MySQL

- 概述
  - 数据库发展历史
    - 萌芽阶段
      - 文件系统
    - 第一代数据库
      - 网状、层次模型的数据库
    - 第二代数据库
      - 关系型数据库和结构化查询语言
    - 新一代数据库
      - 关系-对象型数据库
  - 什么是数据库
    - 存储、管理数据的仓库
    - DataBase System（DBS）
  - 关系模型
    - 把世界看成由实体和联系组成；
    - 实体(行)：现实世界中客观存在、并且相互区分的事务；
    - 实体可以由多个属性(列)描述；
    - 表是关系型数据库的核心单元；
  - 关系型数据库
    - Relational DataBase System（RDBS）
    - 以关系模型建立的数据库；
  - 表之间的联系(外键)
    - 一对一
    - 一对多
    - 多对多
  - 关系型数据库管理系统
    - Relational DataBase Management System（RDBMS
    - 管理操作数据库数据
  - 常见的数据库
    - Oracle
      - Oracle公司
      - 大型数据库
      - 收费昂贵
    - MSSQL(Microsoft SQL Server)
      - 微软
      - 中型数据
      - 收费
    - MySQL
      - 现属于Oracle
      - 小型数据
      - 有开源版(社区版)
  - 命令
    - 安装
      - mysqld -install 服务名称
    - 卸载
      - mysqld -remove 服务名称
    - 启动服务(windows)
      - net start 服务名称
    - 关闭服务(windows)
      - net stop 服务名称
    - 登录MySQL服务
      - mysql -u账号 -p
    - 退出登录
      - exit

- 目录
  - bin(binary)
    - 二进制可执行文件
  - data
    - 数据库数据文件及日志文件
- 配置
  - 端口号：
    - port=3306
  - 编码：
    - 服务器编码:
      - character-set-server=utf8
    - 默认编码：
      - default-character-set=utf8
  - 引擎：
    - default-storage-engine=INNODB
  - 安装目录:
    - basedir = E:\mysql
  - 数据文件目录：
    - datadir = E:\mysql\data
  - 服务ID(ID为1为开机自启动)：
    - server_id = 1
- SQL
  - Structured Query Language
  - 结构化查询语言
  - 一种用于管理关系型数据库，并与数据库中的数据进行通讯的计算机语言。
  - 分类:
    - DDL(data defination language) 数据定义语言
    - DML(data manipulation language) 数据操纵语言
    - DQL(data query language) 数据查询语言
    - DCL(data control language) 数据控制语言

- SQL
  - DDL(数据库、表)
    - 数据库
      - 创建
        - CREATE DATABASE 数据库名称 [CHARACTER SET 编码集];
      - 删除
        - DROP DATABASE 数据库名称;
      - 修改编码集
        - ALTER DATABASE 数据库名称 CHARACTER SET 编码集;
      - 查看MySQL所有数据库
        - SHOW DATABASES;
      - 查看数据库创建语句
        - SHOW CREATE DATABASE 数据库名称;
      - 连接数据库
        - USE 数据库名称;
    - 表
      - 创建
        - CREATE TABLE 表名(
          - 列名 数据类型 [约束],
          - 列名 数据类型 [约束],
          - .....
        -);
        - 数据类型：

- 整数数据类型

类型	大小	范围(有符号)	范围(无符号)	用途
TINYINT	1B	(-128，127)	(0，255)	极小整数值
SMALLINT	2B	(-32 768，32 767)	(0，65 535)	小整数值
MEDIUMINT	3B	(-8 388 608， 8 388 607)	(0，16 777 215)	小整数值
INT	4B	(-2 147 483 648， 2 147 483 647)	(0，4 294 967 295)	整数值
BIGINT	8B	(-9 233 372 036 854 775 808， 9 223 372 036 854 775 807)	(0，18 446 744 073 709 551 615)	大整数值

- 浮点数据类型

类型	大小	范围	精度	用途
FLOAT	4B	(-3.40E-38， 3.40E+38)	7位小数	单精度浮点数
DOUBLE	8B	(-8 388 608， 8 388 607)	15位小数	双精度浮点数

