第3章 关系数据库标准语言SQL

哈尔滨工业大学（威海）

本章内容提要

- ◆3.1 SQL概述
- ◆3.2 学生-课程数据库
- ◆3.3 数据定义
- ◆3.4 数据查询
- ◆3.5 数据更新
- ◆3.6 视图

本章内容提要

- ***3.1 SQL概述***
- **3.2 学生-课程数据库**
- **3.3 数据定义**
- **3.4 数据查询**
- **3.5 数据更新**
- **3.6 视图**

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。

◆SQL是一个通用的、功能极强的关系数据库语言，被国际上绝大多数商品化的关系数据库采用。

3.1 SQL概述

- 关系数据库语言ISBL、QUEL、 QBE
- 1974年，由Boyce和Chamber提出
- 1975-1979年，在System R上实现，由IBM的San Jose研究室研制，称为Sequel

3.1 SQL概述

◆SQL 标准化

➢有关组织

- ✓1986，ANSI(American Natural Standard Institute)
- ✓1987，ISO(International Organization for Standardization)

➢有关标准

- ✓SQL-86
  - ●“数据库语言SQL”
- ✓SQL-89
  - ●“具有完整性增强的数据库语言SQL”，增加了对完整性约束的支持

3.1 SQL概述

◆SQL 标准化

➢有关标准

✓SQL-92

- “数据库语言SQL”，是SQL-89的超集，增加了许多新特性，如新的数据类型，更丰富的数据操作，更强的完整性、安全性支持等。

✓SQL-1999

- 增加了对象关系特征

3.1 SQL概述

◆SQL 标准化

标准	大致页数	发布日期
SQL/86		1986.10
SQL/89(FIPS 127-1)	120页	1989年
SQL/92	622页	1992年
SQL99	1700页	1999年
SQL2003	3600页	2003年

3.1 SQL概述

◆SQL的特点

➢综合统一

- ✓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
- ✓用户数据库投入运行后，可根据需要随时逐步修改模式，不影响数据的运行
- ✓数据操作符统一

3.1 SQL概述

◆SQL的特点

➢高度非过程化

- ✓非关系数据模型的数据操纵语言“面向过程”，必须制定存取路径
- ✓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

3.1 SQL概述

◆SQL的特点

➢面向集合的操作方式

- ✓非关系数据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、更新操作的对象可以是元组的集合

3.1 SQL概述

◆SQL的特点

➢以同一种语法结构提供多种使用方式

- ✓SQL是独立的语言
  - ●能够独立地用于联机交互的使用方式
- ✓SQL又是嵌入式语言
  - ●SQL能够嵌入到高级语言（例如C，C++，Java）程序中，供程序员设计程序时使用

3.1 SQL概述

◆SQL的特点

➢语言简洁，易学易用

✓SQL功能极强，完成核心功能只用了9个动词

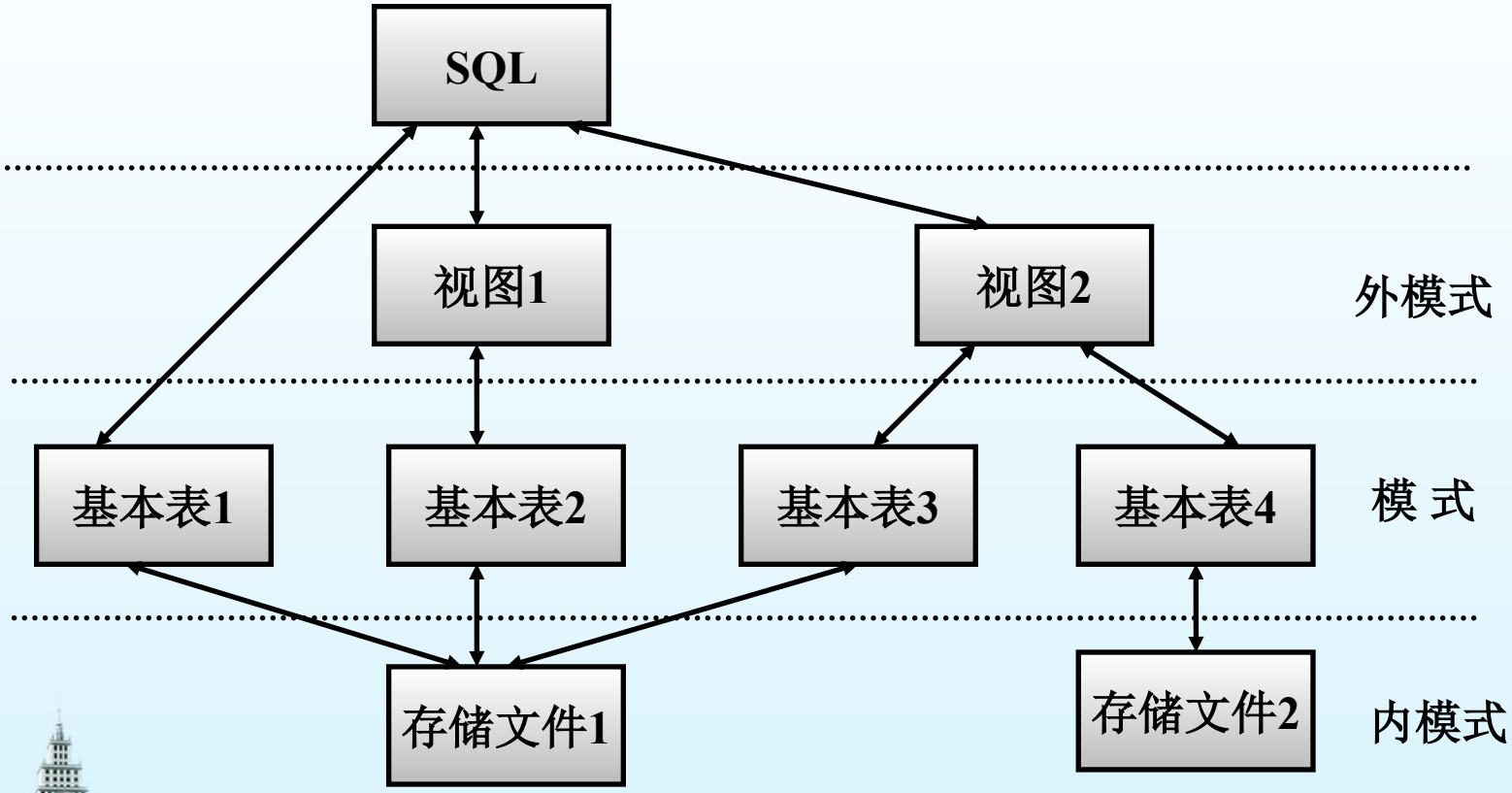
表 3.1　SQL 语言的动词

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

3.1 SQL概述

◆SQL的基本概念

➢SQL支持关系数据库三级模式结构



3.1 SQL概述

◆SQL的基本概念

➢基本表

- ✓本身独立存在的表
- ✓SQL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

➢存储文件

- ✓逻辑结构组成了关系数据库的内模式
- ✓物理结构是任意的，对用户透明

3.1 SQL概述

◆SQL的基本概念

➢视图

- ✓从一个或几个基本表导出的表
- ✓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

本章内容提要

- 3.1 SQL概述
- *3.2 学生-课程数据库*
- 3.3 数据定义
- 3.4 数据查询
- 3.5 数据更新
- 3.6 视图

3.2学生-课程数据库

◆学生-课程模式 S-T

➢院系表

- ✓系号Dno，系名Dname，电话Dtel
- ✓Dept(Dno , Dname , Dtel)

➢学生表

- ✓学号Sno，姓名Sname， 性别Ssex，年龄Sage，所属系别Dno
- ✓Student(Sno,Sname,Ssex,Sage,Dno)

3.2学生-课程数据库

◆学生-课程模式 S-T

➢教师表

- ✓教师编号Pno，教师名Pname，教师年龄Page，所属院系Dno，工资Sal
- ✓Prof(Pno , Pname, Page, Dno , Sal)

3.2学生-课程数据库

◆学生-课程模式 S-T

➢课程表

- ✓课程号Cno，课程名Cname，先行课Cpno，学分Ccredit，任课教师编号Pno
- ✓Course(Cno,Cname,Cpno,Ccredit,Pno)

➢学生选课表

- ✓学号Sno， 课程号Cno，成绩Grade
- ✓SC(Sno,Cno,Grade)

3.2学生-课程数据库

Dept

系号 Dno	系名 Dname	电话 Dtel
CS	计算机	567342
MA	数学	567889
IS	信息	567990
EN	英语	567214

3.2学生-课程数据库

Student

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Dno
20020121	李勇	男	20	EN
20020222	刘晨	女	19	CS
20020323	王敏	女	18	MA
20050425	张立	男	19	IS

3.2学生-课程数据库

Prof

教师编号 Pno	姓 名 Pname	年龄 Page	所在系 Dno	工 资 Sal
04001	王威	男	CS	3000
06007	李亚东	男	EN	1000
08009	李红梅	女	MA	2000
07003	张青	男	IS	4000

3.2学生-课程数据库

Course

课程号 Cno	课程名 Cname	先行课 Cpno	教师编号 Pno	学分 Ccredit
1	数据库	5	04001	4
2	数学		06007	2
3	信息系统	1	08009	4
4	操作系统	6	04003	3
5	数据结构	7	07003	4
6	数据处理		03004	2
7	PASCAL语言	6	04001	4

