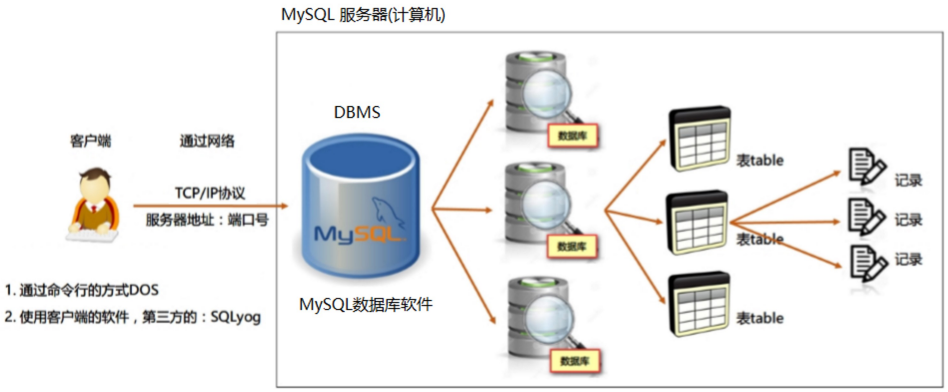
day02_MySQL基础

数据的存储方式：

存储位置	优点	缺点
内存	速度快	不能永久保存，数据是临时存储状态
文件	数据可以永久保存	操作数据不方便
数据库	数据可以永久保存、对数据的管理方便	占用资源、需要购买

- 数据库(DataBase，DB)：用来管理和存储数据的仓库，其本质上还是一个文件系统，它是把数据以文件的方式存储在服务器的电脑上。并且所有的关系型数据库都可以使用通用的 SQL 语句进行管理。
- 数据库管理系统(DataBase Management System，DBMS)：指一种操作和管理多个数据库的大型软件，用于创建、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。用户可以通过数据库管理系统访问、操作数据库中的数据。



一.MySQL数据库软件

见 "参考\MySQL数据库软件的使用.md"

二.SQL

SQL(Structured Query Language)：结构化查询语言，是一种通用的数据库操作语言，是所有关系型数据库的操作规范。SQL 语句的通用语法可以用在不同的数据库中，但是不同数据库的 SQL 语句也有一些区别。SQL 语句的分类如下：

- DDL (Data Definition Language) 数据定义语言：用来定义数据库对象(数据库、表)。关键字：CREATE、DROP、ALTER 等
- 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、改的操作。关键字：INSERT、DELETE、UPDATE 等
- DQL (Data Query Language) 数据查询语言：用来查询数据库中表的数据。关键字：SELECT、WHERE 等
- DCL (Data Control Language) 数据控制语言：用来定义数据库的访问权限、安全级别、以及创建用户。关键字：GRANT、REVOKE 等

#.注意事项：
(1).SQL 语句可以单行或多行书写，以分号结尾
(2).SQL 语句的关键字不区分大小写，关键字建议使用大写
(3).除了数字类型，其他数据类型的值出现在 SQL 语句中时，需要使用引号引起来(单、双引号都可以)
(4).注释方式：
  - 单行注释：-- 注释内容、# 注释内容(mysql 特有)
  - 多行注释： /* 注释内容 */

1.DDL 数据定义语言：对数据库、表进行 CRUD 的操作

(1).对数据库进行 CRUD 的操作

①.创建数据库(Create):

```
CREATE DATABASE [IF NOT EXISTS] 数据库名称 [CHARACTER SET 字符集名];
```

- [IF NOT EXISTS]: 先判断数据库是否已经存在，若存在则不进行创建操作，若不存在再进行创建
- [CHARACTER SET 字符集名]: 创建时指定数据库的字符集

②.查询数据库(Retrieve):

- 查询所有数据库的名称: SHOW DATABASES;
- 查询某个数据库的创建语句，并可以看到该数据库的字符集: SHOW CREATE DATABASE 数据库名称;

③.修改数据库(Update):

- 修改数据库的字符集: ALTER DATABASE 数据库名称 CHARACTER SET 字符集名称;

④.删除数据库(Delete):

```
DROP DATABASE [IF EXISTS] 数据库名称;
```

- [IF EXISTS]: 先判断数据库是否已经存在，若存在则删除，若不存在则不进行删除操作

#.要操作数据库中的表和数据，就要先使用(进入)该数据库:

- 使用(进入)数据库: USE 数据库名称;
- 查询当前正在使用(进入)的数据库名称: SELECT DATABASE();

(2).对表进行 CRUD 的操作: 前提是要使用(进入)某个数据库

①.创建表(Create):

```
CREATE TABLE 表名 (
   列名1  数据类型1,
   列名2  数据类型2,
   ....
   列名n  数据类型n   -- 最后一列没有逗号
);
```

#.注意事项:

a.复制表(只会复制表的结构，不复制表的数据): CREATE TABLE 表名 LIKE 被复制的表名;

b.MySQL 数据库常用的数据类型:

