MySQL

@M了个J

李明杰

https://github.com/CoderMJLee

https://space.bilibili.com/325538782

小码哥教育 SEEMYGO

实力IT教育 www.520it.com

如何长久地存储数据？（持久化）

- 通常会使用数据库（Database）来长久地存储数据

- 使用数据库的明显好处？
  - 可以高效地存储、查询数据
  - 减少重复、冗余数据
  - 提高数据的安全性

常见的数据库

- 数据库分为：关系型数据库（Relational Database）、非关系型数据库（NoSQL Database）

关系型数据库

非关系型数据库

MySQL

- MySQL是一款开源的关系型数据库，有免费版（社区版）、商业版
  - 2008年被Sun公司收购，Sun公司在2009年被Oracle公司收购

- 关于MySQL的发音
  - 官方说明：https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
  - 官方发音：My-S-Q-L，并非My Sequel (/ˈsiːkwəl/)
  - 不过基本都已经习惯发音为My Sequel

MySQL下载

- 首先要下载安装MySQL服务器软件：MySQL Community Server（社区版）
  - 常用版本：5.7、8.0，本课程使用的是5.7.29版本

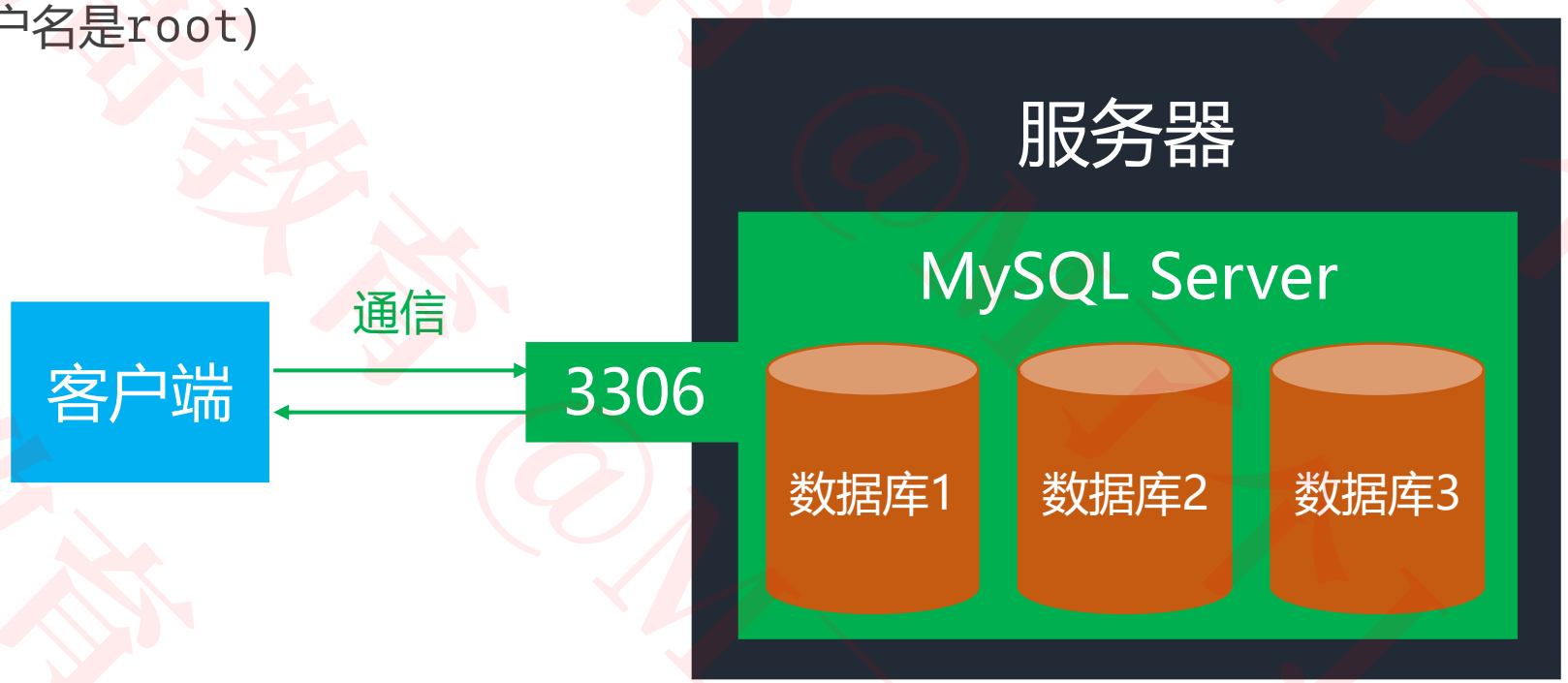
- 下载地址
  - Windows：https://downloads.mysql.com/archives/installer/
  - Mac：https://downloads.mysql.com/archives/community/

