

数据库系统原理与设计

(第 3 版)

人脑不是一个可以灌注知识的容器，而是一个可以点燃的火把。

——古希腊生物学家、散文家：普多塔戈

数据库系统原理与设计

(第3版)

第3章 SQL查询语言

第3章 SQL查询语言

■ 学习目标

- SQL(Structured Query Language, 结构化查询语言)是关系数据库的标准语言
- 几乎所有的关系型数据库管理系统均采用SQL语言标准
- 教学目标主要有两个：
 - 要求读者掌握对数据库的基本操作，并了解数据库管理系统的基本功能
 - 要求读者熟练掌握SQL查询语句，并运用SQL语句完成对数据库的操作

第3章 SQL语言

■ 学习方法

- 要求读者结合课堂讲授的知识，强化上机实训，通过实训加深对课堂上学过的有关概念和知识点的理解，以便达到融会贯通的学习目标。

■ 学习指南

- 重点：3.2、3.3和3.4节
- 难点：3.4节

第3章 SQL语言

■ 本章导读

- SQL查询语句对关系代数运算做了哪些扩展？
- 连接查询包括哪些？它们分别用于什么地方？
- 在使用分组聚合查询时需要注意的地方。
- 相关子查询与非相关子查询的概念。
- 如何理解存在量词以及存在量词在SQL查询中的重要地位。
- 如何理解查询表的概念，查询表与子查询有何异同点？

目 录



SQL 概述



单表查询



连接查询



嵌套子查询



集合查询



SQL 查询一般格式

3.1 SQL概述

- SQL语言于1974年由Boyce等提出，并于1975~1979年在IBM公司研制的System R数据库管理系统上实现，现已成为国际标准。
- 很多数据库厂商都对SQL语句进行了再开发和扩展
- 标准SQL命令包括：
 - 数据操纵语言DML
 - 查询：SELECT
 - 插入：INSERT
 - 修改：UPDATE
 - 删除：DELETE
 - 数据定义语言DDL
 - 创建对象：CREATE
 - 删除对象：DROP
 - 修改对象：ALTER
 - 数据控制语言DCL
 - 权限授予：GRANT
 - 权限收回：REVOKE
- 可被用来完成几乎所有的数据库操作

3.1 SQL概述

- 3.1.1 SQL发展
- 3.1.2 SQL特点
- 3.1.3 SQL查询基本概念

3.1.1 SQL发展

- **SQL-86**: 第一个SQL标准, 由美国国家标准局 (American National Standard Institute, 简称ANSI) 公布, 1987年国际标准化组织(International Organization for Standardization, 简称ISO)通过。该标准也称为SQL-1
- **SQL-92**: 在1992年, 由ISO和ANSI对SQL-86进行重新修订, 该标准也称为SQL-2
- **SQL-99**: 在1999年, 该版本在SQL-2的基础上, 扩展了诸多功能, 包括递归、触发、面向对象技术等。该标准也称为SQL-3
- **SQL-2003**: 该标准是最新的标准, 也称SQL-4, 于2003年发布

3.1.1 SQL发展

■ SQL语言由4部分组成

● 数据定义语言DDL(Data Definition Language)

- 定义数据库的逻辑结构，包括数据库、基本表、视图和索引等，扩展DDL还支持存储过程、函数、对象、触发器等定义
- DDL包括3类语言，即定义、修改和删除

● 数据操纵语言DML(Data Manipulation Language)

- 对数据库的数据进行检索和更新，其中更新操作包括插入、删除和修改数据

● 数据控制语言DCL(Data Control Language)

