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。

常见数据库排行榜：

Rank			DBMS	Database Model	Score		
Mar 2020	Feb 2020	Mar 2019			Mar 2020	Feb 2020	Mar 2019
1.	1.	1.	Oracle	Relational, Multi-model	1340.64	-4.11	+61.50
2.	2.	2.	MySQL	Relational, Multi-model	1259.73	-7.92	+61.48
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1097.86	+4.11	+50.01
4.	4.	4.	PostgreSQL	Relational, Multi-model	513.92	+6.98	+44.11
5.	5.	5.	IBM Db2	Relational, Multi-model	162.56	-2.99	-14.64
6.	6.	6.	Microsoft Access	Relational	125.14	-2.92	-21.07
7.	7.	7.	SQLite	Relational	121.95	-1.41	-2.92
8.	8.	8.	MariaDB	Relational, Multi-model	88.35	+1.01	+4.04
9.	9.	↑ 10.	Hive	Relational	85.38	+1.85	+12.38
10.	10.	↓ 9.	Teradata	Relational, Multi-model	77.84	+1.03	+2.63

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。

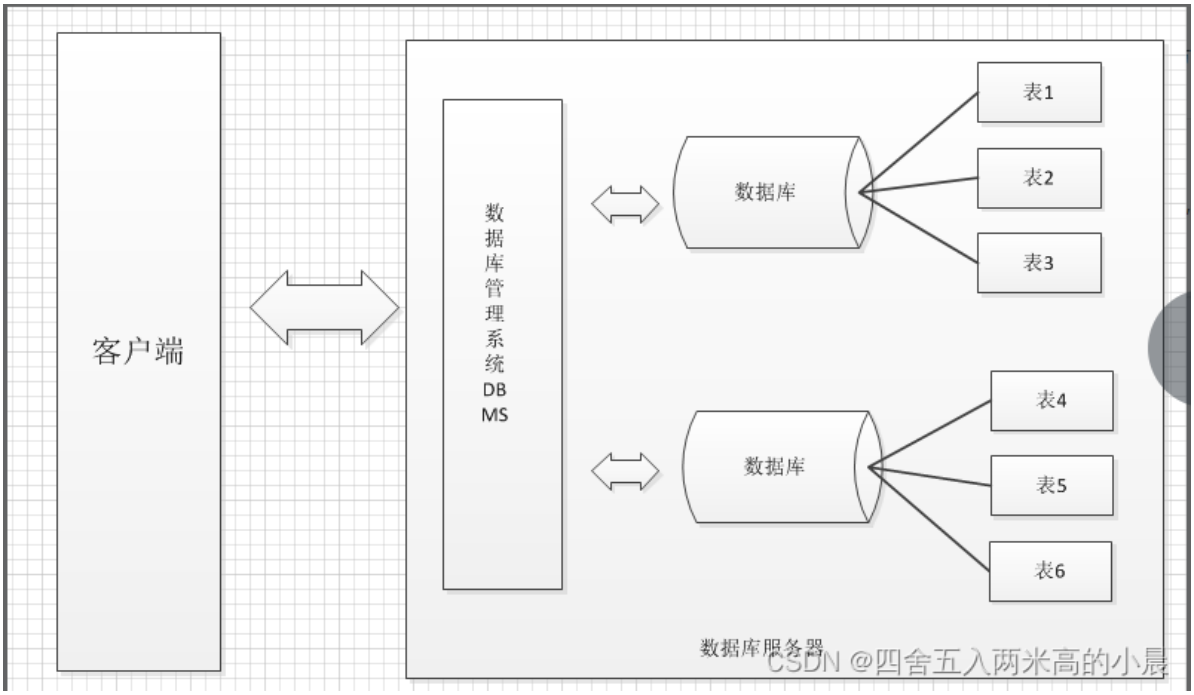
什么是数据库管理系统？

- 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

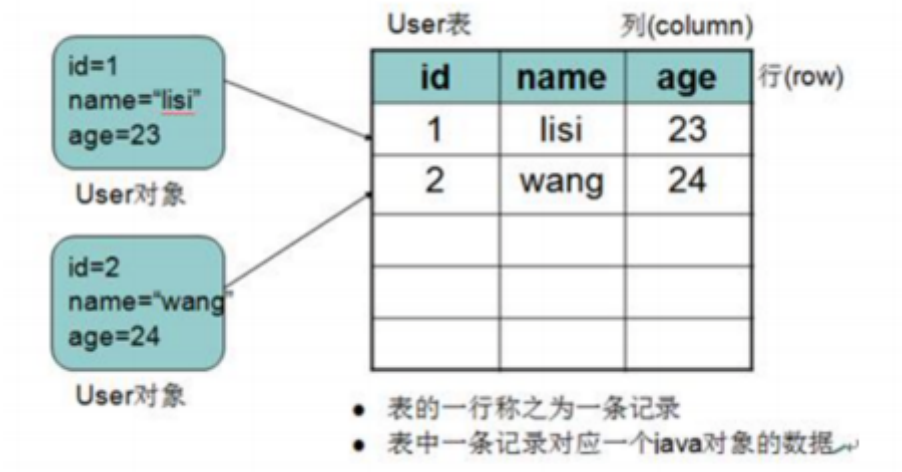
数据库中以表为组织单位存储数据。表中的每个字段都有对应的数据类型。

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)

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 常见的数据库管理系统

- MYSQL： 开源免费的数据库，小型的数据库。已经被Oracle收购了。
- Oracle：收费的大型数据库， Oracle公司的产品。 Oracle收购SUN公司，收购MYSQL。
- DB2： IBM公司的数据库产品，收费的。常应用在银行系统中