- 定点数据类型

类型	大小	范围	精度	用途
DECIMAL(M,D)	17B	(-10的38次方-1， 10的38次方-1)	30位小数	大浮点数

注意：

1. M为总数，D为小数，M必须大于D

- 注意：浮点型效率高但有精度丢失问题，存储近似值；定点效率低，但能保证精度；

- 字符串类型

类型	大小	范围	用途
CHAR(M)	M	0-255	字符型
VARCHAR(M)	M+1B	0-65535	字符型
TINYTEXT	LENGTH+2B	0-255	文本型
TEXT	LENGTH+2B	0-65535	文本型
BINARY(M)	M	0-M	0-M变长字符串
VARBINARY(M)	M+1B	0-M	0-M变长字符串

- CHAR(字符数)与VARCHAR(字符数)区别：

- CHAR效率高，但浪费空间(为存储数据的空间填空格);
- VARCHAR效率低，但节省空间，以实际存储值长度为准;

- 二进制数据类型

类型	大小	范围	用途
TINYBLOB	255	0-255	二进制大对象
BLOB	65K	0-65KB	二进制大对象
MEDIUMBLOB	16M	0-16M	二进制大对象
LONGBLOB	4G	0-4G	二进制大对象

- 日期和时间数据类型

类型	大小	格式	范围
YEAR	1B	YYYY	1901 - 2155
DATE	3B	YYYY-MM-DD	1000-01-01 - 9999-12-31
TIME	3B	HH:MM:SS	-835 : 59 : 59- 838:59 : 59
TIMESTAMP	4B	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:01 - 2038
DATETIME	8B	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00 - 9999-12-31 23:59:59

- TIMESTAMP与DATETIME区别：
  - 表示范围上，TIMESTAMP占4B,DATETIME占8B，DATETIME大于TIMESTAMP;
  - TIMESTAMP为时间戳，可以把当前操作时间作为当前行的此列默认值；

- 删除
  - DROP TABLE 表名;
- 修改
  - 修改表名:
    - ALTER TABLE 旧表名 RENAME 新表名;
  - 添加列:
    - ALTER TABLE 表名 ADD 新列名 数据类型;
  - 修改列：
    - ALTER TABLE 表名 CHANGE 旧列名 新列名 数据类型;
    - ALTER TABLE 表名 MODIFY 列名 数据类型;
  - 删除列:
    - ALTER TABLE 表名 DROP 列名;
- 查看当前数据库所有表
  - SHOW TABLES;
- 查看表的创建语句
  - SHOW CREATE TABLE 表名;
- 复制表
  - 复制表结构和数据
    - CREATE TABLE 新表名 AS (SELECT * FROM 旧表名);

        - 复制表结构
            - CREATE TABLE 新表名 LIKE 旧表名;
- DML
    - 插入
        - 指定插入值
            - INSERT INTO 表名[(列名1,列名2,...)] VALUES(列1值,列2值,...),(列1值,列2值,...),...;
                - INSERT INTO class(id,cname) VALUES(1,"一年级一班"),(2,"一年级二班");
        - 插入值来源于另一张表
            - INSERT INTO 表名[(列名1,列名2,...)] SELECT 列名1,列名2,... FROM 目标表;
    - 更新
        - UPDATE 表名 SET 列名1=值1,列名2=值2 [WHERE 列名=值];
            - UPDATE class SET id=3 WHERE cname="二年级一班";
    - 删除
        - DELETE FROM 表名 [WHERE 列名=值];
            - DELETE FROM class WHERE id=2;
    - 清空表
        - DELETE FROM 表名;
        - TRUNCATE TABLE 表名;
        - 区别：
            - 1、TRUNCATE自增列会重置,DELETE不会;
            - 2、TRUNCATE只能清空表所有数据，DELETE可以只删除部分;
            - 3、TRUNCATE不会生成删除日志，数据不能回滚，但效率较高，DELETE会生成删除日志，但效率低;
- 约束
    - 数据完整性：
        - 准确性+可靠性=数据完整性;
        - 分类：
            - 域完整性；
            - 实体完整性；
            - 引用完整性；
            - 自定义完整性；
    - 概述
        - 非空
            - NOT NULL
            - 该列值不能为NULL
            - 语法：
                - CREATE TABLE 表名(
                    - 列名 数据类型 NOT NULL
                -);
        - 默认
            - DEFAULT
            - 设置默认值
            - 语法：
                - CREATE TABLE 表名(
                    - 列名 数据类型 DEFAULT 值
                -);
        - 唯一
            - UNIQUE
            - 某一列的值在表中的该列必须唯一
            - 语法：
                - CREATE TABLE 表名(
                    - 列名 数据类型,
                    - ...
                    - [CONSTRAINT 约束名称] UNIQUE KEY(列名1，列名2,...)
                -);
        - 主键
            - PRIMARY KEY
            - 表中有一列或几列组合的值能用来唯一地标识表中的每一行
            - 非空+唯一+索引

