目录

[数据库学习 2](#_Toc12758)

[模块1 数据库和SQL 2](#_Toc16823)

[1.1 认识数据库 2](#_Toc19963)

[1.1.1 数据库是什么 3](#_Toc30888)

[1.1.2 DBMS 3](#_Toc16544)

[1.1.3 关系模型 3](#_Toc11627)

[1.2 认识SQL 3](#_Toc741)

[1.2.1 SQL是什么 3](#_Toc12565)

[1.2.2 SQL语句及种类 3](#_Toc21527)

[1.2.3 SQL基本书写规则 4](#_Toc30223)

[1.3 数据库和表的基本操作 4](#_Toc29728)

[1.3.1 创建数据库 4](#_Toc32559)

[1.3.2 删除数据库 4](#_Toc31407)

[1.3.3 数据类型 4](#_Toc27859)

[1、数字型数据：bit 位型 示例：1,0 4](#_Toc30831)

[2、 字符型数据：char 固定长度 5](#_Toc16311)

[3、 日期时间数据：date 日期 示例：’2020-11-11’ 5](#_Toc15095)

[1.3.4 创建表 5](#_Toc26525)

[1.3.5 插入表数据 5](#_Toc4457)

[1.3.6 表的删除和更新 6](#_Toc9210)

[1、更新(修改)表结构 6](#_Toc7895)

[2、 重命名表 7](#_Toc12101)

[3、 复制表 7](#_Toc28891)

[4、 删除表 7](#_Toc12710)

[1.4总结 7](#_Toc1045)

[模块2 查询基础 7](#_Toc7685)

[2.1 列的查询 8](#_Toc28008)

[2.1.1 按列查询 8](#_Toc29915)

[2.1.2 列的别名 8](#_Toc22046)

[2.1.3 查询所有列 8](#_Toc555)

[2.2 查询条件 9](#_Toc10866)

[2.2.1 应用查询条件 9](#_Toc6277)

[2.2.2 算术运算和比较运算 9](#_Toc5380)

[2.2.3 布尔逻辑值 9](#_Toc6417)

[2.2.4 限制行 11](#_Toc30526)

[2.2.5 通配符 11](#_Toc22762)

[2.2.6 NULL值 11](#_Toc19907)

[2.3 聚合查询 11](#_Toc13903)

[2.3.1 使用聚合函数 12](#_Toc10578)

[2.3.2 数值函数 12](#_Toc1151)

[2.3.3 字符函数 12](#_Toc26568)

[2.3.4 日期/时间函数 13](#_Toc27439)

[2.3.5 转换函数 13](#_Toc30242)

[2.4 总结 13](#_Toc32512)

[模块3 查询进阶 13](#_Toc27572)

[3.1 排序和分组 13](#_Toc16994)

[3.1.1 排序数据 13](#_Toc4076)

[3.1.2 分组查询 15](#_Toc31423)

[3.1.3 限定分组查询 15](#_Toc10077)

[3.2 子查询 16](#_Toc25585)

[3.2.1 标量子查询 16](#_Toc2529)

[3.2.2 关联子查询 16](#_Toc23613)

[3.2.3 IN和EXISTS 16](#_Toc21269)

[3.3 连接查询 17](#_Toc13686)

[3.3.1 内连接 18](#_Toc18309)

[3.3.2 外连接 18](#_Toc31735)

[3.4 CASE表达式 19](#_Toc7216)

[3.4.1 CASE表达式的语法 19](#_Toc19083)

[模块4 数据操纵语言和数据定义语言 20](#_Toc31576)

[4.1 复制和修改数据 20](#_Toc14978)

[4.1.1 复制数据 20](#_Toc14528)

[4.1.2 更新数据 20](#_Toc14723)

[4.1.3 删除数据 21](#_Toc30902)

[4.2 使用视图 22](#_Toc6874)

[4.2.1 创建和使用视图 22](#_Toc26787)

[4.2.2 修改和删除视图 23](#_Toc10328)

[1、使用alter view修改视图 23](#_Toc23804)

[2、使用drop view删除视图 23](#_Toc22264)

[4.3 数据定义进阶 23](#_Toc1984)

[4.3.1 添加自增列 24](#_Toc9701)

[4.3.2 添加约束和默认值 24](#_Toc21484)

[4.4 集合逻辑 25](#_Toc8866)

[4.4.1 使用union 运算符 25](#_Toc5517)

[4.4.2 union和union all 26](#_Toc9045)

# 数据库学习

# 模块1 数据库和SQL

## 1.1 认识数据库

认识数据库DB，数据库管理系统DBMS

导入例子：

例1：查询雇员表的所有男性职工的信息

select \* from employee where e\_sex="男";

例2：查询雇员表王姓职工的信息

select \* from employee where e\_name like '王%'; [编辑内嵌]