- 官方文档
  - 英文文档
  - 中文文档

MySQL的使用步骤

- 登录、连接MySQL服务器（管理员账户名是root）
  - mysql –uroot –p密码
  - mysql –uroot –p

- 使用SQL语句对数据库进行CRUD
  - Create：创建，增
  - Read：读取，查
  - Update：更新，改
  - Delete：删除，删
  - 一般在中文中称为：增删改查



- 在Windows中，可以通过以下命令查看占用3306端口号的进程ID
  - netstat –aon|findstr 3306

数据库的内部存储细节

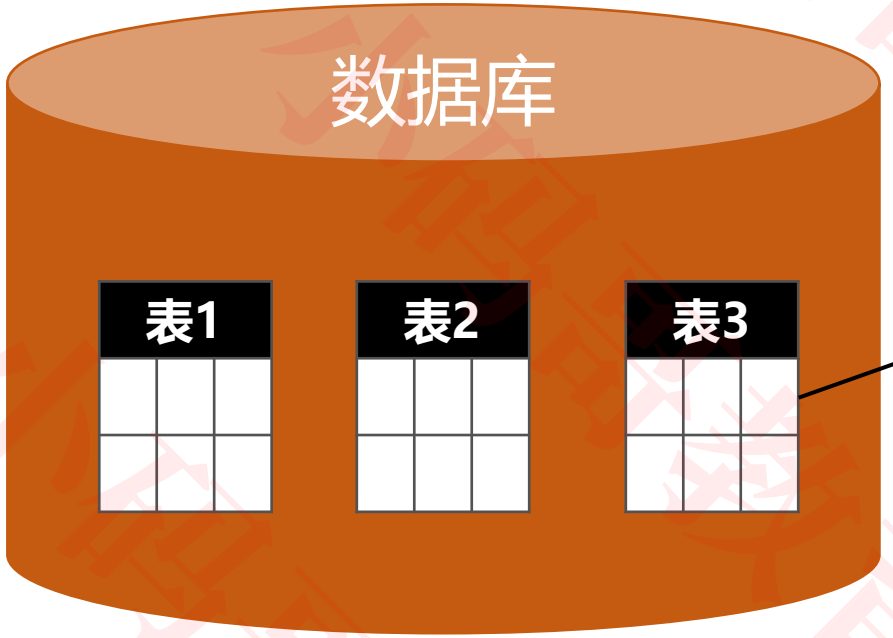


表				
id	name	age	job	column 列
1	小码哥	9	程序员	record 记录
2	大码哥	15	老师	
3	老码哥	18	医生	

- 一个数据库（Database）中可以存放多张表（Table）
- 每个表（Table）中包含一些列（Column，也叫做字段）
- 每个表（Table）中存放的数据，一般称为记录（Record）

SQL语句

■ SQL是Structured Query Language的简称，译为“结构化查询语言“，用于操作关系型数据库

■ SQL语句主要可以分为4大类

◘DDL (Data Definition Language)

✓数据定义语言

✓创建（CREATE）、修改（ALTER）、删除（DROP）数据库\表

◘DQL (Data Query Language)

✓数据查询语言

✓查询记录（SELECT）

◘DML (Data Manipulation Language)

✓数据操纵语言

✓增加（INSERT）、删除（DELETE）、修改（UPDATE）记录

◘DCL (Data Control Language)

✓数据控制语言

✓控制访问权限（GRANT、REVOKE）

SQL语句

- 每一条语句是分号（;）结束

- 不区分大小写，建议：关键字使用大写，其他使用小写，单词之间用下划线连接，比如my_firsname

- 单行注释
  - -- 注释内容（--后要预留至少一个空格）
  - #注释内容

- 多行注释
  - /*注释内容*/

- 参考资料：【官方文档】第13章SQL语句语法

GUI工具

■ 在实际开发过程中，经常会使用图形化界面工具来管理数据库。常用的有（均是付费软件）

□ Navicat Premium

✓ 支持MySQL、MariaDB、MongoDB、SQL Server、Oracle、PostgreSQL、SQLite

□ SQLyog

✓ 支持MySQL

DDL语句 – 数据库

■ 创建

- `CREATE DATABASE 数据库名` # 创建数据库（使用默认的字符编码）
- `CREATE DATABASE 数据库名 CHARACTER SET 字符编码` # 创建数据库（使用指定的字符编码）
- `CREATE DATABASE IF NOT EXISTS 数据库名` # 如果这个数据库不存在，才创建它
- `CREATE DATABASE IF NOT EXISTS 数据库名 CHARACTER SET 字符编码`