- 对数据库的对象进行授权、用户维护(包括创建、修改和删除)、完整性规则定义和事务定义等

● 其它

- 主要是嵌入式SQL语言和动态SQL语言的定义，规定了SQL语言在宿主语言中使用的规则
- 扩展SQL还包括数据库数据的重新组织、备份与恢复等功能

3.1.2 SQL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的功能于一体

■ 高度非过程化：描述做什么，不涉及怎么做。

■ 面向集合的操作方式

- 采用集合操作方式，其操作对象、操作结果都是元组的集合

■ 同一种语法结构提供两种使用方式

- SQL语言既是自含式语言，又是嵌入式语言。在两种不同的使用方式下，其语法结构基本上是一致的

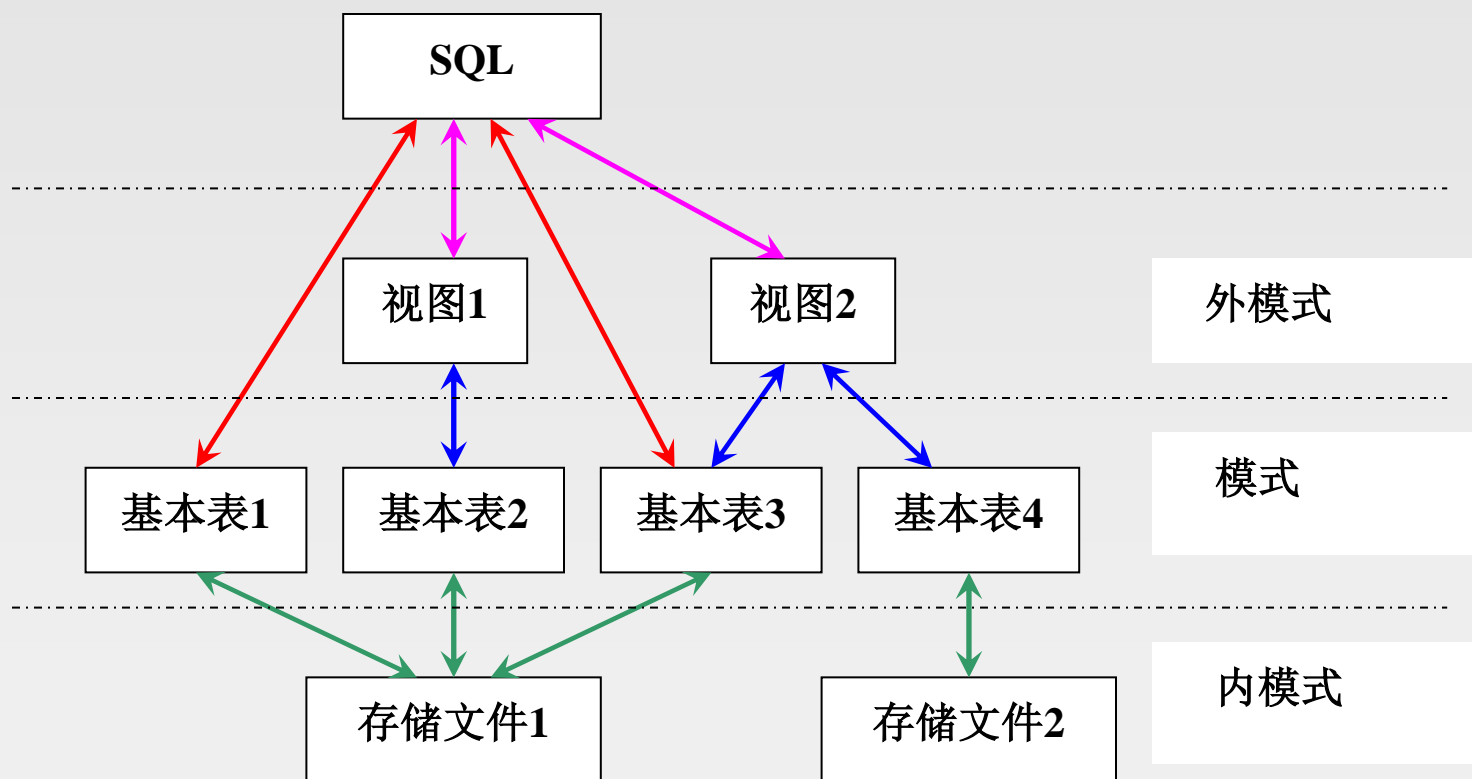
■ 语言简洁，易学易用

- SQL语言的动词非常少，主要包括：

- 数据查询：SELECT；
- 数据更新：INSERT、UPDATE、DELETE；
- 数据定义：CREATE、DROP、ALTER；
- 数据控制：GRANT、REVOKE

3.1.3 SQL查询基本概念

- SQL语言支持三级模式结构，外模式对应视图和部分基本表，模式对应基本表，内模式对应存储文件



3.1.3 SQL查询基本概念

■ 基本表

- 数据库中独立存在的表称为基本表
- 在SQL中一个关系对应一个基本表
- 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
- 索引存放在存储文件中

■ 视图

- 指从一个或几个基本表(或视图)导出的表，是虚表
- 只存放视图的定义而不存放对应数据

■ 查询表

- 指查询结果对应的表

■ 存储文件

- 指数据库中存放关系的物理文件

目 录



SQL 概述



单表查询



连接查询



嵌套子查询



集合查询



SQL 查询一般格式

3.2 单表查询

- 本章所用的数据库为学生成绩管理数据库ScoreDB，其数据库模式如图3-2～图3-6所示，关系数据如图3-8～图3-12所示

班级编号	班级名称	所属学院	年级	班级人数
<u>classNo</u>	className	institute	grade	classNum
char(6)	varchar(30)	varchar(30)	smallint	tinyint

图 3-2 班级表 Class

	<u>classNo</u>	className	institute	grade	classNum
1	CP1601	注册会计 16_01 班	会计学院	2016	NULL
2	CP1602	注册会计 16_02 班	会计学院	2016	NULL
3	CP1603	注册会计 16_03 班	会计学院	2016	NULL
4	CS1501	计算机科学与技术 15-01 班	信息管理学院	2015	NULL
5	CS1502	计算机科学与技术 15-02 班	信息管理学院	2015	NULL
6	CS1601	计算机科学与技术 16-01 班	信息管理学院	2016	NULL
7	ER1501	金融管理 15-01 班	金融学院	2015	NULL
8	IS1501	信息管理与信息系统 15-01 班	信息管理学院	2015	NULL
9	IS1601	信息管理与信息系统 16-01 班	信息管理学院	2016	NULL

图 3-8 班级表 Class 的数据

3.2 单表查询

学号	姓名	性别	出生日期	籍贯	民族	所属班级
<u>studentNo</u>	studentName	sex	birthday	native	nation	<i>classNo</i>
char(7)	varchar(20)	char(2)	datetime	varchar(20)	varchar(30)	char(6)

图 3-3 学生表 Student

课程号	课程名	学分	课时数	先修课程
<u>courseNo</u>	courseName	creditHour	courseHour	<i>priorCourse</i>
char(3)	varchar(30)	numeric	int	char(3)