### 1.1.1 数据库是什么

数据库(Database)是以特定的组织结构（二维表的形式、关系模型）、存放在计算机的存储介质中相互关联的数据集合。

### 1.1.2 DBMS

数据库管理系统，是用来管理数据库的计算机系统。比如：MySQL,SQL Server,Oracle。

### 1.1.3 关系模型

1、使用二维表来组织数据，二维表是由行和列组成

2、记录（行），字段（列）

3、主键（不为空，不重复，能够唯一标识表中一个单独的行） [点击结构，名字后面有钥匙的是主键]

4、外键（在本表中不是主键，但在另一张表中充当主键的字段）。可以配合主键实现表的关联。

## 1.2 认识SQL

介绍SQL的组成，标准SQL [SQL不区分大小写]

### 1.2.1 SQL是什么

SQL是结构化查询语言，是应用于关系数据库管理的标准语言，是让用户和关系型数据库（RDBMS）进行交互的一种计算机语言。

### 1.2.2 SQL语句及种类

3个组成部分：

1、数据定义语言DDL：用于定义数据库对象，对数据库、数据库中的表、视图、索引等数据库对象进行建立和删除。

指令：CREATE 用于创建

ALTER 用于修改

DROP 用于删除

2、数据操纵语言DML：用于对数据库中的数据进行查询、插入、修改、删除等操作。

指令：SELECT 用于查询表中的数据

INSERT 用于向表中插入数据

UPDATE 用于更新表中的数据

DELETE 用于删除表中的数据

1. 数据控制语言DCL：用于控制用户对数据库的操作权限。

指令：COMMIT 确认对数据库中数据进行变更

ROLLBACK 取消对数据库中的数据进行变更

GRANT 赋予用户操作权限

REVOKE 取消用户操作权限

### 1.2.3 SQL基本书写规则

1、SQL语句以分号结尾

2、SQL语句不区分大小写

3、单词之间需要使用半角空格或者换行符进行分隔

4、常数的书写方式是固定的(字符串和日期常数需要用单引号括起来，单引号为定界符。数字常数则无需添加单引号)

## 1.3 数据库和表的基本操作

介绍数据库和表的基本操作

### 1.3.1 创建数据库

格式：create database <数据库名>

示例：创建shop数据库 create database shop;

### 1.3.2 删除数据库

格式：drop database <数据库名>

示例：删除shop数据库 drop database shop;

### 1.3.3 数据类型

1、数字型数据：bit 位型 示例：1,0

int 整型 示例：36

decimal 浮点数 示例：58.66

real 实数 示例：88.667788(可以是近似的值)

1. 字符型数据：char 固定长度

varchar 可变长度

1. 日期时间数据：date 日期 示例：’2020-11-11’

datetime 日期时间 示例：’2020-11-14 08:00:00’

### 1.3.4 创建表

格式：create table <表名>(字段名1 数据类型(长度) 约束, 字段名2.。。。。。。。。)

示例：在shop数据库中创建product表，包含如下字段：

字段名 类型和长度 约束 字段说明

id char(4) not null primary key 商品编号

name varchar(100) not null 商品名称

type varchar(32) not null 商品类型

sale int null(允许为空) 销售单价

purchase int null 进货单价

regist date null 注册日期

建表语句如下：create table product(

id char(4) not null primary key,

name varchar(100) not null,

type varchar(32) not null,

sale int,

purchase int,

regist date

);

总结：建立数据库表的过程是，先建表结构，再录表内容。

### 1.3.5 插入表数据

格式：insert into <表名>[(字段名表)]value(字段值) []可省略

示例：给product表插入数据内容

写法一：

insert into product values(‘0001’,’T恤’,’衣服’,1000,500,’2009-09-20’);

insert into product values(‘0002’,’打孔器’,’办公用品’,500,320,’2009-09-11’);

insert into product values('0003','运动T恤','衣服',4000,2800,'2009-11-10');

insert into product values('0004','菜刀','厨房用具',3000,2800,'2009-9-20');

insert into product values('0005','高压锅','厨房用具',6800,5000,'2009-1-15');

insert into product values('0006','叉子','厨房用具',500,null,'2009-09-20');

insert into product values('0007','菜板','厨房用具',880,790,'2008-04-28');

insert into product values('0008','圆珠笔','办公用品',100,null,'2009-11-11');

写法二：

insert into product values('0001','T恤','衣服',1000,500,'2009-09-20'),

('0002','打孔器','办公用品',500,320,'2009-09-11'),

('0003','运动T恤','衣服',4000,2800,'2009-11-10'),

('0004','菜刀','厨房用具',3000,2800,'2009-9-20'),

('0005','高压锅','厨房用具',6800,5000,'2009-1-15'),

('0006','叉子','厨房用具',500,null,'2009-09-20'),

('0007','菜板','厨房用具',880,790,'2008-04-28'),

('0008','圆珠笔','办公用品',100,null,'2009-11-11');

