2025年秋季课程

数据库原理

哈尔滨工程大学计算机科学与技术学院

李博权

liboquan@hrbeu.edu.cn

第三章 关系数据库标准语言SQL

3.1 SQL概述

❖ SQL（Structured Query Language）

结构化查询语言，是关系数据库的标准语言

■功能包括：数据查询、数据库模式创建、数据库数据的增删改、数据库安全性和完整性定义与控制等。

■在1974年由Boyce和Chamberlin提出，最初的名称是SEQUEL。

SQL概述（续）

3.1.1 SQL 的产生与发展

3.1.2 SQL的特点

3.1.3 SQL的基本概念

3.1.1 SQL的产生与发展

标准	篇幅（约）/页	发布日期/年	标准	大致页数	发布日期/年
SQL 86		1986年	SQL 2003	3 600	2003
SQL 89（FIPS 127-1）	120页	1989年	SQL 2008	3 777	2008
SQL 92	622页	1992年	SQL 2011	3817	2011
SQL 99（SQL 3）	1700页	1999年	SQL2016	4035	2016

SQL标准的发展过程

SQL的产生与发展（续）

- SQL 86和SQL 89是单个文档。
- SQL 92和SQL 99扩展为一系列开放的部分。例如，SQL 92增加了SQL调用接口、SQL永久存储模块；
- SQL 99扩展为框架、SQL基础部分、SQL调用接口、SQL永久存储模块、SQL宿主语言绑定、SQL外部数据的管理和SQL对象语言绑定等
- SQL 2016扩展到了12个部分，引入XML类型、Window函数、TRUNCATE操作、时序数据以及JSON（JavaScript Object Notation）类型等

目前，没有一个关系数据库管理系统能够支持SQL标准的所有概念和特性

3.1 SQL概述

3.1.1 SQL 的产生与发展

3.1.2 SQL的特点

3.1.3 SQL的基本概念

3.1.2 SQL的特点

1.功能综合且风格统一

- 集数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体。
- 可以独立完成数据库生命周期中的全部活动：
  - 创建和删除数据库模式
  - 创建基本表，创建视图
  - 使用数据库，包括查询和增删改数据、事务处理等
  - 数据库控制，包括安全性控制、完整性控制和并发控制等
  - 数据库维护和重构，如修改和删除基本表、数据库备份与恢复等
- 用户在数据库投入运行后，可根据需要随时或逐步创建模式
- 数据操作符统一：实体及实体间联系均用关系表示，克服了操作复杂性

SQL的特点（续）

2. 数据操纵高度非过程化

- 层次、网状模型的数据操纵语言面向过程，必须指定存取路径
- SQL只要提出“做什么”，无须了解存取路径
- 存取路径的选择以及SQL的操作过程由系统自动完成

3. 面向集合的操作方式

- 层次、网状模型采用面向记录的操作方式，操作对象是一条记录
- SQL采用集合操作方式
  - 操作对象、查找结果可以是元组的集合
  - 一次插入、删除、更新操作的对象也可以是元组的集合

SQL的特点（续）

4. 以统一的语法结构提供多种使用方式

- **SQL是独立的语言**

  能够独立用于联机交互的使用方式（在终端对利用SQL命令操作数据库）

- **SQL又是嵌入式语言**

  SQL能够嵌入到高级语言（例如C、C++、Java、Python）程序中，供程序设计使用

5.语言简洁且易学易用

- **SQL功能极强，完成核心功能只用9个动词**

SQL 功能	动词
数据定义	CREATE，DROP，ALTER
数据查询	SELECT
数据操纵	INSERT，UPDATE，DELETE
数据控制	GRANT，REVOKE

3.1 SQL概述

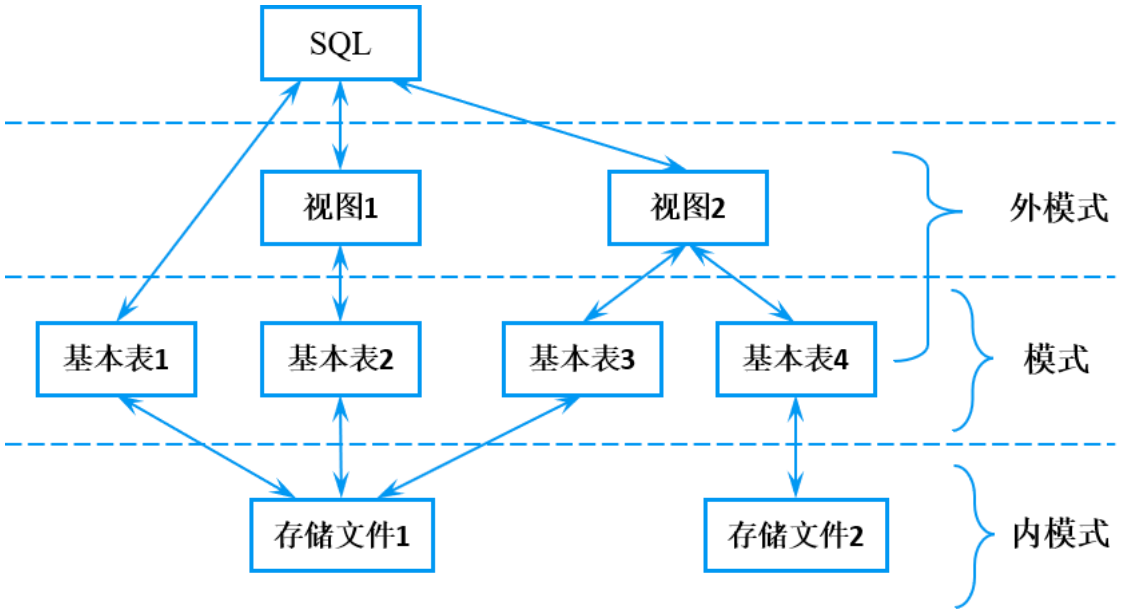
3.1.1 SQL 的产生与发展

3.1.2 SQL的特点

3.1.3 SQL的基本概念

3.1.3 SQL的基本概念

- **SQL支持关系数据库三级模式**
  - 外模式：对应用户能够看见和使用的数据结构，包括视图和部分基本表
  - 模式：对应基本表
  - 内模式：对应存储文件



SQL的基本概念（续）

- 基本表
  - 本身独立存在的表
  - 关系数据库管理系统中一个关系就对应一个基本表
  - 一个或多个基本表对应一个存储文件
  - 一个表可以带若干索引
- 视图
  - 从基本表或其他视图中导出的表（封装）
  - 数据库中只存放视图的定义而不存放视图对应的数据
  - 视图是一个虚表
  - 用户可以在视图上再定义视图（嵌套或链接）
- 存储文件
  - 逻辑结构和物理结构组成了关系数据库的内模式
  - 物理文件结构是由数据库管理系统设计确定的

第三章 关系数据库标准语言SQL

3.1 SQL概述

3.2 数据定义

3.3 数据查询

3.4 数据更新

3.5 空值的处理

3.6 视图

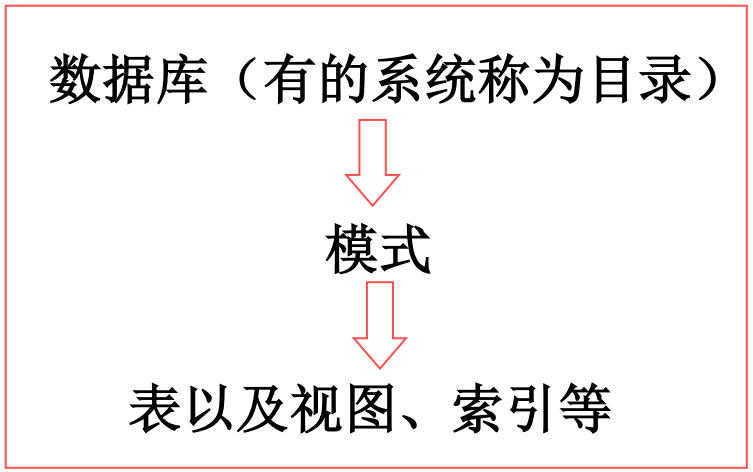
3.2 数据定义

❖SQL的数据定义功能:

- ■ 数据库模式定义
- ■ 表定义
- ■ 视图和索引的定义

操作对象	操作方式		
	创建	删除	修改
数据库模式	CREATE SCHEMA	DROP SCHEMA	*SQL标准无修改语句
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	*CREATE INDEX	*DROP INDEX	*ALTER INDEX

模式



- 定义模式实际上定义了一个命名空间。
  - 在这个空间中可以定义该模式包含的数据库对象，例如基本表、视图、索引等。

- 关系数据库管理系统提供层次化的数据库对象命名机制
  - 一个关系数据库管理系统的实例中可以建立多个数据库
  - 一个数据库中可以建立多个模式
  - 一个模式下通常包括多个表、视图和索引等数据库对象

3.2 数据定义

3.2.1 模式的定义与删除

3.2.2 基本表的定义、删除与修改

3.2.3 索引的建立与删除

3.2.4 数据字典

1. 定义模式

- 在SQL中，定义模式的语句：

  CREATE SCHEMA [<模式名>] AUTHORIZATION <用户名>;

  如果没有指定<模式名>，则 <模式名> 隐含为<用户名>。

  用户名即该模式的所有者。

[例3.1] 为用户**WANG**定义一个“学生选课”模式**S-C-SC**

CREATE SCHEMA “S-C-SC” AUTHORIZATION WANG;

[例3.2] CREATE SCHEMA AUTHORIZATION WANG;

该语句没有指定<模式名>，<模式名>隐含为用户名**WANG**

定义模式（续）

- 定义模式实际上定义了一个命名空间。在这个空间中可以定义该模式包含的数据库对象，例如基本表、视图、索引等，可以用**CREATE**语句定义。

- 在**CREATE SCHEMA**中可以接受**CREATE TABLE**，**CREATE VIEW**和**GRANT**子句，即用户可以在创建模式的同时在该模式定义中进一步创建基本表、视图，定义授权：

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>

[<表定义子句>|<视图定义子句>|<授权定义子句>];

定义模式（续）

［例3.3］为用户ZHANG创建一个模式Test，并且在其中定义一个表Tab1。

```
CREATE SCHEMA Test AUTHORIZATION Zhang
CREATE TABLE Tab1(Col1 SMALLINT,
                  Col2 INT,
                  Col3 CHAR(20),
                  Col4 NUMERIC(10,3),
                  Col5 DECIMAL(5,2)
                  );
```

2. 删除模式

❖ 在SQL中，删除模式的语句：

DROP SCHEMA <模式名> <CASCADE|RESTRICT>

■**CASCADE**（级联）

- 删除模式的同时把该模式中所有的数据库对象全部删除

■**RESTRICT**（限制）

- 如果该模式中定义了数据库对象（如表、视图等），则拒绝该删除语句的执行；仅当该模式中没有任何下属的对象时才能执行

删除模式（续）

[例3.4] DROP SCHEMA Test CASCADE;

删除模式**Test**

该模式中定义的表**Tab1**也被删除

3.2 数据定义

3.2.1 模式的定义与删除

3.2.2 基本表的定义、删除与修改

3.2.3 索引的建立与删除

3.2.4 数据字典

定义基本表

❖ **1.定义基本表**

CREATE TABLE <表名> (<列名> <数据类型> [列级完整性约束]

[,<列名> <数据类型>[列级完整性约束]]

…

[,<表级完整性约束>]);

- **<表名>：所要定义的基本表的名字**
- **<列名>：组成该表的各个属性（列）**
- **<列级完整性约束>：涉及相应属性列的完整性约束**
- **<表级完整性约束>：涉及一个或多个属性列的完整性约束**

- 如果完整性约束涉及该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级。
- 完整性约束具体内容参考第5章。

定义基本表（续）

❖ 以学生选课数据库为例

定义一个“学生选课”模式**S-C-SC**，包括以下三个表：

- “学生”表：**Student (Sno, Sname, Ssex, Sbirthdate, Smajor)**
- “课程”表：**Course (Cno, Cname, Ccredit, Cpno)**
- “学生选课”表：**SC (Sno, Cno, Grade, Semester, Teachingclass)**

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

学生表Student

[例3.5] 建立“学生”表Student

```
CREATE TABLE Student
    (Sno CHAR(8) PRIMARY KEY,          /* 列级完整性约束,Sno是主码*/
     Sname VARCHAR(20) UNIQUE,         /* Sname取唯一值*/
     Ssex CHAR(6),
     Sbirthdate Date,
     Smajor  VARCHAR(40)
     );
```