图 3-4 课程表 Course

学期号	学期描述	备注
<u>termNo</u>	termName	remarks
char(3)	varchar(30)	varchar(10)

图 3-5 学期表 Term

学号	课程号	学期号	成绩
<u>studentNo</u>	<u>courseNo</u>	<u>termNo</u>	score
char(7)	char(3)	char(3)	numeric

图 3-6 成绩表 Score

3.2 单表查询

学号		<u>courseNo</u>	courseName	creditHour	courseHour	priorCourse
<u>student</u>	1	001	大学语文	2	32	NULL
char(7)	2	002	体育	2	32	NULL
	3	003	大学英语	3	48	NULL
课程号	4	004	高等数学	6	96	NULL
<u>courseNo</u>	5	005	C 语言程序设计	4	80	004

	<u>termNo</u>	termName	remarks
1	151	2015-2016 学年第一学期	NULL
2	152	2015-2016 学年第二学期	NULL
3	153	2015-2016 学年第三学期	小学期
4	161	2016-2017 学年第一学期	NULL
5	162	2016-2017 学年第二学期	NULL
6	163	2016-2017 学年第三学期	小学期

005
005
007
008
004
010

图 3-11 学期表 Term 的数据

	<u>studentNo</u>	studentName	sex	birthday	native	nation	classNo
1	1500001	李小勇	男	1998-12-21	南昌	汉族	CS1501
2	1500002	刘方晨	女	1998-11-11	九江	汉族	IS1501
3	1500003	王红敏	女	1997-10-01	上海	汉族	IS1501
4	1500004	张可立	男	1999-05-20	南昌	蒙古族	CS1501
5	1500005	王红	男	2000-04-26	南昌	蒙古族	CS1502
6	1600001	李勇	男	1998-12-21	南昌	汉族	CS1601
7	1600002	刘晨	女	1998-11-11	九江	汉族	IS1601
8	1600003	王敏	女	1998-10-01	上海	汉族	IS1601
9	1600004	张立	男	1999-05-20	南昌	蒙古族	CS1601
10	1600005	王红	男	1999-04-26	南昌	蒙古族	CP1602
11	1600006	李志强	男	1999-12-21	北京	汉族	CP1602
12	1600007	李立	女	1999-08-21	福建	畬族	IS1601
13	1600008	黄小红	女	1999-08-09	云南	傣族	CS1601
14	1600009	黄勇	男	1999-11-21	九江	汉族	CP1602
15	1600010	李宏冰	女	1998-03-09	上海	汉族	CP1602
16	1600011	江宏吕	男	1998-12-20	上海	汉族	CP1602
17	1600012	王立红	男	1998-11-18	北京	汉族	CS1601
18	1600013	刘小华	女	1999-07-16	云南	哈呢族	IS1601
19	1600014	刘宏昊	男	1999-09-16	福建	汉族	IS1601
20	1600015	吴敏	女	1997-01-20	福建	畬族	CP1602

图 3-10 学生表 Student 的数据

学号	姓名
<u>studentNo</u>	studentName
char(7)	varchar(20)

课程号	课程名	学分
<u>courseNo</u>	courseName	credits
char(3)	varchar(30)	numeric(4,1)

图 3-4 设计

	<u>studentNo</u>	<u>courseNo</u>	<u>termNo</u>	score		<u>studentNo</u>	<u>courseNo</u>	<u>termNo</u>	score
1	1500001	001	151	98	41	1500005	011	162	68
2	1500001	002	151	82	42	1600002	001	161	98
3	1500001	003	161	82	43	1600002	002	161	46
4	1500001	004	151	56	44	1600002	003	162	98
5	1500001	004	161	86	45	1600002	004	161	60
6	1500001	005	152	77	46	1600002	005	162	86
7	1500001	006	152	76	47	1600002	010	162	70
8	1500001	007	152	77	48	1600003	001	161	70
9	1500001	008	161	82	49	1600003	002	161	60
10	1500001	009	162	77	50	1600003	004	161	77
11	1500001	010	151	86	51	1600003	005	162	87
12	1500003	001	151	46	52	1600004	001	161	50
13	1500003	002	151	38	53	1600004	001	162	68
14	1500003	002	152	58	54	1600004	002	161	70
15	1500003	005	151	60	55	1600004	003	162	88
16	1500003	006	161	70	56	1600004	004	161	78
17	1500003	007	152	50	57	1600004	010	161	89
18	1500003	007	162	66	58	1600004	011	162	90
19	1500003	008	162	82	59	1600005	001	161	82
20	1500003	009	162	78	60	1600005	002	161	80
21	1500003	010	161	90	61	1600005	003	162	82
22	1500004	001	151	48	62	1600005	004	161	47
23	1500004	001	162	70	63	1600005	010	161	90
24	1500004	002	161	68	64	1600005	011	162	82
25	1500004	003	152	70	65	1600012	001	161	68
26	1500004	004	151	58	66	1600012	002	162	78
27	1500004	005	162	88	67	1600012	003	161	76
28	1500004	006	162	72	68	1600012	004	161	70
29	1500004	007	161	71	69	1600012	005	161	88
30	1500004	008	161	80	70	1600012	006	162	82
31	1500005	001	152	79	71	1600012	007	162	90
32	1500005	002	151	80	72	1600012	010	162	84
33	1500005	003	151	69	73	1600014	001	161	60
34	1500005	004	151	87	74	1600014	002	162	69
35	1500005	005	151	77	75	1600014	003	161	87
36	1500005	006	152	69	76	1600014	004	161	45
37	1500005	007	161	90	77	1600014	004	162	88
38	1500005	008	161	87	78	1600014	005	162	56
39	1500005	009	162	90	79	1600014	010	161	90
40	1500005	010	152	69	80	1600014	011	162	70

