MySQL数据库

- **S1 数据库基本概念与方法**
  - 1 数据库基础知识 27
  - 2 关系数据库：以关系模型作为逻辑数据模型，采用关系作为数据的组织方式。 13
  - 3. 数据库设计基础
    - 3.1 数据库设计步骤
      需求分析 概念结构设计 逻辑结构设计 数据库实施 数据库运行和维护
    - 3.2 关系数据库设计的方法
      遵从数据库设计步骤
      - 概念结构：E-R图来描述现实世界的概念模型
      - 逻辑结构：把已设计好的E-R图转换为关系模型
        - E-R图：实体 实体属性 和实体间联系
          三要素构成；
        - 关系模型：就是一组关系模式的集合
          转换就是需要将E-R图转换为某种关系模式
  - 4 MySQL 概述
    体积小 速度快 开放源代码
    - MySQL服务器安装，使用命令行或者图形化界面来建立与服务器的连接，从而实施各种数据库的操作
      - 命令行方式：执行SQL语句
      - 图形化管理;GUI工具：phpMyAdmin
- **S2 MySQL 编程语言**
  安装MySQL
  - 2.1 结构化查询语言SQL：
    - 是一种专门用来与数据库通信的语言；
      目的就是提供从数据库中读写数据且简单有效的方法。
    - SQL 是所有数据库通用支持的语言
    - 对MySQL数据库交互学习：以SQL为基础，切实多动手多实践
    - SQL语句不区分大小写；
      默认规则：对关键字大写，对列和表的名称使用小写
  - 2.2 MySQL语言组成
    标准SQL语言
    - 数据定义语言DDL：

- 数据定义语言DDL:
  用于对 数据库的对象进行创建 删除 修改等
  - 数据库对象：表 默认约束 规则 视图 触发器 存储过程
  - CREATE：用于创建数据库或数据库对象
  - ALTER: 用于对数据库或数据库对象进行修改
  - DROP: 用于删除数据库或数据库对象
- 数据操纵语言DML:
  主要用于操纵数据库中各种对象，特别是检索和修改数据。
  - SELECT：用于从表中或视图中 检索数据，使用最频繁
  - INSERT：用于将数据插入到表或者视图中
  - DELETE：用于从表或视图中删除数据
- 数据控制语言DCL：
  用于安全管理，管理权限范围等
  - GRANT：用于授予权限
  - REVOKE：用于收回权限，与上面相反

2.3 MySQL函数

- 聚合函数：根据一组数据求出求出一个值
  - 1. COUNT: 返回集合中所有行的数目
  - SUM函数：求出表中某个字段取值的总和
  - AVG()函数：求出某个字段取值的平均值
  - MAX()函数：求出字段中中的最大值
  - MIN()函数：求出字段中的最小值
- 数学函数：处理数字
  - ABS（）函数：求出绝对值
  - FLOOR（）函数：返回小于或等于x的最大数
  - RAND()函数：返回之间随机数
  - TRUNCATE（）：返回保留x后面的几位小数
  - SQRT()函数：求平方根
- 字符串函数
  - UPPER(X)和UCASE(X)函数：将s中所有字母编程小写
  - LEFT()函数：用于返回字符串S中前n个字符
  - SUBSTRING(S,N,LEN):用于从字符串s的第n个位置开始获取长度为len的字符串

- 日期和时间函数

  - CURDATE()和CURRENT_DATE()函数
    用于获取当前日期
  - CURTIME()和CURRENT_TIME()函数
    用于获取当前时间
  - NOW()函数：用于获取当前时间和日期

- 其他函数
  - IF(expr1, v1,v2)：条件判断函数
    - mysql>SELECT studentNo,score,, if(score>85,'优秀'，一般) level from tb_score
  - INPUT(v1,v2)：条件判断函数--表达式v1不为空，则显示v1的值，否则的话显示v2的值
  - VERSION()函数：版本号

