数据库

数据库概述

- **DataBase(DB) 数据库**：存储数据的仓库，数据是有组织的进行存储
- **Database Management SyStem(DBMS) 数据库管理系统**：操纵和管理数据库的大型软件
- **SQL**：操作关系型数据库的编程语言，定义了一套操作关系型数据库统一标准

MySQL 数据库

关系型数据库 (RDBMS)

- **概念**：建立在关系模型上，由多张相互连接的二维表组成的数据库
- **特点**：
    1. 使用表存储数据，格式统一，便于维护
    2. 使用 **SQL** 语言操作，标准统一，使用方便

SQL

- **SQL 通用语法**

    1. SQL 语句可以单行或多行书写，分号结尾
    2. SQL 语句可以使用空格 / 缩进增强语句的可读性
    3. MYSQL 数据库的 SQL 语句不区分大小写
    4. 注释：-- / #

- **SQL 分类**

    1. **DDL (Data Definition)**: 数据定义语言，用来定义数据库对象 (数据库，表，字段)
    2. **DML (Data Manipulate)**: 数据操作语言，用来对数据库表中数据进行增删改
    3. **DQL (Data Query)**: 数据查询语言，用来查询数据库中表的记录
    4. **DCL (Data Control)**：数据控制语言，用来创建数据库用户，控制数据库的访问权限

DDL

DDL - 数据库操作

- 查询所有数据库
  ```
  show databases;
  ```
- 查询当前数据库
  ```
  select database();
  ```
- 创建
  ```
  create database [IF NOT EXISTS] 数据库名 [DEFAULT CHARSET 字符集] [COLLATE 排序规则];
  ```
- 删除
  ```
  drop database [IF EXISTS] 数据库名;
  ```
- 使用
  ```
  use 数据库名;
  ```

DDL - 表操作 - 查询

- 查询当前数据库所有表
  `show tables;`
- 查询表结构
  `desc 表名;`
- 查询指定表的建表语句
  `show create table 表名;`

DDL - 表操作 - 创建

```
create table 表名 (
    字段1 字段1 类型 [COMMENT 字段1 注释],
    字段2 字段2 类型 [COMMENT 字段2 注释],
    )[COMMENT 表注释];
```

> 最后一个字段后面没有逗号

DDL - 表操作 - 数据类型

1. 数值类型
    1. TINYINT: 占用 1 字节
    2. SMALLINT: 占用 2 字节
    3. MEDIUMINT: 占用 3 字节
    4. INT: 占用 4 字节
    5. BIGINT: 占用 8 字节
    6. FLOAT: 占用 4 字节
    7. DOUBLE: 占用 8 字节
    8. DECIMAL: 依赖于 M(精度) 和 D(标度)

    > 可以在数值类型后添加 UNSIGNED 表明为无符号数值
    > 使用 DOUBLE 类型时，要指定整体长度和小数长度。eg: DOUBLE(4, 1)

2. 字符串类型
    1. CHAR: 定长字符串
    2. VARCHAR: 变长字符串
    3. TINYBLOB: 长度不超过 255 的二进制数据
    4. TINYTEXT: 短文本字符串
    5. BLOB: 二进制形式的长文本数据
    6. TEXT: 长文本数据
    7. MEDIUMBLOB: 二进制形式的中等长度文本数据
    8. MEDIUMTEXT: 中等长度文本数据
    9. LONGBLOB: 二进制形式的极大文本数据
    10. LONGTEXT: 极大文本数据

    > CHAR 会根据初始化长度分配空间 (性能好)
    > VARCHAR 会根据存储内容分配合适的空间 (性能较差)

3. 日期类型
    1. `DATE`: 日期值，格式：YYYY-MM-DD
    2. `TIME`: 时间值或持续时间，格式：HH:MM:SS
    3. `YEAR`: 年份值，格式：YYYY
    4. `DATETIME`: 混合日期和时间值，格式：YYYY-MM-DD HH:MM:SS
    5. `TIMESTAMP`: 混合日期和时间值、时间戳，格式：YYYY-MM-DD HH:MM:SS (年份范围到 2038)

DDL - 表操作 - 修改

- 添加字段
  ```
  ALTER TABLE 表名 ADD 字段名 类型(长度) [COMMENT 注释] [约束];
  ```
- 修改字段名和字段类型
  ```
  ALTER TABLE 表名 CHANGE 旧字段名 新字段名 类型(长度) [COMMENT 注释] [约束];
  ```
- 删除字段
  ```
  ALTER TABLE 表名 DROP 字段名;
  ```
- 修改表名
  ```
  ALTER TABLE 旧表名 RENAME TO 新表名;
  ```
- 删除表
  ```
  DROP TABLE [IF EXISTS] 表名;
  ```
- 删除指定表，并重新创建该表
  ```
  TRUNCATE TABLE 表名;
  ```