图 3-12 成绩管理表 Score 的数据

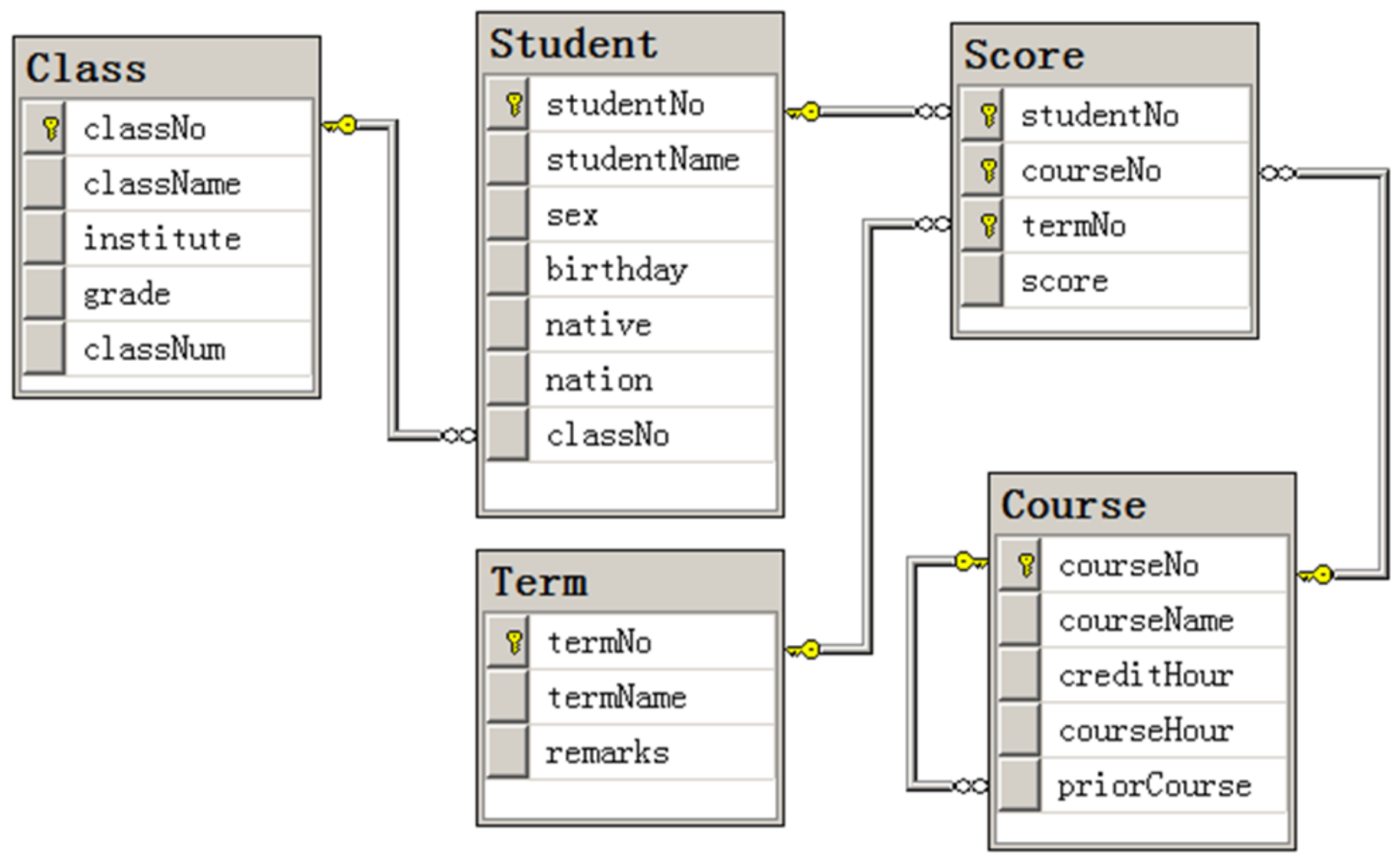


图 3-7 学生成绩管理数据库模式导航图

3.2 单表查询

- 3.2.1 投影运算
- 3.2.2 选择运算
- 3.2.3 排序运算
- 3.2.4 查询表
- 3.2.5 聚合查询

3.2.1 投影运算

■ SQL基本结构包括3个子句：

● SELECT子句

➤ 对应**投影运算**，指定查询结果中所需要的属性或**表达式**

● FROM子句

➤ 对应**笛卡尔积**，给出查询所涉及的表，表可以是基本表、视图或**查询表**

● WHERE子句

➤ 对应**选择运算**（包括连接运算所转化的选择运算），指定查询结果元组所需要满足的选择条件

■ SELECT和FROM是必须的，其他是可选的

3.2.1 投影运算

■ 基本语法为：

```
SELECT  $A_1, A_2, \dots, A_n$   
FROM  $R_1, R_2, \dots, R_m$   
WHERE  $P$ 
```

