIMARY KEY (firstname,lastname)
7 );
```

- 方式三: 创建表之后, 通过修改表结构, 声明指定字段为主键:

- 格式: `ALTER TABLE persons ADD [CONSTRAINT 名称] PRIMARY KEY (字段列表)`

```
1 CREATE TABLE persons_after(
2     firstname varchar(255),
3     lastname varchar(255),
4     address varchar(255),
5     city varchar(255)
6 );
7 ALTER TABLE persons_after ADD PRIMARY KEY (firstname,lastname);
```

删除主键约束

如需撤销 PRIMARY KEY 约束, 请使用下面的 SQL:

```
1 ALTER TABLE persons DROP PRIMARY KEY
```

4.2 自动增长列

我们通常希望在每次插入新记录时, 数据库自动生成字段的值。

我们可以在表中使用 auto_increment (自动增长列) 关键字, 自动增长列类型必须是整形, 自动增长列必须为键(一般是主键)。

- 下列 SQL 语句把 "persons" 表中的 "p_id" 列定义为 auto_increment 主键

```
1 CREATE TABLE persons_id(
2     p_id int PRIMARY KEY AUTO_INCREMENT,
3     lastname varchar(255),
4     firstname varchar(255),
5     address varchar(255),
6     city varchar(255)
7 );
```

- 向persons添加数据时, 可以不为p_id字段设置值, 也可以设置成null, 数据库将自动维护主键值:

```
1 INSERT INTO persons_id (firstname,lastname) VALUES ('Bill','Gates');
2 INSERT INTO persons_id (p_id,firstname,lastname) VALUES
  (NULL,'Bill','Gates');
```

- 默认AUTO_INCREMENT 的开始值是 1，如果希望修改起始值，请使用下列 SQL 语法：

```
1 ALTER TABLE persons AUTO_INCREMENT=100
```

4.3 非空约束

NOT NULL 约束强制列不接受 NULL 值。

NOT NULL 约束强制字段始终包含值。这意味着，如果不向字段添加值，就无法插入新记录或者更新记录。

- 下面的 SQL 语句强制 "id_p" 列和 "lastname" 列不接受 NULL 值：

```
1 CREATE TABLE persons_null(
2   id_p int NOT NULL,
3   lastname varchar(255) NOT NULL,
4   firstname varchar(255),
5   address varchar(255),
6   city varchar(255)
7 );
```

4.4 唯一约束

- UNIQUE 约束唯一标识数据库表中的每条记录。
- UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。
- PRIMARY KEY 拥有自动定义的 UNIQUE 约束。

注意：每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束。

添加唯一约束

与主键添加方式相同，共有3种

- 方式一：创建表时，在字段描述处，声明唯一：

```
1 CREATE TABLE persons(
2   id_p int UNIQUE,
3   lastname varchar(255) NOT NULL,
4   firstname varchar(255),
5   address varchar(255),
6   city varchar(255)
7 );
```

- 方式二：创建表时，在约束区域，声明唯一：

```

1 CREATE TABLE persons(
2     id_p int,
3     lastname varchar(255) NOT NULL,
4     firstname varchar(255),
5     address varchar(255),
6     city varchar(255),
7     CONSTRAINT unique_id_p UNIQUE (Id_P)
8 )

```

- 方式三：创建表后，修改表结构，声明字段唯一：