S3 数据定义 操作实践
已下载安装好MySQL，涉及数据库的创建 选择 查看 修改和删除等操作

- 3.1 定义数据库
  - 1 创建数据库
    语法格式 CREATE DATABASE
    - 语法格式：　CREATE {DATABSE | SCHEMA} [IF NOT EXISTS] db_name
      [[DEFAULT] CHARACTER SET] [=] charset_name]
      [[DEFAULT] COLLATE] [=] collation_name]
  - 2 选择与查看数据库
    - 选择数据库：USE db_name
    - 查看数据库：SHOW {DATABASES | SCHEMAS}
  - 3 修改数据库
    语法格式：ALTER DATABASE
    或 ALTER SCHEMA 用于更改数据库的全局特性
    - 例子
  - 4 删除数据库
    DROP {DATABASE | SCHEMA} [IF EXITS] db_name;
- 3.2 定义表：成功创建数据库后，就可以在数据库中创建数据表；数据表中是数据库中最重要，最基本的数据对象，是数据存储的基本单位。
  - CREATE TABLE table_name：确定表中每个字段的数据类型是创建表中的重要步骤；

- CREATE TABLE productins
  - (product_id CHAR(4) NOT NULL,
  - product_name VARCHAR(100) NOT NULL,
  - product_type VARCHAR(32) NOT NULL,
  5 sale_price INTEGER DEFAULT 0,
  - purchase_price INTEGER ,
  - regist_date DATE ,
  - PRIMARY KEY (product_id));

- 1. MySQL常用数据类型
  - INTEGER型：
    用来指定存储整数的列的数据类型
  - CHAR型：
    用来指定储定长字符串
  - VARCHAR型：
    用来存储可变长度字符串
  - DATE型：
    用来指定存储日期的列的数据类型
- 2. 指定表名和字段名
  - 需要首先选定数据库 或者直接指定名称为
    db_name.tbl_name.
  - 字段名在表中唯一
- 3. 完整性约束条件
  - 实体性完整约束(PRIMARY KEY、UNIQUE)
  - 参照完整性约束(FOREIGN KEY)
  - 用户自定义约束(NOT NULL、DEFAULT、CHECK约束)等

- 3.3 对表的操作
  - 1. 表的删除 修改
    - 1. 表的删除 DROP TABLE tbl_name
    - 2. 添加列 ALTER TABLE <tbl_name> ADD COLUMN <列的定义>
      - ALTER TABLE product ADD COLUMN product_name_pinyin
        VARCHAR(100);
    - 3. 删除列 ALTER TABLE <表名> DROP COLUMN <列名>;
      - ALTER TABLE product DROP COLUMN product_name_pinyin;

- 4. 清空表内容 TRUNTE TABLE TABLE_NAME;
  优点：相比 Drop / Delete， Truncate 用来清除数据时，速度最快

- 2. 表的更新
  - 语法格式：使用 update 时要注意添加 where 条件，否则将会将所有的行按照语句修改
    UPDATE <表名>
    - SET <列名> = <表达式> [, <列名2>=<表达式2>...];
    - WHERE <条件>; -- 可选， 非常重要。
    - ORDER BY 子句; --可选
    5 LIMIT 子句; --可选
    - UPDATE product
      --修改所有的注册时间
      SET regist_date = '2009-10-10';
      --仅修改部分商品的单价
      UPDATE product
      SET sale_price =sale_price * 10
      WHERE product_type ='厨房用具';
    - 特殊的NULL 清空，表达式后面直接写NULL
      UPDATE product
      SET regist_date = NULL
      WHERE product_id ='0008';
      **只有没设置NULL约束和主键约束的列才可以清空为NULL
  - 多列更新　--合并写法
    UPDATE product
    SET sale_price =sale_price * 10,
    　　purchase_price = purchase_price / 2
    WHERE product_type = '厨房用具';
- 3. 向表中插入数据
  - 语法：
    INSERT INTO <表名> (列1 ，列2，列3，....) VALUES(值1，值2，值3....);
    对表进行全列 INSERT 时，可以省略表名后的列清单。
    - --包含列清单
      INSERT INTO
      productins(product_id,product_name,product_type,sale_price,
      purchase_price,regist_date) VALUES('0005','高压锅，'厨房用具'，6800，
      5000，'2009-01-15');
- 4. 查看表
  查看数据表的名称及表结构的定义
  - 查看表名称
    SHOW TABLES [{FROM | IN} db_name];
  - 查看数据表基本结构

查看数据表基本结构
{DESCRIPE | DESC} tb_name;

- 查看数据表的详细结构：存储引擎和编码
  SHOW CREATE TABLE tb_name;