示例：给product表插入如下信息：'0009','毛笔','办公用品'

insert into product(id,name,type) values('0009','毛笔','办公用品');

删除“毛笔”记录

Delete from product where id=’0009’;

结论：字段名表要和字段值(数目和类型)一一对应，如果插入所有字段内容则可以省略字段名表

### 1.3.6 表的删除和更新

1、更新(修改)表结构

格式：alter table <表名> add [column]|change |drop [column] <字段名>

add 添加字段；

change 修改字段；

drop 删除字段

示例：给product表添加一个margin(int)字段

alter table product add column margin int;

或者 alter table product add margin int;

示例：修改product表的id(char(4))字段product\_id(char(4))

alter table product change id product\_id char(4);

示例：修改product表的margin(int)字段为margin(decimal(5,2))

alter table product change margin margin decimal(5,2);

示例：删除product表的margin字段

alter table product drop margin;

1. 重命名表

格式：alter table <旧表名> rename <新表名>

示例：修改product表名为pd

alter table product rename pd;

1. 复制表

格式：create table <新表名> select \* from <旧表名>

示例：复制pd表并命名新表为product

create table product select \* from pd;

1. 删除表

格式：drop table <表名>

示例：删除product表

drop table product;

delect from product;(删除表的内容，表的结构还在，只是里面的数据没了)

## 1.4总结

[primary key(字段1,字段2)]

# 模块2 查询基础

创建数据资源之后就可以使用SQL检索数据，现实工作中数据查询式最常用的SQL功能。

## 2.1 列的查询

按字段名(列)检索数据，格式：select <字段名表> from <表名>

语法说明：当查询的列不止一个时，各字段名之间用逗号分隔

### 2.1.1 按列查询

以下查询以shop数据库为数据源，对product表的字段做如下说明：

id char(4) primary key 商品编号

Name varchar(100) 商品名称

Type varchar(32) 商品类型

Sale int(3) 销售单价

Purchase int 进货单价

Regist date 注册日期

示例1：查询product表中所有商品的商品名称(name)

select name from product;

语法说明：按列查询属于关系运算里的投影

示例2：查询product表中所有商品的商品编号(id),商品名称(name),销售单价(sale)

select id,name,sale from product;

示例3：查询product表中的商品类(type) [查看一共有几种商品类别]

select distinct type from product;

语法说明：可以使用关键字distinct去除查询结果的重复值。

### 2.1.2 列的别名

可以使用关键字as给查询的列定义新的别名，格式：<字段名> [as] <别名>

语法说明：别名可以使用英文或汉字，设定汉字别名时需要使用双引号(“”)

示例4：查询product表中所有商品的商品名称(name)和商品类型(type)，并设定别名

select product\_id as id, name as product\_name,type as product\_type from product;

select name as “商品名称”,type as “商品类型” from product; [as]可以省略

### 2.1.3 查询所有列

可以使用\*代替当前表的所有字段(列)

示例5：select \* from product;

## 2.2 查询条件

前面的例子都是把整列的数据查询出来，但实际应用中往往是查询满足一定条件的数据，这就需要使用查询条件。

相当于关系运算里的选择。

格式：select <字段名表> from <表名> where <条件表达式>

语法说明：在条件表达式中，比较运算的左边一般放字段名或字段名的组合，右边放常数

### 2.2.1 应用查询条件

示例1：查询product表的商品名称(name)和销售单价(sale)，要求销售单价高于2000

select name,sale from product where sale > 2000;

示例2：查询product表的商品信息，要求商品注册日期(regist)在2009-09-01之前

select \* from product where regist < ‘2009-09-01’;

示例3：查询product表的商品信息，要求商品类型(type)为办公用品

select \* from product where type = ‘办公用品’

### 2.2.2 算术运算和比较运算

1、算术运算：+；-；\*；/

2、关系运算(比较运算)：= ； <>不等于；>=大于等于；<=小于等于

3、不同类型的数据常数的比较

数字数据 直接比较值的大小

字符数据 按照asc码的值从左到右依次比较

日期数据 过去小于现在

### 2.2.3 布尔逻辑值

1、通过逻辑运算符，可以将多个查询条件进行结合

2、常用的逻辑运算符有：AND 与；OR 或；NOT 相反

3、AND运算逻辑是：运算符两边都是真，运算结果才是真。只要两边有一个为假，运算结果就是假。其中，“真”表示条件成立，“假”表示条件不成立。

真 AND 真 = 真

真 AND 假 = 假

假 AND 真 = 假

假 AND 假 = 假

4、OR运算逻辑是：运算符两边都是假，运算结果才是假。只要两边有一个为真，运算结果就是真。其中，“真”表示条件成立，“假”表示条件不成立。

真 OR 真 = 真

真 OR 假 = 真

假 OR 真 = 真

假 OR 假 = 假

5、NOT运算逻辑是：不是真就是假

NOT 真 = 假

NOT 假 = 真

运算符优先级：NOT > AND > OR

示例4：查询product表的商品名称(name)，商品类型(type)和销售单价(sale)