3.2学生-课程数据库

SC

学 号 Sno	课程号 Cno	成绩 Grade
20020121	1	92
20020121	2	85
20020221	3	88
20020222	2	90
20020322	3	80

本章内容提要

- 3.1 SQL概述
- 3.2 学生-课程数据库
- *3.3 数据定义*
- 3.4 数据查询
- 3.5 数据更新
- 3.6 视图

3.3 数据定义

◆SQL的数据定义功能: 模式定义、表定义、视图和索引的定义

表 3.2　SQL 的数据定义语句

操作对象	操作方式		
	创建	删除	修改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

3.3 数据定义

- 模式的定义与删除
- 基本表的定义、删除与修改
- 索引的建立与删除

3.3 数据定义-模式的定义与删除

◆模式的定义

- ➢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；

CREATE SCHEMA <模式名>
AUTHORIZATION <用户名>

3.3 数据定义-模式的定义与删除

◆模式的定义

- **[例1]定义一个学生-课程模式S-T**

  CREATE SCHEMA “S-T” AUTHORIZATION WANG

  为用户WANG定义了一个模式S-T

- **[例2]CREATE SCHEMA AUTHORIZATION WANG；**

  <模式名>隐含为用户名WANG

  ✓如果没有指定<模式名>，那么<模式名>隐含为<用户名>

3.3 数据定义-模式的定义与删除

◆模式的定义

➢在CREATE SCHEMA中可以接受CREATE TABLE，CREATE VIEW和GRANT子句。

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>]

3.3 数据定义-模式的定义与删除

◆模式的定义

➢[例3]

```
CREATE SCHEMA TEST AUTHORIZATION
  ZHANG
CREATE TABLE TAB1 ( COL1 SMALLINT，
                    COL2 INT，
                    COL3 CHAR(20)，
                    COL4 NUMERIC(10，3)，
                    COL5 DECIMAL(5，2) )；
```

✓为用户ZHANG创建了一个模式TEST，并在其中定义了一个表TAB1。

3.3 数据定义-模式的定义与删除

◆删除模式

- ***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下属的数据库对象（如表、视图等），则拒绝该删除语句的执行。当该模式中没有任何下属的对象时 才能执行

3.3 数据定义-模式的定义与删除

◆删除模式

- [例4] DROP SCHEMA ZHANG CASCADE；
  - ✓删除模式ZHANG
  - ✓同时该模式中定义的表TAB1也被删除

3.3 数据定义-基本表的定义、删除与修改

◆定义基本表

➢ ***CREATE TABLE*** <表名>

(<列名> <数据类型>[<列级完整性约束条件>]
[，<列名> <数据类型>[<列级完整性约束条件>]]

…

[，<表级完整性约束条件>])；

✓如果完整性约束条件涉及到该表的多个属性列，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。

3.3 数据定义-基本表的定义、删除与修改

◆数据类型

- 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
  - ✓取值范围
  - ✓要做哪些运算

3.3 数据定义-基本表的定义、删除与修改

◆数据类型

数据类型	含义
CHAR(n)	长度为n的定长字符串
VARCHAR(n)	最大长度为n的变长字符串
INT	长整数（也可以写作INTEGER）
SMALLINT	短整数
NUMERIC(p，d)	定点数，由p位数字（不包括符号、小数点）组成，小数后面有d位数字
REAL	取决于机器精度的浮点数
Double Precision	取决于机器精度的双精度浮点数
FLOAT(n)	浮点数，精度至少为n位数字
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

3.3 数据定义-基本表的定义、删除与修改

◆数据类型

- 现行商用DBMS的数据类型有时和上面有些差异，请注意：和高级语言的数据类型，总体上是一致的，但也有些差异

3.3 数据定义-基本表的定义、删除与修改

◆ SQL Server的数据类型

	语法	存储长度	适用范围	备注
二进制型	Binary	N+4字节	表示二进数据长度基本相同时可以使用	N为1~8000，最后用检索输出的是二进制
	Varbinary	实际长度	二进制数据的长度未知或变化较大时可用	存放8000字节内可变长数据
图形	image	实际大小	照片、图、画	最大可存储2^{31}

3.3 数据定义-基本表的定义、删除与修改

◆ SQL Server的数据类型

	语法	存储长度	适用范围	备注
整型数据类型	Int	4个字节	2^{31}～(2^{31}-1)内所有正负整数数	存储可直接运算的数值
	Smallint	2个字节	-2^{15}～(2^{15}-1)内所有正负整数数	
	Tinyint	1个字节	0~255范围的所有正整数	
文本型	Text	实际大小	最大可存储2^{31}-1	存储文本数据
	Ntext	实际大小	最大可存储2^{30}-1	数据可直接输出

3.3 数据定义-基本表的定义、删除与修改

◆ SQL Server的数据类型

	语法	存储长度	适用范围	备注
浮点数据类型	Real	4个字节	精确到7位小数	存储十进制小数。采用只入不舍的方式存储。
	Float	8个字节	精确到15位小数	
	Decimal	实际存储空间	Decimal (p,s), p表示总位数，s表示小数点后的位数。	
	Numeric			
货币型	Money	8个字节	用于存储货币	精确度为万分之一
	Samllmoney	4个字节	范围比Money小	
位型	Bit	1个字节	常用作逻辑变量表示真假	只能输入0、1

3.3 数据定义-基本表的定义、删除与修改

◆ SQL Server的数据类型

	语法	存储长度	适用范围	备注
字符型	Char(n)	N字节	输入字符少于n,以空格填满。若超长则截掉。	N为1-8000范围
	Varchar(n)	实际长度	N为最大长度小于N时不加空格	可节省空间
	Nchar(n)	N字节	Unicode标准，两个字节为存储单位，容纳量增加了。	N为1-4000范围
	Nvarchar(n)	实际可变值		
日期时间	Datetime	8个字节精度	MM DD YYYY hh:mm	存储日期和时间的结合体, 引用时用单引号
	Small datetime	4个字节 精度1分钟	1900.1.1~2079.6.6	

3.3 数据定义-基本表的定义、删除与修改

◆ SQL Server的数据类型

	语法	存储长度	适用范围	备注
特殊型	Timestamp	8个字节	提供数据库范围内的唯一值	单调上升的计数器
	Unique identifier	16字节	存储一个16字节长的二进制数	全局惟一标识符
新增型	Bigint	8个字节	2^{63} ~ (2^{63}-1)范围内的所有正负数	
	Sql_variant	存储除文本、图形数据和timestamp类型数据外的其他任何合法的SQL server数据。		
	table	用于存储对表或视图处理后的结果集。		
注：全局惟一标识符(GUID),由计算机网卡和CPU时钟产生的，每台机器不会重复。Newid()函数可求出。如：Select newid()				

3.3 数据定义-基本表的定义、删除与修改

◆定义基本表

➢[例5] 建立“学院”表Dept，学院号是主码，学院名不可为空。

```
CREATE  TABLE  Dept
( Dno   CHAR（2）,
  Dname  CHAR（20）  NOT NULL,
  Dtel   CHAR（10）,
  PRIMARY KEY (Dno)
);
```

主码

3.3 数据定义-基本表的定义、删除与修改

◆定义基本表

➢[例6] 建立“学生”表Student，学号是主码，姓名取值唯一。

```
CREATE TABLE Student
(Sno   CHAR(9) PRIMARY KEY,    /* 列级完整性约束条件*/
Sname  CHAR(20) UNIQUE,        /* Sname取唯一值*/
Ssex   CHAR(2),
Sage   SMALLINT,
Dno  CHAR(2),
 FOREIGN KEY (Dno)
        REFERENCES   Dept(Dno));
```

主码

表级完整性约束条件， Dno是外码，被参照表是Dept

3.3 数据定义-基本表的定义、删除与修改

◆定义基本表

➢[例7] 建立一个“教师”表Prof

```
CREATE TABLE  Prof
  ( Pno  CHAR（10）,
    Pname CHAR（20） NOT NULL,
    Sal   SMALLINT,
    Page  SMALLINT,
    Dno   CHAR（2）,
    PRIMARY KEY (Pno),
    FOREIGN KEY (Dno)
        REFERENCES  Dept(Dno));
```

3.3 数据定义-基本表的定义、删除与修改

◆定义基本表

➢[例8] 建立一个“课程”表Course

```
CREATE TABLE  Course
    ( Cno      CHAR(4) PRIMARY KEY ,
      Cname  CHAR(40),
      Cpno     CHAR(4) ,
      Ccredit  SMALLINT,
      Pno  CHAR（10） ,
       FOREIGN KEY (Cpno) REFERENCES
            Course(Cno) ,
       FOREIGN KEY (Pno) REFERENCES
            Prof(Pno) );
```

3.3 数据定义-基本表的定义、删除与修改

◆定义基本表

➢[例9] 建立一个“学生选课”表SC

```
CREATE TABLE  SC
(Sno  CHAR(9),
Cno  CHAR(4),
Grade   SMALLINT,
PRIMARY KEY (Sno ，Cno)，
FOREIGN KEY (Sno) REFERENCES Student(Sno)，
FOREIGN KEY (Cno) REFERENCES Course(Cno)
);
```

主码由两个属性构成，必须作为表级完整性进行定义

3.3 数据定义-基本表的定义、删除与修改

◆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-T”.Student（.....）；

    Create table “S-T”.Cource（.....）；

    Create table “S-T”.SC（.....）；
  - ✓方法二：在创建模式语句中同时创建表
  - ✓方法三：设置所属的模式

3.3 数据定义-基本表的定义、删除与修改

◆修改基本表

➢ALTER TABLE <表名>

[ADD <新列名> <数据类型> [完整性约束]]