3.4 练习题

- 1. 创建数据表
- 2. 添加列
- 3. 删除表
- 4. 恢复已删除掉的表

S4 数据查询
SELECT 语句查询

1 从表中选取数据
通过SELECT语句查询并选取出必要数据的过程
称为匹配查询或 查询。

- 基本SELECT 语句语法：
  SELECT <列名>,
  FROM <表名>;
- 从表中选取符合条件的数据：
  WHERE语句 当不需要取出全部数据，而是选取出满足一定条件的数据。通过WHERE来指定查询数据的条件。
  SELECT <列名>,.....
  FROM <表名>
  WHERE <条件表达式>;
  - --用来选取product_type列为 衣服 的记录的
    SELECT product_name,product_type
    FROM product
    where product_type ='衣服';
    --也可以选取出不是查询条件的列(条件列与输出列不同)
    SELECT product_name
    FROM product
    WHERE product_type='衣服'
- 相关法则：
  * 星号(*)代表全部列的意思；
  * SQL中可以随意使用换行符，不影响语句执行
  * 设定汉语别名时需要使用双引号("")括起来
  * 注释符号 :行注释"--" 和多行注释/* */两种。

- -- 要查询全部列时，可以使用代表所有列的星号
  SELECT *
  FROM <表名>
  --SQL语句使用AS 关键字为列设定 别名
  SELECT product_id AS id,
  product_name AS name,
  product_price AS "进货单价"
  FROM product;
  --使用DISTINCT删除列中重复数据
  SELECT DISTINCT product_type
  FROM product

2 运算符

- 算术运算符：加减乘除
- 比较运算符:
  SELECT product_name, product_type
  FROM product
  WHERE sale_price =500;
  - 例子：
    --SQL语句中使用运算表达式
    SELECT product_name,sale_price,sale_price *2
    AS "sale_price x2"
    FROM product;
    --WHERE子句的表达式中也可以使用计算表达式
    SELECT product_name,sale_price,purchase_price
    FROM product
    WHERE sale_price-purchase_price >=500;
  - 例子2：
    --DDL：创建表
    CREATE TABLE chars
    (chr CHAR(3) NOT NULL,
    PRIMARY KEY (chr));
    --选取出大于'2'的数据的SELECT语句
    SELECT chr
    FROM chars
    WHERE chr > '2';
    --选取NULL的记录
    SELECT product_name.purchase_price
    FROM product
    WHERE purchase_price IS NULL;
    --选取不为NULL的记录
    SELECT product_name,purchase_price
    FROM product

FROM product
WHERE purchase_price IS NOT NULL;

- 逻辑运算符
  - NOT 运算符
    - 例子：
      --选取出销售单价大于等于1000日元的记录
      SELECT product_name,product_type,sale_price
      FROM product
      WHERE sale_price >=1000;
      --向代码清单的查询条件中添加NOT运算符
      SELECT product_name,product_type,sale_price
      FROM product
      WHERE NOT sale_price >=1000;
  - AND运算符 OR 运算符
    - 例子：
      --SELECT product_name,product_type,regist_data
      FROM product
      WHERE product_type =' 办公用品'
          AND(regist_data ='2009-09-11' OR regist_date ='2009-09-20');
  - 借助 工具真值表 使用
    - 画真值表来理清逻辑关系
- 第一部分练习题
  - 3.1 解：
    SELECT product_name, regist_data
    FROM product
    WHERE regist_date >"2009-04-28"
  - 3.2
    1. 解：
    返回段名 价格为空时的包含所有段名记录
    2. 解：
    返回段名 价格不为空时的包含所有段名记录
  - 3.3 解：
    SELECT product_name ,sale_price,purchase_price
    FROM product
    WHERE sale_price - purchase_price >5000;

- 3.4 解：
  --增加一个段名 利润
  ALTER TABLE product ADD COLUMN profit INTEGER;
  --怎么对段名添加数据呢

  --开始选择查询
  SELECT product-name,product_type，profit
  FROM product
  WHERE sale_price * 0.9 -pruchase_price > 100;

3. 对表进行分组聚合查询

- 3.1 聚合函数：
  COUNT : 用于计算表中的记录数
  SUM : 计算表中数值列的数据合计值
  AVG: 计算表中数值列中数据的平均值
  MAX: 求出表中任意列中数据的最大值
  MIN ： 求出表中任意列中数据的最小值