■ 查询

- `SHOW DATABASES` # 查询所有的数据库
- `SHOW CREATE DATABASE 数据库名` # 查询数据库的创建语句
- `USE 数据库名` # 使用数据库
- `SELECT DATABASE()` # 查询正在使用的数据库

DDL语句 – 数据库

■ 修改

- `ALTER DATABASE 数据库名 CHARACTER SET 字符编码  # 修改数据库的字符编码`

■ 删除

- `DROP DATABASE 数据库名`
- `DROP DATABASE IF EXISTS 数据库名  # 如果这个数据库存在，才删除它`

DDL语句 – 表

■ 创建（基本语法）

```
CREATE TABLE 表名(
    列名1 数据类型1,
    列名2 数据类型2,
    列名3 数据类型3,
    ...
    列名n 数据类型n
)
```

■ 查询

- SHOW TABLES # 查询当前数据库的所有表
- DESC 表名 # 查看表结构

DDL语句 – 表

■ 删除

- DROP TABLE 表名
- DROP TABLE IF EXISTS 表名 # 如果这个表存在，才删除它

■ 修改

- ALTER TABLE 表名 RENAME TO 新表名 # 修改表名
- ALTER TABLE 表名 CHARACTER SET 字符编码 # 修改表格的字符编码
- ALTER TABLE 表名 ADD 列名 数据类型 # 增加新的一列
- ALTER TABLE 表名 MODIFY 列名 新数据类型 # 修改某一列的数据类型
- ALTER TABLE 表名 CHANGE 列名 新列名 新数据类型 # 修改某一列的列名、数据类型
- ALTER TABLE 表名 DROP 列名 # 删除某一列

常用数据类型 – 数字类型

类型	字节	有符号取值范围	无符号取值范围
TINYINT\BOOL\BOOLEAN	1	-128 ~ 127	0 ~ 255
SMALLINT	2	-32768 ~ 32767	0 ~ 65535
MEDIUMINT	3	-8388608 ~ 8388607	0 ~ 16777215
INT\INTEGER	4	-2147483648 ~ 2147483647	0 ~ 4294967295
BIGINT	8	-9223372036854775808 ~ 9223372036854775807	0 ~ 18446744073709551615

- 上述整数类型的使用格式为：TYPE[(M)] [UNSIGNED] [ZEROFILL]
  - UNSIGNED：无符号
  - ZEROFILL：等价于UNSIGNED ZEROFILL，表示在显示数值时，若数字不足M位，就在前面用0填充

常用数据类型 – 数字类型

类型	字节	描述
FLOAT	4	存储的小数是近似值
DOUBLE\DOUBLE PRECISION	8	
DECIMAL\DEC\NUMERIC\FIXED	看情况	存储的小数可以更加精确

- 参考资料
  - 【官方文档】11.1.1数字类型概述
  - 【官方文档】11.8数据类型存储要求

常用数据类型 – 字符串类型

类型	描述
CHAR	长度可以指定为0~255 查询数据时，会省略后面的空白字符
VARCHAR	长度可以指定为0~65535（较常使用） 查询数据时，不会省略后面的空白字符
BLOB	用于存储二进制数据（照片、文件、大文本等）
TEXT	用于存储大文本

- 参考资料
  - 【官方文档】11.4.1 CHAR和VARCHAR类型

常用数据类型 – 日期和时间类型

类型	字节 MySQL 5.6.4之前	字节 MySQL 5.6.4开始	显示格式
YEAR	1	1	YYYY
DATE	3	3	YYYY-MM-DD
TIME	3	3 + 小数秒存储	HH:MM:SS[.fraction]
DATETIME	8	5 + 小数秒存储	YYYY-MM-DD HH:MM:SS[.fraction]
TIMESTAMP	4	4 + 小数秒存储	YYYY-MM-DD HH:MM:SS[.fraction]

- 从MySQL 5.6.4开始，允许TIME\DATETIME\TIMESTAMP有小数部分，需要0~3字节的存储
- DATETIME支持的范围：1000-01-01 00:00:00.000000到9999-12-31 23:59:59.999999
- TIMESTAMP支持的范围：1970-01-01 00:00:01.000000到2038-01-19 03:14:07.999999