[DROP <完整性约束名>]

[ALTER COLUMN<列名> <数据类型>];

3.3 数据定义-基本表的定义、删除与修改

◆修改基本表

➢[例11]向Student表增加“入学时间”列，其数据类型为日期型。

```
ALTER TABLE Student ADD S_entrance DATE;
```

✓不论基本表中原来是否已有数据，新增加的列一律为空值。

```
CREATE TABLE Student
(Sno  NUMERIC(6)
 CONSTRAINT C1 CHECK (Sno BETWEEN
  90000 AND 99999),
 Sname  CHAR(20)
 CONSTRAINT C2 NOT NULL,
 Sage  NUMERIC(3)
 CONSTRAINT C3 CHECK (Sage < 30),
 Ssex  CHAR(2)
 CONSTRAINT C4 CHECK (Ssex IN ( '男'，'女')),
 CONSTRAINT StudentKey PRIMARY KEY(Sno)
 );
```

CREATE TABLE Student

```
(Sno  NUMERIC(6)
 CONSTRAINT C1 CHECK (Sno BETWEEN
  90000 AND 99999),
 Sname  CHAR(20)
 CONSTRAINT C2 NOT NULL,
 Sage  NUME
 CONSTRAIN
 Ssex  CHAR(2)
 CONSTRAINT C4 CHECK (Ssex IN ( '男'，'女')),
 CONSTRAINT StudentKey PRIMARY KEY(Sno)
 );
```

```
ALTER TABLE Student
 DROP CONSTRAINT C1;
```

3.3 数据定义-基本表的定义、删除与修改

◆修改基本表

➢[例12]将年龄的数据类型由字符型（假设原来的数据类型是字符型）改为整数。

```
ALTER TABLE Student
        ALTER COLUMN Sage INT;
```

➢[例13]增加课程名称必须取唯一值的约束条件。

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

3.3 数据定义-基本表的定义、删除与修改

◆删除基本表

➢DROP TABLE <表名>

［RESTRICT| CASCADE］;

✓RESTRICT：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

3.3 数据定义-基本表的定义、删除与修改

◆删除基本表

➢[例14] 删除Student表

```
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

3.3 数据定义-基本表的定义、删除与修改

◆删除基本表

➢［例15］若表上建有视图，选择RESTRICT时表不能删除

```
CREATE VIEW IS_Student
AS
SELECT Sno，Sname，Sage
FROM  Student
WHERE Dno='IS';

DROP TABLE Student RESTRICT;
--ERROR: cannot drop table Student because other
          objects depend on it
```

3.3 数据定义-基本表的定义、删除与修改

◆删除基本表

➢[例16]如果选择CASCADE时可以删除表，视图也自动被删除

```
DROP TABLE Student CASCADE;

--NOTICE: drop cascades to view IS_Student

SELECT * FROM IS_Student;

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

3.3 数据定义-基本表的定义、删除与修改

◆DROP TABLE时，SQL99 与 3个RDBMS的处理策略比较

序号	标准及主流数据库的处理方式 / 依赖基本表的对象	SQL99		Kingbase ES		ORACLE 9i		MS SQL SERVER 2000
		R	C	R	C		C	
1	索引	无规定		√	√	√	√	√
2	视图	×	√	×	√	√ 保留	√ 保留	√ 保留

序号	标准及主流数据库的处理方式 / 依赖基本表的对象	SQL99		Kingbase ES		ORACLE 9i		MS SQL SERVER 2000
		R	C	R	C		C	
3	DEFAULT，PRIMARY KEY，CHECK（只含该表的列）NOT NULL 等约束	√	√	√	√	√	√	√
4	Foreign Key	×	√	×	√	×	√	×
5	TRIGGER	×	√	×	√	√	√	√
6	函数或存储过程	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留

3.3 数据定义-索引的建立与删除

◆建立索引的目的：加快查询速度

顺序查找

学号	姓名	性别	年龄
20060101	张建	男	18
20060201	王欢	女	17
20060102	田静	男	18
20060202	鲜恒	男	19
20060103	田静	女	17

3.3 数据定义-索引的建立与删除

◆建立索引的目的：加快查询速度

索引

索引域	地址
20060101	
20060102	
20060103	
20060201	
20060202	

数据

学号	姓名	性别	年龄
20060101	张建	男	18
20060201	王欢	女	17
20060102	田静	男	18
20060202	鲜恒	男	19
20060103	田静	女	17

3.3 数据定义-索引的建立与删除

- ◆谁可以建立索引
  - ➢DBA 或 表的属主（即建立表的人）
  - ➢DBMS一般会自动建立以下列上的索引
    - PRIMARY KEY
    - UNIQUE
- ◆谁维护索引
  - ➢DBMS自动完成
- ◆使用索引
  - ➢DBMS自动选择是否使用索引以及使用哪些索引

3.3 数据定义-索引的建立与删除

◆索引

- ➢索引是关系数据库的内部实现技术，属于内模式的范畴
- ➢CREATE INDEX语句定义索引时，可以定义索引是唯一索引、非唯一索引或聚簇索引

3.3 数据定义-索引的建立与删除

◆索引

- ➢唯一索引：每个索引值只对应唯一的数据记录。
- ➢聚簇索引：索引项的顺序与表中记录的物理顺序一致的索引组织。

3.3 数据定义-索引的建立与删除

◆建立索引

➢CREATE [UNIQUE] [CLUSTER]

INDEX <索引名>

ON <表名>

(<列名>[<次序>][,<列名>[<次序>]]…);

3.3 数据定义-索引的建立与删除

◆建立索引

- [例17] CREATE CLUSTER INDEX Stusname
  ON Student(Sname)；

在Student表的Sname（姓名）列上建立一个聚簇索引

- 在最经常查询的列上建立聚簇索引以提高查询效率
- 一个基本表上最多只能建立一个聚簇索引
- 经常更新的列不宜建立聚簇索引

3.3 数据定义-索引的建立与删除

◆建立索引

- [例18]为学生-课程数据库中的Student，Course，SC三个表建立索引。

要求： Student表按学号升序建唯一索引
Course表按课程号升序建唯一索引
SC表按学号升序和课程号降序建唯一索引

```
CREATE UNIQUE INDEX  Stusno ON Student(Sno);
CREATE UNIQUE INDEX  Coucno ON Course(Cno);
CREATE UNIQUE INDEX  SCno ON SC(Sno ASC，Cno DESC);
```

3.3 数据定义-索引的建立与删除

◆删除索引

- *DROP* INDEX <索引名>;

- [例19] 删除Student表的Stusname索引

  DROP INDEX Stusname;

3.3 数据定义-索引的建立与删除

- 索引是有用的（可以加速数据访问速度。如没有索引，在查找一个主题的信息时，必须一次一页地扫描整个文本），但索引要耗用磁盘空间，并引起系统开销和增加维护的费用。在使用索引时要考虑以下事实和要点：
  - 修改一个索引列上的数据时，要更新相关的索引
  - 维护索引需要时间和资源，所以不要创建不经常使用的索引
  - 小表上的索引没有多少好处。
  - 如果数据增加、删除、修改频繁，系统需要花费许多时间来维护索引，这时需要删除一些不必要的索引。

本章内容提要

- 3.1 SQL概述
- 3.2 学生-课程数据库
- 3.3 数据定义
- *3.4 数据查询*
- 3.5 数据更新
- 3.6 视图

3.4 数据查询

◆语句格式

➢*SELECT* [ALL|DISTINCT] <目标列表达式>

[，<目标列表达式>] …

FROM <表名或视图名>[， <表名或视图名>] …

[*WHERE* <条件表达式>]

[*GROUP BY* <列名1> [*HAVING* <条件表达式>]]

[*ORDER BY* <列名2> [ASC|DESC]]；

3.4 数据查询

- ◆单表查询
- ◆连接查询
- ◆嵌套查询
- ◆集合查询
- ◆Select语句的一般形式

3.4 数据查询

- ***单表查询***
- 连接查询
- 嵌套查询
- 集合查询
- Select语句的一般形式

3.4 数据查询-单表查询

- 查询仅涉及一个表
  - 选择表中的若干列
  - 选择表中的若干元组
  - ORDER BY子句
  - 聚集函数
  - GROUP BY子句

3.4 数据查询-单表查询

- 查询仅涉及一个表
  - ***选择表中的若干列***
  - 选择表中的若干元组
  - ORDER BY子句
  - 聚集函数
  - GROUP BY子句

3.4 数据查询-单表查询

◆查询指定列

- [例1] 查询全体学生的学号与姓名

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

- [例2] 查询全体学生的姓名、学号、所在系

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

投影的列可以重新排定顺序

3.4 数据查询-单表查询

◆查询指定列

➢[例3] 查询所有老师的姓名

```
SELECT Pname
FROM Prof;
```

3.4 数据查询-单表查询

◆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4] 查询全体学生的详细记录

```
SELECT Sno，Sname，Ssex，Sage，Dno
FROM Student;
```

或

```
SELECT *
FROM Student;
```

3.4 数据查询-单表查询

◆查询全部列

➢[例5]给出所有老师的所有信息

```
SELECT Pno , Pname, Page, Dno , Sal
FROM Prof;
```

或

```
SELECT  *
FROM Prof;
```

3.4 数据查询-单表查询

◆查询经过计算的值

- ➢SELECT子句的<目标列表达式>可以为：
  - ✓算术表达式
  - ✓字符串常量
  - ✓函数（特殊函数的使用需结合各自DBMS的说明书）
  - ✓列别名