要求：商品类型为厨房用具，销售单价大于2000

select name,type,sale from product where type=’厨房用具’ and sale >2000;

示例5：查询product表的商品信息，要求：商品编号(id)是’0001’或’0003’

select \* from product where product\_id=’0001’ OR product\_id=’0003’;

或者

select \* from product where product\_id in(‘0001’,’0003’);

示例6：查询product表的商品名称(name)和销售单价(sale)

要求销售单价在[2000,5000]之间

select name,sale from product where sale>=2000 and sale<=5000;

或者

select name,sale from product where sale between 2000 and 5000;

示例7：查询product表的商品名称(name)和销售单价(sale)

要求销售单价在小于2000或者大于5000

select name,sale from product where sale<2000 OR sale>5000;

或者

select name,sale from product where sale not between 2000 and 5000;

示例8：查询product表的商品名称，商品类型，注册日期

要求商品类型为办公用品，注册日期是2009-9-11或者2009-9-20

select name,type,regist from product where type=’办公用品’ AND regist in(‘2009-9-11’,’2009-9-20’);

或者

select name,type,regist from product where type=’办公用品’ AND (regist = ’2009-9-11’ or regist = ‘2009-9-20’);

### 2.2.4 限制行

有时需要显示查询表的一个小子集，比如显示当前表的前5条记录，此时需要使用限制行，关键字limit,放置到select语句的最后。

示例9：查询product表的前5条记录

select \* from product limit 5;

示例10：查询product表商品注册时间最早的3个商品的信息

select \* from product order by regist asc limit 3; [desc 降序；asc 升序]

补充语法：限制行一般可以配合排序共同使用

### 2.2.5 通配符

1、可以使用关键字like进行模糊匹配字符串常数，这时需要使用通配符

2、格式：<字段名> like 字符串常数

3、常用的通配符有：“%” 代表0个或多个任意字符

“\_” 代表任意一个字符

示例11：查询product表商品名称里有”锅”这个字的商品信息

select \* from product where name like ‘%锅%’;

示例12：查询product表商品名称的第二个字是’刀’的商品信息

select \* from product where name like ‘\_刀%’;

### 2.2.6 NULL值

1、NULL值是指不确定的值。要选择空值需要使用IS NULL或者IS NOT NULL

2、NULL值不参与运算

示例13：查询product表进货单价(purchase)为空的商品名称(name)和进货单价(purchase)

select name,purchase from product where purchase is NULL;

示例14：查询product表进货单价(purchase)不为空的商品名称(name)和进货单价(purchase)

select name,purchase from product where purchase is not NULL;

## 2.3 聚合查询

1、聚合查询是指可以进行聚合计算的查询

2、常见的聚合计算有：求和，求平均，求最大值，求最小值，计数

3、在数据库中可以使用聚合函数实现聚合计算

### 2.3.1 使用聚合函数

常见的聚合函数有：

sum()

avg() 求平均

max() min() count()

示例：统计product表所有商品的销售单价的合计值

select sum(sale) as “销售单价合计” from product;

示例：统计product表所有商品的销售单价平均值，要求商品类型为厨房用具

select avg(sale) as “销售单价平均值” from product where type=”厨房用具”;

示例：统计product表的商品的最高售价和最低售价

select max(sale) as “最高售价”,min(sale) as “最低售价” from product;

示例：统计product表的记录总数

select count(\*) as “记录总数” from product;

或者

统计主键的个数(因为这里主键唯一)

select count(id) as “记录总数” from product;

示例：统计product表商品类型的数目(去重)

select count(distinct type) as “商品类别数目” from product;

### 2.3.2 数值函数

1、返回圆周率 select pi();

2、求绝对值 select abs();

3、进行求余运算 select mod(31,8);

4、返回随机数 select rand();

5、进行四舍五入运算 select round(1.38,1);

### 2.3.3 字符函数

1、计算字符串的长度 select length(‘egg’),length(‘西安’)

2、连接字符串concat() select concat(‘MySql’,’5.7’)

select concat(‘MySql’,null,’5.7’)

1. 生成空格函数space() select concat(‘(‘,space(6),’)’);
2. 返回字符串左边的字符left() select left(‘football’,4);
3. 返回字符串右边的字符right() select right(‘football’,4);
4. 重复生成相同字符串repeat() select repeat(‘MySql’,,3);

### 2.3.4 日期/时间函数

1、使用日期函数获取当前系统日期 select curdate()|current\_date();