分类	类型名称	类型说明
整数类型	tinyint	微整型: 很小的整数，占8位二进制
	smallint	小整型: 小的整数，占16位二进制
	mediumint	中整型: 中等长度的整数，占24位二进制
	int(integer)	整型: 整数类型，占 32 位二进制
小数类型	float(m, n)	单精度浮点数，占4个字节，m为数字的总长度，n为小数位长度
	double(m, n)	双精度浮点数，占8个字节，m为数字的总长度，n为小数位长度
日期类型	date	只包含年月日的日期: yyyy-MM-dd
	datetime	包含年月日时分秒的日期: yyyy-MM-dd HH:mm:ss
	timestamp	包含年月日时分秒的日期: yyyy-MM-dd HH:mm:ss
字符串类型	char(m)	固定长度的字符串，无论使用几个字符都占满全部，m为0~255之间的整数
	varchar(m)	可变长度的字符串，使用几个字符就占用几个，m为0~65535之间的整数

*若一个字段为 timestamp 类型，那么在添加一条记录时不给这个字段赋值、或赋值为 null，则默认使用当前的系统时间，来自动赋值。并且这个字段里的时间数据会随着这条记录的修改而自动刷新，所以 timestamp 类型的字段可以存放这条记录最后被修改的时间。

②.查询表(Retrieve):

- 查询数据库中所有的表名称: SHOW TABLES;
- 查询某个表结构: DESC 表名;
- 查询某个表的创建语句，并可以看到该表的字符集: SHOW CREATE TABLE 表名;

③.修改表(Update):

- 修改表名: ALTER TABLE 表名 RENAME TO 新的表名;
- 修改表的字符集: ALTER TABLE 表名 CHARACTER SET 字符集名称;
- 添加一列: ALTER TABLE 表名 ADD 列名 数据类型;
- 修改某列的数据类型: ALTER TABLE 表名 MODIFY 列名 新数据类型;
- 同时修改某列的名称、数据类型: ALTER TABLE 表名 CHANGE 列名 新列名 新数据类型;
- 删除某列: ALTER TABLE 表名 DROP 列名;

④.删除表(Delete):

```
DROP TABLE [IF EXISTS] 表名;
```

- [IF EXISTS]：先判断表是否已经存在，若存在则删除，若不存在则不进行删除操作

2.DML 数据操作语言：对数据库中表的数据进行增、删、改的操作

(1).添加数据：

```
INSERT INTO 表名 (列名1, 列名2, … , 列名n) values (值1, 值2, … , 值n);
```

- 列名和值要一一对应。不写某些列名时，则默认赋值为 NULL
- 表名后面可以不定义列名，则默认给所有列添加值：INSERT INTO 表名 VALUES (值1, 值2, … , 值n);

(2). 删除数据：

```
DELETE FROM 表名 [WHERE 条件列表];
```

- 若不加限定条件，则会删除表中所有记录
- 删除表中所有记录的两种方法：

```
DELETE FROM 表名;      -- 有多少条记录就会执行多少次删除操作，效率低下，不推荐使用
TURNCAT TABLE 表名;    -- 先删除表，然后再创建一张一样的空数据表，效率高，推荐使用
```

(3).修改数据：

```
UPDATE 表名 SET 列名1 = 值1, 列名2 = 值2, ... [WHERE 条件列表];
```

- 如果不加限定条件，则会修改表中的所有记录

3.DQL 数据查询语言：查询数据库中表的数据(单表查询)

```
SELECT 字段列表 FROM 表名列表
[WHERE 条件列表]
[GROUP BY 分组字段]
[HAVING 分组之后的条件]
[ORDER BY 排序字段]
[LIMIT 分页限定];
```

(1).基础查询

①.多个字段的查询：

```
SELECT 字段名1, 字段名2, ... FROM 表名;
```

- 查询表中的所有字段，可以使用 * 替代字段列表：SELECT * FROM 表名;

②.去除重复记录：

```
SELECT DISTINCT 字段名1, 字段名2, ... FROM 表名;
```

- 当查询多个字段时，DISTINCT 只能放在最前面，并且只有当 "字段1、字段2 ..." 的值都相同时，才能算做重复的记录

③.计算列：

```
SELECT 字段名1, 字段名2, 字段名1 + 字段名2, ... FROM 表名;
```

- 一般是在多个列之间进行四则运算，形成一个新的列，并且参与运算的列必须是数值类型。
- 如果某条数据的某列值为 NULL，那么四则运算后的结果将还是 NULL，此时就可以使用 IFNULL 函数将 NULL 替换为其他数据。
  IFNULL(表达式1, 表达式2)：如果表达式1为 NULL 值，则返回表达式2；如果表达式1不为 NULL，则返回表达式1
  SELECT name, math, english, IFNULL(math, 0) + IFNULL(english, 0) FROM student; -- 查询每位学生的数学和英语成绩之和