- A_1, A_2, \dots, A_n 代表需要查找的属性或表达式
- R_1, R_2, \dots, R_m 代表查询所涉及的表
- P 代表谓词(即选择条件)，如果省略WHERE子句，表示 P 为真
- SQL的查询结果中允许包含重复元组

■ SQL执行过程（逻辑上的理解）：

- 首先对 R_1, R_2, \dots, R_m 执行笛卡尔积
- 然后在笛卡尔积中选择使得谓词 P 为真的记录
- 再在 A_1, A_2, \dots, A_n 属性列中进行投影运算，不消除重复元组
 - 如需消除重复元组，必须使用关键字DISTINCT

3.2.1 投影运算

■ 基本语法为：

```
SELECT  $A_1, A_2, \dots, A_n$   
FROM  $R_1, R_2, \dots, R_m$   
WHERE  $P$ 
```

- 刚才描述的SQL查询执行过程只是**逻辑上**的，在具体执行时会进行**优化处理**，查询优化的内容详见第7章。

■ SQL执行过程（**逻辑上的理解**）：

- 首先对 R_1, R_2, \dots, R_m 执行笛卡尔积
- 然后在笛卡尔积中选择使得谓词 P 为真的记录
- 再在 A_1, A_2, \dots, A_n 属性列中进行投影运算，不消除重复元组
 - 如需消除重复元组，必须使用关键字**DISTINCT**

3.2.1 投影运算

■ 查询指定列

- 选取表中的全部列或指定列，通过SELECT确定要查询的属性
- [例3.1] 查询所有班级的班级编号、班级名称和所属学院

SELECT classNo, className, institute

FROM Class

➤ 该查询的执行过程是：

- ✓ 从Class表中依次取出每个元组
- ✓ 对每个元组仅选取classNo、className和institute三个属性的值，形成一个新元组
- ✓ 最后将这些新元组组织为一个结果关系输出
- ✓ 该查询的结果如图3-13所示

3.2.1 投影运算

■ 查询指定列

- 选取表中的全部列或指定列，通过SELECT确定要查询的属性
- [例3.1] 查询所有班级的班级编号、班级名称和所属学院

SELECT classNo, className, institute

FROM Class

➤ 该查询的执行过程是

- ✓ 从Class表中依次取
- ✓ 对每个元组仅选取值，形成一个新元
- ✓ 最后将这些新元组
- ✓ 该查询的结果如图

	classNo	className	institute
1	CP1601	注册会计师16_01班	会计学院
2	CP1602	注册会计师16_02班	会计学院
3	CP1603	注册会计师16_03班	会计学院
4	CS1501	计算机科学与技术15-01班	信息管理学院
5	CS1502	计算机科学与技术15-02班	信息管理学院
6	CS1601	计算机科学与技术16-01班	信息管理学院
7	ER1501	金融管理15-01班	金融学院
8	IS1501	信息管理与信息系统15-01班	信息管理学院
9	IS1601	信息管理与信息系统16-01班	信息管理学院

3.2.1 投影运算

■ 消除重复元组

- 需要消除重复元组，使用**DISTINCT**关键字
- [例3.2] 查询所有学院的名称。

SELECT institute

FROM Class

✓ 上述查询不消除重复元组，其查询结果如图3-14所示

- 消除重复元组，查询结果如图3-15所示

SELECT DISTINCT institute

FROM Class

	institute
1	会计学院
2	会计学院
3	会计学院
4	信息管理学院
5	信息管理学院
6	信息管理学院
7	金融学院
8	信息管理学院
9	信息管理学院

图 3-14

	institute
1	会计学院
2	金融学院
3	信息管理学院

图 3-15

3.2.1 投影运算

■ 查询所有列

- 可使用两种方法:

- 将所有的列在SELECT子句中列出(可以改变列的显示顺序);
- 使用*符号, *表示所有属性, 按照表定义时的顺序显示所有属性

- [例3.3] 查询所有班级的全部信息。

```
SELECT classNo, className, classNum, grade, institute  
FROM Class
```

- 或

```
SELECT *  
FROM Class
```

3.2.1 投影运算

■ 给属性列取别名

- 可为属性列取一个便于理解的列名，如用中文来显示列名
- 为属性列取别名特别适合经过计算的列
- [例3.4] 查询所有班级的所属学院、班级编号和班级名称，要求用中文显示列名

```
SELECT institute 所属学院, classNo 班级编号,  
        className 班级名称  
FROM Class
```

- 查询结果如图3-16所示。该查询可使用AS关键字取别名：

```
SELECT institute AS 所属学院, classNo AS 班级编号,  
        className AS 班级名称  
FROM Class
```

3.2.

