# 数据库3-多表联查-左右连接-存储过程

- 多表查询：在关系型数据库中，可以通过表与表之间的共有字段，将多个表关联，可以更广泛的查询数据
- 数据库中有数据表，表中有字段
- 在sql语句中，.可以理解成一个层级关系
  - select 表.字段 from 数据库.表；
- 内连接：多表通过共有字段关联，取公共部分
- 关键字：inner join on
- 命令公式：select 字段 from 表1 inner join 表2 on 表1.字段=表2.字段；

```
27    SELECT
28      test08.id,
29      test08.name,
30      test08.math,
31      test09.hobby
32    FROM
33      test08
34      INNER JOIN test09
35        ON test08.name = test09.name
36    WHERE test09.hobby = '钓鱼' ;
37
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

（只读）

| id | name | math | hobby |
|---|---|---|---|
| 1 | 张三 | 90 | 钓鱼 |

- *多表联查需要添加关联条件，如果不加会得到笛卡尔积：两个集合相乘的结果

```
21    SELECT
22      *
23    FROM
24      test08
25      INNER JOIN test09 ;
```

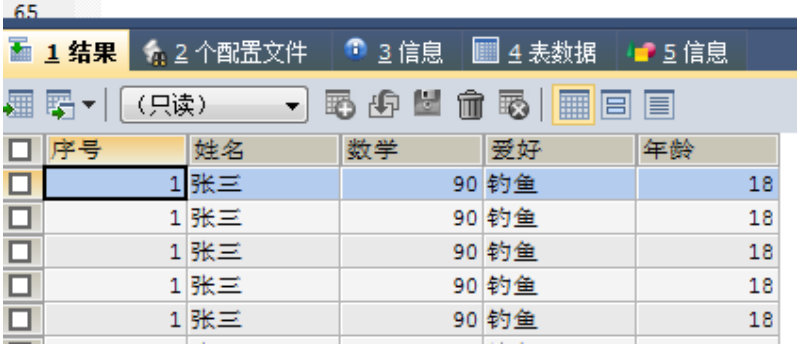
1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

（只读）

| id | NAME | chinese | math | english | id | name | gender | hobby |
|---|---|---|---|---|---|---|---|---|
| 1 | 张三 | 83 | 90 | 76 | 1 | 张三 | 男 | 钓鱼 |
| 2 | 李四 | 76 | 92 | 81 | 1 | 张三 | 男 | 钓鱼 |
| 3 | 王五 | 88 | 78 | 81 | 1 | 张三 | 男 | 钓鱼 |
| 4 | 白云 | 75 | 93 | 82 | 1 | 张三 | 男 | 钓鱼 |
| 5 | 黑土 | 89 | 79 | 90 | 1 | 张三 | 男 | 钓鱼 |
| 1 | 张三 | 83 | 90 | 76 | 2 | 李四 | 男 | 游泳 |
| 2 | 李四 | 76 | 92 | 81 | 2 | 李四 | 男 | 游泳 |
| 3 | 王五 | 88 | 78 | 81 | 2 | 李四 | 男 | 游泳 |
| 4 | 白云 | 75 | 93 | 82 | 2 | 李四 | 男 | 游泳 |
| 5 | 黑土 | 89 | 79 | 90 | 2 | 李四 | 男 | 游泳 |
| 1 | 张三 | 83 | 90 | 76 | 3 | 王五 | 女 | 逛街 |
| 2 | 李四 | 76 | 92 | 81 | 3 | 王五 | 女 | 逛街 |
| 3 | 王五 | 88 | 78 | 81 | 3 | 王五 | 女 | 逛街 |
| 4 | 白云 | 75 | 93 | 82 | 3 | 王五 | 女 | 逛街 |

  - 命令公式：select 字段 from ( 表1 inner join 表2 on 表1.字段=表2.字段) inner join 表3 on 表2.字段=表3.字段；