2、返回日期中的年月日: select year(curdate()),month(‘2020-11-1’),day(‘2020-11-24’)

### 2.3.5 转换函数

1、cast() 类型转换函数 select cast(‘0001’ as singed int),cast(‘2020-11-24’ as date)

## 2.4 总结

# 模块3 查询进阶

本模块继续介绍sql查询的用法，包括排序和分组，子查询，连接查询等内容

## 3.1 排序和分组

排序ORDER BY ,分组 GROUP BY

### 3.1.1 排序数据

随机顺序的数据是难以处理的，所以我们需要给数据排序。可以使用ORDER BY子句进行排序

格式：select <字段名> from <表名> [where <条件>] order by <排序键> [asc]|desc

语法说明：

1. asc表示升序，是缺省值，desc表示降序。
2. order by 可以指定多个排序键，各排序键之间用逗号间隔
3. 排序键中包含NULL时，会在开头或者末尾进行汇总
4. 不论何种情况，order by 子句都需要写在select 语句的末尾，因为排序是在结果即将返回时执行
5. 可以使用列的别名排序
6. 可以使用聚合函数排序
7. 排序可以和限制行(limit)配合使用
8. 如果想在多个列上进行降序排序，必须对每个列指定DESC关键字

示例：查询product 表的所有内容,查询结果以销售单价的升序排列

Select \* from product order by sale asc;

也可以这样写：

Select \* from product order by sale;

示例：查询product表所有厨房用具的信息，查询结果以销售单价的降序排列

Select \* from product where type=’厨房用具’ order by sale desc;

示例：查询product表售价在3000元以上的商品信息，查询结果以注册日期降序排序

Select \* from product where purchase >3000 order by regist desc;

示例：查询product表的所有内容，查询结果先以注册日期升序排列，再以销售单价降序排列

Select \* from product order by regist asc,sale dsec;

语法注释：当有多个排序键时，先按第一排序键排列，当第一排序键有重复值时，再按第二排序键排列

示例：查询product表的商品编号，商品名称，销售单价，进货单价，

要求：查询结果以进货单价的升序排列

Select id,name,sale,purchase from product order by purchase;

示例：可以使用列的别名排序，有语句如下：

Select id as “商品编号”,name as “商品名称”,sale as “销售单价”,purchase as “进货单价” from product order by 进货单价desc,商品编号;

示例：可以使用聚合函数排序，有语句如下：

Select type,count(\*) from product group by type order by count(\*) desc;

示例：查询product表的商品名称和销售单价

要求：查询销售单价最高的前3个商品

Select name,sale from product order by sale desc limit 3;

示例：查询product表销售单价最低的商品信息

Select \* from product order by sale asc limit 1;

示例：查询product表销售单价排名第二的商品信息

Select \* from product order by sale desc limit 1 offset 1;

### 3.1.2 分组查询

可以使用关键字group by将查询数据分组，格式如下：

Select <字段名表> from <表名> [where <条件>] group by <分组字段> [having <分组条件>]

语法说明：

1. 不能使用列的别名
2. 分组经常和聚合函数一起使用
3. 一般需要把分组字段放入字段名表

示例：查询product 表每类商品的销售单价合计值

Select type,sum(sale) as “销售单价合计” from product group by type;

示例：查询product表每类商品的最高售价和最低售价

Select type,max(sale) as “最高售价”,min(sale) as “最低售价” from product group by type;

示例：查询product表每类商品的销售单价平均值，查询结果以销售单价平均值降序排序

Select type,avg(sale) as “平均售价” from product group by type order by avg(sale) desc;

也可以这样写：

Select type,avg(sale) as “平均售价” from product group by type order by 平均售价desc;

示例：统计product 表每类商品的个数

Select type,count(\*) as “商品数目” from product group by type;

也可以这样写：

Select type,count(id) as “商品数目” from product group by type;

### 3.1.3 限定分组查询

1、可以使用having句子来限定分组条件

2、having子句不能单独使用，需要写在group by子句之后

3、having 子句可以使用聚合函数，where 子句则不能使用聚合函数

示例：按商品类型查询product表的商品类别和每类商品的个数

要求：每类至少有3个商品

Select type,count(\*) as “商品数目” from product group by type having count(\*)>=3;

示例：按商品类型查询product表每类商品的平均销售单价在2000以上的商品类型和商品的平均销售单价

Select type,avg(sale) as “平均售价” from product group by type having avg(sale)>2000;

## 3.2 子查询

复杂的查询问题需要用到嵌套查询语句，这时被嵌套的查询语句就是子查询。子查询就是一次性的视图，不同的是视图被创建之后一直存在，而子查询在语句执行完毕后就会消失

### 3.2.1 标量子查询

标量子查询就是只能返回一行一列结果的子查询

示例：查询product表的商品名称和销售单价，要求：销售单价高于所有商品的平均销售单价

Select name,sale from product where sale>(select avg(sale) from product);