        - 注意：
            - 1、每张表都应该设置主键列;
            - 2、每张表只能有一个主键;
            - 3、选择主键列时应遵循最少(最好单列)、较稳(更新频率较低)原则；
        - 语法：
            - 列级定义：
                - CREATE TABLE 表名(
                - 列名 数据类型 PRIMARY KEY
                -);
            - 表级定义：
                - CREATE TABLE 表名(
                    - 列名 数据类型,
                    - ...
                    - [CONSTRAINT 约束名称] PRIMARY KEY(列名1，列名2,...)
                -);
    - 自增
        - AUTO_INCREMENT
        - 该列值自动设置值，由1开始，每次自增1；
        - 在整型的主键列上添加自增;
        - 语法：
            - CREATE TABLE 表名(
            - 列名 数据类型 PRIMARY KEY AUTO_INCREMENT
            -);
    - 外键
        - FOREIGN KEY
        - 该列的值必须来源于另外一张表的主键或者唯一键
            - 其中“另外一张表”称主表；
            - 当前表称从表;
        - 注意：
            - 删除数据或表时，先删从表或从表数据，再删主表或主表数据;
            - 创建表时，先创建主表，再创建从表;
        - 语法：
            - CREATE TABLE 表名(
                - 列名 数据类型,
                - ...
                - [CONSTRAINT 约束名称] FOREIGN KEY(外键列名) REFERENCES 表名(列名);
            -);
- 修改表的方式添加约束
    - 非空
        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型 NOT NULL;
    - 默认
        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型 DEFAULT 值;
    - 唯一
        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型 UNIQUE;
    - 主键
        - 语法:
            - ALTER TABLE 表名 ADD PRIMARY KEY(列名);
    - 自增
        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型 PRIMARY KEY AUTO_INCREMENT;
    - 外键
        - 语法:
            - ALTER TABLE 表名 ADD CONSTRAINT 约束名称 FOREIGN KEY(列名) REFERENCES 表名(列名);
- 删除约束
    - 非空

        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型;
    - 默认
        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型;
    - 唯一
        - 语法：
            - ALTER TABLE 表名 DROP INDEX 约束名称;
    - 主键
        - 语法(删除主键时，要先删除自增):
            - ALTER TABLE 表名 DROP PRIMARY KEY;
    - 自增
        - 语法：
            - ALTER TABLE 表名 MODIFY 列名 数据类型;
    - 外键
        - 语法：
            - ALTER TABLE 表名 DROP FOREIGN KEY 约束名称;
- 三大范式
    - 第一范式：
        - 列的原子性；
    - 第二范式：
        - 行的唯一性(主键);
        - 非主键列依赖全部主键列，不是依赖部分(针对联合主键);
        - 第二范式前提先满足第一范式;
    - 第三范式：
        - 非主键列与主键列直接相关，而不是间接相关；
            - 非主键列直接依赖于主键列，不存在传递依赖;
        - 第三范式前提先满足第二范式;