- SQLServer： MicroSoft 公司收费的中型的数据库。 C#、 .net等语言常使用。
- 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。
- 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。

第二章 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。
  。。。

2.2 安装

推荐参考视频

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. 登陆格式1：mysql -u用户名 -p密码

   例如：

   ```
   mysql -uroot -proot
   ```

   后输入密码方式:

   ```
   mysql -u用户名 -p回车
   ```

   ```
   密码
   ```

   2.登录格式2：`mysql -hip地址 -u用户名 -p密码`

   ```
   例如：`mysql -h127.0.0.1 -uroot -proot`
   ```

2. 登录格式3：`mysql --host=ip地址 --user=用户名 --password=密码`

   例如：`mysql --host=localhost --user=root --password=root`

   4.退出MySQL：exit

第三章 SQL语句

3.1 SQL的概念

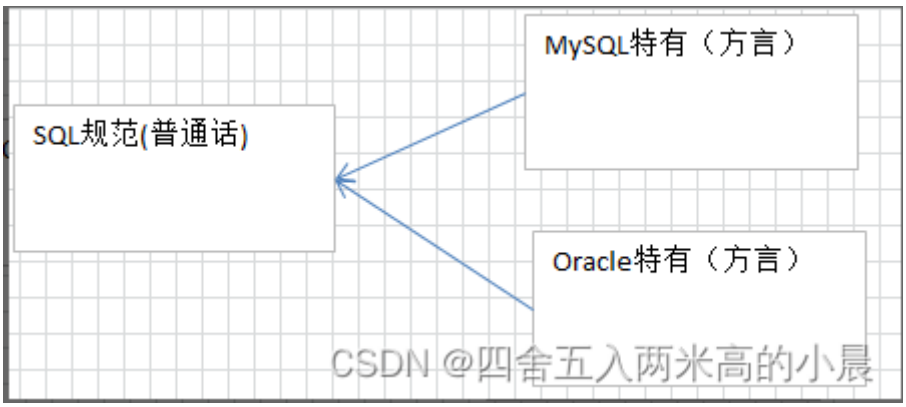
什么是SQL？

结构化查询语言(Structured Query Language)简称SQL,SQL语句就是对数据库进行操作的一种语言。

SQL作用

通过SQL语句我们可以方便的操作数据库中的数据库、表、数据。

SQL是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。



SQL语句分类

1. DDL(Data Definition Language)数据定义语言
   用来定义数据库对象：数据库，表，列等。关键字：create, drop,alter等
2. DML(Data Manipulation Language)数据操作语言
   用来对数据库中表的数据进行增删改。关键字：insert, delete, update等
3. DQL(Data Query Language)数据查询语言
   用来查询数据库中表的记录(数据)。关键字：select, where等
4. DCL(Data Control Language)数据控制语言(了解)
   用来定义数据库的访问权限和安全级别，及创建用户。关键字：GRANT，REVOKE等