示例：查询product 表的商品类型和每类商品的平均销售单价，要求：该类商品的平均销售单价高于所有商品的平均销售单价

Select type,avg(sale) as “每类商品平均售价” from product group by type having avg(sale)>(select avg(sale) from product);

### 3.2.2 关联子查询

在细分的组内进行比较时，需要使用关联子查询

示例：查询product表的商品名称，商品类型和销售单价。要求：该商品的销售单价高于该商品所属商品种类的平均销售单价

Select name,type,sale from product as p1 where sale>(select avg(sale) from product as p2 where p1.type=p2.type group by type);

### 3.2.3 IN和EXISTS

1、IN和EXISTS都属于谓词，IN表示在某一个范围内，EXISTS用来判断是否存在满足某种条件的记录

2、IN和NOT IN，EXISTS和NOT EXISTS

3、基本上可以使用IN来代替EXISTS

shopproduct表结构如下：

字段 字段类型 字段说明

Shop\_id char(4) not null 商店编号

Shop\_name varchar(200)not null 商品名称

Id char(4) not null 商品编号

Quantity int() not null 数量

示例：在product表中查询东京商店(shop\_name)和销售的商品名称(name)，售价(sale)和进价(purchase)

使用IN

Select name,sale,purchase from product where id in (select id from shopproduct where shop\_name=’东京’);

使用EXISTS

Select name,sale,purchase from product where exists (select \* from shopproduct where shop\_name=’东京’ and product.id=shopproduct.id);

说明：可以先在shopproduct表找到’东京’商店销售的商品编号，再用商品编号去找product表的相关查询内容

示例：查询商店商品销售单价在3000以上的商品编号，商店名称和数量

使用IN

Select shop\_id,shop\_name,quality from shopproduct where id in (select id from product where sale>3000);

使用EXISTS

Select shop\_id,shop\_name,quality from shopproduct where exists (select \* from product where sale>3000 and product.id=shopproduct.id);

示例：查询不在东京商店(shop\_name)和销售的商品名称(name)，售价(sale)和进价(purchase)

使用IN

Select name,sale,purchase from product where id not in (select id from shopproduct where shop\_name=’东京’);

使用EXISTS

Select name,sale,purchase from product where not exists (select \* from shopproduct where shop\_name=’东京’ and product.id=shopproduct.id);

## 3.3 连接查询

1、连接查询一般用于多表查询中，也就是查询的内容和条件属于多张表

2、有内连接和外连接两种，其中内连接是应用最广泛的连接运算

3、进行连接时需要在FROM子句中使用多张表

4、查询的字段需要按照<表名>.<列名>的格式书写，例如：product.name

### 3.3.1 内连接

使用<表1>[linner] join <表2> on <连接字段相等的表达式>

示例： 查询东京商店(shop\_name)的商品编号(id)，商品名称(name)，商店名称(shop\_name)和数量(quantity)

Select product.id,product.name,shopproduct.shop\_name,shopproduct.quantity from product join shopproduct on product.id=shopproduct.id where shopproduct.shop\_name=’东京’;

也可以这样写：给表起别名来简化指令

Select p.id,p.name,sp.shop\_name,sp.quantity from product as p join shopproduct as sp on p.id=sp.id where sp.shop\_name=’东京’;

也可以进一步简化

Select p.id,name,shop\_name,quantity from product as p join shopproduct as sp on p.id=sp.id where shop\_name=’东京’;

示例：查询商品名称(name),商品名称(shop\_name) ,进货单价(purchase)和注册日期(reegist)

要求：注册日期在2009年

Select p.name,sp.shop\_name,p.purchase,p.regist from product as p join shopproduct as sp on p.id=sp.id where year(p.regist)=2009;

### 3.3.2 外连接

使用 <表1>left|right outer jion <表2> on <连接字段相等的表达式>

语法说明：

1. 外连接中可以使用left、right来指定主表
2. 外连接的结果是两表都有的记录，外加主表的若干记录
3. 在实际业务中，如果要生成固定行数的单据，常常需要使用外连接

示例：使用外连接查询商品编号(id)，商品名称(name)，商品编号(shop\_id)，商店名称(shop\_name)

Select p.id,p.name,sp.shop\_id,sp.shop\_name from product as p left outer join shopproduct as sp on p.id=sp.id;

或者也可以这样写：

Select p.id,p.name,sp.shop\_id,sp.shop\_name from shopproduct as sp right outer join product as p on p.id=sp.id;