```

1 ALTER TABLE persons ADD [CONSTRAINT 名称] UNIQUE (Id_P)

```

删除唯一约束

- 如需撤销 UNIQUE 约束，请使用下面的 SQL：

```

1 ALTER TABLE persons DROP INDEX 名称

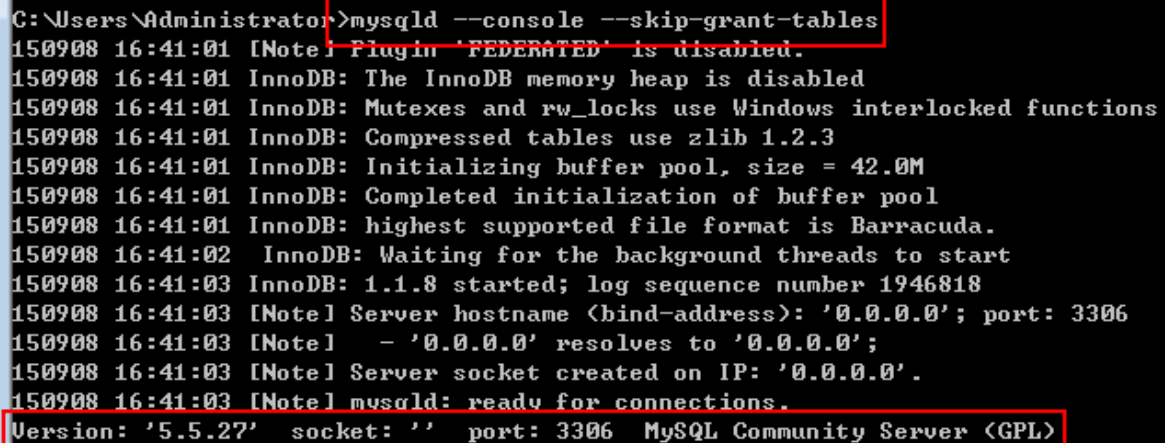
```

- 如果添加唯一约束时，没有设置约束名称，默认是当前字段的字段名。

常见问题解答

01-MySQL数据库密码重置

1. 停止mysql服务器运行输入services.msc 停止mysql服务
2. 在cmd下,输入mysql --console --skip-grant-tables 启动服务器,出现一下页面,不要关闭该窗口



```

C:\Users\Administrator>mysql --console --skip-grant-tables
150908 16:41:01 [Note] Plugin 'FEDERATED' is disabled.
150908 16:41:01 InnoDB: The InnoDB memory heap is disabled
150908 16:41:01 InnoDB: Mutexes and rw_locks use Windows interlocked functions
150908 16:41:01 InnoDB: Compressed tables use zlib 1.2.3
150908 16:41:01 InnoDB: Initializing buffer pool, size = 42.0M
150908 16:41:01 InnoDB: Completed initialization of buffer pool
150908 16:41:01 InnoDB: highest supported file format is Barracuda.
150908 16:41:02 InnoDB: Waiting for the background threads to start
150908 16:41:03 InnoDB: 1.1.8 started; log sequence number 1946818
150908 16:41:03 [Note] Server hostname (bind-address): '0.0.0.0'; port: 3306
150908 16:41:03 [Note] - '0.0.0.0' resolves to '0.0.0.0';
150908 16:41:03 [Note] Server socket created on IP: '0.0.0.0'.
150908 16:41:03 [Note] mysqld: ready for connections.
Version: '5.5.27' socket: '' port: 3306 MySQL Community Server <GPL>

```

3. 新打开cmd,输入mysql -uroot 不需要密码

```

1 use mysql;
2 update user set password=password('root') WHERE user='root';

```

4. 关闭两个cmd窗口

02-DOS操作数据乱码解决

我们在dos命令行操作中文时，会报错

```
1 insert into category(cid,cname) values('c010','中文');
2 ERROR 1366 (HY000): Incorrect string value: '\xB7\xFE\xD7\xB0' for column
  'cname' at row 1
```

错误原因：因为mysql的客户端设置编码是utf8，而系统的cmd窗口编码是gbk

1. 查看MySQL内部设置的编码

```
1 show variables like 'character%'; 查看所有mysql的编码
```

Variable_name	Value
character_set_client	utf8
character_set_connection	utf8
character_set_database	utf8
character_set_filesystem	binary
character_set_results	utf8
character_set_server	utf8
character_set_system	utf8

2. 需要修改client、connection、results的编码一致（GBK编码）

```
1 解决方案1：在cmd命令窗口中输入命令，此操作当前窗口有效，为临时方案。
2 set names gbk;
```

```
1 解决方案2：安装目录下修改my.ini文件，重启服务所有地方生效。
```