DATETIME\TIMESTAMP的自动设置

- DEFAULT CURRENT_TIMESTAMP
  - 当插入记录时，如果没有指定时间值，就设置时间为当前的系统时间

- ON UPDATE CURRENT_TIMESTAMP
  - 当修改记录时，如果没有指定时间值，就设置时间为当前的系统时间

DML语句

■ 增加

- INSERT INTO 表名(列名1，列名2，...，列名n) VALUES (值1，值2，...，值n)
  - ✓非数字类型的值，一般需要用引号括住（单引号或双引号，建议使用单引号）
- INSERT INTO 表名 VALUES (值1，值2，...，值n) # 从左至右按顺序给所有列添加值

■ 修改

- UPDATE 表名 SET 列名1 = 值1，列名2 = 值2，...，列名n = 值n [WHERE 条件]
- 如果没有添加条件，将会修改表中的所有记录

■ 删除

- DELETE FROM 表名 [WHERE 条件]
  - ✓如果没有添加条件，将会删除表中的所有记录

TRUNCATE

- 如果要删除表中的所有数据（保留表结构），有2种常见做法
  - DELETE FROM 表名 # 逐行删除每一条记录
  - TRUNCATE [TABLE] 表名 # 先删除后重新创建表（效率高）
    - ✓ TRUNCATE归类为DDL语句，而不是DML语句
    - ✓ 为了实现高性能，它绕过了删除数据的DML方法。因此，它不能被回滚，不会导致ON DELETE触发器触发，并且不能对InnoDB具有父子外键关系的表执行

- 参考资料
  - 【官方文档】13.1.34 TRUNCATE TABLE语法

DQL语句 – SELECT语句

```
SELECT [DISTINCT] 列名1, 列名2, ..., 列名n
FROM 表名
[WHERE ...]
[GROUP BY ...]
[HAVING ...]
[ORDER BY ...]
[LIMIT ...]
```

- `SELECT * FROM customer  # 查询表中的所有记录`

- `SELECT DISTINCT * FROM customer  # 查询表中的所有记录（去除了重复的记录）`

聚合函数（Aggregate Function）

- `SELECT COUNT(*) FROM customer  # 查询表中的记录总数`
- `SELECT COUNT(phone) FROM customer  # 查询表中phone的总数（不包括NULL）`
- `SELECT COUNT(DISTINCT phone) FROM customer`
  - 查询表中phone的总数（不包括NULL，去除了重复的记录）

- `SELECT SUM(salary) FROM customer  # 计算所有salary的总和`
- `SELECT MIN(age) FROM customer  # 查询最小的age`
- `SELECT MAX(age) FROM customer  # 查询最大的age`
- `SELECT AVG(salary) FROM customer  # 计算所有salary的平均值`

- 参考资料：【官方文档】12.19.1聚合函数描述

常见的WHERE子句

■ 比较运算

```
WHERE age > 18   # age大于18
WHERE age <= 30   # age小于等于30
WHERE age = 20   # age等于20
WHERE name = '张三'   # age等于20
WHERE age != 25   # age不等于25
WHERE age <> 25   # age不等于25
```

■ NULL值判断（不能用=、!=、<>）

```
WHERE phone IS NULL   # phone的值为NULL
WHERE phone IS NOT NULL   # phone的值不为NULL
```

常见的WHERE子句

■ 逻辑运算

```
WHERE age > 18 AND age <= 30  # age大于18并且小于等于30
WHERE age > 18 && age <= 30  # age大于18并且小于等于30
WHERE age BETWEEN 20 AND 30  # age大于等于20并且小于等于30

WHERE age = 18 OR age = 20 OR age = 22  # age等于18或者等于20或者等于22
WHERE age IN (18, 20, 22)  # age等于18或者等于20或者等于22

WHERE NOT (age < 18)  # age大于等于18
WHERE ! (age < 18)  # age大于等于18
```

常见的WHERE子句

- 模糊查询（_代表单个任意字符，%代表任意个任意字符）

```
WHERE name LIKE '_码_'   # name是3个字符并且中间是'码'字
WHERE name LIKE '___'    # name是3个字符
WHERE name LIKE '李%'    # name以'李'字开头
WHERE name LIKE '_码%'   # name的第2个字符是'码'字
WHERE name LIKE '%码%'   # name中包含'码'字
```