3.4 数据查询-单表查询

◆查询经过计算的值

➢[例6] 查全体学生的姓名及其出生年份。

```
SELECT Sname，2015-Sage
FROM Student;
```

输出结果：

Sname	2015-Sage
李勇	1987
刘晨	1988
王敏	1989
张立	1988

3.4 数据查询-单表查询

◆查询经过计算的值

➢[例7]查询所有老师的姓名及税后工资额。

```
select    Pname，sal * 0.95
from      Prof
```

3.4 数据查询-单表查询

◆查询经过计算的值

- [例8] 查询全体学生的姓名、出生年份和所在院系，要求用小写字母表示所有系号

```
SELECT Sname，‘Year of Birth: ’，
                2015-Sage，LOWER(Dno)
 FROM Student；
```

3.4 数据查询-单表查询

◆查询经过计算的值

➢[例8] 查询全体学生的姓名、出生年份和所在系，要求用小写字母表示所有系号

输出结果：

Sname	'Year of Birth:'	2015-Sage	LOWER(Dno)
李勇	Year of Birth:	1987	cs
刘晨	Year of Birth:	1988	is
王敏	Year of Birth:	1989	ma
张立	Year of Birth:	1988	en

3.4 数据查询-单表查询

◆使用列别名改变查询结果的列标题

```
SELECT Sname NAME，'Year of Birth: ' BIRTH，
       2014-Sage BIRTHDAY，
       LOWER(Sdept)  DEPARTMENT
FROM Student;
```

3.4 数据查询-单表查询

◆使用列别名改变查询结果的列标题

输出结果：

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1987	cs
刘晨	Year of Birth:	1988	is
王敏	Year of Birth:	1989	ma
张立	Year of Birth:	1988	is

3.4 数据查询-单表查询

- 查询仅涉及一个表
  - 选择表中的若干列
  - ***选择表中的若干元组***
  - ORDER BY子句
  - 聚集函数
  - GROUP BY子句

3.4 数据查询-单表查询

◆选择表中的若干元组

- ***结果唯一性问题***。尽管关系模型要求无重复元组出现在数据库中，但现实DBMS操作中，是允许检索结果出现重复元组的，但也允许无重复元组。
- 在关系(存储的Table)中要求无重复元组是通过定义***主键***或*Unique*来保证的，在检索结果中要求无重复元组, 是通过*DISTINCT*保留字的使用来实现的
  - ✓如果没有指定DISTINCT关键词，则缺省为ALL
  - ✓指定DISTINCT关键词，去掉表中重复的行

3.4 数据查询-单表查询

◆选择表中的若干元组

➢[例9] 查询选修了课程的学生学号。

```
SELECT Sno
FROM SC;
```

等价于：

```
SELECT ALL  Sno
FROM SC
```

输出结果：

Sno
200215121
200215121
200215121
200215122
200215122

3.4 数据查询-单表查询

◆选择表中的若干元组

➢[例9] 查询选修了课程的学生学号。

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

输出结果：

Sno
200215121
200215122

◆查询满足条件的元组

常用的查询条件

查 询 条 件	谓　词
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

3.4 数据查询-单表查询

◆比较大小

- [例10] 查询CS系全体学生的名单。

```
SELECT Sname
FROM Student
WHERE Dno='CS';
```

- [例11] 查询所有年龄在20岁以下的学生姓名及其年龄。

```
SELECT Sname，Sage
FROM   Student
WHERE Sage < 20;
```

3.4 数据查询-单表查询

◆比较大小

➢[例12] 查询考试成绩有不及格的学生的学号。

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

3.4 数据查询-单表查询

◆比较大小

➢[例13]查询工资低于2000的职工的姓名、工资、系号

```
SELECT Pname , Sal , Dno
FROM  Prof
WHERE Sal < 2000 ;
```

3.4 数据查询-单表查询

◆确定范围

- 谓词： BETWEEN … AND …

  NOT BETWEEN … AND …

- [例14] 查询年龄在20~23岁（包括20岁和23岁）之间的学生的姓名、系号和年龄

```
SELECT Sname，Dno，Sage
FROM   Student
WHERE  Sage BETWEEN 20 AND 23；
```

3.4 数据查询-单表查询

◆确定范围

➢[例15] 查询年龄不在20~23岁之间的学生姓名、系号和年龄

```
SELECT Sname，Dno，Sage
FROM   Student
WHERE Sage NOT BETWEEN 20 AND 23；
```

3.4 数据查询-单表查询

◆确定范围

➢[例16] 查询工资在500~800之间的老师姓名

```
SELECT Pname
FROM   Prof
WHERE sal  between  500   and  800 ;
```

3.4 数据查询-单表查询

◆确定集合

➢谓词： IN <值表>，NOT IN <值表>

➢[例17]查询CS系、MA系和IS系学生的姓名和性别。

```
SELECT Sname，Ssex
FROM  Student
WHERE Dno IN ( ‘CS'，‘MA'，‘IS' );
```

3.4 数据查询-单表查询

◆确定集合

➢[例18]查询张三、王五同学的所有信息。

```
SELECT *
FROM Student
WHERE Sname  in  ('张三', '王五');
```

3.4 数据查询-单表查询

◆确定集合

➢[例19]查询既不是CS系、MA系，也不是IS系的学生的姓名和性别。

```
SELECT Sname，Ssex
FROM Student
WHERE Dno NOT IN ( ‘CS'，‘MA'，‘IS' );
```

3.4 数据查询-单表查询

◆字符匹配

➢谓词： [NOT] LIKE ‘<匹配串>’

[ESCAPE ‘ <换码字符>’]

3.4 数据查询-单表查询

◆字符匹配

- 匹配串为固定字符串
- [例20] 查询学号为200215121的学生的详细情况。

```
SELECT *
FROM  Student
WHERE  Sno LIKE '200215121';
```

等价于：

```
SELECT  *
FROM  Student
WHERE Sno = '200215121';
```

3.4 数据查询-单表查询

◆字符匹配

- ➢匹配串为含通配符的字符串
  - ✓“%” 匹配零个或多个字符
  - ✓“_” 匹配任意单个字符

3.4 数据查询-单表查询

◆字符匹配

- 匹配串为含通配符的字符串
- [例21] 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%’；
```

3.4 数据查询-单表查询

◆字符匹配

➢[例22] 查询姓"欧阳"且全名为三个汉字的学生的姓名。

```
SELECT Sname
FROM   Student
WHERE  Sname LIKE '欧阳_ ';
```

3.4 数据查询-单表查询

◆字符匹配

➢[例23] 查询名字中第2个字为"阳"字的学生的姓名和学号。

```
SELECT Sname，Sno
FROM Student
WHERE Sname LIKE '_ 阳%';
```

3.4 数据查询-单表查询

◆字符匹配

➢[例24] 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%';
```

3.4 数据查询-单表查询

◆字符匹配

➢[例25] 查询姓名以“张”打头的教师的所有信息。

```
SELECT *
FROM Prof
WHERE Pname  like ‘张%’;
```

3.4 数据查询-单表查询

◆字符匹配

- ➢使用换码字符将通配符转义为普通字符
  - ✓ESCAPE '\' 表示“ \ ”为换码字符
- ➢ [例26] 查询DB_Design课程的课程号和学分。

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

3.4 数据查询-单表查询

◆字符匹配

- [例27] 查询名称中含有4个字符以上，且倒数第3个字符是d，倒数第2个字符是_的教师的所有信息。

```
SELECT *
 FROM Prof
 WHERE Pname  like  '% _d \_ _' ESCAPE '\';
```

3.4 数据查询-单表查询

◆字符匹配

- [例28] 查询以"DB_"开头，且倒数第3个字符为 i的课程的详细情况

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

3.4 数据查询-单表查询

◆涉及空值的查询

- 空值是其值不知道、不确定、不存在的值
- 数据库中有了空值，会影响许多方面，如影响聚集函数运算的正确性，不能参与算术、比较或逻辑运算等

◆涉及空值的查询

➢例如：右下图所示表SC, 如果有某一记录为空值，则求001号课程的平均成绩？会是多少呢？

SC

Sno	Cno	Grade
98030101	001	92
98040202	001	55
98030102	001	?