课程表Course

[例3.6] 建立一个“课程”表Course

```
CREATE TABLE  Course
    (Cno CHAR(5) PRIMARY KEY,          /* 列级完整性约束,Cno是主码*/
     Cname  VARCHAR(40) NOT NULL,  /* 列级完整性约束,Cname不能为空*/
     Ccredit  SMALLINT，
     Cpno  CHAR(5),
     FOREIGN KEY (Cpno) REFERENCES Course(Cno)
     /*表级完整性约束，Cpno是外码，被参照表是Course，被参照列是Cno*/
     );
```

注：被参照表和参照表相同的例子参考第2.3.2章（教材43-44页）

学生选课表SC

[例3.7] 建立“学生选课”表SC

```
CREATE TABLE SC
        (Sno CHAR(8),
         Cno CHAR(5),
         Grade SMALLINT,              /*成绩*/
         Semester CHAR(5),            /*开课学期*/
         Teachingclass CHAR(8),       /*学生选修某一门课所在的教学班*/
         PRIMARY KEY (Sno,Cno),    /*主码由两个属性构成，必须作为表级完整性进行定义*/
         FOREIGN KEY (Sno) REFERENCES Student(Sno),
          /*表级完整性约束，Sno是外码，被参照表是Student*/
         FOREIGN KEY (Cno) REFERENCES Course(Cno)
          /*表级完整性约束，Cno是外码，被参照表是Course*/
        );
```

2.数据类型

❖2.数据类型

- 关系模型中每一个属性来自一个域，它的取值必须是域中的值。SQL中域的概念用数据类型来实现。
- 定义表的属性时需要指明其数据类型及长度
- 选用哪种数据类型根据实际情况来决定：
  - 取值范围
  - 要做哪些运算

数据类型（续）

表 3.4　SQL 标准常用的数据类型

数据类型	含义
CHAR(n)，CHARACTER(n)	长度为 n 的定长字符串
VARCHAR(n)，CHARACTERVARYING(n)	最大长度为 n 的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT，INTEGER	整数(4 字节)，取值范围是[-2 147 483 648，2 147 483 647]
SMALLINT	短整数(2 字节)，取值范围是[-32 768，32 767]
BIGINT	大整数(8 字节)，取值范围是[-2^63，2^63-1]
NUMERIC(p，d)	定点数，由 p 位数字(不包括符号、小数点)组成，小数点后面有 d 位数字
DECIMAL(p，d)，DEC(p，d)	同 NUMERIC 类似，但数值精度不受 p 和 d 的限制
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(n)	可选精度的浮点数，精度至少为 n 位数字
BOOLEAN	逻辑布尔量
DATE	日期，包含年、月、日，格式为 YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为 HH：MM：SS
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

3. 模式与表

❖ 3.模式与表

■ 每一个基本表都属于某一个模式，一个模式包含多个基本表

■ 定义基本表所属模式

● 方法一：在表名中明显地给出模式名

```
Create table "S-C-SC".Student(…);    /*Student所属的模式是S-C-SC*/
Create table "S-C-SC".Course(…);     /*Course所属的模式是S-C-SC*/
Create table "S-C-SC".SC(…);         /*SC所属的模式是S-C-SC*/
```

● 方法二：在创建模式语句中同时创建表

● 方法三：设置所属的（默认）模式，这样在创建表时表名中不必给出模式名

3. 模式与表

- 创建基本表（其他数据库对象）时，若没有指定模式，系统根据搜索路径来确定该对象所属的模式。搜索路径包含一组模式列表。
- 关系数据库管理系统的搜索路径：使用模式列表中第一个存在的模式作为数据库对象的模式名 。
- 若搜索路径中的模式名都不存在，系统将给出错误。

■ 显示当前搜索路径：**SHOW SEARCH_PATH**

- 搜索路径的默认值是“**$user, Public**”，表示首先搜索与用户名相同的模式名，如果该模式名不存在，则使用**PUBLIC**模式。
- **PUBLIC**：默认的预定义模式，用于存储没有指定模式的对象。

模式与表（续）

■数据库管理员可以设置搜索路径，例如：

```
SET SEARCH_PATH TO "S-C-SC", PUBLIC;
```

- 随后定义基本表：

```
Create table Student (.....);
```

- 实际结果是建立了**S-C-SC.Student**基本表，因为关系数据库管理系统发现搜索路径中第一个模式名**S-C-SC**，就把该模式作为基本表**Student**所属的模式。

4. 修改基本表

```
ALTER TABLE <表名>
    [ ADD [COLUMN] <新列名> <数据类型> [ 完整性约束 ] ]
    [ ADD <表级完整性约束>]
    [ DROP [ COLUMN ] <列名> [ CASCADE| RESTRICT] ]
    [ DROP CONSTRAINT<完整性约束名>[ RESTRICT | CASCADE ] ]
    [ RENAME COLUMN <列名> TO <新列名> ]
    [ ALTER COLUMN <列名> TYPE <数据类型> ];
```

- <表名>是要修改的基本表
- ADD子句用于增加新列、新的列级完整性约束和新的表级完整性约束
- DROP COLUMN子句用于删除表中的列
  - 如果指定了CASCADE短语，则自动删除引用了该列的其他对象
  - 如果指定了RESTRICT短语，则如果该列被其他对象引用，关系数据库管理系统将拒绝删除该列
- DROP CONSTRAINT子句用于删除指定的完整性约束
- RENAME COLUMN子句用于修改列名
- ALTER COLUMN子句用于修改列的数据类型

修改基本表（续）

[例3.8] 向Student表增加“邮箱地址”列Semail，其数据类型为字符型

```
ALTER TABLE Student ADD Semail VARCHAR(30);
```

注意不论基本表中原来是否已有数据，新增加的列一律为空值

[例3.9] 将Student表中出生日期Sbirthdate的数据类型由DATE型改为字符型

```
ALTER TABLE Student ALTER COLUMN Sbirthdate TYPE VARCHAR(20);
```

注意DATE类型占用19字节，所以修改为VARCHAR时长度要大于等于19

[例3.10] 增加课程名称必须取唯一值的约束条件

```
ALTER TABLE Course ADD UNIQUE(Cname);
```

5.删除基本表

DROP TABLE <表名>［RESTRICT| CASCADE］;

■ **RESTRICT（默认）：删除表是有限制的**

- 欲删除的基本表不能被其他表的约束所引用
- 如果存在依赖该表的对象，则此表不能被删除

■ **CASCADE：删除该表没有限制**

- 在删除基本表的同时，相关的依赖对象一起删除

[例3.11] 删除**Student**表，选择**CASCADE**选项

DROP TABLE Student CASCADE;

基本表定义被删除，数据被删除，表上建立的索引、视图、触发器等一般也将被删除。

删除基本表（续）

[例3.12] 删除Student表，若表上建有视图，选择RESTRICT时表不能删除。

```
CREATE VIEW CS_Student                /* Student表上建立计算机科学与技术专业学生视图*/
AS
SELECT Sno,Sname,Ssex,Sbirthdate,Smajor
FROM Student
WHERE Smajor='计算机科学与技术';

DROP TABLE Student RESTRICT;          /*删除Student表*/
--ERROR: cannot drop table Student because other objects depend on it    /* 系统返回错误信息，存在依赖该表的对象，此表不能被删除*/
```

删除基本表（续）

[例3.12续] 选择CASCADE时可以删除表，视图也自动被删除。

```
DROP TABLE Student CASCADE;                    /*删除Student表*/

--NOTICE: drop cascades to view CS_Student     /*系统返回提示，此表上的视图也被删除*/


SELECT * FROM CS_Student;                      /* CS_Student视图不存在*/

--ERROR: relation " CS_Student " does not exist
```

上机练习

❖ 1. 建立数据库

- ■ 在Mysql中建立数据库等效于建立模式：
  - ● **CREATE DATABASE DB1;**
  - ● **CREATE SCHEMA DB2;**

❖ 2. 基本表的建立、删除与修改

- ● [例 3.5]
- ● [例 3.6]
- ● [例 3.7]
- ● [例 3.8]
- ● [例 3.9]
- ● [例 3.10]
- ● [例 3.11]
- ● [例 3.12]

3.2 数据定义

3.2.1 模式的定义与删除

3.2.2 基本表的定义、删除与修改

3.2.3 索引的建立与删除

3.2.4 数据字典

3.2.3 索引的建立与删除

- 索引
  - 目的：加快查询速度。
  - 类似于图书的目录，能快速定位到需要查询的内容。
  - 用户可以根据应用环境在基本表上建立一个或多个索引，以提供多种存取路径，加快查找速度。
- 数据库常见索引（参考教材第9章）
  - 顺序文件上的索引
  - B+树索引
  - 哈希（hash）索引
  - 位图索引

索引的建立与删除

- 谁可以建立与删除索引：数据库管理员或表的属主（即建立表的人）。
- 谁维护索引：关系数据库管理系统自动完成 。
- 使用索引：关系数据库管理系统自动选择合适的索引作为存取路径，用户不必自主地选择索引。

- 注意：
  - 索引虽然能够加速查询数据，但需要占用一定的存储空间，当基本表更新时，索引也要进行相应的维护，这些都会增加数据库的负担，因此要根据实际应用的需要有选择地创建索引。
  - 目前SQL标准中没有涉及索引，但商用关系数据库管理系统一般都支持索引机制，但不同DBMS支持的索引类型不相同。

1. 建立索引

❖ 语句格式

CREATE [UNIQUE] [CLUSTER] INDEX <索引名>

ON <表名>(<列名>[<次序>][,<列名>[<次序>]]...) ;

- **<表名>：要建索引的基本表的名字**
- **索引：可以建立在该表的一列或多列上，各列名之间用逗号分隔**
- **<次序>：指定索引值的排列次序，升序：ASC，降序：DESC。默认值：ASC**
- **UNIQUE：此索引的每一个索引值只对应唯一的数据记录**
- **CLUSTER：表示要建立的索引是聚簇索引**

建立索引（续）

[例3.13]为“学生选课”数据库中的Student、Course和SC三个表建立索引。其中Student表按学生姓名升序建唯一索引，Course表按课程名升序建唯一索引，SC表按学号升序和课程号降序建唯一索引（即先按照学号升序，对同一个学号再按课程号降序）

```
CREATE UNIQUE INDEX Idx_StuSname ON Student(Sname);
                                    /*保证了Sname取唯一值的约束*/
CREATE UNIQUE INDEX Idx_CouCname ON Course(Cname);
                                    /*加上Cname取唯一值的约束*/
CREATE UNIQUE INDEX Idx_SCCno ON SC(Sno ASC,Cno DESC);
```

2. 修改索引

- 语句格式：

ALTER INDEX <旧索引名> RENAME TO <新索引名>

[例3.14] 将SC表的Idx_SCCno索引名改为Idx_SCSnoCno

ALTER INDEX Idx_SCCno RENAME TO Idx_SCSnoCno;

3. 删除索引

❖ 语句格式

DROP INDEX <索引名>;

删除索引时，系统会从数据字典中删去有关该索引的描述

[例3.15] 删除Student表的Idx_StuSname索引

DROP INDEX Idx_StuSname;

3.2 数据定义

3.2.1 模式的定义与删除

3.2.2 基本表的定义、删除与修改

3.2.3 索引的建立与删除

3.2.4 数据字典

3.2.4 数据字典

- 数据字典是关系数据库管理系统内部的一组系统表，它记录了数据库中所有定义信息：
  - 关系模式定义
  - 视图定义
  - 索引定义
  - 完整性约束定义
  - 各类用户对数据库的操作权限
  - 统计信息等
- 关系数据库管理系统在执行SQL的数据定义语句时，就是在更新数据字典表中的相应信息
- 查询优化和查询处理时，关系数据库管理系统要根据数据字典中的信息执行处理算法和优化算法

第3章 关系数据库标准语言SQL

3.1 SQL概述

3.2 数据定义

3.3 数据查询

3.4 数据更新

3.5 空值的处理

3.6 视图

3.3 数据查询

❖ 语句格式

SELECT [ALL | DISTINCT] <目标列表达式> [别名] [,<目标列表达式> [别名]] …

FROM <表名或视图名> [别名] [,<表名或视图名> [别名]] … | (<SELECT语句>)