## 3.4 CASE表达式

1、CASE表达式称为条件分支

2、CASE表达式分为简单CASE表达式和搜索CASE表达式

3、CASE表达式中的END不能省略

4、使用CASE表达式能够将SELECT语句的结果进行组合

5、最好不要省略ELSE

### 3.4.1 CASE表达式的语法

1、简单CASE表达式格式：

Case <表达式>

When <表达式> then <表达式>

When <表达式> then <表达式>

。。。。。。

Else <表达式>

End

1. 搜索CASE表达式格式：

Case When <表达式> then <表达式>

When <表达式> then <表达式>

。。。。。。

Else <表达式>

End

示例：简单CASE表达式

Select name,

(case type

When ‘衣服’ then concat(‘A:’,type)

When ‘办公用品’ then concat(‘B:’,type)

When ‘厨房用具’ then concat(‘C:’,type)

Else null

end) as abc\_type

From product;

示例：搜索CASE表达式

Select name,

(case When ‘衣服’ then concat(‘A:’,type)

When ‘办公用品’ then concat(‘B:’,type)

When ‘厨房用具’ then concat(‘C:’,type)

Else null

end) as abc\_type

From product;

# 模块4 数据操纵语言和数据定义语言

本模块主要补充介绍DDL和DML的基本操作

## 4.1 复制和修改数据

实际应用中，除了进行数据查询，还经常需要对数据进行复制、更新和删除

### 4.1.1 复制数据

复制表数据的方法1：

格式：create table <新表名> selec \* from <旧表名>

语法说明：此方法复制的新表没有旧表的主键、索引、外键等信息

示例：复制product表到新表productcopy

Create table productcopy select \* from product;

复制表数据方法2：分两步

格式：

1. create table <新表名> like <旧表名>
2. Insert into <新表名> select \* from <旧表名>

语法说明：此方法复制的新表带有旧表的主键、索引、外键等信息

示例：复制product表到新表productcopy

Create table productcopy like product;

Insert into productcopy select \* from product;

### 4.1.2 更新数据

更新数据update

格式说明：update <表名> set <字段名> =<表达式> [where <条件>]

语法说明：

1. update默认更新全部数据行，带where时可以对数据行部分更新
2. Update 可以将列的值更新为null
3. 同时更新多列时，可以在set子句中使用逗号分隔更新对象的多个列

示例：将productcopy表的所有商品的注册日期更新为2009-10-1

Update productcopy set rigist=’2009-10-1’;

示例：将productcopy 表的销售单价更新为原来的10倍，要求：商品类型为厨房用具

Update productcopy set sale=sale\*10 where type=’厨房用具’;

示例：将productcopy 表的销售单价打6折，要求：商品类型为厨房用具

Update productcopy set sale=sale\*0.6 where type=’厨房用具’;

示例：将productcopy 表的销售单价提升20%，要求：商品类型是除了“厨房用具”以外的商品

Update productcopy set sale=sale\*1.2 where type<>’厨房用具’;

示例：将productcopy 表的进货单价增加500元，要求：商品类型为厨房用具

Update productcopy set purchase=purchase+500 where type=’厨房用具’;

说明：本题中null值不参与运算，所以没有被更新

示例：将productcopy 表商品编号为0006的商品进货单价设置为500元

Update productcopy set purchase=500 where id=’0006’;

示例：将productcopy 表商品编号为0008的商品注册日期更新为null

Update productcopy set regist = null where id=’0008’;

示例：将productcopy 表销售单价降低20%，进货单价增加500元，要求：商品类型为厨房用具

Update productcopy set sale=sale\*0.8 ,purchase=purchase+500 where type=’厨房用具’;

### 4.1.3 删除数据

删除数据内容是delete

格式：delete from <表名> [where <条件>]

语法说明：

1. delete只能删除表内容，不影响表结构，drop table可以将整个表删除
2. Delete默认删除全部数据行，带where时可以对数据行部分删除

示例：删除productcopy表商品编号为0003，0006，0008的商品记录

Delete from productcopy where id in(‘0003’,’0006’,’0008’);

也可以这样写：

Delete from productcopy where id=’0003’ or id=’0006’ or id=’0008’;

示例：删除productcopy表的全部记录

Delete from productcopy;

示例：删除productcopy表

Drop table productcopy;

## 4.2 使用视图

1、视图是一张“虚”表，因为表中存储的是实际的数据，而视图中保存的是select语句

2、使用视图，可以轻松地完成跨很多表查询数据等复杂操作

3、可以将常用的select语句做成视图

4、应避免在视图的基础上再创建视图

5、应避免对视图进行更新

### 4.2.1 创建和使用视图

使用create view语句创建视图

格式：create view <视图名> (<视图列名1>,<视图列名2>...)