- DQL
    - 简单查询
        - 格式：
            - SELECT 列名1,列名2,... FROM 表名 [WHERE 条件] [ORDER BY 列名 [ASC|DESC]] [LIMIT 起始索引,数据条数];
        - 取别名：
            - 格式:
                - SELECT 列名1 AS "别名1",列名2 AS "别名2",... FROM 表名;
                    - SELECT sname AS "姓名",tel AS "电话" FROM student;
                    - SELECT 工资 + 奖金 AS '收入' FROM Employees
                - 注意：AS关键字可省;
        - 限制行数(分页)：
            - 格式:
                - SELECT 列名1,列名2,... FROM 表名 LIMIT [起始索引,]数据条数;
                    - SELECT * FROM student LIMIT 2;
                    - SELECT * FROM student LIMIT 0,2;
        - 结果去重：
            - 格式：
                - SELECT DISTINCT 列名1,列名2,... FROM 表名;
                    - SELECT DISTINCT id,sex FROM student;
            - 注意：只有当查询结果的所有列值相同时，才认为是重复数据被去掉，不是真的某一列查询结果;
        - 多条件查询：
            - SELECT 列名1,列名2,... FROM 表名 WHERE 列名1=值1 AND|OR 列名2=值2;
                - SELECT sname,tel FROM student WHERE sex="女" AND age<25;
        - 范围查询：
            - 格式：
                - SELECT 列名1,列名2,... FROM 表名 WHERE 列名1>=值1 AND 列名1<=值2;
                    - SELECT sname FROM student WHERE age>=25 AND age<=30;
                - SELECT 列名1,列名2,... FROM 表名 WHERE 列名1 BETWEEN 值1 AND 值2;
                    - SELECT sname FROM student WHERE age BETWEEN 25 AND 30;
        - 集合查询：

        - 格式：
            - SELECT 列名1,列名2,... FROM 表名 WHERE 列名1 IN (值1,值2,...);
                - SELECT * FROM student WHERE sname IN ("张三","王八");
    - 模糊查询：
        - 格式：
            - SELECT 列名1,列名2,... FROM 表名 WHERE 列名1 LIKE "通配符表达式";
                - 通配符:
                    - "_":任意一个字符；
                    - "%":任意长度的字符串；
                - SELECT * FROM student WHERE sname LIKE "王_";
                - SELECT * FROM student WHERE sname LIKE "王%";
    - NULL查询：
        - 格式：
            - SELECT 列名1,列名2,... FROM 表名 WHERE 列名1 IS NULL;
                - SELECT * FROM student WHERE sname IS NULL;
    - 常量列：
        - 格式：
            - SELECT 列名1,列名2,...,"常量值" AS "常量列名" FROM 表名;
                - SELECT id,sname,"蜗牛学院" AS "学校名称" FROM student;
    - 查询结果排序：
        - 格式：
            - SELECT 列名1,列名2,... FROM 表名 ORDER BY 列名1 [ASC|DESC],列名2 [ASC|DESC],...;
                - ASC:默认排序，升序;
                - DESC:降序；
                - SELECT * FROM student ORDER BY age ASC;
                - SELECT * FROM student ORDER BY age DESC LIMIT 5;
                - SELECT * FROM student WHERE sex="女" ORDER BY age DESC;
                - SELECT * FROM student ORDER BY age DESC,cid DESC;
- 聚合(统计)函数
    - COUNT(DISTINCT 列)
        - 统计结果集或者组内的数据行数
        - 注意：
            - COUNT(DISTINCT 列)与COUNT(*)区别：
                - 1、COUNT(DISTINCT 列)可以对指定列去重后计数；
                - 2、COUNT(DISTINCT 列)如果指定列值为NULL，该行不会作为计数目标，COUNT(*)会；
    - AVG
        - 统计结果集或者组内中指定列的平均值
    - SUM
        - 统计结果集或者组内中指定列的总和
    - MAX
        - 统计结果集或者组内中指定列的最大值
    - MIN
        - 统计结果集或者组内中指定列的最小值
    - 注意：使用了聚合函数或者分组时，SELECT查询不要出现普通列，可以出现：
        - 1.分组的依据列
        - 2.为每个分组返回一个值的表达式，如聚合函数
- 分组
    - SELECT 列名1,列名2,... FROM 表 WHERE 条件 GROUP BY 列名1,列名2,... HAVING 条件;
    - 将结果集数据按分组依据列的值进行分组(值相同为一组)，对组按HAVING的条件进行筛选；
- 关系型数据库管理系统在执行一条SQL时，按照如下顺序执行各子句。
    - SELECT 列 FROM 表 WHERE 条件 GROUP BY 列 HAVING 条件 ORDER BY 列 LIMIT 条数;
    - 1、首先执行FROM子句，将FROM子句中的表做为中间表；
    - 2、如果有WHERE子句，则根据其中的过滤条件，从中间表中去掉不满足过滤条件的行。
    - 3、根据GROUP BY子句中指定的分组列，对中间表中的数据进行分组。
    - 4、为每个组计算SELECT子句聚合函数的值，并为每组生成查询结果中的一行。
    - 5、如果有HAVING子句，则根据HAVING子句的过滤条件，分组计算聚合计算的结果再次过滤。
    - 6、如果有ORDER BY子句中，则根据ORDER BY子句中的列，对结果集进行排序。
    - 7、如果有LIMIT，只留下符合规则的数据条数