[AS] <别名>

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC | DESC]]

LIMIT <行数1> [OFFSET <行数2>];

注：中括号内的关键字为可选。

- SELECT语句：指定要显示的属性列。
- FROM子句：指定查询对象（基本表或视图）。
- WHERE子句：指定查询条件。
- GROUP BY子句：结果按照<列名1>的值进行分组，该属性列值相等的元组为一个组，通常在每组中作用聚集函数。HAVING短语：只有满足指定条件的组才予以输出。
- ORDER BY子句：对查询结果表按<列名1>的值的升序或降序排序。
- LIMIT子句:限制SELECT语句查询结果的数量为<行数1>行，OFFSET <行数2>，表示在计算<行数1>行前忽略<行数2>行。

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5 基于派生表的查询

3.3.1 单表查询

- 查询仅涉及一个表
  - 1.选择表中的若干列
  - 2.选择表中的若干元组
  - 3.ORDER BY子句
  - 4.聚集函数
  - 5.GROUP BY子句
  - 6.LIMIT子句

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

1.选择表中的若干列

❖ （1）查询指定列

[例3.16] 查询全体学生的学号与姓名

```
SELECT Sno, Sname
FROM Student;
```

[例3.17] 查询全体学生的姓名、学号、主修专业。

```
SELECT Sname, Sno, Smajor
FROM Student;
```

选择表中的若干列（续）

❖ （2）查询全部列

■ 选出所有属性列：

- 在SELECT关键字后面列出所有列名
- 将<目标列表达式>指定为 *

[例3.18] 查询全体学生的详细记录

```
SELECT  Sno, Sname, Ssex, Sbirthdate, Smajor
FROM Student;
```

或

```
SELECT  *
FROM Student;
```

查询经过计算的值（续）

❖ （3）查询经过计算的值

■ SELECT子句的<目标列表达式>可以是算术表达式、字符串常量、函数等

[例3.19] 查全体学生的姓名及其年龄

KingbaseES语句为：

```
SELECT Sname, (extract(year from current_date) - extract(year from Sbirthdate)) "年龄"
FROM Student;
```

输出结果：

Sname	年龄
李勇	21
刘晨	22
王敏	20
张立	21
陈新奇	20
赵明	21
王佳佳	20

EXTRACT：从日期或时间值中提取特定部分的函数。

EXTRACT(field FROM source)

注：不同数据库管理系统支持的函数库不同，参考教材246，247页二维码

查询经过计算的值（续）

[例3.20] 查询全体学生的姓名、出生日期和主修专业

```
SELECT Sname, 'Date of Birth:', Sbirthdate, Smajor
FROM Student;
```

输出结果：

Sname	Date of Birth：	Sbirthdate	Smajor
李勇	Date of Birth：	2000-3-8	信息安全
刘晨	Date of Birth：	1999-9-1	计算机科学与技术
王敏	Date of Birth：	2001-8-1	计算机科学与技术
张立	Date of Birth：	2000-1-8	计算机科学与技术
陈新奇	Date of Birth：	2001-11-1	信息管理与信息系统
赵明	Date of Birth：	2000-6-12	数据科学与大数据技术
王佳佳	Date of Birth：	2001-12-7	数据科学与大数据技术

3.3.1 单表查询

- 查询仅涉及一个表：
  1. 选择表中的若干列
  2. 选择表中的若干元组
  3. ORDER BY子句
  4. 聚集函数
  5. GROUP BY子句
  6. LIMIT子句

2. 选择表中的若干元组

❖ （1）消除取值重复的行

如果没有指定DISTINCT关键词，则缺省为ALL

[例3.21] 查询选修了课程的学生学号。

```
SELECT Sno
FROM SC;
```

等价于：

```
SELECT ALL Sno
FROM SC;
```

消除取值重复的行（续）

- 该查询结果里包含了许多重复的行。如想去掉结果表中的重复行，必须指定**DISTINCT**：
-

```
SELECT DISTINCT Sno
FROM SC;
```

执行结果：

Sno
20180001
20180002
20180003
20180004
20180005

（2）查询满足条件的元组

常用的查询条件（教材第83页）

查询条件	谓词（逻辑条件）
比较	=，>，<，>=，<=，!=，<>，!>，!<；NOT+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件（逻辑运算）	AND，OR，NOT

① 比较大小

[例3.22] 查询主修计算机科学与技术专业全体学生的姓名

```
SELECT Sname
FROM   Student
WHERE  Smajor='计算机科学与技术';
```

[例3.23] 查询所有2000年后（包括2000年）出生的学生姓名及其性别

```
SELECT Sname, Ssex
FROM   Student
WHERE  extract(year from Sbirthdate)>=2000;
        /*函数extract(year from Sbirthdate)从出生日期中抽取出年份*/
```

[例3.24] 查询考试成绩不及格的学生的学号

```
SELECT DISTINCT Sno;
FROM  SC
WHERE Grade<60;
```

② 确定范围

❖ 谓词： **BETWEEN … AND …**

NOT BETWEEN … AND …

[例3.25] 查询年龄在**20~23**岁（包括**20**岁和**23**岁）之间的学生的学生的姓名、出生年月和主修专业

```
SELECT Sname, Sbirthdate, Smajor
FROM Student
WHERE extract(year from current_date) - extract(year from Sbirthdate) BETWEEN 20 AND 23;
```

[例3.26]查询年龄不在**20~23**岁之间的学生姓名、出生年月和主修专业

```
SELECT Sname,Sbirthdate,Smajor
FROM Student
WHERE extract(year from current_date) - extract(year from Sbirthdate) NOT BETWEEN 20 AND 23;
```

③ 确定集合

❖ 谓词：IN （集合），NOT IN （集合）

[例3.27] 查询计算机科学与技术专业和信息安全专业学生的姓名和性别

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor IN ( '计算机科学与技术' ,'信息安全' );
```

[例3.28] 查询既不是计算机科学与技术专业也不是信息安全专业学生的姓名和性别

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor NOT IN ( '计算机科学与技术' ,'信息安全' );
```

④ 字符匹配

❖ 谓词： [NOT] LIKE ‘<匹配串>’ [ESCAPE ‘ <换码字符>’]

<匹配串>：一个完整的字符串或含有通配符%和 _

■ %（百分号）：任意长度（长度可以为0）的字符串

- 例如a%b表示以a开头，以b结尾的任意长度的字符串

■ _（下横线）：任意单个字符。

- 例如a_b表示以a开头，以b结尾的长度为3的任意字符串

字符匹配（续）

- 匹配串为固定字符串

[例3.29] 查询学号为20180003的学生的详细情况

```
SELECT *
FROM  Student
WHERE  Sno LIKE '20180003';
```

等价于：

```
SELECT  *
FROM  Student
WHERE Sno = '20180003';
```

字符匹配（续）

■ 匹配串为含通配符的字符串

[例3.30]查询所有姓刘学生的姓名、学号和性别

```
SELECT Sname, Sno, Ssex
FROM Student
WHERE  Sname LIKE '刘%';
```

[例3.31]查询2018级学生的学号和姓名

```
SELECT Sno,Sname
FROM Student
WHERE Sno LIKE '2018%';       /*学号的数据类型是字符，用字符匹配*/
```

字符匹配（续）

[例3.32] 查询课程号为81开头，最后一位是6的课程名称和课程号

```
SELECT Cname,Cno
FROM Course
WHERE Cno LIKE '81_ _6';
```

/* 注意课程关系中课程号为固定长度，占5个字符大小 */

[例3.33] 查询所有不姓刘的学生姓名、学号和性别

```
SELECT Sname, Sno, Ssex
FROM    Student
WHERE  Sname NOT LIKE '刘%';
```

字符匹配（续）

- 使用换码字符将通配符转义为普通字符

[例3.34] 查询DB_Design课程的课程号和学分

```
SELECT Cno，Ccredit
FROM    Course
WHERE  Cname LIKE 'DB\_Design' ESCAPE '\' ;
```

[例3.35] 查询以“DB_”开头，且倒数第三个字符为i的课程的详细情况。

```
SELECT *
FROM Course
WHERE Cname LIKE 'DB\_%i_ _' ESCAPE '\';
```

- ESCAPE '\' 表示 '\' 为换码字符
- 第一个_前面有换码字符\，被转义为普通的_字符
- i后面的两个_ 的前面均没有换码字符\，仍作为通配符

⑤ 涉及空值的查询

❖谓词： IS NULL 或 IS NOT NULL

■ “IS” 不能用 “=” 代替

[例3.36] 某些学生选修课程后没有参加考试，有选课记录，但没有考试成绩。查询缺少成绩的学生的学号和相应的课程号。

```
SELECT Sno, Cno
FROM SC
WHERE Grade IS NULL;          /*分数Grade是空值*/
```

[例3.37] 查所有有成绩的学生学号和课程号

```
SELECT Sno，Cno
FROM    SC
WHERE  Grade IS NOT NULL;
```

⑥多重条件查询

❖ 逻辑运算符：AND和 OR来连接多个查询条件

■ AND的优先级高于OR，可以用括号改变优先级

[例3.38] 查询主修计算机科学与技术专业2000年（包括2000年）以后出生的学生学号、姓名和性别。

```
SELECT Sno,Sname,Ssex
FROM Student
WHERE Smajor='计算机科学与技术' AND extract(year from Sbirthdate)>=2000;
```

例3.27中的IN谓词实际上是多个OR运算符的缩写：

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor='计算机科学与技术' OR Smajor='信息安全';
```

[例3.27] 查询计算机科学与技术专业和信息安全专业学生的姓名和性别

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor IN ( '计算机科学与技术' ,'信息安全' );
```

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

3. ORDER BY子句

❖ **ORDER BY子句**

■可以按一个或多个属性列排序升序（**ASC，Ascending**）降序（**DESC，Descending**）排列，**默认值为升序ASC**

■对于空值，排序时显示的次序由具体系统实现决定

ORDER BY子句（续）

[例3.39]查询选修了81003号课程的学生的学号及其成绩，查询结果按分数的降序排列

```
SELECT Sno, Grade
FROM SC
WHERE Cno='81003’
ORDER BY Grade DESC;
```

[例3.40]查询全体学生选修课程情况，查询结果先按照课程号升序排列，同一课程中按成绩降序排列。

```
SELECT *
FROM SC
ORDER BY Cno ASC, Grade DESC;        /*ASC可以省略，DESC需要明确写出来*/
```

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

4. 聚集函数

❖ 聚集函数：

■统计元组个数：**COUNT(*)**

■统计一列中值的个数：**COUNT([DISTINCT|ALL] <列名>)**

■计算一列值的总和（此列必须为数值型）：**SUM([DISTINCT|ALL] <列名>)**

■计算一列值的平均值（此列必须为数值型）：**AVG([DISTINCT|ALL] <列名>)**

■求一列中的最大值和最小值

●**MAX([DISTINCT|ALL] <列名>)**

●**MIN([DISTINCT|ALL] <列名>)**

聚集函数（续）

[例3.41] 查询学生总人数

```
SELECT COUNT(*)      /*统计元组数*/
FROM  Student;
```

[例3.42] 查询选修了课程的学生人数

```
SELECT COUNT(DISTINCT Sno)
FROM SC;
```

[例3.43] 计算选修81001号课程的学生平均成绩

```
SELECT AVG(Grade)
FROM  SC
WHERE Cno= ‘ 81001 ';
```

聚集函数（续）

[例3.44] 查询选修81001号课程的学生最高分数

```
SELECT MAX (Grade)
FROM SC
WHERE Cno='81001';
```

[例3.45] 查询学号为20180003学生选修课程的总学分数

```
SELECT SUM (Ccredit)
FROM  SC, Course
WHERE Sno='20180003' AND SC.Cno=Course.Cno;
```

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

5. GROUP BY子句

- **GROUP BY子句分组：**
  - 按指定的一列或多列值分组，值相等的为一组
- 查询结果分组的目的之一：细化聚集函数的作用对象
  - 如果未对查询结果分组，聚集函数将作用于整个查询结果
  - 分组后，聚集函数将作用于每一个组

GROUP BY子句（续）

[例3.46] 求各个课程号及选修该课程的人数

```
SELECT Cno，COUNT(Sno)     /*COUNT作用于由Cno区分的每一组*/
FROM SC
GROUP BY Cno;
```