■ 给属性列取别名

- 可为属性列取一个便于记忆的别名
- 为属性列取别名特别适用于多表连接查询
- [例3.4] 查询所有班级要求用中文显示列名

	所属学院	班级编号	班级名称
1	会计学院	CP1601	注册会计师16_01班
2	会计学院	CP1602	注册会计师16_02班
3	会计学院	CP1603	注册会计师16_03班
4	信息管理学院	CS1501	计算机科学与技术15-01班
5	信息管理学院	CS1502	计算机科学与技术15-02班
6	信息管理学院	CS1601	计算机科学与技术16-01班
7	金融学院	ER1501	金融管理15-01班
8	信息管理学院	IS1501	信息管理与信息系统15-01班
9	信息管理学院	IS1601	信息管理与信息系统16-01班

SELECT institute 所属学院, classNo 班级编号,
 className 班级名称

FROM Class

- 查询结果如图3-16所示。该查询可使用AS关键字取别名：

SELECT institute AS 所属学院, classNo AS 班级编号,
 className AS 班级名称

FROM Class

3.2.1 投影运算

■ 查询经过计算的列

- 可使用属性、常数、函数和表达式
- [例3.5]查询每门课程的课程号、课程名以及周课时(周课时为课时数除以16), 并将课程名中大写字母改为小写字母输出。

```
SELECT courseNo 课程号, lower(courseName) AS 周课时  
FROM Course
```

- 函数lower()将大写字母改为小写字母
- 查询结果如图3-17所示

	课程号	课程名	周课时
1	001	大学语文	2
2	002	体育	2
3	003	大学英语	3
4	004	高等数学	6
5	005	c语言程序设计	5
6	006	计算机原理	4
7	007	数据结构	6
8	008	操作系统	4
9	009	数据库系统原理	5
10	010	会计学原理	4
11	011	中级财务会计	5

3.2.2 选择运算

■ WHERE子句可实现关系代数中的选择运算

■ WHERE常用的查询条件有：

- 比较运算：>、>=、<、<=、=、<>(或!=)
- 范围查询：[NOT] BETWEEN <值1> AND <值2> []
- 集合查询：[NOT] IN <集合>
- 空值查询：IS [NOT] null
- 字符匹配查询：[NOT] LIKE <匹配字符串>
- 逻辑查询：AND、OR、NOT

3.2.2 选择运算

■ 比较运算

- 使用比较运算符>、>=、<、<=、=、<>(或!=)
- [例3.6] 查询2015级的班级编号、班级名称和所属学院。

SELECT classNo, className, institute

FROM Class

WHERE grade=2007

该选择运算执行
顺序是？

- 其查询结果如图3-18所示

	classNo	className	institute
1	CS1501	计算机科学与技术15-01班	信息管理学院
2	CS1502	计算机科学与技术15-02班	信息管理学院
3	ER1501	金融管理15-01班	金融学院
4	IS1501	信息管理与信息系统15-01班	信息管理学院

3.2.2 选择运算

- 该查询的执行过程可能有多种方法：

- 全表扫描法

- ✓ 依次取出Class表中的每个元组
- ✓ 判断该元组的grade属性值是否等于2015
- ✓ 若是则将该元组的班级编号、班级名称和所属学院属性取出，形成一个新元组
- ✓ 最后将所有新元组组织为一个结果关系输出
- ✓ 该方法适用于小表，或者该表未在grade属性列上建索引

- 索引搜索法

- ✓ 如果该表在grade属性列上建有索引，且满足条件的记录不多，则可使用索引搜索法来检索数据
- 具体使用何种方法由数据库管理系统的查询优化器来选择，详见第8章内容

3.2.2 选择运算

- [例3.7] 在学生Student表中查询年龄大于或等于19岁的同学学号、姓名和出生日期。

```
SELECT studentNo, studentName, birthday
```

```
FROM Student
```

```
WHERE year(getdate()) - year(birthday) >= 19
```

➤ 函数getdate()获取当前系统的日期

➤ 函数year()提取日期中的年份

■ WHERE子句可实现关系代数中的选择运算

■ WHERE常用的查询条件有：

- 比较运算：>、>=、<、<=、=、<>(或!=)
- 范围查询：[NOT] BETWEEN <值1> AND <值2>
- 集合查询：[NOT] IN <集合>
- 空值查询：IS [NOT] null
- 字符匹配查询：[NOT] LIKE <匹配字符串>
- 逻辑查询：AND、OR、NOT

3.2.2 选择运算

■ 范围查询

- **BETWEEN...AND**用于查询属性值在某一个范围内的元组
- **NOT BETWEEN...AND**用于查询属性值不在某一个范围内的元组
- **BETWEEN**后是属性的下限值，**AND**后是属性的上限值
- [例3.8] 在选课Score表中查询成绩在80~90分之间的同学学号、课程号和相应成绩

SELECT studentNo, courseNo, score

FROM Score

WHERE score **BETWEEN** 80 **AND** 90

➤该查询也可以使用**逻辑运算AND**实现，见例3.22

3.2.2 选择运算

- [例3.9] 在选课Score表中查询成绩不在80~90分之间的同学学号、课程号和相应成绩。

SELECT studentNo, courseNo, score

FROM Score

WHERE score NOT BETWEEN 80 AND 90

- 该查询也可以使用逻辑运算OR实现，见例3.23

3.2.2 选择运算

■ 集合查询

- **IN**用于查询属性值在某个集合内的元组
- **NOT IN**用于查询属性值不在某个集合内的元组
- **IN**后面是集合，可以是具体的集合，也可以是**查询出来的元组集合**(该部分内容详见3.4节的内容)。
- [例3.10] 在选课Score表中查询选修了“001”、“005”或“003”课程的同学学号、课程号和相应成绩。

```
SELECT studentNo, courseNo, score
```

```
FROM Score
```

```
WHERE courseNo IN ('001', '005', '003')
```