  > 删除表时，表中的全部数据也会被删除

DML

DML - 字段操作 - 添加数据

1. 给指定字段添加数据
   ```
   INSERT INTO 表名(字段名1, 字段名2, ...) VALUES(值1, 值2, ...);
   ```
2. 给全部字段添加数据
   ```
   INSERT INTO 表名 VALUES(值1, 值2, ...);
   ```
3. 批量添加数据
   ```
   INSERT INTO 表名(字段名1, 字段名2, ...) VALUES(值1, 值2, ...),(值1, 值2, ...);
   INSERT INTO 表名 VALUES(值1, 值2, ...),(值1, 值2, ...);
   ```

   > - 插入数据时，指定的字段顺序需要和值的顺序一一对应
   > - 字符串和日期型数据需要包含再引号中
   > - 插入数据大小应该在字段的规定范围内

DML - 数据操作 - 插入

1. 给指定字段添加数据
   ```
   INSERT INTO 表名(字段名1,字段名2,...) VALUES(值1,值2,...);
   ```

2. 给全部字段添加数据
   `INSERT INTO 表名 VALUES(值1,值2,...);`

3. 批量添加数据
   `INSERT INTO 表名(字段名1,字段名2,...),(字段名1,字段名2,...) VALUES(值1,值2,...),(值1,值2,...);`
   `INSERT INTO 表名 VALUES (值1,值2,...),(值1,值2,...);`

DML - 数据操作 - 修改

1. 修改数据
   `UPDATE 表名 SET 字段名1 = 值1,字段名2 = 值2,... [WHERE 条件];`

   > 修改语句的条件非必需，若没有条件，则修改整张表的数据

2. 删除数据
   `DELETE FROM 表名 [WHERE 条件];`

   > - `DELETE` 语句的条件非必需，若没有条件，则会删除整张表的所有数据
   > - `DELETE` 语句不能删除某一个字段的值 (可以使用`UPDATE`)

DQL

- 语法

```
SELECT
	字段列表
FROM
	表名列表
WHERE
	条件列表
GROUP BY
	分组字段列表
HAVING
	分组后条件列表
ORDER BY
	排序字段列表
LIMIT
	分页参数
```

DQL - 基本查询

1. 查询多个字段
   `SELECT 字段1, 字段2, 字段3... FROM 表名;`
   `SELECT * FROM 表名;`
2. 设置别名 (增强字段的可读性)
   `SELECT 字段1 [AS 别名1], 字段2 [AS 别名2]...FROM 表名;`
3. 去除重复记录
   `SELECT DISTINCT 字段列表 FROM 表名;`

DQL - 条件查询

1. 语法
   `SELECT 字段列表 FROM 表名 WHERE 条件列表;`
2. 条件：
   - 比较运算符：
     - 大于 / 大于等于：`> >=`
     - 小于 / 小于等于：`< <=`
     - 等于：`=`
     - 不等于：`<>` 或 `!=`
     - 某个范围内 []：`BETWEEN...AND...`
     - 在 IN 后的条件列表中的值，多选一：`IN(...)`
     - 是否为 NULL：`IS NULL`
     - 模糊匹配：`LIKE 占位符` (_ 匹配单个字符，% 匹配任意个字符)
   - 逻辑运算符：
     - 并且：`AND` 或 `&&`
     - 或者：`OR` 或 `||`
     - 非：`NOT` 或 `!`

DQL - 聚合函数

1. 概念：将一列数据作为一个整体，进行纵向计算
2. 常见聚合函数：
   1. **count**：统计数量
   2. **max**：最大值
   3. **min**：最小值
   4. **avg**：平均值
   5. **sum**：求和
3. 语法
   `SELECT 聚合函数(字段列表) FROM 表名;`

> 所有 **null** 值不参与聚合函数的统计

DQL - 分组查询

1. 语法：
   `SELECT 字段列表 FROM 表名 [WHERE 条件] GROUP BY 分组字段名 [HAVING 分组后过滤条件];`
2. `WHERE` 和 `HAVING` 的区别
   - 执行时机不同：`WHERE` 是分组之前进行过滤，不满足 `WHERE` 条件不参与分组； `HAVING` 是分组之后对结果进行过滤
   - 判断条件不同：`WHERE` 不能对聚合函数进行判断，`HAVING` 可以

> 注意：
>
> - 分组之后，查询的字段一般未聚合函数和分组字段，查询其他字段无任何意义
> - 执行顺序：`WHERE` > 聚合函数 > `HAVING`
> - 支持多字段分组，语法为 `GROUP BY 字段1, 字段2`