SQL通用语法

1. SQL语句可以单行或多行书写，以分号结尾。
2. 可使用空格和缩进来增强语句的可读性。
3. MySQL数据库的SQL语句不区分大小写，关键字建议使用大写。

```
SELECT * FROM student;
select * from student;
SELECT * FROM student;
```

3.2 DDL语句

DDL(Data Definition Language)数据定义语言
用来定义数据库对象：数据库，表，列等。关键字：create, drop,alter等

3.2.1 DDL操作数据库

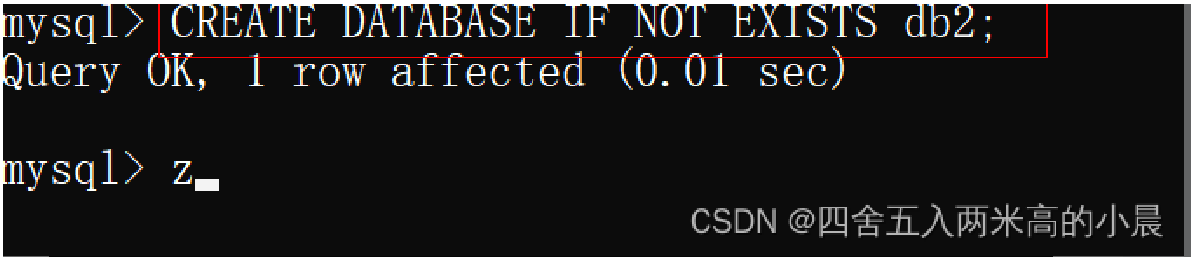
创建数据库

1. 直接创建数据库

```
CREATE DATABASE 数据库名;
```

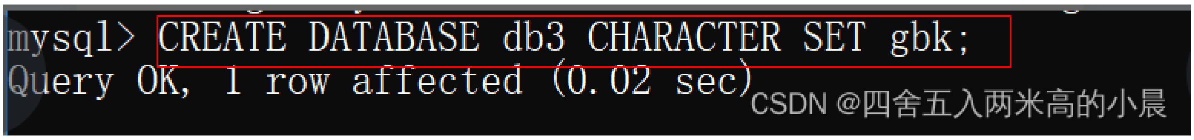
2. 判断是否存在并创建数据库

```
CREATE DATABASE IF NOT EXISTS 数据库名;
```



3. 创建数据库并指定字符集(编码表)

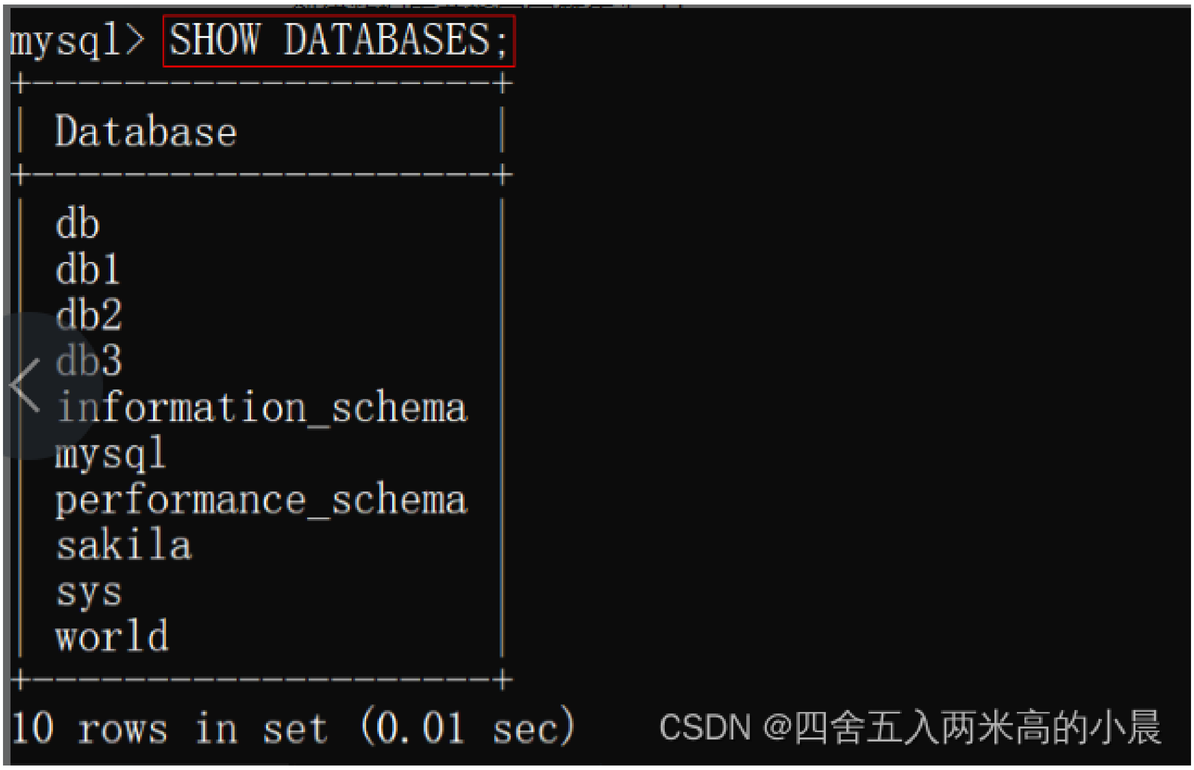
```
CREATE DATABASE 数据库名 CHARACTER SET 字符集;
```