```sql
SELECT
  t1.id 序号,
  t1.name 姓名,
  t1.math 数学,
  t2.hobby 爱好,
  t3.age 年龄
FROM
  (
    test08 t1
    INNER JOIN test09 t2
      ON t1.name = t2.name
  )
  INNER JOIN test07 t3
    ON t3.name = t2.name
WHERE t2.hobby = '钓鱼' ;
```



1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

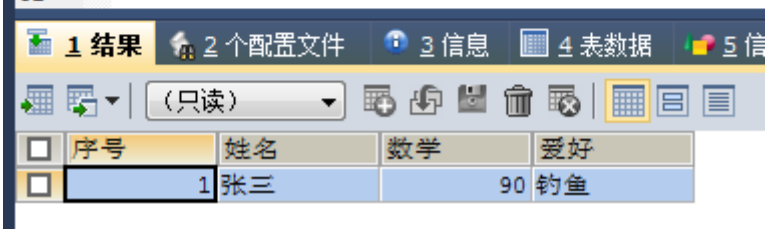
(只读)

| 序号 | 姓名 | 数学 | 爱好 | 年龄 |
|---|---|---|---|---|
| 1 | 张三 | 90 | 钓鱼 | 18 |
| 1 | 张三 | 90 | 钓鱼 | 18 |
| 1 | 张三 | 90 | 钓鱼 | 18 |
| 1 | 张三 | 90 | 钓鱼 | 18 |
| 1 | 张三 | 90 | 钓鱼 | 18 |

- 取别名：通过关键字as来给表或者字段定义一个临时变量名
  - as可以省略
- ***隐式内连接：不使用inner join on 关键字来实现内连接
  - 实现方式：通过where来实现
  - 命令公式：select 字段 from 表1，表2，...，表n where 表1.字段=表2.字段 and 表2.字段=表n.字段 and 其他筛选条件；

```sql
SELECT
  t1.id 序号,
  t1.name 姓名,
  t1.math 数学,
  t2.hobby 爱好
FROM
  test08 t1,
  test09 t2
WHERE t1.name = t2.name
  AND t2.hobby = '钓鱼' ;

SELECT
  *
```



1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信

(只读)

| 序号 | 姓名 | 数学 | 爱好 |
|---|---|---|---|
| 1 | 张三 | 90 | 钓鱼 |

  - 查询爱好为钓鱼的人的数学成绩
- 
- 左/右连接：以左/右表为基准，右/左表与之匹配，右表中多余的数据不显示，缺少的数据用null补充

- 关键字：left/right join on
- 命令公式：select 字段 from 表1 left/right join 表2 on 表1.字段=表2.字段；

```
SELECT
  *
FROM
  test08
  LEFT JOIN test09
    ON test08.name = test09.name ;
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

（只读）

| id | NAME | chinese | math | english | id | name | gender | hobby |
|---|---|---|---|---|---|---|---|---|
| 1 | 张三 | 83 | 90 | 76 | 1 | 张三 | 男 | 钓鱼 |
| 2 | 李四 | 76 | 92 | 81 | 2 | 李四 | 男 | 游泳 |
| 3 | 王五 | 88 | 78 | 81 | 3 | 王五 | 女 | 逛街 |
| 4 | 白云 | 75 | 93 | 82 | (NULL) | (NULL) | (NULL) | (NULL) |
| 5 | 黑土 | 89 | 79 | 90 | (NULL) | (NULL) | (NULL) | (NULL) |

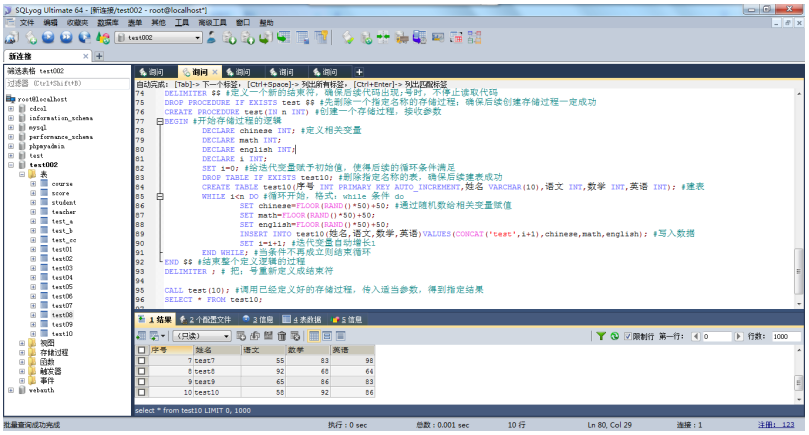
```
SELECT
  *
FROM
  test08
  RIGHT JOIN test09
    ON test08.name = test09.name ;
```