  - 例子：
    --计算全部数据的行数(包含NULL)
    SELECT COUNT(*)
    FROM product；
    --计算NULL以外数据的行数
    SELECT COUNT(purchase_price)
    FROM product ;
    --计算销售单价和进货单价的合计值
    SELECT SUM(sale_price),SUM(purchase_price)
    FROM product;
    --计算销售单价和进货单价的平均值
    SELECT AVG(sale_price),AVG(purchase_price)
    FROM product;
    --MAX和MIN也可用于非数值型数据
    SELECT MAX(regist_date),MIN(regist_date)
    FROM product;

- 常用法则：
  * COUNT函数的结果会根据参数的不同而不同。COUNT(*) 会得到包含 NULL 的数据行数，而 COUNT(<列名 >) 会得到 NULL 之外的数据行数。
  * 聚合函数会将NULL排除在外。但COUNT(*)例外，并不会排除NULL
  * MAX/MIN 函数几乎适用于所有数据类型的列。SUM/AVG 函数只适用于数值类型的列。
  * 想要计算值的种类的时，可以在COUNT函数的参数中使用DISTINCT
  * 在聚合函数的参数中使用DISTINCT，可以删除重复数据

4. 对表进行分组
GROUP BY 子句指定的列称为聚合键

GROUP BY 子句指定的列称为聚合键
或者 分组列

- 1. GROUP BY 分组汇总
  SELECT <列1> <列2> <列3>....
  FROM <表名>
  GROUP BY <列1>，<列2>，<列3>，....;
  - 例子：
    --按照商品种类统计数据行数
    SELECT product_type, COUNT(*)
    FROM product
    GROUP BY product_type;
    -- 不含GROUP BY
    SELECT product_type,COUNT(*)
    FROM product
- SELECT <列名1>,<列名2>,<列名3>,....
  FROM <表名>
  GROUP BY <列名1>， <.列名2>,<列名3> , ...;
  看一看是否使用GROUP BY 语句的差异；
  - 聚合键中包含NULL时
    SELECT purchase_price, COUNT(*)
    FROM product
    GROUP BY purchase_price;
- 书写顺序：
  1 SELECT 2 FROM 3 WHERE 4 GROUP BY
  - SELSCT purchase_price ,COUNT(*)
    FROM product
    WHERE product_type = '衣服'
    GROUP BY purchase_price;

- 5. HAVING 特点 GROUP BY + HAVING
  HAVING 子句用于对分组进行过滤，可以使用数字、聚合函数和 GROUP BY 中指定的列名（聚合键）
  - 例子：
    --SELECT product_type,COUNT(*)
    FROM product
    GROUP BY product_type
    HAVING COUNT(*) =2;
    --错误形式，因为product_name不在GROUP BY 聚合键中
    SELECT product_type, COUNT(*)
    FROM product
    GROUP BY product_type
    HAVING product_name='圆珠笔';
  - SELECT product_id,

- 6. 对查询结果进行排序
  ORDER BY：SQL中的执行结果是随机排列的，需要特定排序时，可使用该子句；
  是调节 行(记录)的顺序
  - SELECT <列名1>,<列名2>, <列名3>,...
    FROM <表名>
    ORDER BY <排序基准列1 >， <排序基准列2>,
    - 例子：
      -- 降序排列
      SELECT prodct_id，product_name, sale_price, purchase_price
      FROM product
      ORDER BY sale_price DESC;

      --多个排序列
      SELECT product-id, product_name,sale_price,purcahse
      FROM product
      ORDER BY sale_price, product_id;

      --当用于排序的列名中含有NULL时，NULL会在开头或末尾斤西瓜汇总
      SELECT product_id, product_name,purchase_price
      FROM product
      ORDER BY purchase_price ;
  - ORDER BY 中可使用SELECT设置的别名
    SQL在使用HAVING子句时SELECT语句的顺序为
    FROM WHERE GROUP BY HAVING SELECT ORDER BY
- 7 总结 语法格式：单表查询
  SELECT [ALL|DISTINCT|DISTINCTROW] 列名...
  FROM tbl_name
  WHERE <条件表达式>
  GROUP BY <上面列名> [HAVING <条件表达式>]
  ORDER BY <列名2> [ASC | DESC]
  LIMIT [m,] n ;
  - SELECT 子句：用于指定要现实的字段或表达式
  - FROM 子句用于指定数据来源于哪些表或视图
  - WHERE子句用于指定对记录的过滤条件；
    字段别名不允许出现在WHERE子句中
    - 比较大小：> < =
      范围：BETWEEN ....AND....
    - IN 关键字查询
    - 带有LIKE关键字字符串匹配查询