➤该查询也可以使用**逻辑运算OR**实现，见例3.19

3.2.2 选择运算

- [例3.11] 在学生 Student表中查询籍贯不是“南昌”或“上海”的同学姓名、籍贯和所属班级编号。

SELECT studentName, native, classNo

FROM Student

WHERE native **NOT IN** ('南昌', '上海')

- 该查询也可以使用逻辑运算**AND**实现，见例3.21

3.2.2 选择运算

■ 空值查询

- 空值表示未知或不确定的值，空值表示为null
- IS null用于查询属性值为空值
- IS NOT null用于查询属性值不为空值
- IS不能用“=”替代
- [例3.12] 在课程Course表中查询先修课程为空值的课程信息.

is null
is not null

```
SELECT *  
FROM Course  
WHERE priorCourse IS NULL
```

	courseNo	courseName	creditHour	courseHour	priorCourse
1	001	大学语文	2	32	NULL
2	002	体育	2	32	NULL
3	003	大学英语	3	48	NULL
4	004	高等数学	6	96	NULL

3.2.2 选择运算

- [例3.13] 在课程Course表中查询有先修课程的课程信息。

SELECT *

FROM Course

WHERE priorCourse IS NOT NULL

3.2.2 选择运算

■ 字符匹配查询

- LIKE用于字符匹配查询，语法格式为：
 - [NOT] LIKE <匹配字符串> [ESCAPE <换码字符>]
- 查询的含义是：
 - 如果在LIKE前没有NOT，则查询指定的属性列值与<匹配字符串>相匹配的元组；
 - 如果在LIKE前有NOT，则查询指定的属性列值不与<匹配字符串>相匹配的元组。
 - <匹配字符串>可以是一个具体的字符串，也可以包括通配符%和_
 - ✓ %表示任意长度的字符串
 - » ab%，表示所有以ab开头的任意长度的字符串；
 - » zhang%ab，表示以zhang开头，以ab结束，中间可以是任意个字符的字符串。
 - ✓ 符号_(下划线)表示任意一个字符
 - » ab_，表示所有以 ab开头的3个字符的字符串，其中第3个字符为任意字符；
 - » a__b表示所有以a开头，以b 结束的4个字符的字符串，且第2、3个字符为任意字符。

3.2.2 选择运算

- [例3.14] 在班级Class表中查询班级名称中含有会计的班级信息

```
SELECT *  
FROM Class
```

```
WHERE className LIKE '%会计%'
```

	classNo	className	institute	grade	classNum
1	CP1601	注册会计师16_01班	会计学院	2016	NULL
2	CP1602	注册会计师16_02班	会计学院	2016	NULL
3	CP1603	注册会计师16_03班	会计学院	2016	NULL

➤注意：匹配字符串必须用一对引号括起来

- [例3.15] 在学生Student表中查询所有姓王且全名为3个汉字的同学学号和姓名

```
SELECT studentNo, studentName  
FROM Student  
WHERE studentName LIKE '王__'
```

➤注意：在中文SQL-Server中，如果匹配字符串为汉字，则一个下划线代表一个汉字；如果是西文，则一个下划线代表一个字符。

3.2.2 选择运算

- [例3.16] 在学生Student表中查询名字中不含有“福”的同学学号和姓名。

SELECT studentNo, studentName

FROM Student

WHERE studentName **NOT LIKE** '%福%'

3.2.2 选择运算

- [例3.17] 在学生Student表中查询蒙古族的同学学号和姓名

```
SELECT studentNo, studentName
```

```
FROM Student
```

```
WHERE nation LIKE '蒙古族'
```

➤注意：如果匹配字符串中不含有%和_，则LIKE与比较运算符“=”的查询结果一样

➤该查询等价于下面的查询：

```
SELECT studentNo, studentName
```

```
FROM Student
```

```
WHERE nation='蒙古族'
```

3.2.2 选择运算

- 如果查询字符串中本身要包含%和_，必须使用“**ESCAPE** <换码字符>”短语，对通配符进行转义处理。

- [例3.18] 在班级Class表中查询班级名称中含有“16_”符号的班级名称

SELECT className

FROM Class

WHERE className **LIKE** '%16_%' **ESCAPE** '\'

	className
1	注册会计师16_01班
2	注册会计师16_02班
3	注册会计师16_03班

➤ “**ESCAPE** ‘\’”表示\为换码字符

➤ 紧跟在\符号后的_不是通配符，而是普通的用户要查询的符号

3.2.2 选择运算

➤如果将#字符作为换码字符，则该查询可改写为：

SELECT className

FROM Class

WHERE className **LIKE** '%16#_%' **ESCAPE** '#'

	className
1	注册会计师16_01班
2	注册会计师16_02班
3	注册会计师16_03班

3.2.2 选择运算

■ 逻辑查询

- SQL提供AND、OR和NOT逻辑运算符分别实现逻辑与、逻辑或和逻辑非运算
- [例3.19] 在选课Score表中查询选修了“001”、“005”或“003”课程的同学学号、课程号和相应成绩

```
SELECT studentNo, courseNo, score
```

```
FROM Score
```

```
WHERE courseNo='001' OR courseNo='005' OR courseNo='003'
```