3.4 数据查询-单表查询

◆涉及空值的查询

- 以前，很多DBMS将空值按默认值处理，如字符串类型则以空格来表示，而如数值类型则以0来表示，这也将会引起统计、计算上的不正确性
- 在SQL标准中和许多现流行的DBMS中，空值被用一种特殊的符号Null来标记，使用特殊的空值检测函数来获得某列的值是否为空值。

3.4 数据查询-单表查询

◆涉及空值的查询

- 谓词： IS NULL 或 IS NOT NULL

  “IS” 不能用 “=” 代替

  （空值是不能进行运算的）

- [例29] 某些学生选修课程后没有参加考试，所以有选课记录，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
```

3.4 数据查询-单表查询

◆涉及空值的查询

➢[例30] 查询所有有成绩的学生学号和课程号。

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

3.4 数据查询-单表查询

◆现行DBMS的空值处理小结

- ➢除is [not] null之外，空值不满足任何查找条件
- ➢如果null参与算术运算，则该算术表达式的值为null
- ➢如果null参与比较运算，则结果可视为false。在SQL-92中可看成unknown
- ➢如果null参与聚集运算，则除count(*)之外其它聚集函数都忽略null

◆现行DBMS的空值处理小结

例：select sum(score) 350
from SC

例：select count(*) 6
from SC

Sno	Cno	Score
s1	c1	80
s1	c2	90
s1	c3	95
s2	c1	85
s2	c2	null
s3	c2	null

3.4 数据查询-单表查询

◆多重条件查询

- ➢逻辑运算符：AND和 OR来联结多个查询条件
  - ✓AND的优先级高于OR
  - ✓可以用括号改变优先级
  - ✓可用来实现多种其他谓词
    - [NOT] IN
    - [NOT] BETWEEN … AND …

3.4 数据查询-单表查询

◆多重条件查询

➢[例31] 查询CS系年龄在20岁以下的学生姓名。

```
SELECT Sname
FROM  Student
WHERE Dno= 'CS' AND Sage<20;
```

3.4 数据查询-单表查询

◆多重条件查询

➢改写[例17] 查询CS系、MA系和IS系学生的姓名和性别

```
SELECT Sname，Ssex
  FROM Student
  WHERE Dno IN ( ‘CS'，‘MA'，‘IS' )
```

可改写为：

```
SELECT Sname，Ssex
FROM   Student
WHERE Dno = ' CS ' OR Dno = ' MA' OR
            Dno = ' IS ';
```

3.4 数据查询-单表查询

◆多重条件查询

➢改写[例18]查询张三、王五同学的所有信息。

```
SELECT *
FROM Student
WHERE Sname  in  (“张三”, “王五”);
```

可改写为:

```
SELECT *
FROM   Student
WHERE Sname = “张三”  or  Sname = “王五;
```

3.4 数据查询-单表查询

◆多重条件查询

➢改写[例16] 查询工资在500~800之间的老师姓名

```
SELECT Pname
FROM   Prof
WHERE sal  between  500   and  800 ;
```

可改写为：

```
SELECT Pname
FROM   Prof
WHERE sal  <=800   and  sal>=500 ;
```

3.4 数据查询-单表查询

◆多重条件查询

➢[例32]检索教师表中所有工资少于1500元或者工资大于2000元，并且是MA系的教师姓名？如何理解与书写检索条件？

```
SELECT Pname
FROM   Prof
WHERE Sal< 1500  or  Sal  > 2000
          and  Dno = 'MA';
```

3.4 数据查询-单表查询

◆多重条件查询

➢[例32]检索教师表中所有工资少于1500元或者工资大于2000元，并且是MA系的教师姓名？如何理解与书写检索条件？

```
SELECT Pname
FROM   Prof
WHERE (Sal<1500  or  Sal>2000)
      and  Dno = 'MA';
```

3.4 数据查询-单表查询

◆多重条件查询

➢[例33]求或者学过1号课程, 或者学过2号课程的学生的学号

```
SELECT Sno
FROM   SC
WHERE  Cno='1' or Cno='2' ;
```

3.4 数据查询-单表查询

◆多重条件查询

- [例]求既学过1号课程, 又学过2号课程的学生的学号？
- 如下书写SQL语句会得到正确结果吗？它能得到什么结果？怎样正确书写？

```
SELECT Sno
FROM   SC
WHERE  Cno='1' and  Cno='2' ;
```

3.4 数据查询-单表查询

- 查询仅涉及一个表
  - 选择表中的若干列
  - 选择表中的若干元组
  - ***ORDER BY子句***
  - 聚集函数
  - GROUP BY子句

3.4 数据查询-单表查询

◆ORDER BY子句

- ➢可以按一个或多个属性列排序
- ➢升序：ASC；降序：DESC；缺省值为升序
- ➢当排序列含空值时
  - ✓ASC：排序列为空值的元组最后显示
  - ✓DESC：排序列为空值的元组最先显示

3.4 数据查询-单表查询

◆ORDER BY子句

➢[例34] 查询选修了3号课程的学生的学号及其成绩，查询结果按分数降序排列。

```
SELECT Sno，Grade
FROM  SC
WHERE  Cno= ' 3 '
ORDER BY Grade DESC;
```

3.4 数据查询-单表查询

◆ORDER BY子句

➢[例35] 查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列。

```
SELECT  *
FROM  Student
ORDER BY Dno，Sage DESC；
```

◆ORDER BY子句

➢[例36] 按系号升序列出老师姓名，所在系别，同一系中老师按姓名降序排列。

```
SELECT  Dno,Pname
FROM  Prof
ORDER BY Dno，Pname DESC；
```

3.4 数据查询-单表查询

- 查询仅涉及一个表
  - 选择表中的若干列
  - 选择表中的若干元组
  - ORDER BY子句
  - ***聚集函数***
  - GROUP BY子句

3.4 数据查询-单表查询

◆聚集函数

- ➢计数
  - ✓COUNT（[DISTINCT | ALL] *）
  - ✓COUNT（[DISTINCT | ALL] <列名>）
- ➢计算总和
  - ✓SUM（[DISTINCT | ALL] <列名>）
- ➢计算平均值
  - ✓AVG（[DISTINCT | ALL] <列名>）

3.4 数据查询-单表查询

◆聚集函数

➢最大最小值

- ✓MAX（[DISTINCT | ALL] <列名>）
- ✓MIN（[DISTINCT | ALL] <列名>）

3.4 数据查询-单表查询

◆聚集函数

➢ [例37] 查询学生总人数。

```
SELECT COUNT(*)
FROM  Student;
```

➢[例38] 查询选修了课程的学生人数。

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

3.4 数据查询-单表查询

◆聚集函数

➢[例39] 计算1号课程的学生平均成绩。

```
SELECT AVG(Grade)
FROM SC
WHERE Cno= ' 1 ';
```

➢ [例40] 查询选修1号课程的学生最高分数。

```
SELECT MAX(Grade)
 FROM SC
 WHER Cno= ' 1 '  ;
```

3.4 数据查询-单表查询

◆聚集函数

➢［例41］查询工号为04001的老师所授课程数

```
SELECT COUNT(Cno)
FROM  COURSE
WHERE Pno='04001';
```

3.4 数据查询-单表查询

◆聚集函数

➢［例42］查询先修课是3号课程的课程的总学分数

```
SELECT SUM(Ccredit)
FROM  Course
WHER Cpno='3';
```

3.4 数据查询-单表查询

◆聚集函数

- 下面的查询请求该如何表达呢？
  - ✓求每一门课程的平均成绩
  - ✓求每一个学生的平均成绩
  - ✓以上问题，将在分组查询中解决

3.4 数据查询-单表查询

- 查询仅涉及一个表
  - 选择表中的若干列
  - 选择表中的若干元组
  - ORDER BY子句
  - 聚集函数
  - *GROUP BY子句*

3.4 数据查询-单表查询

◆GROUP BY子句：细化聚集函数的作用对象

- ➢按指定的一列或多列值分组，值相等的为一组
- ➢未对查询结果分组，聚集函数将作用于整个查询结果
- ➢对查询结果分组后，聚集函数将分别作用于每个组

◆GROUP BY子句

➢[例43]求每一个学生的平均成绩。

Group by Sno

Sno	Cno	Grade
98030101	001	92
98030101	002	85
98030101	003	88
98040202	002	90
98040202	003	80
98040202	001	55
98040203	003	56
98030102	001	54
98030102	002	85
98030102	003	48

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例43]求每一个学生的平均成绩。

```
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
```

◆GROUP BY子句

➢[例44]求每一门课程的平均成绩。

Group by Cno

Sno	Cno	Grade
98030101	001	92
98040202	001	55
98030102	001	54
98030101	002	85
98030102	002	85
98040202	002	90
98040202	003	80
98030101	003	88
98040203	003	56
98030102	003	48

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例44]求每一门课程的平均成绩。

```
SELECT Cno, AVG(Grade)
FROM SC
GROUP BY Cno
```

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例45] 求各个课程号及相应的选课人数。

```
SELECT Cno，COUNT(Sno)
FROM   SC
GROUP BY Cno;
```

查询结果：

Cno	COUNT(Sno)
1	22
2	34
3	44
4	33
5	48

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例46]列出各系的老师的最高、最低、平均工资。

```
SELECT Dno，MAX(sal)，MIN(sal)，AVG(sal)
FROM prof
GROUP BY Dno
```

◆GROUP BY子句

➢[例47] 求不及格课程超过两门的同学的学号，下述写法正确吗？

```
Select  Sno
From  SC
Where  Grade < 60  and  Count(*)>2
Group by  Sno;
```

◆GROUP BY子句

- 前述写法是不正确的，聚集函数是不允许用于Where子句中的：Where子句是对每一元组进行条件过滤，而不是对集合进行条件过滤
- 若要对集合(即分组)进行条件过滤，可使用Having短语
- Having短语，又称为分组过滤短语。需要有Group by子句支持，换句话说，没有Group by子句，便不能有Having短语

3.4 数据查询

◆语句格式

➢SELECT [ALL|DISTINCT] <目标列表达式>

[，<目标列表达式>] …

FROM <表名或视图名>[， <表名或视图名>] …

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]]；

◆GROUP BY子句

➢[例47] 求不及格课程超过两门的同学的学号

```
Select  Sno
From  SC
Where  Grade < 60
Group by  Sno
Having Count(*)>2
```

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例48] 查询选修了3门以上课程的学生学号。

```
SELECT Sno
FROM  SC
GROUP BY Sno
 HAVING  COUNT(*) >3;
```

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例49]列出各系年龄大于60岁的老师的平均工资。