查询结果可能为：

Cno	COUNT(Sno)
81001	42
81002	44
81003	44
81004	33
81005	48
81006	45
81007	48
81008	39

GROUP BY子句（续）

- 如果分组后还要按一定要求对这些组进行筛选，最终只输出满足指定条件的组，则可以使用HAVING短语指定筛选条件。

[例3.47] 查询2019年第2学期选修了10门以上课程的学生学号

```
SELECT Sno
FROM  SC
WHERE Semester='20192'          /*先求出2019年第2学期选课的所有学生*/
GROUP BY Sno                    /*用GROUP BY子句按Sno进行分组*/
HAVING COUNT(*) >10;            /* 聚集函数COUNT对每一组计数 */
```

GROUP BY子句（续）

[例3.48]查询平均成绩大于等于90分的学生学号和平均成绩

```
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade)>=90;
```

下面的语句是错误的：WHERE子句不能用聚集函数作为条件表达式。

```
SELECT Sno, AVG(Grade)
FROM SC
WHERE AVG(Grade)>=90
GROUP BY Sno;
```

GROUP BY子句（续）

- **HAVING短语与WHERE子句的区别：**
  - **WHERE子句作用于基表或视图，从中选择满足条件的元组**
  - **HAVING短语作用于组，从中选择满足条件的组**

6.LIMIT子句

LIMIT子句用于限制SELECT语句查询结果的（元组）数量

LIMIT <行数1> [OFFSET <行数2>];

- 语义是忽略前<行数2>行，然后取<行数1>作为查询结果数据
- OFFSET可以省略，代表不忽略任何行
- LIMIT子句经常和ORDER BY子句一起使用

LIMIT子句（续）

[例3.49]查询选修了数据库系统概论课程的成绩排名前10名的学生学号

```
SELECT Sno
FROM SC,Course
WHERE Course.Cname='数据库系统概论' AND SC.Cno=Course.Cno
ORDER BY Grade DESC
LIMIT 10;              /*取前10行数据为查询结果*/
```

LIMIT子句（续）

[例3.50] 查询平均成绩排名在3-5名的学生学号和平均成绩

```
SELECT Sno,AVG(Grade)
FROM SC
GROUP BY Sno
ORDER BY AVG(Grade) DESC
LIMIT 5 OFFSET 2;  /*取5行数据，忽略前2行，之后为查询结果数据*/
```

单表查询上机练习

- 基于教材第73页二维码中的示例数据，构建三个基本表，练习并思考：
- [例 3.16—例3.50]

- 注意保存构建的表便于后续练习。

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

3.3.2 连接查询

- 连接查询：同时涉及两个以上的表的查询
- 连接条件或连接谓词：用来连接两个表的条件

  一般格式：

  - [<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>
  - [<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>
- 连接字段：连接谓词中的列名称
  - 连接条件中的各连接字段类型必须是可比的，但名字不必相同

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

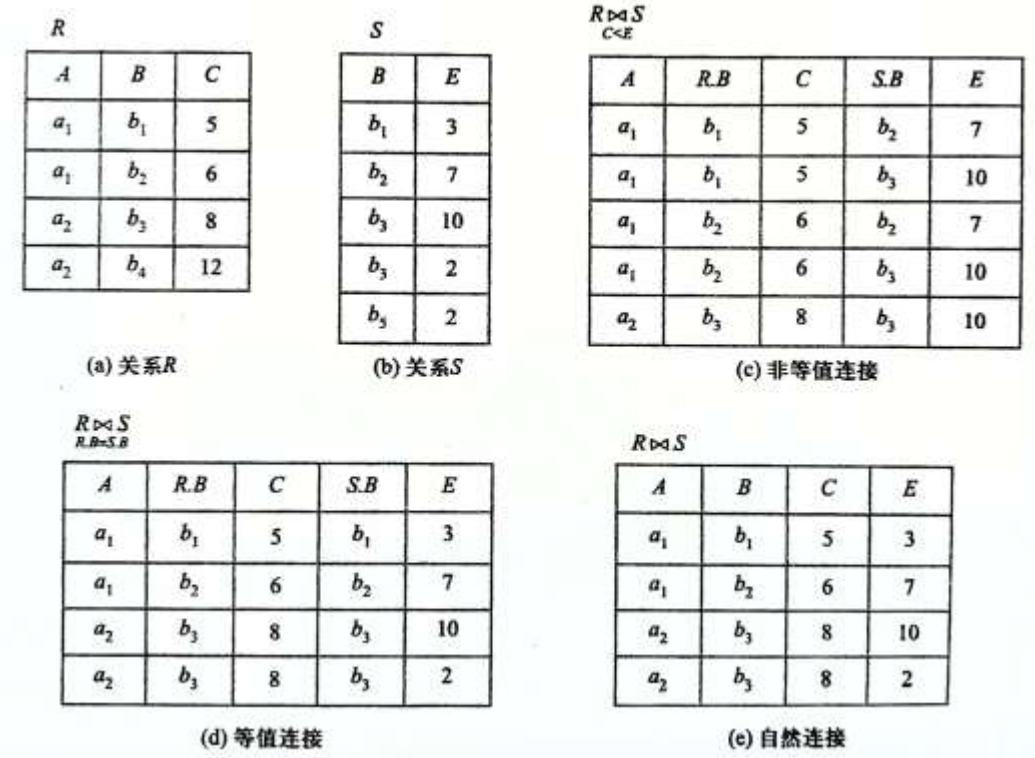
6.多表连接查询

1. 等值与非等值连接查询

❖ 等值连接：连接运算符为 =

[例3.51]查询每个学生及其选修课程的情况

```
SELECT Student.*, SC.*
FROM   Student, SC
WHERE Student.Sno=SC.Sno
/* 将Student与SC中同一学生的元组连接起来 */
```



R

A	B	C
a_1	b_1	5
a_1	b_2	6
a_2	b_3	8
a_2	b_4	12

(a) 关系R

S

B	E
b_1	3
b_2	7
b_3	10
b_3	2
b_5	2

(b) 关系S

$R \bowtie S$ (C<E)

A	R.B	C	S.B	E
a_1	b_1	5	b_2	7
a_1	b_1	5	b_3	10
a_1	b_2	6	b_2	7
a_1	b_2	6	b_3	10
a_2	b_3	8	b_3	10

(c) 非等值连接

$R \bowtie S$ (R.B=S.B)

A	R.B	C	S.B	E
a_1	b_1	5	b_1	3
a_1	b_2	6	b_2	7
a_2	b_3	8	b_3	10
a_2	b_3	8	b_3	2

(d) 等值连接

$R \bowtie S$

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2

(e) 自然连接

思考：学习过程对照教材第50-51页基于关系代数的连接操作

等值与非等值连接查询（续）

假设Student表、SC表的数据如第2章图2.1所示，本查询的执行结果如下图所示

Student.Sno	Sname	Ssex	Sbirthdate	Smajor	SC.Sno	Cno	Grade	Semester	Teachingclass
20180001	李勇	男	2000-3-8	信息安全	20180001	81001	85	20192	81001-01
20180001	李勇	男	2000-3-8	信息安全	20810001	81002	96	20201	81002-01
20180001	李勇	男	2000-3-8	信息安全	20810001	81003	87	20202	81003-01
20180002	刘晨	女	1999-9-1	计算机科学与技术	20180002	81001	80	20192	81001-01
20180002	刘晨	女	1999-9-1	计算机科学与技术	20180002	81002	98	20201	81002-01
20180002	刘晨	女	1999-9-1	计算机科学与技术	20810002	81003	71	20202	81003-01
。。。。	。。。	。。。	。。。	。。。	。。。	。。。	。。。	。。。	。。。

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

2. 自然连接查询

- 若在等值连接中把目标列中重复的属性列去掉则为自然连接，如[例3.52]中去掉了SC表中的Sno

[例3.52] 查询每个学生的学号、姓名、性别、出生日期、主修专业及该学生选修课程的课程号与成绩。

```
SELECT Student.Sno,Sname,Ssex,Sbirthdate,Smajor,Cno,Grade
FROM  Student,SC
WHERE Student.Sno=SC.Sno;
```

- Sname，Ssex，Sbirthdate，Smajor，Cno和Grade属性列在Student表与SC表中唯一，引用时可以去掉表名前缀
- Sno在两个表都出现，因此SELECT子句和WHERE子句在引用时必须加上表名前缀

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

3.复合条件连接查询

- 复合条件连接：WHERE子句是由连接谓词和选择谓词组成的复合条件

[例3.53]查询选修81002号课程且成绩在90分以上的所有学生的学号和姓名

```
SELECT Student.Sno,Sname
FROM Student,SC
WHERE Student.Sno=SC.Sno  AND                    /*连接谓词 */
      SC.Cno='81002' AND SC.Grade>90;            /*其他选择条件*/
```

- 优化（高效）执行过程
- 先从SC中挑选出Cno='81002'并且Grade>90的元组形成一个中间关系
- 再和Student中满足连接条件的元组进行连接得到最终的结果关系

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

4. 自身连接

- 自身连接：一个表与其自己进行连接（复制自身）
- 需要给表起别名以示区别
- 由于所有属性名都是同名属性，因此必须使用别名前缀

[例3.54]查询每一门课的间接先修课（即先修课的先修课）

```
SELECT FIRST.Cno, SECOND.Cpno
FROM Course FIRST, Course SECOND
WHERE FIRST.Cpno=SECOND.Cno AND SECOND.Cpno IS NOT NULL;
```

自身连接（续）

FIRST表（Course表）

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与C语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

SECOND表（Course表）

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与C语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

自身连接（续）

❖ 分析

- 查询每一门课的间接先修课，必须先对一门课找到其直接先修课Cpno
- 再按此先修课的课程号查找它的先修课程
- 相当于将Course表与其自身连接后，取第一个副本的课程号与第二个副本的先修课号做为目标列中的属性。
- 可以为Course表取两个别名：FIRST和SECOND
- 这就是Course表与其自身连接

❖ 查询结果

Cno	Cpno
81003	81001
81004	81002
82006	81001
81008	81002

[例3.54]查询每一门课的间接先修课（即先修课的先修课）

```
SELECT FIRST.Cno, SECOND.Cpno
FROM Course FIRST, Course SECOND
WHERE FIRST.Cpno=SECOND.Cno AND
SECOND.Cpno IS NOT NULL;
```

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

5. 外连接查询

❖ 外连接与普通连接的区别

■普通连接操作只输出满足连接条件的元组

■外连接操作以指定表为连接主体，将主体表中不满足连接条件的元组一并输出

■ 左外连接

●列出左边关系中所有的元组

■ 右外连接

●列出右边关系中所有的元组

R

A	*B*	*C*
a_1	b_1	5
a_1	b_2	6
a_2	b_3	8
a_2	b_4	12

(a) 关系*R*

S

B	*E*
b_1	3
b_2	7
b_3	10
b_3	2
b_5	2

(b) 关系*S*

A	*B*	*C*	*E*
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
a_2	b_4	12	NULL
NULL	b_5	NULL	2

(a) 外连接

A	*B*	*C*	*E*
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
a_2	b_4	12	NULL

(b) 左外连接

A	*B*	*C*	*E*
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	2
NULL	b_5	NULL	2

(c) 右外连接

外连接查询（续）

[例3.55] 以Student表为主体列出每个学生的基本情况及其选课情况，若某个学生没有选课，则只输出其基本情况的数据，而把选课信息填为空值NULL

```
SELECT Student.Sno,Sname,Ssex,Sbirthdate,Smajor,Cno,Grade
FROM Student LEFT OUTER JOIN SC ON (Student.Sno=SC.Sno);
```

（左外连接 保留学生关系的所有元组）

Student.Sno	Sname	Ssex	Sbirthdate	Smajor	Cno	Grade
20180001	李勇	男	2000-3-8	信息安全	81001	85
20180001	李勇	男	2000-3-8	信息安全	81002	96
20180001	李勇	男	2000-3-8	信息安全	81003	87
20180002	刘晨	女	1999-9-1	计算机科学与技术	81001	80
20180002	刘晨	女	1999-9-1	计算机科学与技术	81002	98
20180002	刘晨	女	1999-9-1	计算机科学与技术	81003	71
20180003	王敏	女	2001-8-1	计算机科学与技术	81001	81
20180003	王敏	女	2001-8-1	计算机科学与技术	81002	76
20180004	张立	男	2000-1-8	计算机科学与技术	81001	56
20180004	张立	男	2000-1-8	计算机科学与技术	81002	97
20180205	陈新奇	男	2001-11-1	信息管理与信息系统	81003	68
20180306	赵明	男	2000-6-12	数据科学与大数据技术	NULL	NULL
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术	NULL	NULL