查看数据库

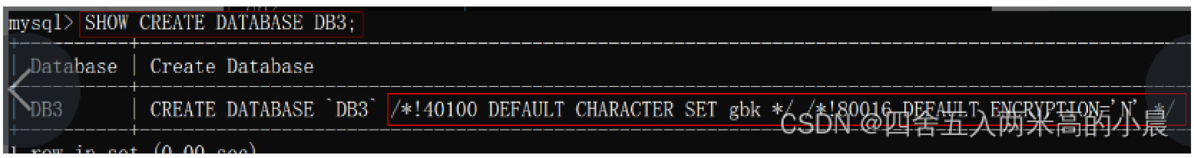
1. 查看所有的数据库

```
SHOW DATABASES;
```



2. 查看某个数据库的定义信息

```
SHOW CREATE DATABASE 数据库名;
```



修改数据库

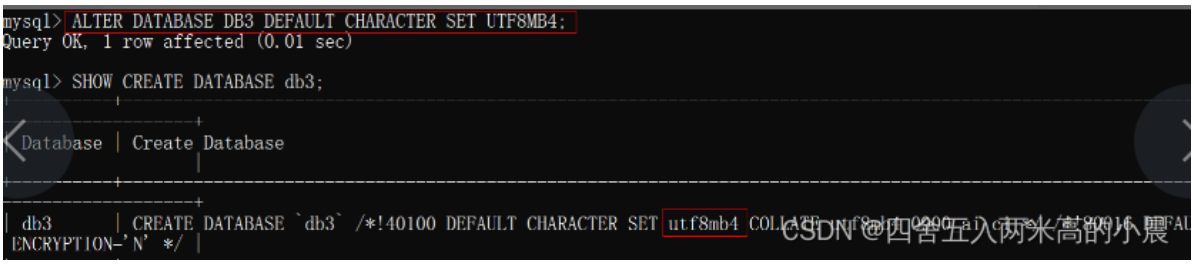
修改数据库字符集格式

```
ALTER DATABASE 数据库名 DEFAULT CHARACTER SET 字符集;
```