```
SELECT Dno，AVG(sal)
FROM prof
GROUP BY Dno
HAVING age>60
```

```
SELECT Dno，AVG(sal)
FROM prof
WHERE age>60
GROUP BY Dno
```

◆GROUP BY子句

➢HAVING短语与WHERE子句的区别

- ✓作用对象不同
- ✓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。

◆GROUP BY子句

➢HAVING短语与WHERE子句的区别

每一分组检查满足与否的条件要用Having子句表达。

注意：不是每一行都检查，所以使用Having子句一定要有Group by子句

S#	C#	Score
98030101	001	92
98030101	002	85
98030101	003	88
98040202	002	90
98040202	003	80
98040202	001	55
98040203	003	56
98030102	001	54
98030102	002	85
98030102	003	48

每一行都要检查满足与否的条件要用WHERE子句表达

3.4 数据查询-单表查询

◆GROUP BY子句

➢[例50]列出每一年龄组中男学生（超过50人）的人数。

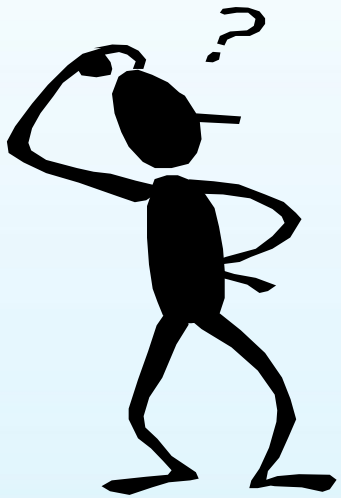
```
SELECT Sage，COUNT(Sno)
FROM Student
Where Sex = '男'
GROUP BY Sage
HAVING count(*) > 50
```

◆GROUP BY子句

➢[例51]求有两门以上不及格课程的同学的学号及其平均成绩

```
Select   Sno, Avg(Score)
From  SC
Where  Score < 60
Group by   Sno
Having  Count(*)>2;
```

分组查询仍需要注意语义问题



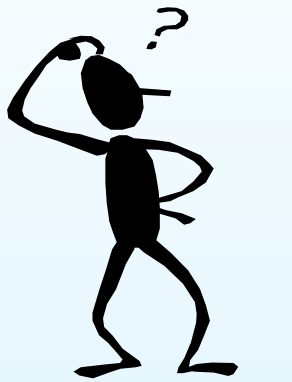
上述写法正确吗？

◆GROUP BY子句

➢[例52]列出所有课程及格的学生的平均成绩。

```
select   Sno，AVG(Grade)
from     SC
group by  Sno
having  MIN(Grade) >= 60
```

```
select    Sno，AVG(Grade)
from      SC
where     Grade >=60
group by  Sno
```



哪个正确？

3.4 数据查询-单表查询

◆下列写法正确吗？

①求选修了课程的学生人数

```
select    count (Sno)
from      SC
```

②

```
select    PNAME，max(SAL)
from      PROF
```

3.4 数据查询-单表查询

求各个课程号及相应的选课人数。

```
SELECT Cno，COUNT(Sno)
FROM   SC
GROUP BY Cno;
```

3.4 数据查询

- 单表查询
- *连接查询*
- 嵌套查询
- 集合查询
- Select语句的一般形式

3.4 数据查询-连接查询

- 连接查询：同时涉及多个表的查询
- 连接条件或连接谓词：用来连接两个表的条件
- 连接条件的一般格式
  - [<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，但名字不必是相同的

3.4 数据查询-连接查询

- 等值与非等值连接查询
- 自身连接
- 外连接
- 复合条件连接

3.4 数据查询-连接查询

- ***等值与非等值连接查询***
- 自身连接
- 外连接
- 复合条件连接

3.4 数据查询-连接查询

◆等值与非等值连接查询

➢等值连接：连接运算符为=

✓[例53] 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；
```

3.4 数据查询-连接查询

查询结果：

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
200215121	李勇	男	20	CS	200215121	1	92
200215121	李勇	男	20	CS	200215121	2	85
200215121	李勇	男	20	CS	200215121	3	88
200215122	刘晨	女	19	CS	200215122	2	90
200215122	刘晨	女	19	CS	200215122	3	80

3.4 数据查询-连接查询

◆等值与非等值连接查询

➢自然连接

✓出现在结果关系中的两个连接关系的元组在公共属性上取值相等，且公共属性只出现一次

3.4 数据查询-连接查询

◆等值与非等值连接查询

➢自然连接

✓[例54] 查询每个学生及其选修课程的情况，用自然连接完成

```
SELECT  Student.Sno，Sname，
                Ssex，Sage，Dno，Cno，Grade
FROM    Student，SC
WHERE  Student.Sno = SC.Sno；
```

3.4 数据查询-连接查询

◆连接操作的执行过程

➢嵌套循环法(NESTED-LOOP)

- ✓首先在表1中找到第一个元组，然后从头开始扫描表2，逐一查找满足连接条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组。
- ✓表2全部查找完后，再找表1中第二个元组，然后再从头开始扫描表2，逐一查找满足连接条件的元组，找到后就将表1中的第二个元组与该元组拼接起来，形成结果表中一个元组。
- ✓重复上述操作，直到表1中的全部元组都处理完毕

3.4 数据查询-连接查询

◆连接操作的执行过程

➢索引连接(INDEX-JOIN)

- ✓对表2按连接字段建立索引
- ✓对表1中的每个元组，依次根据其连接字段值查询表2的索引，从中找到满足条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组

3.4 数据查询-连接查询

- 等值与非等值连接查询
- ***自身连接***
- 外连接
- 复合条件连接

3.4 数据查询-连接查询

- **自身连接：一个表与其自己进行连接**
  - 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55]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;
```

3.4 数据查询-连接查询

FIRST表（Course表）

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL	6	4

SECOND表（Course表）

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL	6	4

查询结果

Cno	Cpno
1	7
3	5
5	6

3.4 数据查询-连接查询

◆自身连接：一个表与其自己进行连接

➢［例56］求有薪水差额的任意两位教师

```
Select  P1.Pname, P2.Pname
From  Prof  P1,  Prof  P2
Where   P1.Salary > P2.Salary ;
```

3.4 数据查询-连接查询

◆自身连接：一个表与其自己进行连接

➢［例57］求年龄有差异的任意两位同学的姓名

```
Select  S1.Sname, S2.Sname
 From  Student  S1,  Student  S2
Where  S1.Sage > S2.Sage ;
```

3.4 数据查询-连接查询

- 等值与非等值连接查询
- 自身连接
- *外连接*
- 复合条件连接

3.4 数据查询-连接查询

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
200215121	李勇	男	20	CS	200215121	1	92
200215121	李勇	男	20	CS	200215121	2	85
200215121	李勇	男	20	CS	200215121	3	88
200215122	刘晨	女	19	CS	200215122	2	90
200215122	刘晨	女	19	CS	200215122	3	80

3.4 数据查询-连接查询

◆外连接

- ➢外连接与普通连接(内连接)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

3.4 数据查询-连接查询

◆外连接

- ➢左外连接
  - ✓左边关系的任何元组t都会出现在结果表中，如右边关系中有满足连接条件的元组s, 则t与s连接；否则t与空值元组连接；
- ➢右外连接
  - ✓右边关系的任何元组t都会出现在结果表中，如左边关系中有满足连接条件的元组s, 则t与s连接；否则t与空值元组连接；
- ➢全外连接
  - ✓是前两者的并

3.4 数据查询-连接查询

◆外连接

R

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

S

C	D
c1	d1
c2	d2
c4	d3

$R.C = S.C$

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2

3.4 数据查询-连接查询

◆外连接

R

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

S

C	D
c1	d1
c2	d2
c4	d3

R left out join S on R.C = S.C

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	null

3.4 数据查询-连接查询

◆外连接

R

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

S

C	D
c1	d1
c2	d2
c4	d3

R right out join S **on** *R.C = S.C*

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
null	null	c4	d3

3.4 数据查询-连接查询

◆外连接

R

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

S

C	D
c1	d1
c2	d2
c4	d3

R full out join S on R.C = S.C

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	null
null	null	c4	d3

3.4 数据查询-连接查询

◆外连接

➢[例58]改写[例53]查询每个学生及其选修课程的情况

```
SELECT Student.Sno，Sname，
       Ssex，Sage，Dno，Cno，Grade
FROM  Student  LEFT OUT JOIN SC ON
       (Student.Sno=SC.Sno)；
```

3.4 数据查询-连接查询

执行结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
200215121	李勇	男	20	CS	1	92
200215121	李勇	男	20	CS	2	85
200215121	李勇	男	20	CS	3	88
200215122	刘晨	女	19	CS	2	90
200215122	刘晨	女	19	CS	3	80
200215123	王敏	女	18	MA	NULL	NULL
200215125	张立	男	19	IS	NULL	NULL

3.4 数据查询-连接查询

◆外连接

➢[例59]列出老师的教工号、姓名、工资、所教课程号

```
SELECT PROF.Pno，Pname，Sal，Cno
FROM  Prof  LEFT OUT JOIN COURSE ON
                (Prof.Pno=COURSE.Pno)；
```

3.4 数据查询-连接查询

- 等值与非等值连接查询
- 自身连接
- 外连接
- ***复合条件连接***

3.4 数据查询-连接查询

◆复合条件连接：WHERE子句中含多个连接条件

➢[例60]查询选修2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= '2' AND SC.Grade > 90;
                /* 其他限定条件 */
```

3.4 数据查询-连接查询

◆复合条件连接：WHERE子句中含多个连接条件

➢［例61］求既学过1号课程, 又学过2号课程的学生的学号

```
Select   S1.Sno
From  SC   S1, SC   S2
Where   S1.Sno = S2.Sno
            and  S1.Cno= '1'
           and  S2.Cno= '2 ' ;
```