④.给列、表起别名：

```
SELECT 字段名1 AS 别名1, 字段名2 AS 别名2, ... FROM 表名 AS 表别名;
```

- 字段名的别名会显示在结果集中，而表的别名多用于多表查询
- AS 可以省略，直接在字段名、表名后面写别名即可

(2).条件查询

```
SELECT 字段列表 FROM 表名列表 WHERE 条件列表;
```

- 比较运算符：> 、< 、>=、<= 、= 、<>
  <> 在 SQL 中表示不等于，在 mysql 中也可以使用 != 表示不等于，没有 == 运算符

```
-- 查询年龄大于20岁的学生信息
SELECT * FROM student WHERE age > 20;
-- 查询年龄大于等于20岁的学生信息
SELECT * FROM student WHERE age >= 20;
-- 查询年龄等于20岁的学生信息
SELECT * FROM student WHERE age = 20;
```

```
-- 查询年龄不等于20岁的学生信息
SELECT * FROM student WHERE age != 20;
SELECT * FROM student WHERE age <> 20;
```

- BETWEEN...AND：在一个范围之内，左右的值均包含在内。

```
-- 查询年龄大于等于20小于等于30的学生信息
SELECT * FROM student WHERE age BETWEEN 20 AND 30;
```

- IN(集合)：集合表示多个值，使用逗号分隔

```
-- 查询年龄为18岁、22岁、25岁的学生信息
SELECT * FROM student WHERE age IN (18, 22, 25);
```

- LIKE 模糊查询：　　通配符"_"：匹配单个任意字符　　通配"%"：匹配多个任意字符

```
-- 查询姓马的学生信息
SELECT * FROM student WHERE NAME LIKE '马%';
-- 查询姓名第二个字是"化"的学生信息
SELECT * FROM student WHERE NAME LIKE "_化%";
-- 查询姓名是3个字的学生信息
SELECT * FROM student WHERE NAME LIKE '___';
-- 查询姓名中包含"德"的学生信息
SELECT * FROM student WHERE NAME LIKE '%德%';
```

IS NULL、IS NOT NULL：判断是否为空

```
-- 查询英语成绩为 null 的学生信息
SELECT * FROM student WHERE english = NULL; -- 错误，null 值不能使用 =、!= 判断
SELECT * FROM student WHERE english IS NULL;
-- 查询英语成绩不为 null 的学生信息
SELECT * FROM student WHERE english  IS NOT NULL;
```

- 逻辑运算符：and 或 &&(与)、or 或 ||(或)、not 或 !(非)，用来连接多个条件表达式

```
-- 查询年龄大于等于20小于等于30的学生信息
SELECT * FROM student WHERE age >= 20 &&  age <=30;
-- 查询年龄为18岁、22岁、25岁的学生信息
SELECT * FROM student WHERE age = 18 OR age = 22 OR age = 25;
```

(3).排序查询

* 语法：order by 子句
    * order by 排序字段1 排序方式1 ，　排序字段2 排序方式2...
* 排序方式：
    * ASC：升序，默认的。
    * DESC：降序。
* 注意：
    * 如果有多个排序条件，则当前边的条件值一样时，才会判断第二条件。

(4).聚合函数：将一列数据作为一个整体，进行纵向的计算。

1. count：计算个数
    1. 一般选择非空的列：主键
    2. count(*)
2. max：计算最大值
3. min：计算最小值
4. sum：计算和
5. avg：计算平均值

* 注意：聚合函数的计算，排除null值。
    解决方案：
        1. 选择不包含非空的列进行计算
        2. IFNULL函数

(5). 分组查询：

1. 语法：group by 分组字段;
2. 注意：
    1. 分组之后查询的字段：分组字段、聚合函数

如果现在要进行分组的话，则SELECT子句之后，只能出现**分组的字段和统计函数**，其他的字段不能出现

2. where 和 having 的区别?

1. where 在分组之前进行限定，如果不满足条件，则不参与分组。having在分组之后进行限定，如果不满足结果，则不会被查询出来

2. where 后不可以跟聚合函数，having可以进行聚合函数的判断。

```
-- 按照性别分组。分别查询男、女同学的平均分
SELECT sex , AVG(math) FROM student GROUP BY sex;

-- 按照性别分组。分别查询男、女同学的平均分,人数

SELECT sex , AVG(math),COUNT(id) FROM student GROUP BY sex;

--  按照性别分组。分别查询男、女同学的平均分,人数 要求：分数低于70分的人，不参与分组
SELECT sex , AVG(math),COUNT(id) FROM student WHERE math > 70 GROUP BY sex;

--  按照性别分组。分别查询男、女同学的平均分,人数 要求：分数低于70分的人，不参与分组,分组之后。人数要大于2个人
SELECT sex , AVG(math),COUNT(id) FROM student WHERE math > 70 GROUP BY sex HAVING COUNT(id) > 2;

SELECT sex , AVG(math),COUNT(id) 人数 FROM student WHERE math > 70 GROUP BY sex HAVING 人数 > 2;
```

(6).分页查询

1. 语法：limit 开始的索引,每页查询的条数;
2. 公式：开始的索引 = （当前的页码 - 1） * 每页显示的条数

```
-- 每页显示3条记录
SELECT * FROM student LIMIT 0,3; -- 第1页

SELECT * FROM student LIMIT 3,3; -- 第2页

SELECT * FROM student LIMIT 6,3; -- 第3页
```

3. limit 是一个MySQL"方言"

4.DQL 数据查询语言：查询数据库中表的数据(多表查询)