1 结果 | 2 个配置文件 | 3 信息 | 4 表数据 | 5 信息

（只读）

| id | NAME | chinese | math | english | id | name | gender | hobby |
|---|---|---|---|---|---|---|---|---|
| 1 | 张三 | 83 | 90 | 76 | 1 | 张三 | 男 | 钓鱼 |
| 2 | 李四 | 76 | 92 | 81 | 2 | 李四 | 男 | 游泳 |
| 3 | 王五 | 88 | 78 | 81 | 3 | 王五 | 女 | 逛街 |
| (NULL) | (NULL) | (NULL) | (NULL) | (NULL) | 4 | 王朝 | 女 | 咖啡 |
| (NULL) | (NULL) | (NULL) | (NULL) | (NULL) | 5 | 马汉 | 男 | 睡觉 |

- **什么时候使用左/右连接：确保某一个表的数据完整性的时候
-
- 存储过程：自定义一个向数据库读取或写入数据的逻辑
- 例如：定义一个逻辑，传入大于零的整数n，自动生成n行随机的成绩表
- 定义存储过程的关键字：CREATE PROCEDURE
- 定义存储过程的通用过程
  - 先通过DELIMITER定义一个临时的结束符号，使得；号暂时不作为结束符，只充当当前行代码结束
  - 通过CREATE PROCEDURE 存储过程的名称（是否有参数出入），来定义一个存储过程的名称
    - 存储过程的名称不能重复
  - 通过BEGIN关键字，开始存储过程的逻辑
    - 存储过程的逻辑中可能涉及定义新的变量、循环、判断等逻辑分支
    - DECLARE:定义新的局部变量
    - SET:给变量赋值
    - WHILE:基于条件成立的循环
  - 通过END关键字来结束逻辑的定义
  - 最后把；重新定位成结束符，并且通过call关键字调用定义好的存储过程，传入参数实现逻辑



- DELIMITER $$ #定义一个新的结束符，确保后续代码出现;号时，不停止读取代码
- DROP PROCEDURE IF EXISTS test $$ #先删除一个指定名称的存储过程；确保后续创建存储过程一定成功
- CREATE PROCEDURE test(IN n INT) #创建一个存储过程，接收参数
- BEGIN #开始存储过程的逻辑
- DECLARE chinese INT; #定义相关变量
- DECLARE math INT;
- DECLARE english INT;
- DECLARE i INT;
- SET i=0; #给迭代变量赋予初始值，使得后续的循环条件满足
- DROP TABLE IF EXISTS test10; #删除指定名称的表，确保后续建表成功
- CREATE TABLE test10(序号 INT PRIMARY KEY AUTO_INCREMENT,姓名 VARCHAR(10),语文 INT,数学 INT,英语 INT); #建表
- WHILE i<n DO #循环开始，格式：while 条件 do
- SET chinese=FLOOR(RAND()*50)+50; #通过随机数给相关变量赋值
- SET math=FLOOR(RAND()*50)+50;
- SET english=FLOOR(RAND()*50)+50;
- INSERT INTO test10(姓名,语文,数学,英语)VALUES(CONCAT('test',i+1),chinese,math,english); #写入数据
- SET i=i+1; #迭代变量自动增长1
- END WHILE; #当条件不再成立则结束循环
- END $$ #结束整个定义逻辑的过程
- DELIMITER ; # 把；号重新定义成结束符
- CALL test(10); #调用已经定义好的存储过程，传入适当参数，得到指定结果
- SELECT * FROM test10;
-
- DDL/DML/DQL/DCL

- DDL:Data Definition Language的缩写，即数据定义语言
  - 数据定义语言是由SQL语言集中负责数据结构定义与数据库对象定义的语言，并且由CREATE、ALTER、DROP和TRUNCATE四个语法组成
  - TRUNCATE：只删除数据不删除表结构
    - 与delete的区别
      - TRUNCATE速度较快，delete速度较慢
      - TRUNCATE删除的数据不可以回滚，delete删除的数据可以恢复
- DML:Data Manipulation Language的缩写，数据操纵语言，主要有insert、update、delete语法组成
- DQL:Data Query Language的缩写，数据库查询语言，即最常用的select语句
- DCL:Data Control Language的缩写，数据控制语言，用来授权或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果，能够对数据库进行监视。