3.4 数据查询-连接查询

◆复合条件连接：WHERE子句中含多个连接条件

➢［例62］查询学生200215012选修课程的总学分数

```
SELECT SUM(Ccredit)
FROM  SC，  Course
WHER Sno='200215012'
        AND SC.Cno=Course.Cno;
```

3.4 数据查询-连接查询

◆复合条件连接：WHERE子句中含多个连接条件

➢［例63］求“1”号课成绩比“2”号课成绩高的所有学生的学号

```
Select   S1.Sno
From  SC   S1, SC   S2
Where   S1.Sno = S2.Sno
          and   S1.Cno= '1'
          and  S2.Cno= '2 '
          and  S1.Grade > S2.Grade;
```

3.4 数据查询-连接查询

◆复合条件连接

➢[例64]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   /*多表连接*/
  WHERE Student.Sno = SC.Sno
            and SC.Cno = Course.Cno;
```

3.4 数据查询-连接查询

◆复合条件连接

➢[例65]查询教授“哲学”课程的老师的教工号及姓名

```
select    Prof. Pno , Pname
from      Prof , Course
where     Prof.Pno = COURSE.Pno
          and    Course.Cname = ‘哲学’
```

3.4 数据查询

- 单表查询
- 连接查询
- ***嵌套查询***
- 集合查询
- Select语句的一般形式

3.4 数据查询-嵌套查询

◆嵌套查询

- ➢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*

```
SELECT Sname    /*外层查询/父查询*/
FROM Student
WHERE Sno IN
        （SELECT Sno      /*内层查询/子查询*/
          FROM SC
          WHERE Cno= ' 2 '）；
```

3.4 数据查询-嵌套查询

◆嵌套查询

- ➢层层嵌套方式反映了 SQL语言的结构化
- ➢子查询的限制
  - ✓不能使用ORDER BY子句
- ➢有些嵌套查询可以用连接运算替代

3.4 数据查询-嵌套查询

◆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。

3.4 数据查询-嵌套查询

◆嵌套查询求解方法

- ➢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

3.4 数据查询-嵌套查询

- 带有IN谓词的子查询
- 带有比较运算符的子查询
- 带有ANY（SOME）或ALL谓词的子查询
- 带有EXISTS谓词的子查询

3.4 数据查询-嵌套查询

- ***带有IN谓词的子查询***
- **带有比较运算符的子查询**
- **带有ANY（SOME）或ALL谓词的子查询**
- **带有EXISTS谓词的子查询**

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

➢[例66] 查询与“刘晨”在同一个系学习的学生。

此查询要求可以分步来完成

① 确定“刘晨”所在系

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

结果为： CS

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

➢[例67] 查询与“刘晨”在同一个系学习的学生。

② 查找所有在CS系学习的学生。

```
SELECT  Sno，Sname，Dno
FROM    Student
WHERE  Dno= ‘CS';
```

Sno	Sname	Dno
200215121	李勇	CS
200215122	刘晨	CS

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

➢[例67] 查询与“刘晨”在同一个系学习的学生。

将第一步查询嵌入到第二步查询的条件中

```
SELECT Sno，Sname，Dno
FROM Student
WHERE Dno  IN
                (SELECT Dno
          FROM Student
          WHERE Sname= ‘刘晨’)；
```

此查询为不相关子查询。

3.4 数据查询-嵌套查询

用自身连接完成[例67]查询要求

```
SELECT  S1.Sno，S1.Sname，S1.Dno
FROM    Student S1，Student S2
WHERE   S1.Dno= S2.Dno  AND
        S2.Sname = '刘晨';
```

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

- [例68]查询选修了课程名为“信息系统”的学生学号和姓名

3.4 数据查询-嵌套查询

```
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'
        );
```

③ 最后在Student关系中取出Sno和Sname

② 然后在SC关系中找出选修了3号课程的学生学号

① 首先在Course关系中找出“信息系统”的课程号，为3号

3.4 数据查询-嵌套查询

用连接查询实现[例68]

```
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';
```

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

➢对[例51]“求有两门以上不及格课程的同学的学号及其平均成绩”给出正确语句

```
Select   Sno, AVG(grade)
From  SC
Where  Sno  in
      ( Select   Sno  From  SC
      Where  grade < 60
      Group by   Sno
      Having  Count(*)>2 )
 Group by Sno;
```

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

➢重新改写[例61]“求既学过1号课程, 又学过2号课程的学生的学号”，用嵌套查询实现

```
Select   Sno
From  SC
Where Cno = ‘1’  and
   Sno in
      (  Select   Sno
         From  SC
          Cno = ‘2’
      );
```

3.4 数据查询-嵌套查询

◆带有IN谓词的子查询

➢[例69] 列出没学过李明老师讲授课程的所有同学的姓名

```
Select   Sname
From Student
Where Sno not in
      (  Select   Sno
         From   SC,Prof,Course
        Where   Prof.Pname = '李明'  and
         Prof.Pno = Course.Pno  and
         SC.Cno =  Course.Cno );
```

3.4 数据查询-嵌套查询

- 带有IN谓词的子查询
- ***带有比较运算符的子查询***
- 带有ANY（SOME）或ALL谓词的子查询
- 带有EXISTS谓词的子查询

3.4 数据查询-嵌套查询

◆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< >）。

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

➢例：假设一个学生只可能在一个系学习，并且必须属于一个系，则在[例67]可以*用 = 代替IN*：

```
SELECT Sno，Sname，Dno
FROM   Student
WHERE Dno=
        (SELECT Dno
         FROM   Student
         WHERE Sname= '刘晨');
```

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

子查询一定要跟在比较符之后

错误的例子：

```
SELECT  Sno，Sname，Dno
FROM    Student
WHERE ( SELECT Dno
        FROM Student
        WHERE Dno=‘刘晨’)
        = Dno;
```

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

➢［例70］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 ) ;
```

此查询为相关子查询

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

➢可能的执行过程

✓(1)从外层查询中取出SC的一个元组x，将元组x的Sno值（200215121）传送给内层查询。

```
SELECT  AVG(Grade)
 FROM  SC  y
 WHERE  y.Sno='200215121';
```

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

➢可能的执行过程

✓(2)执行内层查询，得到值88（近似值），用该值代替内层查询，得到外层查询：

```
SELECT Sno，  Cno
FROM  SC x
WHERE Grade >=88;
```

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

- ➢可能的执行过程
  - ✓(3) 执行这个查询，得到

    （200215121，1）

    （200215121，3）

3.4 数据查询-嵌套查询

◆带有比较运算符的子查询

➢可能的执行过程

✓(4) 外层查询取出下一个元组重复做上述1至3步骤，直到外层的SC元组全部处理完毕。结果为:

（200215121，1）

（200215121，3）

（200215122，2）

3.3 数据查询-嵌套查询

- 带有比较运算符的子查询
  - [例75]找出所有课程都不及格的学生姓名。

```
SELECT Sname
    FROM Student
    WHERE 60>
                (SELECT max(Grade)
                 FROM SC
                  WHERE Sno=Sudent.Sno);
```

3.4 数据查询-嵌套查询

- 带有IN谓词的子查询
- 带有比较运算符的子查询
- *带有ANY（SOME）或ALL谓词的子查询*
- 带有EXISTS谓词的子查询

3.4 数据查询-嵌套查询

◆带有ANY（SOME）或ALL谓词的子查询

- ➢谓词语义
  - ✓ANY：任意一个值
  - ✓ALL：所有值

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - 需要配合使用比较运算符

> 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

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - 需要配合使用比较运算符

<= 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
= 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

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - [例71] 查询其他系中比1系某一学生年龄小的学生姓名和年龄

```
SELECT Sname，Sage
FROM   Student
WHERE Sage < ANY
                (SELECT  Sage
                 FROM   Student
                 WHERE Dno= ' 1 ')
AND Dno <>  '1 ' ;
```

3.4 数据查询-嵌套查询

结果：

Sname	Sage
王敏	18
张立	19

执行过程：

1. RDBMS执行此查询时，首先处理子查询，找出1系中所有学生的年龄，构成一个集合(20，19)
2. 处理父查询，找所有不是1系且年龄小于20 或19的学生

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - 用聚集函数实现[例71]

```
SELECT Sname，Sage
    FROM   Student
    WHERE Sage <
                (SELECT MAX(Sage)
                 FROM Student
                 WHERE Dno= '1 ')
        AND Dno <> ' 1 ' ;
```

3.4 数据查询-嵌套查询

◆ 带有ANY（SOME）或ALL谓词的子查询

- **[例72] 查询其他系中比1系*所有*学生年龄都小的学生姓名及年龄。**

方法一：用ALL谓词

```
SELECT Sname，Sage
FROM Student
WHERE Sage < ALL
                (SELECT Sage
                 FROM Student
                 WHERE Dno = ' 1 ')
      AND Dno<> ' 1';
```

3.4 数据查询-嵌套查询

◆ 带有ANY（SOME）或ALL谓词的子查询

➢ **[例72] 查询其他系中比1系*所有*学生年龄都小的学生姓名及年龄。**

方法二：用聚集函数

```
SELECT Sname，Sage
FROM Student
WHERE Sage <
            (SELECT MIN(Sage)
             FROM Student
             WHERE Dno = ' 1 ')
AND Dno <>' 1 ';
```

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - 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

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - [例73]找出工资最低的教师姓名。

```
SELECT Pname
    FROM Prof
    WHERE Sal <=all
                (SELECT Sal
                 FROM Prof);
```