具体操作：

- 将db3数据库的字符集改成utf8

```
ALTER DATABASE db3 DEFAULT CHARACTER SET utf8;
```



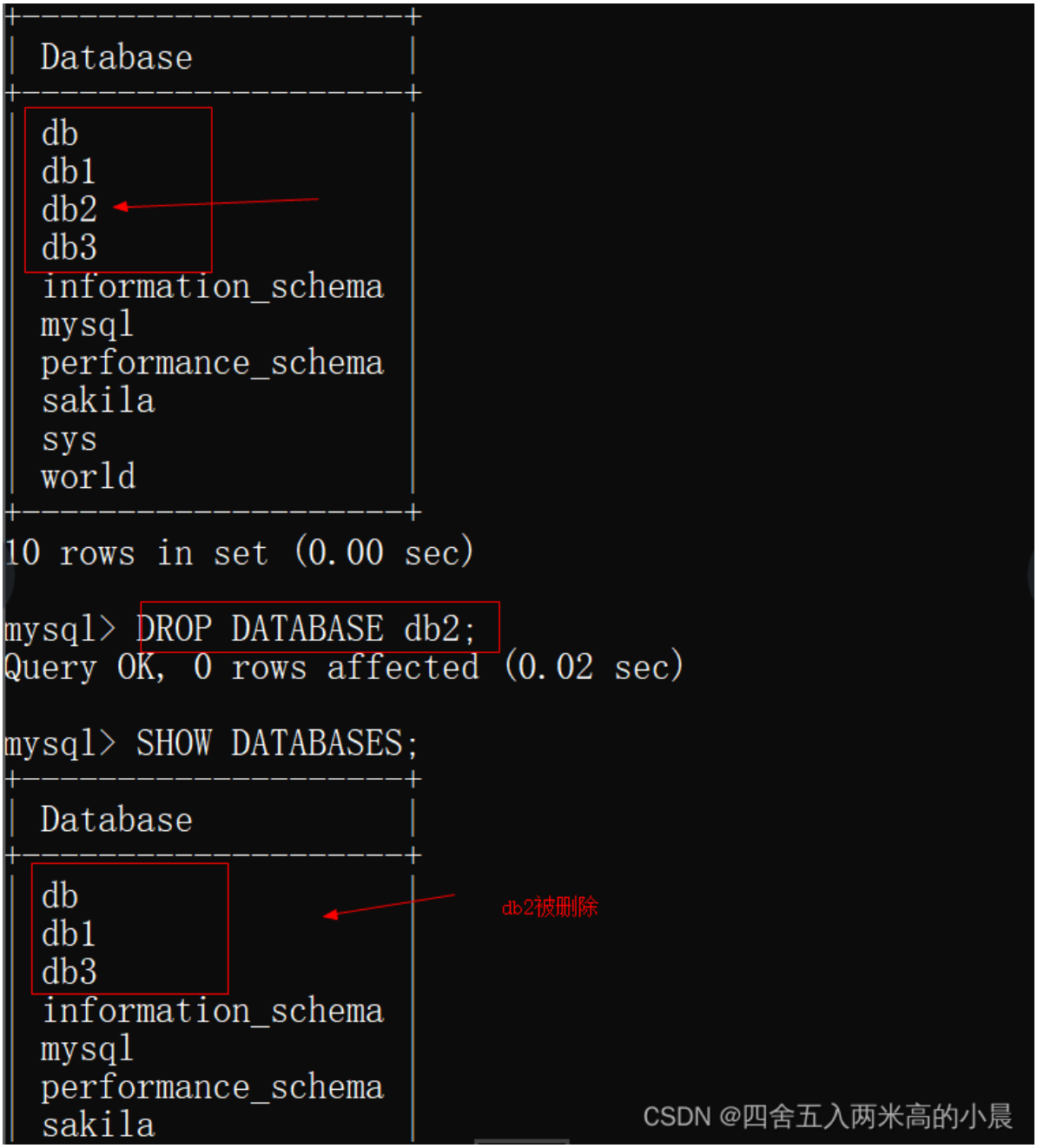
删除数据库

```
DROP DATABASE 数据库名;
```

具体操作：

- 删除db2数据库

```
DROP DATABASE db2;
```



使用数据库

1. 查看正在使用的数据库

```
SELECT DATABASE();
```

2. 使用/切换数据库

```
USE 数据库名;
```

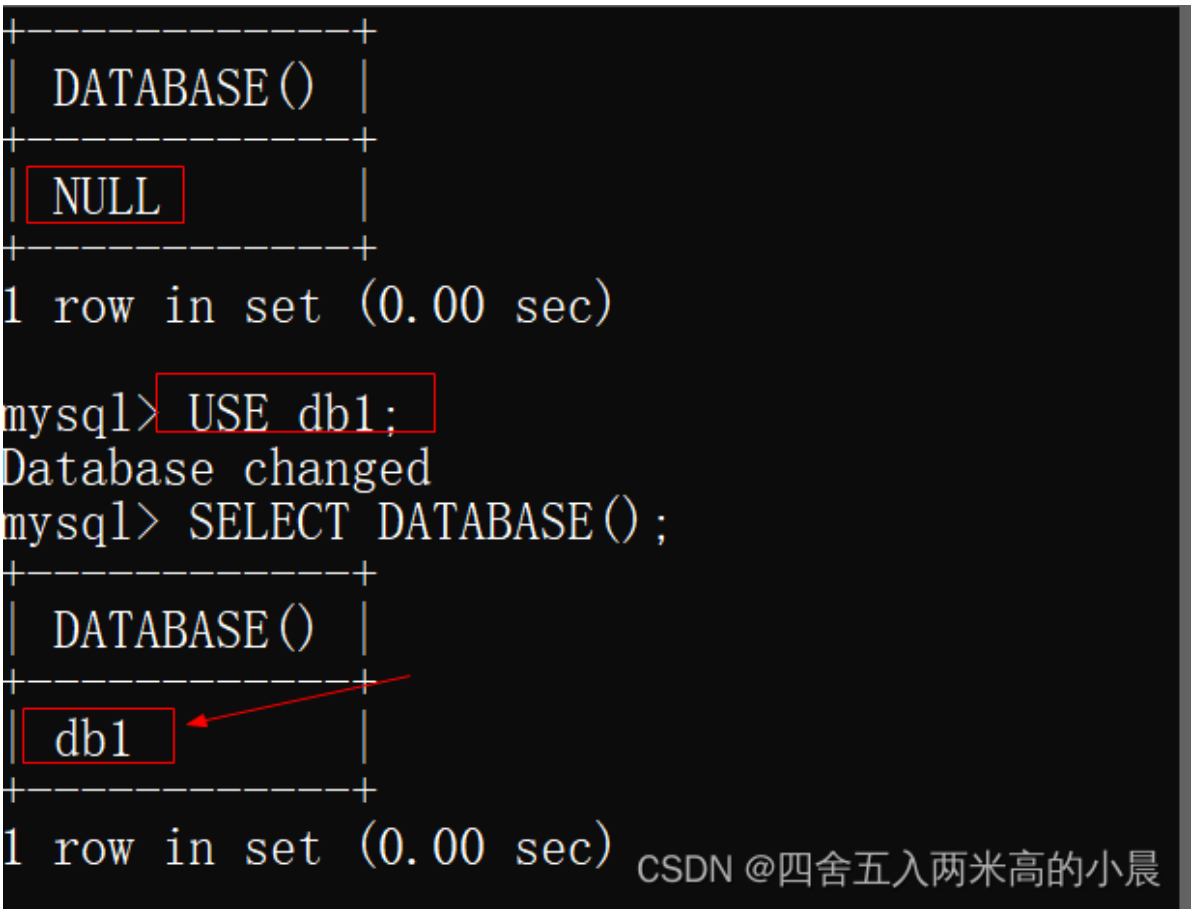
具体操作：

- 查看正在使用的数据库

```
SELECT DATABASE();
```

- 使用db1数据库

```
USE db1;
```



3.2.2 DDL操作表

> 前提是先使用某个数据库

创建表

表的结构和EXECL相似

编号	姓名	年龄	性别
1	zhangsan	19	男
2	lisi	17	女

语法：

```
CREATE TABLE 表名 (字段名1 字段类型1, 字段名2 字段类型2…);
```

关键字说明：

> CREATE -- 表示创建
> TABLE -- 表示创建一张表
>
> 建议写成：

```
CREATE TABLE 表名 (
字段名1 字段类型1,
字段名2 字段类型2
);
```

MySQL数据类型

常用的数据类型如下：

类型	描述
int	整型
double	浮点型
varchar	字符串型
date	日期类型：yyyy-MM-dd

查看表

```
>查看数据库中的所有表：show tables;
>查看表结构：desc 表名;
>查看创建表的 SQL 语句：show create table 表名;
```

案例：

```
use hello;       -- 使用 hello 数据库
show tables;     -- 查看所有表
desc user;       -- 查看 user 表的结构
show create table user; -- 查看 user 表的创建语句
```

快速创建一个表结构相同的表

```
>create table 新的表名 like 旧的表名;
```

案例

```
create table tb_user like user;
desc tb_user;
```

删除表

> drop table 表名;
> drop table if exists 表名 ;