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

6. 多表连接查询

❖ 多表连接：两个以上的表进行连接

[例3.56]查询每个学生的学号、姓名、选修的课程名及成绩。

```
SELECT Student.Sno,Sname,Cname,Grade
FROM Student,SC,Course
WHERE Student.Sno=SC.Sno AND SC.Cno=Course.Cno;
```

❖ 可能的执行方式

- 先将Student表与SC表进行连接，得到每个学生的学号、姓名、所选课程号和相应的成绩
- 再将其与Course表进行连接，得到最终结果

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

嵌套查询（续）

❖ 嵌套查询概述

■ 一个SELECT-FROM-WHERE语句称为一个查询块

■ 将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中的查询称为嵌套查询

➢ 例如：查询选修81003号课程的所有学生姓名。

```
SELECT Sname                    /*外层查询或父查询*/
FROM Student
WHERE Sno IN
        (SELECT Sno             /*内层查询或子查询*/
        FROM SC
        WHERE Cno= '81003');
```

嵌套查询（续）

- 上层的查询块称为外层查询或父查询
- 下层的查询块称为内层查询或子查询
- SQL语言允许多层嵌套查询
  - 即一个子查询中还可以嵌套其他子查询
- 子查询的限制
  - SELECT语句不能使用ORDER BY子句

嵌套查询求解方法

- 不相关子查询：子查询的查询条件不依赖于父查询
  - 由里向外逐层处理。即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件。

- 相关子查询：子查询的查询条件依赖于父查询
  - 首先取外层查询中表的第一个元组，根据它与内层查询相关的属性值处理内层查询，若WHERE子句返回值为真，则取此元组放入结果表。
  - 然后再取外层表的下一个元组。
  - 重复这一过程，直至外层表全部检查完为止。

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

1. 带有IN谓词的子查询

[例3.57]查询与“刘晨”在同一个主修专业的学生学号、姓名和主修专业

此查询要求可以分步来完成

① 确定“刘晨”所在系名

```
SELECT  Smajor
FROM    Student
WHERE  Sname= ' 刘晨 ';
```

结果为： 计算机科学与技术

② 查找所有主修计算机科学与技术专业的学生

```
SELECT   Sno, Sname, Smajor
FROM     Student
WHERE   Smajor= ' 计算机科学与技术 ';
```

Sno	Sname	Smajor
20180002	刘晨	计算机科学与技术
20180003	王敏	计算机科学与技术
20180004	张立	计算机科学与技术

带有IN谓词的子查询（续）

[例3.57]查询与“刘晨”在同一个主修专业的学生学号、姓名和主修专业

- 构造嵌套查询语句：

```
SELECT Sno, Sname, Smajor        /*例3.57的解法一*/
FROM Student
WHERE Smajor IN
      (SELECT Smajor
      FROM Student
      WHERE Sname='刘晨');
```

■ 子查询的查询条件不依赖于父查询，为不相关子查询

带有IN谓词的子查询（续）

[例3.57]查询与“刘晨”在同一个主修专业的学生学号、姓名和主修专业

- 用自身连接：

```
/*例3.57的解法二 */
SELECT S1.Sno,S1.Sname,S1.Smajor
FROM Student S1,Student S2
WHERE S1.Smajor=S2.Smajor AND S2.Sname='刘晨';
```

Sno	Sname	Smajor
20180002	刘晨	计算机科学与技术
20180003	王敏	计算机科学与技术
20180004	张立	计算机科学与技术

Sno	Sname	Smajor
20180002	刘晨	计算机科学与技术
20180003	王敏	计算机科学与技术
20180004	张立	计算机科学与技术

带有IN谓词的子查询（续）

[例3.58]查询选修了课程名为“信息系统概论”的学生的学号和姓名

```
SELECT Sno,Sname
FROM Student
WHERE Sno IN
        (SELECT Sno
         FROM SC
         WHERE Cno IN
                (SELECT Cno
                FROM Course
                WHERE Cname='信息系统概论'
                )
        );
```

③最后在Student关系中取出Sno和Sname

②然后在SC关系中找出选修81004号课程的学生学号

①首先在Course关系中找出“信息系统概论”的课程号，结果为81004

带有IN谓词的子查询（续）

[例3.58]查询选修了课程名为“信息系统概论”的学生的学号和姓名

连接查询：

```
SELECT Student.Sno, Sname
FROM Student,SC,Course
WHERE Student.Sno=SC.Sno AND
      SC.Cno=Course.Cno AND
      Course.Cname='信息系统概论';
```

思考：连接查询和嵌套查询的效率？

而嵌套查询（子查询）可能需要多次扫描或临时表操作，导致性能下降。

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

2. 带有比较运算符的子查询

❖ 当能确切知道内层查询返回单值时，可用比较运算符（>，<，=，>=，<=，!=或< >）

[例3.57] 查询与“刘晨”在同一个主修专业的学生学号、姓名和主修专业

- 由于一个学生只可能在一个系学习，则可以用 = 代替IN：

```
SELECT Sno, Sname, Smajor          /*例3.57的解法三 */
FROM  Student
WHERE Smajor  =
            (SELECT Smajor
             FROM  Student
             WHERE Sname= '刘晨');
```

带有比较运算符的子查询（续）

[例3.59]找出每个学生超过他选修课程平均成绩的课程号。

```
SELECT Sno, Cno
FROM  SC x
WHERE Grade >= (SELECT AVG (Grade)
                FROM  SC y
                WHERE y.Sno = x.Sno);
```

- 子查询的查询条件依赖于父查询，为相关子查询。

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semeste	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
…	…	…	…	…

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semeste	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
…	…	…	…	…

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

带有ANY（SOME）或ALL谓词的子查询

使用ANY或ALL谓词时必须同时使用比较运算符

语义为：

> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值
<= ANY	小于等于子查询结果中的某个值
<= ALL	小于等于子查询结果中的所有值

带有ANY（SOME）或ALL谓词的子查询（续）

使用ANY或ALL谓词时必须同时使用比较运算符

语义为：

= ANY	等于子查询结果中的某个值
= ALL	等于子查询结果中的所有值（通常没有实际意义）
!=（或<>）ANY	不等于子查询结果中的某个值
!=（或<>）ALL	不等于子查询结果中的任何一个值

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.60] 查询非计算机科学技术专业中比计算机科学技术专业任意一个学生年龄小（出生日期晚）的学生的姓名、出生日期和主修专业

```
SELECT Sname, Sbirthdate, Smajor
FROM   Student
WHERE Smajor <> '计算机科学与技术'          /*父查询块中的条件 */
      AND Sbirthdate > ANY (SELECT Sbirthdate
                            FROM Student
                            WHERE Smajor= '计算机科学与技术');
```

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.60] 查询非计算机科学技术专业中比计算机科学技术专业任意一个学生年龄小（出生日期晚）的学生的姓名、出生日期和主修专业

- 用聚集函数实现

```
SELECT Sname, Sbirthdate, Smajor
FROM   Student
WHERE Smajor <>'计算机科学与技术'
          AND Sbirthdate > (SELECT MIN (Sbirthdate)
                            FROM Student
                            WHERE Smajor= '计算机科学与技术') ;
```

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.61]查询非计算机科学与技术专业中比计算机科学与技术专业所有学生年龄都小（出生日期晚）的学生的姓名及出生日期。

```
SELECT Sname,Sbirthdate
FROM Student
WHERE Sbirthdate > ALL (SELECT Sbirthdate
                        FROM Student
                        WHERE Smajor='计算机科学与技术')
AND Smajor <> '计算机科学与技术';
```

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.61]查询非计算机科学与技术专业中比计算机科学与技术专业所有学生年龄都小（出生日期晚）的学生的姓名及出生日期。

- 用聚集函数实现：

```
SELECT Sname,Sbirthdate
FROM Student
WHERE Sbirthdate >
        (SELECT MAX (Sbirthdate)   /*计算机科学与技术专业所有学生中最晚出生日期*/
         FROM Student
         WHERE Smajor='计算机科学与技术')
AND Smajor <>'计算机科学与技术';
```

带有ANY（SOME）或ALL谓词的子查询（续）

- ANY（或SOME），ALL谓词与聚集函数、IN谓词的等价转换关系

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>= MIN
ALL	--	NOT IN	<MIN	<= MIN	>MAX	>= MAX

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

带有EXISTS谓词的子查询

- **EXISTS谓词**
  - 存在量词 ∃
  - 带有**EXISTS**谓词的子查询不返回任何数据，只产生逻辑真值“**True**”或逻辑假值“**False**”。
    - 若内层查询结果非空，则外层的**WHERE**子句返回真值
    - 若内层查询结果为空，则外层的**WHERE**子句返回假值
  - 由**EXISTS**引出的子查询，其目标列表达式通常都用 *，因为带**EXISTS**的子查询只返回真值或假值，给出列名无实际意义。

带有EXISTS谓词的子查询（续）

- **NOT EXISTS谓词**
  - ■若内层查询结果非空，则外层的**WHERE**子句返回假值
  - ■若内层查询结果为空，则外层的**WHERE**子句返回真值

带有EXISTS谓词的子查询（续）

[例3.62]查询所有选修了81001号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE EXISTS
        (SELECT *
         FROM SC
         WHERE SC.Sno=Student.Sno AND Cno='81001');
```

带有EXISTS谓词的子查询（续）

[例3.63]查询没有选修81001号课程的学生姓名

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
        (SELECT *
         FROM SC
         WHERE Sno=Student.Sno AND Cno='81001');
```

上机练习

- 基于教材第73页二维码中的示例数据，构建三个基本表，练习并思考：
- 连接查询：例3.51-例3.56
- 嵌套查询：例3.57-例3.63

- 注意保存构建的表便于后续练习。

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

3.3.4 集合查询

- 集合操作的种类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT
- 集合操作
  - 各查询结果的列数必须相同;
  - 对应项的数据类型也必须相同。

集合查询（续）

[例3.66]查询计算机科学与技术专业的学生及年龄不大于19岁（包括等于19岁）的学生

```
SELECT * FROM Student WHERE Smajor='计算机科学与技术'
UNION
SELECT * FROM Student
WHERE (extract(year from current_date) - extract(year from Sbirthdate)) <=19;
```

- UNION：将多个查询结果合并起来时，系统自动去掉重复元组
- UNION ALL：将多个查询结果合并起来时，保留重复元组

集合查询（续）

[例3.67]查询2020年第2学期选修了课程81001或者选修了课程81002的学生

```
SELECT Sno
FROM SC
WHERE Semester='20202' AND Cno='81001’
UNION
SELECT Sno
FROM SC
WHERE Semester='20202' AND Cno='81002';
```

集合查询（续）

[例3.68] 查询计算机科学与技术专业的学生与年龄不大于19岁的学生的交集

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术’
INTERSECT
SELECT *
FROM Student
WHERE(extract(year from current_date)-extract(year from Sbirthdate) ) <=19;
```

集合查询（续）

[例3.68] 查询计算机科学与技术专业中年龄不大于**19**岁的学生

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术' AND
        (extract(year from current_date) - extract(year from Sbirthdate) )<=19;
```

集合查询（续）

[例3.69]查询既选修了课程81001又选修了课程81002的学生。

```
SELECT Sno
FROM SC
WHERE Cno='81001’
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='81002';
```

集合查询（续）

[例3.69]查询既选修了课程81001又选修了课程81002的学生。

也可用嵌套查询的方式：

```
SELECT Sno
FROM SC
WHERE Cno='81001' AND Sno IN
                    (SELECT Sno
                    FROM SC
                    WHERE Cno='81002');
```

集合查询（续）

[例3.70]查询计算机科学与技术专业的学生与年龄不大于19岁的学生的差集。

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术’
EXCEPT
SELECT *
FROM Student
WHERE(extract(year from current_date) - extract(year from Sbirthdate) )<=19;
```

集合查询（续）

[例3.70] 查询计算机科学与技术专业中年龄大于**19**岁的学生。

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术' AND
      (extract(year from current_date) - extract(year from Sbirthdate) )>19；
```