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - [例74]找出1号课成绩不是最高的所有学生的学号。

```
SELECT Sno
    FROM SC
    WHERE Cno='1' and
            Grade<some
                (SELECT Grade
                 FROM SC
                 WHERE Cno='1');
```

3.4 数据查询-嵌套查询

- 带有ANY（SOME）或ALL谓词的子查询
  - [例75] 查询平均成绩最高的学生号

```
select    Sno
 from     SC
 group by   Sno
 having    avg(Grade) >=  all
   （select    avg(Grade)
    from     SC
        group  by   Sno）
```

3.4 数据查询-嵌套查询

- 带有IN谓词的子查询
- 带有比较运算符的子查询
- 带有ANY（SOME）或ALL谓词的子查询
- ***带有EXISTS谓词的子查询***

3.4 数据查询-嵌套查询

◆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

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢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

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例76]查询所有选修了1号课程的学生姓名。

思路分析：

✓本查询涉及Student和SC关系

✓在Student中依次取每个元组的Sno值，用此值去检查SC关系

✓若SC中存在这样的元组，其Sno值等于此Student.Sno值，并且其Cno= '1'，则取此Student.Sname送入结果关系

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例76]查询所有选修了1号课程的学生姓名。

用嵌套查询

```
SELECT Sname
FROM Student
WHERE EXISTS
        (SELECT *
         FROM SC
         WHERE Sno=Student.Sno
                 AND Cno= ' 1 ');
```

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例76]查询所有选修了1号课程的学生姓名。

用连接运算

```
SELECT Sname
FROM Student, SC
WHERE Student.Sno=SC.Sno
        AND SC.Cno= '1';
```

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例77] 查询没有选修1号课程的学生姓名。

```
SELECT Sname
 FROM Student
 WHERE NOT EXISTS
        (SELECT *
         FROM SC
         WHERE Sno = Student.Sno
                AND Cno='1');
```

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

- ➢不同形式的查询间的替换
  - ✓一些带EXISTS或NOT EXISTS谓词的子查询不能被其他形式的子查询等价替换
  - ✓所有带IN谓词、比较运算符、ANY和ALL谓词的子查询都能用带EXISTS谓词的子查询等价替换

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢例：[例67]查询与“刘晨”在同一个系学习的学生。

可以用带EXISTS谓词的子查询替换：

```
SELECT Sno，Sname，Dno
FROM Student S1
WHERE EXISTS
        (SELECT *
         FROM Student S2
         WHERE S2.Dno = S1.Dno
                 AND S2.Sname = '刘晨');
```

用EXISTS谓词代替IN谓词

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

- ➢用EXISTS/NOT EXISTS实现全称量词(难点)
  - ✓SQL语言中没有全称量词∀ （For all）
  - ✓可以把带有全称量词的谓词转换为等价的带有存在量词的谓词： $(\forall x)P \equiv \neg (\exists x(\neg P))$

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例78] 查询选修了全部课程的学生姓名。

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

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

- ➢用EXISTS/NOT EXISTS实现逻辑蕴函(难点)
  - ✓SQL语言中没有蕴函(Implication)逻辑运算
  - ✓可以利用谓词演算将逻辑蕴函谓词等价转换为：$p \rightarrow q \equiv \neg p \vee q$

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例79]查询至少选修了学生200215122选修的全部课程的学生号码。

解题思路：

✓用逻辑蕴函表达：查询学号为x的学生，对所有的课程y，只要200215122学生选修了课程y，则x也选修了y。

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例79]查询至少选修了学生200215122选修的全部课程的学生号码。

解题思路：

✓形式化表示：

用P表示谓词 “学生200215122选修了课程y”

用q表示谓词 “学生x选修了课程y”

则上述查询为: $(\forall y)\ p \rightarrow q$

3.4 数据查询-嵌套查询

◆带有EXISTS谓词的子查询

➢[例79]查询至少选修了学生200215122选修的全部课程的学生号码。

等价变换：

$$(\forall y)p \rightarrow q \equiv \neg (\exists y (\neg(p \rightarrow q)))$$
$$\equiv \neg (\exists y (\neg(\neg p \vee q)))$$
$$\equiv \neg \exists y(p \wedge \neg q)$$

变换后语义：不存在这样的课程y，学生200215122选修了y，而学生x没有选。

3.4 数据查询-嵌套查询

用NOT EXISTS谓词表示：

```
SELECT DISTINCT Sno
FROM SC SCX
WHERE NOT EXISTS
        (SELECT *
         FROM SC SCY
         WHERE SCY.Sno = ' 200215122 '  AND
                NOT EXISTS
                (SELECT *
                 FROM SC SCZ
                 WHERE SCZ.Sno=SCX.Sno AND
                        SCZ.Cno=SCY.Cno));
```

3.4 数据查询

- 单表查询
- 连接查询
- 嵌套查询
- *集合查询*
- Select语句的一般形式

3.4 数据查询-集合查询

- 集合操作的种类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT
- 参加集合操作的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

3.4 数据查询-集合查询

[例81]查询CS系的学生及年龄不大于19岁的学生

方法一：

```
SELECT *
FROM Student
WHERE Dno= 'CS'
UNION
SELECT *
FROM Student
WHERE Sage<=19;
```

3.4 数据查询-集合查询

- 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

3.4 数据查询-集合查询

[例81]查询CS系的学生及年龄不大于19岁的学生

方法二：

```
SELECT  DISTINCT  *
 FROM Student
 WHERE Dno= '1'  OR  Sage<=19;
```

3.4 数据查询-集合查询

[例82] 查询选修了课程1或者选修了课程2的学生。

```
SELECT Sno
 FROM SC
WHERE Cno=' 1 '
UNION
SELECT Sno
FROM SC
WHERE Cno= ' 2 ';
```

3.4 数据查询-集合查询

[例83] 查询CS系的学生与年龄不大于19岁的学生的交集

```
SELECT *
FROM Student
WHERE Dno='CS'
INTERSECT
SELECT *
FROM Student
WHERE Sage<=19
```

3.4 数据查询-集合查询

[例83] 查询CS系的学生与年龄不大于19岁的学生的交集

实际上就是查询CS系中年龄不大于19岁的学生。

```
SELECT *
FROM Student
WHERE Dno= 'CS'  AND  Sage<=19;
```

3.4 数据查询-集合查询

[例84] 查询既选修了课程1又选修了课程2的学生。
选修课程1的学生集合与选修课程2的学生集合的交集

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
 INTERSECT
SELECT Sno
FROM SC
WHERE Cno='2 ';
```

3.4 数据查询-集合查询

[例84] 查询选修课程1的学生集合与选修课程2的学生集合的交集

实际上是查询既选修了课程1又选修了课程2的学生

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
        AND Sno IN
                (SELECT Sno
                 FROM SC
                 WHERE Cno=' 2 ')
```

3.4 数据查询-集合查询

[例85] 查询CS系的学生与年龄不大于19岁的学生的差集。

```
SELECT *
FROM Student
WHERE Dno='CS'
EXCEPT
SELECT  *
FROM Student
WHERE Sage <=19;
```

3.4 数据查询-集合查询

[例85] 查询CS系的学生与年龄不大于19岁的学生的差集。

实际上是查询CS系中年龄大于19岁的学生

```
SELECT *
FROM Student
WHERE Dno= '1' AND  Sage>19;
```

3.4 数据查询-集合查询

[例86]求选修了1或2号而没有选3号课程的学生号。

```
(select    Sno
 from      SC
 where     Cno = '1'  or  Cno = '2' )
 except
 (select    Sno
 from      SC
 where     Cno = '3')
```

3.4 数据查询

- ◆单表查询
- ◆连接查询
- ◆嵌套查询
- ◆集合查询
- ◆*Select语句的一般形式*

3.4 数据查询- SELECT语句的一般格式

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]] …

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]]

3.4 数据查询-SELECT语句的一般格式

◆目标列表达式格式

- [<表名>.] *
- COUNT ([DISTINCT|ALL] *)
- [<表名>.]<属性列名表达式>[，[<表名>.]<属性列名表达式>] …
  - ✓其中<属性列名表达式>可以是由***属性列***、作用于属性列的***函数***和***常量***的任意算术运算（+，-，*，/）组成的运算公式。

3.4 数据查询-SELECT语句的一般格式

◆聚集函数的一般格式为

{ COUNT
  SUM
  AVG
  MAX
  MIN } ([DISTINCT | ALL] <列名>)

3.4 数据查询-SELECT语句的一般格式

◆WHERE子句的条件表达式有以下可选格式

(1)

<属性列名> θ { <属性列名> | <常量> | [ANY|ALL] (SELECT语句) }

3.4 数据查询-SELECT语句的一般格式

◆WHERE子句的条件表达式有以下可选格式

(2)

<属性列名> [NOT] BETWEEN

{ <属性列名> | <常量> | (SELECT语句) } AND { <属性列名> | <常量> | (SELECT语句) }

3.4 数据查询-SELECT语句的一般格式

◆WHERE子句的条件表达式有以下可选格式

（3）

<属性列名> [NOT] IN { (<值1>[，<值2>] …) | (SELECT语句) }

(4) <属性列名> [NOT] LIKE <匹配串>

(5) <属性列名> IS [NOT] NULL

(6) [NOT] EXISTS (SELECT语句)

3.4 数据查询-SELECT语句的一般格式

◆WHERE子句的条件表达式有以下可选格式

(7)

<条件表达式> { AND / OR } <条件表达式> { AND / OR } <条件表达式> …