小码哥教育 SEEMYGO

表的复制

- 创建一张拥有相同表结构的空表（只复制表结构，不复制记录）
  - `CREATE TABLE new_table LIKE old_table`

- 创建一张拥有相同表结构、相同记录的表（复制表结构、复制记录）
  - `CREATE TABLE new_table AS (SELECT * FROM old_table)`
  - 可以省略AS

列的常用属性

- NOT NULL：不能设置为NULL值
- COMMENT：注释
- DEFAULT：默认值（BLOB、TEXT、GEOMETRY、JSON类型不能有默认值）
- AUTO_INCREMENT：自动增长
  - 适用于INT、FLOAT、DOUBLE类型
  - 在插入记录时，如果不指定此列的值或设置为NULL，会在此前的基础上自动增长1
  - 不能有默认值（不能使用DEFAULT）
  - 在一个表格中，最多只能有一列被设置为AUTO_INCREMENT
  - 这一列必须被索引（UNIQUE、PRIMARY KEY、FOREIGN KEY等）

UNIQUE索引

- 一旦某一列被设置了UNIQUE索引
  - 该列的所有值必须是唯一的
  - 允许存在多个NULL值

- 2种常见写法
  - 列名 数据类型 UNIQUE [KEY]
  - UNIQUE [KEY] (列名)

```
CREATE TABLE student(
        id   INT UNIQUE,
        name VARCHAR(20),
        UNIQUE (name)
);
```

主键（PRIMARY KEY）

- 主键的作用：可以保证在一张表中的每一条记录都是唯一的
  - 如果将某一列设置为主键，那么这一列相当于加上了NOT NULL UNIQUE
  - 建议每一张表都有主键
  - 主键最好跟业务无关，常设置为INT AUTO_INCREMENT

- 2种常见写法
  - 列名 数据类型 RIMARY KEY
  - RIMARY KEY（列名）

```
CREATE TABLE company(
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(20) NOT NULL UNIQUE
);

CREATE TABLE company(
        id INT AUTO_INCREMENT,
        name VARCHAR(20) NOT NULL UNIQUE,
        PRIMARY KEY (id)
);
```

外键（FOREIGN KEY）

- 一般用外键来引用其他表的主键
- 常见写法
  - FOREIGN KEY（列名）REFERENCE 表名(列名)

```
CREATE TABLE company(
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(20) NOT NULL UNIQUE
);


CREATE TABLE customer(
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(20) NOT NULL,
        company_id INT NOT NULL,
        FOREIGN KEY (company_id) REFERENCES company(id)
);
```

级联（CASCADE）

- 定义外键时，可以设置级联

- ON DELETE CASCADE

✓ 当删除被引用的记录时，引用了此记录的其他所有记录都会被自动删除

- ON UPDATE CASCADE

✓ 当修改被引用的记录时，引用了此记录的其他所有记录都会被自动更新

多表查询

- 内连接
  - INNER JOIN、CROSS JOIN、JOIN
    - ✓ 在MySQL中，它们是等价的；但是在标准SQL中，它们并不是等价的
- 外连接
  - LEFT [OUTER] JOIN、RIGHT [OUTER] JOIN
- 并集
  - UNION
    - ✓ MySQL并不支持标准SQL中的“FULL [OUTER] JOIN”，但可以用UNION来替代实现
- 参考教程：13.2.9.2 JOIN语法

多表查询

- ON和WHERE后面都可以跟着条件，它们的区别是
  - ON：配合JOIN语句使用，用以指定如何连接表的条件
  - WHERE：限制哪些记录出现在结果集中

- INNER JOIN和逗号（,）在没有连接条件的情况下，语义上是等价的
  - 都在指定的表之间产生笛卡尔乘积（Cartesian Product）
  - 也就是说，第一个表中的每一行都连接到第二个表中的每一行

排序、分页

■ 排序

□ ORDER BY 字段 [ASC | DESC]

■ 分页

□ LIMIT {[offset,] row_count | row_count OFFSET offset}

□ offset是记录的偏移量（最小值是0），从哪一条记录开始选择

□ row_count是希望选择的记录数量

□ 比如

✓ LIMIT 10, 20 或 LIMIT 20 OFFSET 10

✓ 表示从第10条记录开始，选择20条记录

子查询

- 当一个查询是另一个查询的条件时，称之为子查询

- 举例

```
SELECT * FROM customer WHERE company_id = (SELECT id FROM company WHERE
  name = '腾讯')
```