As

<select 语句>

示例：下列语句根据product表创建视图p1

Create view p1

As

Select type,count(\*) from product group by type;

示例：按商店名称统计各商店的总销售单价和总进货单价，并计算各商店的利润（使用视图完成）

Create view p3(商店名称，总销售单价，总进货单价，利润)

As

Select sp.shop\_name,sum(p.purchase),sum(p.sale)-sum(p.purchase) from product as p join shopproduct as sp on p.id=sp.id group by sp.shop\_name;

问“东京”商店的总销售单价，总进货单价和利润

Select \* from p3 where 商店名称=‘东京’；

示例：查询商品编号，商品名称，商店名称和商品数量

要求：商店名称为“大阪”，数量在60以上，并且用视图完成

Create view p4(商品编号，商品名称，商店名称，商品数量)

As

Select p.id p.name,sp.shop\_name,sp.quantity from product as p join shopproduct as sp on p.id=sp.id where sp.shop\_name=’大阪’ and sp.quantity>60;

Select \* from p4

### 4.2.2 修改和删除视图

1、使用alter view修改视图

格式：alter view <视图名> (<视图列名1>,<视图列名2>,...)

As

<select 语句>

示例：下列语句修改视图p4，给p4添加‘销售单价’列

Alter view p4(商品编号，商品名称，商店名称，商品数量,销售单价)

As

Select p.id p.name,sp.shop\_name,sp.quantity,p.sale from product as p join shopproduct as sp on p.id=sp.id where sp.shop\_name=’大阪’ and sp.quantity>60;

Select \* from p4

2、使用drop view删除视图

格式：drop view <视图名>

示例：删除视图p1，p3，p4

Drop view p1,p3,p4;

## 4.3 数据定义进阶

本节讲述加强版的create table语句

创建两张表：employee和dept

1. 创建dept表语句如下：

Create table dept

(

D\_no int not null auto\_increment,

D\_name varchar(50),

D\_location varchar(100),

Primary key(d\_no)

);

1. 创建employee表语句如下：

Create table employee

(

E\_no int not null primary key,

E\_name varchar(100) not null,

E\_sex char(2) not null default ‘男’, (添加默认值)

D\_no int not null,

E\_job varchar(100) not null,

E\_salary int not null,

Hiredate date,

Foreign key(d\_no) references dept(d\_no) 创建外键连接别的表

);

### 4.3.1 添加自增列

使用关键字auto\_increment添加列的自增约束，即实现自动编号功能

示例：给dept表插入记录

Insert into dept values(10,’财务部’,’上海’);

Insert into dept(d\_name,d\_location) values(’生产部’,’郑州’),(’客服部’,’西安’),(’销售部’,’深证’),(’产品部’,’北京’);

### 4.3.2 添加约束和默认值

1、使用关键字defaul添加默认值约束

2、使用primary key 添加主键约束

这里不合适，应该小于主表的取值范围

3、使用foreign key添加外键约束

示例：给employee表插入记录

Insert into employee values(1001,’张军’,default,10,’营业员’,800,’2005-11-12’);

Insert into employee values(1002,’张丽’,’女’,30,’营业员’,800,’2005-11-12’);错误

Insert into employee values(1002,’张丽’,’女’,11,’营业员’,800,’2005-11-12’);

说明：

1. 第一条语句使用了默认值，运行成功
2. 主表是连接字段作为主键的表，子表是连接字段作为外键的表，本例的连接字段是d\_no
3. 主表和子表连接规则需满足：主表主键字段取值范围>=子表外键字段取值范围
4. 第2条语句执行不成功，原因是违反了外键约束。本例中，dept是主表，employee是子表
5. 本例中主表dept中并没有d\_no=30的记录，也就是子表的外键字段想插入一个主表主键字段没有的值，这违反了连接规则，因此无法正常插入。

## 4.4 集合逻辑

1、集合运算可以去除重复行(union)

2、如果希望集合运算保留重复行，需要使用all关键字

创建product2表，格式和内容如下：

Create table product2(

Id char(4) not null primary key,

Name varchar(100) not null,

Type varchar(32) not null,

Sale int,

Purchase int,

Regist date

);

Insert into product2 values(‘0001’,’T恤’,’衣服’,1000,500,’2009-9-20’),

(‘0002’,’打孔器’,’办公用品’,500,320,’2009-9-11’),

(‘0003’,’运动T恤’,’衣服’,40002800,’2009-11-10’),

(‘0009’,’手套’,’衣服’,800,500,null),

(‘0010’,’冰壶’,’厨房用具’,2000,1700,’2009-9-20’);

### 4.4.1 使用union 运算符

1、并集运算

示例：执行下列语句

Select id,name from product

Union

Select id,name from product2;

分析：该语句是将两个select语句的查询结果做了并集运算，并且去掉重复记录

2、集合运算对象的列数必须相同

示例：执行下列语句

Select id,name from product

Union

Select id,name,sale from product2; 不能执行

分析：该语句两个select语句的查询列数不一致，不能执行

### 4.4.2 union和union all

可以使用关键字all包含重复行

示例：执行下列语句

Select id,name from product

Union all

Select id,name from product2;

分析：包含重复记录的并集运算