- 子查询(嵌套查询)
  - 嵌入到另一条SQL(SELECT、UPDATE、DELETE)语句中的查询语句；
    - SELECT … FROM 表1 WHERE 字段1 >(子查询)
      - 外面的查询称为父查询，括号中嵌入的查询称为子查询
    - SELECT … FROM (子查询) 别名
      - 子查询可以出现在WHERE后，也可出现在FROM后；
  - 子查询常用的关键字：
    - IN
      - 在子查询结果中取一个值
    - NOT IN
      - 不在子查询结果中的值
    - EXISTS
      - 如果子查询有查询结果(一条及以上)，则为true，否则为false；
    - ANY
      - 满足任意一个子查询结果
    - ALL
      - 同时满足所有子查询结果
- 联合查询
  - 关键字：UNION 或 UNION ALL
  - 将多张结构类似的表内容，合为一个结果集展示;
  - 查询语句1 UNION 查询语句2;
    - 结果去重；
    - SELECT sname FROM student UNION SELECT tname FROM teacher;
  - 查询语句1 UNION ALL 查询语句2;
    - 结果不去重
    - SELECT sname FROM student UNION ALL SELECT tname FROM teacher;
- 视图
  - 由一条SQL查询语句结果形成的虚拟表；
  - 其本身不存储数据，数据来源于基本表；
  - 可以对视图进行各种数据操作(INSERT、UPDATE、DELETE、SELECT)，但直推荐做SELECT操作，做修改操作时直接修改基本表；
  - 语法：
    - CREATE VIEW 视图名称
    - AS
      - SELECT语句;
- 连接查询
  - 笛卡尔积
    - 假设集合A={a, b}，集合B={0, 1, 2}，则两个集合的笛卡尔积为{(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)}
  - SELECT * FROM a,b;
    - a表与b数据由笛卡尔积生成结果集；
  - 分类：
    - 内连(INNER JOIN)
      - 等值连接，笛卡尔积结果筛选符合连接条件的值
      - 只返回两个表中连接字段相等的行
      - 语法：
        - SELECT 列名1,... FROM 表1 INNER JOIN 表2 ON 连接条件 ...;
    - 外连
      - 左外连(LEFT JOIN)
        - 返回包括左表中的所有记录和右表中连接字段相等的记录
        - 语法：
        - SELECT 列名1,... FROM 表1 LEFT JOIN 表2 ON 连接条件 ...;
      - 右外连(RIGHT JOIN)
        - 返回包括右表中的所有记录和左表中连接字段相等的记录
        - 语法：
        - SELECT 列名1,... FROM 表1 RIGHT JOIN 表2 ON 连接条件 ...;
      - 全外连(FULL JOIN)
        - MySQL暂时不支持全外连(full join)，但是我们用左右外连配合着union使用，可以达到全外连的效果