- 带有LIKE关键字的单匹配查询
  - 带有正则表达式查询
  - 带有AND 和OR的逻辑表达式查询
- GROUP BY 子句用于将查询结果集按指定的字段分组
- HAVING子句用于指定分组结果集的过滤条件
- ORDER BY 将查询结果按照指定字段值进行升序或者降序
- LIMIT 用于指定查询结果集包含的记录数

- 8. 连接查询
- 9. 子查询

- 8. 练习习题
  - 1. 语言改错题
    SUM后面的括号需要使用英文括号
    单行注释后面需要加一个空格 " "
    GROUP BY 和WHERE的顺序不对
    - GRROUP BY 后就变为组形式；WHERE就只能进行行行过滤；
      若是对组过滤，使用HAVING;或者WHERE提至GROUP VY之前，即在查询结果未分组(仍为行形式)
      通常情况下，GROUP BY后的product_type应该在SELECT后出现，否则看不到分组依据；
  - 2. 解
    SELECT product_type,SUM(sale_price) as 'sum',SUM(purchase_price) as 'sum'
    FROM product
    WHERE SUM(sale_price) > 1.5 * SUM(purchase_price)
    GROUP BY product_type；
    - 若要是不加 GRUOP BY 会怎么样

S5 表的更新 索引 视图

- 表的更新：
  插入数据 INSERT INTO
  修改数据 UPDATE
  删除数据 DELETE
  - 1 插入数据
    INSER INTO tbl_name col_list VALUES val_list；
    ** 要求字段和数据值的数量必须相同，并要保证每个插入值的类型 顺序 与对应字段定义的数据类型 顺序匹配。
    - 1. 插入记录 例子：
      INSERT INTO tb_student(studentNo,
      studentName,sex,birthday,native,nation,classNo)
      VALUE('2013110102'，'赵婷婷'，'女'，'1996-11-30'，'天津'，'汉'，'AC1301')

- 2. 将SELECT查询结果插入新表中
  INSERT INTO tb_name1 (col_list1)
  SELECT col_list2 FROM tbl_name WHERE(condition)；
- 3. 使用REPLACE 语句插入表数据
  若待插入的表中存在有 PRIMARY KEY或UNIIQUE约束，且待插入行也包含这些约束，则无法插入。需使用REPLACE语句来实现。

- 2. 修改数据记录
  UPDATE tb_name
  SET col1=val1,col2 = val2,....,coln=valn
  [WHERE <condition>]; -- 用于限定表中要修改的行
  - 1. 修改特定数据记录
    UPDATE db_school.tb_student
    SET studentName='黄涛'，native = '湖北'，nation='汉'
    WHERE studentNo = '2014210101';
  - 2. 修改所有数据记录
    UPDATE db_school.tb_score
    SET score=score * 1.05;
  - 3. 带子查询的修改
    WHERE 子句中使用嵌套子查询，在修改的表和条件的表不想同时

- 3. 删除数据记录
  DELETE FROM tbl_name
  [WHERE <conditions>]:
  - 1. 删除特定查询记录
    DELETE FROM db_student
    WHERE studentName = '王一敏'
  - 2. 带子查询的删除
    WHERE子句中嵌套子循环，构造删除条件；在删除数据数据的表与设置删除条件的表不想同时
  - 3. 删除所有数据记录

- 索引：建立索引的目的就是加快数据库检索速度

Windows下安装MySQL

1. 安装包安装
   - 下载安装包：
     百度云：https://pan.baidu.com/s/1SOtMoVqqRXwa2qD0siHcIg]
     密码：80lf
   - 安装教程：DataWhale 学习教程
     CSDN上教程

CSDN上教程

- 目录内容

  - bin目录下保存 MySQL常用的工具命令及管理工具
  - data目录是MySQL默认用来保存数据文件以及日志文件的地方
  - docs目录下是MySQL上的帮助文档
  - include和lib目录是MySQL依赖的头文件以及库文件
  - share目录下保存目录文件以及日志文件

- 用于连接MySQL并执行查询
  命令行程序或其他有好的应用程序

  - 命令行方式：打开第二个并输入密码进行连接

    

  - 可视化客户端：MySQL Workbench HeidiSQL DBeaver ； 商业软件：Navicat SQLyog免费社区版 DataGrip等

- MySQL Workbench 客户端工具
  既可以用于设计数据库；也可以用于连接数据进行查询；

  - 照着DataWhale的界面进行连接