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

回顾

[例3.59]找出每个学生超过他选修课程平均成绩的课程号。

```
SELECT Sno, Cno
FROM  SC x
WHERE Grade >= (SELECT AVG (Grade)
                FROM  SC y
                WHERE y.Sno = x.Sno);
```

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semeste	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
…	…	…	…	…

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semeste	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
…	…	…	…	…

3.3.5 基于派生表的查询

- 子查询不仅可以出现在**WHERE**子句中，还可以出现在**FROM**子句中，子查询生成的临时派生表（**Derived Table**）成为主查询的查询对象。

[例3.59]找出每个学生超过他自己选修课程平均成绩的课程号

```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade)
          FROM SC
          GROUP BY Sno)
          AS Avg_SC (Avg_sno, Avg_grade)
WHERE SC.Sno = Avg_SC.Avg_sno AND SC.Grade >= Avg_SC.Avg_grade;
```

基于派生表的查询（续）

❖ 如果子查询中没有聚集函数，派生表可以不指定属性列，子查询SELECT子句后面的列名为其缺省属性。

[例3.62]查询所有选修了81001号课程的学生姓名

```
SELECT Sname
FROM Student, (SELECT Sno
               FROM SC
               WHERE Cno=‘ 81001 ') AS SC1
WHERE  Student.Sno=SC1.Sno;
```

- FROM子句生成派生表时，AS关键字可省略，必须为派生关系指定一个别名（不可省略）。
- 派生表是一个中间结果表，查询完成后派生表将被系统自动清除。

上机练习

- 基于教材第73页二维码中的示例数据，构建三个基本表，练习并思考：
- 集合查询：例3.66-例3.70
- 基于派生表的查询：例3.59；例3.62

- 注意保存构建的表便于后续练习。

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

第3章 关系数据库标准语言SQL

3.1 SQL概述

3.2 数据定义

3.3 数据查询

3.4 数据更新

3.5 空值的处理

3.6 视图

3.4 数据更新

3.4.1 插入数据

3.4.2 修改数据

3.4.3 删除数据

插入数据

1.插入一个元组

2.插入子查询结果（插入多个元组）

1. 插入一个元组

- 语句格式

  INSERT INTO <表名> [(<属性列1>[,<属性列2 >] …)]

  VALUES (<常量1> [,<常量2>]…);

- 功能
  - 将一个新元组插入指定表中

插入一个元组（续）

- **INTO子句**
  - 没有指明任何属性列名，新插入的元组必须在每个属性列上均有指定值
  - 没有出现的属性列，新元组在这些列上将取空值
  - 定义表时指定了相应属性列的缺省值，新元组在这些列上将取缺省值
  - 在表定义时说明了**NOT NULL**的属性列不能取空值
- **VALUES子句**
  - 提供的值必须与**INTO**子句匹配
    - 值的个数
    - 值的类型

插入一个元组（续）

[例3.71] 将一个新学生元组（学号：**20180009**，姓名：陈冬，性别：男，出生日期：**2000-5-22**，主修专业：信息管理与信息系统）插入到**Student**表中。

```
INSERT INTO Student (Sno,Sname,Ssex,Smajor,Sbirthdate)
VALUES ('20180009', '陈冬', '男','信息管理与信息系统', '2000-5-22');
```

- **INTO**子句中指出表名**Student**，并指出新增加的元组在哪些属性上要赋值，属性的顺序可以与**CREATE TABLE**中的顺序不一样
- **VALUES**子句按照**INTO**子句指定的属性次序对新元组的各属性赋值
- 字符串常数要用单引号（英文符号）括起来

插入一个元组（续）

[例3.72] 将学生张成民的信息插入到**Student**表中

```
INSERT INTO Student
VALUES ('20180008', '张成民', '男','2000-4-15','计算机科学与技术');
```

- **INTO**子句中只指出了表名，没有指出属性名，表示**VALUES**子句要在表的所有属性列上都指定值。
- 属性列的次序与**CREATE TABLE**中的次序相同。

插入一个元组（续）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

[例3.73] 插入一条选课记录

```
INSERT INTO SC(Sno,Cno,Semester,Teachingclass)
VALUES ('20180005','81004','20202','81004-01');
```

- 数据库管理系统将在新插入记录的Grade列上自动地赋空值

或者：

```
INSERT INTO SC
VALUES ('20180005','81004',NULL,'20202','81004-01');
```

- INTO子句没有指出SC的属性名，所以在VALUES子句中对应的Grade列上要明确给出空值

2. 插入子查询结果

❖ 语句格式

INSERT INTO <表名> [(<属性列1> [,<属性列2>…)]

子查询;

插入子查询结果（续）

[例3.74] 对每一个专业，求学生平均年龄，并把结果存入数据库。

首先在数据库中建立一个新表，其中一列存放专业名，另一列存放学生的平均年龄。

```
CREATE TABLE Smajor_age
        (Smajor VARCHAR(20),
         Avg_age SMALLINT);
```

对Student表按专业分组求平均年龄，再把专业名和平均年龄存入新表中。

```
INSERT INTO Smajor_age(Smajor, Avg_age)
SELECT Smajor, AVG(extract(year from current_date) - extract(year from Sbirthdate))
FROM Student
GROUP BY Smajor;
```

3.4 数据更新

3.4.1 插入数据

3.4.2 修改数据

3.4.3 删除数据

3.4.2 修改数据

- 修改操作又称为更新操作，其语句的一般语法为

  UPDATE <表名>

  SET <列名>=<表达式> [,<列名>=<表达式>] …

  [WHERE <条件>] ;

■功能：修改指定表中满足**WHERE**子句条件的元组

■**SET**子句给出**<表达式>**的值用于取代相应的属性列值

■如果省略**WHERE**子句，则表示要修改表中的所有元组

插入数据（续）

1.修改某一个元组的值

2.修改多个元组的值

3.带子查询的修改语句

1. 修改某一个元组的值

[例3.75] 将学生20180001的出生日期改为2001年3月18日

```
UPDATE Student
SET Sbirthdate = ‘2001-3-18'
WHERE Sno = '20180001';
```

2. 修改多个元组的值

[例3.76]将2020年第1学期选修81002课程所有学生的成绩减少5分

```
UPDATE SC
SET Grade = Grade-5
WHERE Semester='20201' AND Cno='81002'
```

3. 带子查询的修改语句

- 子查询也可以嵌套在**UPDATE**语句中，用以构造修改的条件。

[例3.77] 将计算机科学与技术专业学生成绩置零

```
UPDATE SC
SET Grade = 0
WHERE Sno IN
        (SELECT Sno
         FROM Student
         WHERE Smajor = '计算机科学与技术');
```

3.4 数据更新

3.4.1 插入数据

3.4.2 修改数据

3.4.3 删除数据

3.4.3 删除数据

❖ 语句格式

DELETE FROM <表名>

[WHERE <条件>];

❖ 功能

■删除指定表中满足**WHERE**子句条件的元组

❖ **WHERE**子句

■指定要删除的元组

■缺省表示要删除表中的全部元组，表的定义仍在字典中

删除数据

1.删除某一个元组的值

2.删除多个元组的值

3.带子查询的删除语句

1. 删除某一个元组的值

[例3.78] 删除学号为20180007的学生记录。

```
DELETE FROM Student
WHERE Sno='20180007';
```

2. 删除多个元组的值

[例3.79] 删除所有的学生选课记录。

```
DELETE FROM SC;
```

- 这条DELETE语句将使SC成为空表，它删除了SC的所有元组。

3. 带子查询的删除语句

[例3.80] 删除计算机科学与技术专业所有学生的选课记录

```
DELETE FROM SC
WHERE Sno IN
        (SELECT Sno
         FROM Student
         WHERE Smajor= '计算机科学与技术' );
```

第3章 关系数据库标准语言SQL

3.5 空值的处理

- 空值就是“不知道”、“不存在”或“无意义”的值。
- 一般有以下几种情况：
  - ■①该属性应该有一个值，但目前不知道它的具体值
  - ■②该属性不应该有值
  - ■③由于某种原因不便于填写
- 空值是一个很特殊的值，含有不确定性。对关系运算带来特殊的问题，需要做特殊的处理。

1. 空值的产生

[例3.81] 向SC表中插入一个元组，学生号是“20180006”，课程号是“81004”，选课学期2021年第1学期，选课班没有确定，成绩还没有。

```
INSERT INTO SC(Sno,Cno,Grade,Semester,Teachingclass)
        VALUES('20180006', '81004',NULL, '20211',NULL);
            /*在插入时该学生还没有选定教学班，没有考试成绩，都要取空值*/
```

或

```
INSERT INTO SC(Sno,Cno,Semester)
        VALUES('20180006', '81004','20211');
         /*在插入语句的INTO字句中没有指定的属性，系统自动置空值*/
```

空值的产生（续）

[例3.82] 将Student表中学生号为“20180006”的学生主修专业改为空值。

```
UPDATE Student
SET Smajor = NULL
WHERE Sno='20180006’;
```

■外连接也会产生空值，参考[例3.55]

■ 空值的关系运算也会产生空值

2. 空值的判断

❖ 判断一个属性的值是否为空值，用IS NULL或IS NOT NULL来表示

[例3.83] 从Student表中找出漏填了数据的学生信息。

```
SELECT *
FROM Student
WHERE Sname IS NULL OR Ssex IS NULL OR Sbirthdate IS NULL OR Smajor IS NULL;
```

- 不包含Sno，因为其是主码，不允许取空值。

3. 空值约束

❖在创建基本表时，如果属性定义（或者域定义）为NOT NULL约束，则该属性不能取空值。

❖主码的属性不能取空值

- SC表的主码是（Sno,Cno），Sno和Cno（主属性）都不能取空值。
- Student表的主码是Sno，不能取空值。

4. 空值的算术运算、比较运算和逻辑运算

- 空值与另一个值（包括另一个空值）的算术运算的结果为空值。
- 空值与另一个值（包括另一个空值）的比较运算的结果为**UNKNOWN**。
- 有**UNKNOWN**后，传统的逻辑运算中二值（**TRUE**，**FALSE**）逻辑就扩展成了三值（**TRUE**，**FALSE**，**UNKNOWN**）逻辑。

空值的算术运算、比较运算和逻辑运算(续)

x y	x AND y	x OR y	NOT x
T T	T	T	F
T U	U	T	F
T F	F	T	F
U T	U	T	U
U U	U	U	U
U F	F	U	U
F T	F	T	T
F U	F	U	T
F F	F	F	T

T表示TRUE，U表示UNKNOWN（二者之间），F表示FALSE

空值的算术运算、比较运算和逻辑运算（续）

[例3.84] 找出选修**81001**号课程且成绩不及格的学生。

```
SELECT Sno
FROM SC
WHERE Grade < 60 AND Cno='81001’;
```

■ 选出的学生是那些参加了考试而成绩不及格（**Grade**属性为非空值）的学生，不包括缺考（**Grade**属性为空值）的学生。

空值的算术运算、比较运算和逻辑运算（续）

[例3.85] 选出选修81001号课程且成绩不及格的学生以及缺考的学生。

```
SELECT Sno
FROM SC
WHERE Grade < 60 AND Cno='81001'
UNION
SELECT Sno
FROM SC
WHERE Grade IS NULL AND Cno='81001';
```

或

```
SELECT Sno
FROM SC
WHERE Cno='81001' AND (Grade < 60 OR Grade IS NULL);
```

上机练习

- 基于教材第73页二维码中的示例数据，构建三个基本表，练习并思考：
- 数据更新：例3.71-例3.80
- 空值的处理：例3.81-例3.85

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ceredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

第3章 关系数据库标准语言SQL

3.1 SQL概述

3.2 数据定义

3.3 数据查询

3.4 数据更新

3.5 空值的处理

3.6 视图

3.6 视图

❖ 视图的特点

■虚表，是从一个或几个基本表（或视图）导出的表

■只存放视图的定义，而不存放视图对应的数据

■基本表中的数据发生变化，从视图中查询出的数据也随之改变

3.6 视图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

1. 建立视图

❖语句格式

CREATE VIEW <视图名> [(<列名> [,<列名>]…)]

AS <子查询>

[WITH CHECK OPTION];

建立视图（续）

[例3.86] 建立信息管理与信息系统专业学生的视图

```
CREATE VIEW IS_Student
AS
SELECT Sno, Sname, Ssex, Sbirthdate, Smajor
FROM Student
WHERE Smajor='信息管理与信息系统';
```