案例

```
-- 删除用户表
drop table user;
```

修改表

```
>-- 1.修改表添加列
alter table 表名 add 列名 类型(长度) 约束;
-- 2.修改表修改列的类型长度及约束
alter table 表名 modify 列名 类型(长度) 约束;
-- 3.修改表修改列名
alter table 表名 change 旧列名 新列名 类型(长度) 约束;
-- 4.修改表删除列
alter table 表名 drop 列名;
-- 5.修改表名
rename table 表名 to 新表名;
```

案例

```
-- 修改表添加列
alter table user add address varchar(50);
-- 修改表修改列的类型长度及约束
alter table user modify address int(30);
-- 修改表修改列名
alter table user change address addr varchar(50);
-- 修改表删除列
alter table user drop addr;
-- 修改表名
rename table user to tb_user;
```

3.3 DML数据操作语言

插入记录：insert

```
-- 1.向表中插入某些列
insert into 表 (列名1,列名2,列名3..) values (值1,值2,值3..);
-- 2.向表中插入所有列
insert into 表 values (值1,值2,值3..);

-- 3.从另外一张表查某些列的结果插入当前表
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
-- 4.从另外一张表查所有列的结果插入当前表
insert into 表 values select * from 表
```

案例

```
-- 向表中插入某些列，必须写列名
insert into user (uid, uname) values (001, 'cuihua');
-- 向表中插入所有列
insert into user values (002, 'aqiang');
```

五个注意事项：

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）

更新记录：update

语法格式：update更新，set修改的列值，where指定条件。

```
-- 1.不指定条件，会修改表中当前列所有数据
update 表名 set 字段名=值 , 字段名=值;
-- 2.指定条件，符合条件的才会修改
update 表名 set 字段名=值 , 字段名=值 where 条件;
```

案例

```
-- 更新所有字段的值
update user set uname='xiaodong';
-- 根据指定的条件来更新
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

删除记录：delete & truncate

语法格式：

```
delete from 表名 [where 条件];
```

案例

```
-- 删除表中所有数据
delete from user;
-- 删除 uid 为 1 的用户
delete from user where uid = 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：

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

第四章：SQL约束

- 什么是约束？

  约束其实就是一种限制条件，让你不能够超出这个控制范围。公路上有速度限制，车距限制，鸣笛限制。而在数据库中的约束，就是指表中的数据内容不能胡乱填写必须按照要求填写，保证数据的完整性与安全性 。

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

```
CREATE TABLE persons(
id_p int PRIMARY KEY,--设置id_p为主键
lastname varchar(255),
firstname varchar(255),
address varchar(255),
city varchar(255)
);
```

- 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（给主键起别名）] 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。

```
CREATE TABLE persons_cons(
firstname varchar(255),
lastname varchar(255),
address varchar(255),
city varchar(255),
CONSTRAINT pk_personID PRIMARY KEY (firstname,lastname)
);--添加主键约束，多个字段，我们称为联合主键
```

- 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)

```
CREATE TABLE persons_after(
firstname varchar(255),
lastname varchar(255),
address varchar(255),
city varchar(255)
);
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：

```
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

```
CREATE TABLE persons_id(
p_id int PRIMARY KEY AUTO_INCREMENT,
lastname varchar(255),
firstname varchar(255),
address varchar(255),
city varchar(255)
);
```

- 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：

```
INSERT INTO persons_id (firstname,lastname) VALUES ('Bill','Gates');
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：

```
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：

```
CREATE TABLE persons_null(
id_p int NOT NULL,
lastname varchar(255) NOT NULL,
firstname varchar(255),
address varchar(255),
city varchar(255)
);
```

删除非空约束

-格式：ALTER TABLE 表名 MODIFY 字段名 数据类型[长度]

```
ALTER TABLE persons_null MODIFY id_p int;
ALTER TABLE persons_null MODIFY lastname varchar(255);
```

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。

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：

```
CREATE TABLE persons_unique(
id_p int UNIQUE,
lastname varchar(255) NOT NULL,
firstname varchar(255),
address varchar(255),
city varchar(255)
)
```

- 方式二：创建表时，在约束区域，声明唯一：

```
CREATE TABLE persons(
id_p int,
lastname varchar(255) NOT NULL,
firstname varchar(255),
address varchar(255),
city varchar(255),
CONSTRAINT unique_id_p UNIQUE (Id_P)
)
```

- 方式三：创建表后，修改表结构，声明字段唯一：

```
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:

```
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。