- 省略视图IS_Student的列名，表示隐含由子查询中SELECT子句中的五个列名组成。
- 关系数据库管理系统执行CREATE VIEW语句的结果只是把视图的定义存入数据字典，并不执行其中的SELECT语句。只有在对视图查询时，才按视图的定义从基本表中将数据查出。

建立视图（续）

❖ 组成视图的属性列名或者全部省略或者全部指定

■全部省略:

- 由子查询中SELECT子句目标列中的诸字段组成

■下列情况必须明确指定组成视图的所有列名:

- ①某个目标列不是单纯的属性名，而是聚集函数或列表达式
- ②多表连接时选出了几个同名列作为视图的字段
- ③需要在视图中为某个列启用新的更合适的名字

建立视图（续）

- 子查询可以是任意的SELECT语句，是否可以含有ORDER BY子句和DISTINCT短语，则取决于具体系统的实现。

- WITH CHECK OPTION表示对视图进行UPDATE，INSERT和DELETE操作时要保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）。

建立视图（续）

[例3.87] 建立信息管理与信息系统专业学生的视图，并要求进行插入、修改和删除操作时仍需保证该视图只有信息管理与信息系统专业的学生。

```
CREATE VIEW IS_Student
AS
SELECT Sno,Sname,Ssex,Sbirthdate, Smajor
FROM Student
WHERE Smajor='信息管理与信息系统'
WITH CHECK OPTION;
```

建立视图（续）

- 定义IS_Student视图时加了WITH CHECK OPTION子句，对该视图进行插入、修改和删除操作时，关系数据库管理系统会自动检查Smajor='信息管理与信息系统'的条件。
  - INSERT INTO IS_Student values ('20180010','贾明','男','2001-11-1','信息管理与信息系统’);
    插入成功
  - INSERT INTO IS_Student values (‘20180011’,‘王伟’,‘男’,‘2003-11-1’,‘计算机科学与技术’);
    插入失败，不符合WITH CHECK OPTION条件

- 若一个视图是从单个基本表导出，并且只是去掉了基本表的某些行和某些列，但保留了主码，则称这类视图为行列子集视图。
  - IS_Student视图就是一个行列子集视图。

建立视图（续）

❖ 视图也可以建立在多个基本表上

[例3.88] 建立信息管理与信息系统专业选修了81001号课程的学生的视图（包括学号、姓名、成绩属性）。

```
CREATE VIEW IS_C1(Sno, Sname, Grade)
AS
SELECT Student.Sno, Sname, Grade
FROM Student, SC
WHERE Smajor='信息管理与信息系统' AND Student.Sno=SC.Sno AND SC.Cno='81001';
```

建立视图（续）

❖ 视图也可以建立在一个或多个已定义好的视图上，或建立在基本表与视图上。

[例3.89] 建立信息管理与信息系统专业选修了81001号课程且成绩在90分以上的学生的视图（包括学号、姓名、成绩属性）。

```
CREATE VIEW IS_C2
AS
SELECT Sno, Sname, Grade
FROM  IS_C1
WHERE Grade>=90;
```

建立视图（续）

❖ 带表达式的视图

[例3.90]将学生的学号、姓名、年龄定义为一个视图。

```
CREATE VIEW S_AGE(Sno, Sname, Sage)
AS
SELECT Sno, Sname, (extract(year from current_date)-extract(year from Sbirthdate) )
FROM Student;
```

建立视图（续）

❖ 分组视图

[例3.91] 将学生的学号及平均成绩定义为一个视图。

```
CREATE VIEW S_GradeAVG(Sno, Gavg)
AS
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;
```

建立视图（续）

- 存在问题的设计

[例3.92] 将**Student**表中所有女生记录定义为一个视图。

```
CREATE VIEW F_Student(Fsno, Fname, Fsex, Fbirthdate, Fmajor)
AS
SELECT *
FROM Student
WHERE Ssex='女';
```

- 如果以后修改基表**Student**的结构后，**Student**表与**F_Student**视图的映象关系被破坏，导致该视图不能正确工作。
- 因此在修改基本表后，可以删除由该基本表导出的视图，并重建视图。

2. 删除视图

❖ 语句的格式：

DROP VIEW <视图名>[CASCADE];

■视图定义从数据字典中删除。

■如果该视图上还导出了其他视图，使用CASCADE级联删除语句，把该视图和由它导出的所有视图一起删除。

■删除基本表后，由该基本表导出的所有视图均无法使用，但并未从字典中清除。删除这些视图定义需要显式地使用DROP VIEW语句删除。

删除视图（续）

[例3.93] 删除视图S_AGE和视图IS_C1：

```
DROP VIEW S_AGE;          /*成功执行*/
DROP VIEW IS_C1;          /*报告错误*/
```

- 执行DROP VIEW IS_C1语句时，由于IS_C1视图上还导出了IS_C2视图，所以该语句执行时会报告错误：视图 IS_C2依赖于视图IS_C1。
- 如果导出视图也确定可以删除，则使用级联删除语句：

  DROP VIEW IS_C1 CASCADE;

  - 执行此语句不仅删除了IS_C1视图，还级联删除了由它导出的IS_C2视图

3.6 视图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

3.6.2 查询视图

❖ 用户角度：查询视图与查询基本表相同

❖ 关系数据库管理系统实现视图查询的方法

■视图消解法（View Resolution）

- 进行有效性检查
- 转换成等价的对基本表的查询
- 执行修正后的查询

查询视图（续）

[例3.94]在信息管理与信息系统专业学生的视图中，找出年龄小于等于20岁的学生（包括学生的学号和出生日期）。

```
SELECT Sno, Sbirthdate
FROM IS_Student
WHERE (extract(year from current_date)-extract(year from Sbirthdate) )<=20;
```

查询视图（续）

- 关系数据库管理系统实现视图查询的方法：视图消解：

1.先找到视图IS_Student的定义（找到定义视图来源基本表上的查询条件）

```
CREATE VIEW IS_Student
AS
SELECT Sno,Sname,Ssex,Sbirthdate
FROM Student
WHERE Smajor='信息管理与信息系统'
WITH CHECK OPTION;
```

查询视图（续）

2.进行视图消解，转换后的查询语句为：

```
SELECT Sno, Sbirthdate
FROM Student
WHERE Smajor='信息管理与信息系统' AND
(extract(year from current_date)-extract(year from Sbirthdate) )<=20;
```

查询视图（续）

[例3.95]查询选修了81001号课程的信息管理与信息系统专业学生。

```
SELECT IS_Student.Sno, Sname
FROM IS_Student, SC
WHERE IS_Student.Sno=SC.Sno AND SC.Cno='81001’;
```

关系数据库管理系统先从数据字典中取出视图IS_Student的定义，然后进行视图消解，把上面查询转换为：

```
SELECT Student.Sno, Sname
FROM  Student, SC
WHERE Student.Sno=SC.Sno AND SC.Cno='81001' AND Smajor='信息管理与信息系统';
```

查询视图（续）

❖ 视图消解法的局限：有些情况下，视图查询的转换不能直接进行，查询会出问题

[例3.96]在S_GradeAVG视图（例3.91中定义的视图）中查询平均成绩在90分以上的学生学号和平均成绩。

```
SELECT *
FROM  S_GradeAVG     /*视图S_GradeAVG*/
WHERE Gavg>=90;
```

❖ 视图消解

❖ 1. 视图S_GradeAVG的定义为：

```
CREATE VIEW S_GradeAVG(Sno,Gavg)
AS
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;
```

❖ 2.直观的视图消解过程

```
SELECT Sno, AVG(Grade)
FROM  SC
WHERE  AVG(Grade)>=90
GROUP BY Sno;
```

查询视图（续）

错误（WHERE子句不能用聚集函数作为条件表达式）：

```
SELECT Sno, AVG(Grade)
FROM    SC
WHERE  AVG(Grade)>=90
GROUP BY Sno;
```

正确：

```
SELECT  Sno, AVG(Grade)
FROM  SC
GROUP BY Sno
HAVING AVG(Grade)>=90;
```

- 目前多数关系数据库管理系统对行列子集视图的查询均能正确转换。但对非行列子集视图的查询不一定能正确转换，因此这类查询应该直接对基本表进行。

查询视图（续）

例3.96也可以用如下SQL语句完成：

```
SELECT *
FROM (SELECT Sno, AVG(Grade)       /*子查询生成一个派生表S_GradeAVG*/
        FROM SC
        GROUP BY Sno ) AS S_GradeAVG(Sno, Gavg)
WHERE Gavg>=90;
```

■ 对视图查询与基于派生表查询的区别

- 视图一旦定义，将保存在数据字典，之后的所有查询都可以直接引用该视图
- 派生表只是在语句执行时临时定义，语句执行后该派生表定义即被删除

3.6 视图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

3.6.3 更新视图

- 更新视图：通过视图来插入（INSERT）、删除（DELETE）和修改（UPDATE）数据。
- 视图的更新操作通过视图消解，转换为对基本表的更新操作。
- 为防止用户有意无意地对不属于视图范围内的基本表数据进行操作，可在定义视图时加上**WITH CHECK OPTION**子句：在视图上增、删、改数据时，关系数据库管理系统会检查视图定义中的条件，若不满足条件则拒绝执行该操作

更新视图（续）

[例3.97]将信息管理与信息系统专业学生视图IS_Student中学号为“20180005”的学生姓名改为“刘新奇”。

```
UPDATE IS_Student
SET Sname=‘刘新奇'
WHERE Sno='20180005’;
```

视图消解后，对视图IS_Student的更新语句就转换为对基本表Student的更新：

```
UPDATE Student
SET Sname=‘刘新奇'
WHERE Sno='20180005' AND Smajor='信息管理与信息系统';
```

更新视图（续）

[例3.98]向信息管理与信息系统专业学生视图IS_Student中插入一个新的学生记录

(20180207，赵新，男，2001-7-19)

```
INSERT INTO IS_Student
VALUES('20180207','赵新','男','2001-7-19','信息管理与信息系统');
```

视图消解转换为对基本表的更新：

```
INSERT INTO Student(Sno,Sname,Ssex,Sbirthdate,Smajor)
VALUES('20180207','赵新','男','2001-7-19,'信息管理与信息系统' );
```

更新视图（续）

[例3.99]删除信息管理与信息系统专业学生视图IS_Student中学号为“20180207”的记录。

```
DELETE FROM IS_Student
WHERE Sno='20180207’;
```

视图消解转换为对基本表的更新：

```
DELETE FROM Student
WHERE Sno='20180207' AND Smajor='信息管理与信息系统';
```

更新视图（续）

❖ 并不是所有的视图都是可更新的

例3.91定义的视图S_GradeAVG是由学号和平均成绩两个属性列组成的，平均成绩是由Student表中对元组分组后计算平均值得来。

```
CREATE VIEW S_GradeAVG(Sno,Gavg)
AS
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;
```

更新视图（续）

如果想把视图S_GradeAVG中学号为“20180001”学生的平均成绩改成90分

```
UPDATE S_GradeAVG
SET Gavg=90
WHERE Sno='20180001’;
```

- 对视图的更新是无法转换成对基本表SC的更新，因为系统无法修改各科成绩，以使20180001学生平均成绩成为90
- 所以S_GradeAVG视图是不可更新的

更新视图（续）

- 一般地，行列子集视图是可更新的
- 除行列子集视图外，有些视图理论上是可更新的，还有些视图从理论上就是不可更新的
- 不可更新的视图与不允许更新的视图不同
  - 前者指理论上已证明是不可更新视图
  - 后者指实际系统中不支持其更新，但它本身有可能是可更新的视图

3.6 视图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

3.6.4 视图的作用

❖ 1.视图能够对机密数据提供安全保护

■对不同的用户定义不同的视图，使机密数据不出现在不应看到这些数据的用户视图上，自动提供了对机密数据的安全保护功能

■例如，Student表涉及全校30个院系的学生数据

- ●可以在其上定义30个视图
- ●每个视图只包含一个院系的学生数据
- ●只允许每个院系的主任查询和修改本院系的学生视图

视图的作用（续）

❖ **2.视图对重构数据库提供了一定程度的逻辑独立性**

■数据的逻辑独立性是指当数据库重构造时，如增加新的关系或对原有关系增加新的字段等，用户的应用程序不会受影响

例：将学生关系**Student(Sno,Sname,Ssex,Sbirthdate,Smajor)** 重构:

“垂直”地分成两个基本表：

SX(Sno,Sname,Sbirthdate)

SY(Sno,Ssex,Smajor)

视图的作用（续）

- 通过建立一个视图**Student**：

```
CREATE VIEW  Student(Sno,Sname,Ssex,Sbirthdate,Smajor)
AS
SELECT  SX.Sno,SX.Sname,SY.Ssex,SX.Sbirthdate,SY.Smajor
FROM  SX,SY
WHERE  SX.Sno=SY.Sno;
```

■通过建立视图，用户的应用程序通过视图仍然能够查找数据（不受影响）

■视图只能在一定程度上提供数据的逻辑独立性：对视图的更新是有条件的，因此应用程序中修改数据的语句可能仍会因基本表结构的改变而相应修改

视图的作用（续）

- **3.视图能够简化用户的操作**
- 适当的利用视图可以更清晰的表达查询
  - 经常查询“对每个同学找出他获得最高成绩的课程号”
  - 可以先定义一个视图，求出每个同学获得的最高成绩

```
CREATE VIEW VMGrade
AS
SELECT Sno, MAX(Grade)  Mgrade
FROM  SC
GROUP BY Sno;
```

  - 然后用如下的查询语句完成查询：

```
SELECT SC.Sno,Cno
FROM SC,VMGrade
WHERE SC.Sno=VMGrade.Sno AND
      SC.Grade=VMGrade .Mgrade;
```

视图的作用（续）

❖4.视图使用户能以多种角度看待同一数据

- ■视图机制能使不同用户以不同方式看待同一数据，适应数据库共享的需要
- ■例如：希望了解学生的平均成绩，学生的最高成绩和最低成绩，都可以在基本表SC上定义自己感兴趣的视图，直接对这些视图查询

上机练习

- 基于教材第73页二维码中的示例数据，构建三个基本表，练习并思考：
- 例3.86-例3.99

学生选课数据库 3 个关系的示例数据

（第 2 章图 2.1）

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180205	陈新奇	男	2001-11-1	信息管理与信息系统
20180306	赵明	男	2000-6-12	数据科学与大数据技术
20180307	王佳佳	女	2001-12-7	数据科学与大数据技术

（a）

Course

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与 C 语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python 语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

（b）

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180205	81003	68	20202	81003-01

（c）

作业1

请根据给出的关系数据库模式，按要求完成作答。

现有学生选课数据库

学生表 Student(Sno,Sname,Ssex,Sbirth,Sdeptno)

学院表 Dept(Deptno,Deptname)

课程表 Course(Cno,Cname,Teachername)

选课表 SC(Sno,Cno,Grade)

试完成下列要求：

1、查询至少选修“人工智能”和“算法设计”两门课程的学生学号及姓名。（关系代数）

2、查询有 50 人以上成绩超过 90 分的课程课号及课名。（SQL 语言）

3、查询“计算机科学与技术”学院选修了“数据库”课程且成绩 90 分以上的男生的学号和姓名。请给出对应的关系代数表达式。

4、对第 3 题中关系代数表达式进行代数优化，给出语法树。

5、创建反映“计算机科学与技术”学院选修了两门及以上张丽老师所授课程的学生的学号及这些课程的平均成绩的视图 CZ-AvgG(Sno, G)。（SQL 语言）

6、把查询Student表和修改学生学号的权限授予所有用户。（SQL语言）

把对表SC的全部操作权限授予用户User，并允许将此权限再授予其他用户。（SQL语言）

参考答案

要求1

$$\prod_{Sno,Sname}(Student \infty (\prod_{Sno,\ Cno} SC \div \prod_{Cno}（\sigma_{Cname='人工智能' \vee Cname='算法设计'}(Course))))$$

学生表 Student(Sno,Sname,Ssex,Sbirth,Sdeptno)

学院表 Dept(Deptno,Deptname)

课程表 Course(Cno,Cname,Teachername)

选课表 SC(Sno,Cno,Grade)

试完成下列要求：

1、查询至少选修“人工智能”和“算法设计”两门课程的学生学号及姓名。（关系代数）

2、查询有 50 人以上成绩超过 90 分的课程课号及课名。（SQL 语言）

3、查询“计算机科学与技术”学院选修了“数据库”课程且成绩 90 分以上的男生的学号和姓名。请给出对应的关系代数表达式。

4、对第 3 题中关系代数表达式进行代数优化，给出语法树。

5、创建反映“计算机科学与技术”学院选修了两门及以上张丽老师所授课程的学生的学号及这些课程的平均成绩的视图 CZ-AvgG(Sno, G)。（SQL 语言）

6、把查询Student表和修改学生学号的权限授予所有用户。（SQL语言）

把对表SC的全部操作权限授予用户User，并允许将此权限再授予其他用户。（SQL语言）

参考答案

要求2

```
SELECT Course.Cno, Cname
FROM Course, SC
WHERE Course.Cno=SC.Cno AND
Grade>=90
GROUP BY Course.Cno HAVING
COUNT(*)>=50
```

或采用嵌套查询方式。

学生表 Student(Sno,Sname,Ssex,Sbirth,Sdeptno)

学院表 Dept(Deptno,Deptname)

课程表 Course(Cno,Cname,Teachername)

选课表 SC(Sno,Cno,Grade)

试完成下列要求：

1、查询至少选修“人工智能”和“算法设计”两门课程的学生学号及姓名。（关系代数）

2、查询有 50 人以上成绩超过 90 分的课程课号及课名。（SQL 语言）

3、查询“计算机科学与技术”学院选修了“数据库”课程且成绩 90 分以上的男生的学号和姓名。请给出对应的关系代数表达式。

4、对第 3 题中关系代数表达式进行代数优化，给出语法树。

5、创建反映“计算机科学与技术”学院选修了两门及以上张丽老师所授课程的学生的学号及这些课程的平均成绩的视图 CZ-AvgG(Sno, G)。（SQL 语言）

6、把查询Student表和修改学生学号的权限授予所有用户。（SQL语言）
把对表SC的全部操作权限授予用户User，并允许将此权限再授予其他用户。（SQL语言）

参考答案

学生表 Student(Sno,Sname,Ssex,Sbirth,Sdeptno)

学院表 Dept(Deptno,Deptname)

课程表 Course(Cno,Cname,Teachername)

选课表 SC(Sno,Cno,Grade)

试完成下列要求：

1、查询至少选修“人工智能”和“算法设计”两门课程的学生学号及姓名。（关系代数）

2、查询有 50 人以上成绩超过 90 分的课程课号及课名。（SQL 语言）

3、查询“计算机科学与技术”学院选修了“数据库”课程且成绩 90 分以上的男生的学号和姓名。请给出对应的关系代数表达式。

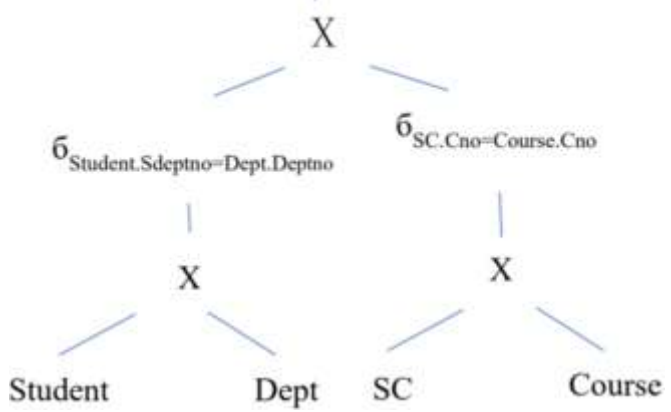
4、对第 3 题中关系代数表达式进行代数优化，给出语法树。

5、创建反映“计算机科学与技术”学院选修了两门及以上张丽老师所授课程的学生的学号及这些课程的平均成绩的视图 CZ-AvgG(Sno, G)。（SQL 语言）

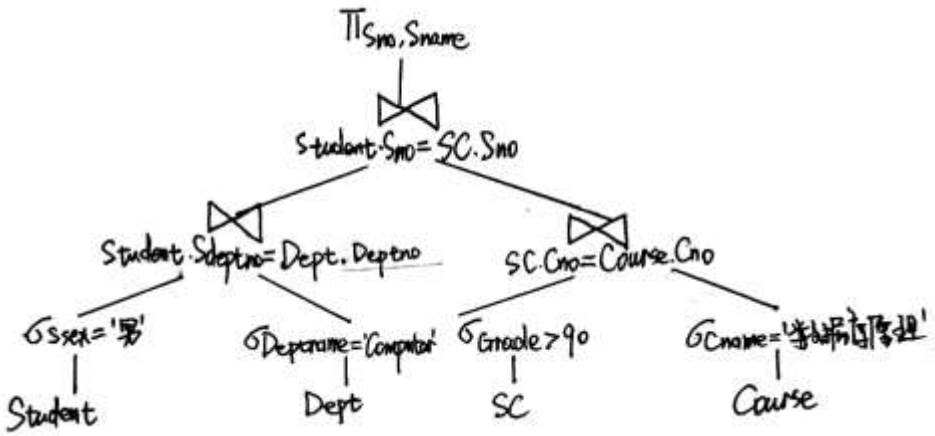
6、把查询Student表和修改学生学号的权限授予所有用户。（SQL语言）
把对表SC的全部操作权限授予用户User，并允许将此权限再授予其他用户。（SQL语言）

要求3

$$\Pi_{Sno,\ Sname}\left(\sigma_{Deptname='Computer'\wedge Cname='数据库'\wedge Ssex='男'\wedge Grade>90}(Course\infty SC\infty Student\infty Dept)\right)$$



要求4



参考答案

要求5

```
create view Cm-AG( Sno, avgG)
    as
       select Sno, AVG(grade)
        from Dept, Course, SC, Student
        where Deptname='计算机科学与技术'
             and Teachername='张丽'
             and Dept.Deptno=Student.Sdeptno
    and Course.Cno=SC.Cno
    and Student.Sno=SC.Sno
     group by Sno
             having count(*)>=2;
```

学生表 Student(Sno,Sname,Ssex,Sbirth,Sdeptno)

学院表 Dept(Deptno,Deptname)

课程表 Course(Cno,Cname,Teachername)

选课表 SC(Sno,Cno,Grade)

试完成下列要求：

1、查询至少选修“人工智能”和“算法设计”两门课程的学生学号及姓名。（关系代数）

2、查询有 50 人以上成绩超过 90 分的课程课号及课名。（SQL 语言）

3、查询“计算机科学与技术”学院选修了“数据库”课程且成绩 90 分以上的男生的学号和姓名。请给出对应的关系代数表达式。

4、对第 3 题中关系代数表达式进行代数优化，给出语法树。

5、创建反映“计算机科学与技术”学院选修了两门及以上张丽老师所授课程的学生的学号及这些课程的平均成绩的视图 CZ-AvgG(Sno, G)。（SQL 语言）

6、把查询Student表和修改学生学号的权限授予所有用户。（SQL语言）

把对表SC的全部操作权限授予用户User，并允许将此权限再授予其他用户。（SQL语言）

参考答案

要求6

```
GRANT UPDATE (Sno), SELECT
ON TABLE Student
TO PUBLIC;

GRANT ALL PRIVILEGES
ON TABLE SC
TO User
WITH GRANT OPTION;
```

学生表 Student(Sno,Sname,Ssex,Sbirth,Sdeptno)

学院表 Dept(Deptno,Deptname)

课程表 Course(Cno,Cname,Teachername)

选课表 SC(Sno,Cno,Grade)

试完成下列要求：

1、查询至少选修“人工智能”和“算法设计”两门课程的学生学号及姓名。（关系代数）

2、查询有 50 人以上成绩超过 90 分的课程课号及课名。（SQL 语言）

3、查询“计算机科学与技术”学院选修了“数据库”课程且成绩 90 分以上的男生的学号和姓名。请给出对应的关系代数表达式。

4、对第 3 题中关系代数表达式进行代数优化，给出语法树。

5、创建反映“计算机科学与技术”学院选修了两门及以上张丽老师所授课程的学生的学号及这些课程的平均成绩的视图 CZ-AvgG(Sno, G)。（SQL 语言）

6、把查询Student表和修改学生学号的权限授予所有用户。（SQL语言）

把对表SC的全部操作权限授予用户User，并允许将此权限再授予其他用户。（SQL语言）

感谢各位同学聆听！