- 在例3.10中使用的是集合运算，本例中采用逻辑“或”运算

3.2.2 选择运算

- [例3.20] 在Student表中查询1998年出生且民族为“汉族”的同学学号、姓名、出生日期。

```
SELECT studentNo, studentName, birthday
FROM Student
WHERE year(birthday)=1998 AND nation='汉族'
```

- **注意：**在逻辑运算中，不可以对同一个属性进行逻辑“与”的等值运算

- 如在选课Score表中查询同时选修了“001”和“002”课程的同学的选课信息，**如下查询是错误的，得不到结果：**

```
SELECT *
FROM Score
WHERE courseNo='001' AND courseNo='002'
```

- 要实现该查询，需要使用**连接运算**或**嵌套子查询**
- 通过连接运算表示该查询，参见例3.34、例3.37
- 通过嵌套子查询，参见例3.45、例3.46

studentNo	courseNo	termNo	score
1500001	001	151	98.0
1500001	002	151	82.0
1500001	003	161	82.0
1500001	004	151	56.0
1500001	004	161	86.0
1500001	005	152	77.0
1500001	006	152	76.0
1500001	007	152	77.0
1500001	008	161	82.0
1500001	009	162	77.0
1500001	010	151	86.0
1500003	001	151	46.0
1500003	002	151	38.0
1500003	002	152	58.0

3.2.2 选择运算

- [例3.21] 在Student表中查询籍贯既不是“南昌”也不是“上海”的同学姓名、籍贯和所属班级编号。

```
SELECT studentName, native, classNo
```

```
FROM Student
```

```
WHERE native!='南昌' AND native!='上海'
```

- [例3.22] 在选课Score表中查询成绩在80～90分之间的同学学号、课程号和相应成绩。

```
SELECT studentNo, courseNo, score
```

```
FROM Score
```

```
WHERE score>= 80 AND score<=90
```

3.2.2 选择运算

- [例3.23] 在选课Score表中查询成绩不在80~90分之间的同学学号、课程号和相应成绩。

SELECT studentNo, courseNo, score

FROM Score

WHERE score<80 **OR** score>90

小结

- 投影，选择查询（比较运算,范围查询,集合查询,空值查询,字符匹配查询,逻辑查询）
- 查询次序：
 - Selet
 - From
 - where

3.2.3 排序运算

■ 使用**ORDER BY** 子句实现排序运算，其语法为：

ORDER BY <表达式1> [**ASC** | **DESC**]

[, <表达式2> [**ASC** | **DESC**] ...]

● 其中：

- <表达式1>, <表达式2>, ... 可以是属性、函数或表达式
- 缺省按升序(**ASC**)排序
- 按降序排序，必须指明**DESC**选项

● 该运算含义是：

- 在查询结果中首先按<表达式1>的值进行排序
- 在<表达式1>值相等的情况下再按<表达式2>值排序
- 依此类推

3.2.3 排序运算

- [例3.24] 在学生 Student表中查询籍贯既不是“南昌”也不是“上海”的同学姓名、籍贯和所属班级编号，并按籍贯的降序排序输出。

SELECT studentName, native, classNo

FROM Student

WHERE native!='南昌'

AND native!='上海'

ORDER BY native **DESC**

	studentName	native	classNo
1	黄小红	云南	CS1601
2	刘小华	云南	IS1601
3	黄勇	九江	CP1602
4	刘方晨	九江	IS1501
5	刘晨	九江	IS1601
6	刘宏昊	福建	IS1601
7	吴敏	福建	CP1602
8	李立	福建	IS1601
9	李志强	北京	CP1602
10	王立红	北京	CS1601

3.2.3 排序运算

- [例3.25] 在学生Student表中查询“女”学生的学号、姓名、所属班级编号和出生日期，并按班级编号的升序、出生日期的月份降序排序输出。

```
SELECT studentNo, studentName, classNo, birthday
FROM Student
WHERE sex='女'
ORDER BY classNo, month(birthday) DESC
```

- 其中：month()函数表示提取日期表达式的月份
- 查询结果如图3-22所示

3.2.3 排序运算

- [例3.25] 在学生Student表中查询“女”学生的学号、姓名、所属班级编号和出生日期，并按班级编号的升序、出生日期的月份降序排序输出。

```
SELECT studentNo, studentName, classNo, birthday
```

```
FROM Student
```

```
WHERE sex='女'
```

```
ORDER BY classNo, month(birthday) DESC
```

➤ 其中：month()函数表示月份

➤ 查询结果如图3-22所示

	studentNo	studentName	classNo	birthday
1	1600010	李宏冰	CP1602	1998-03-09 00:00:00.000
2	1600015	吴敏	CP1602	1997-01-20 00:00:00.000
3	1600008	黄小红	CS1601	1999-08-09 00:00:00.000
4	1500002	刘方晨	IS1501	1998-11-11 00:00:00.000
5	1500003	王红敏	IS1501	1997-10-01 00:00:00.000
6	1600002	刘晨	IS1601	1998-11-11 00:00:00.000
7	1600003	王敏	IS1601	1998-10-01 00:00:00.000
8	1600007	李立	IS1601	1999-08-21 00:00:00.000
9	1600013	刘小华	IS1601	1999-07-16 00:00:00.000

3.2 简单查询

■ 3.2.1 投影运算

■ 3.2.2 选择运算

■ 3.2.3 排序运算

ORDER BY <表达式1> [ASC | DESC]
[, <表达式2> [ASC | DESC], ...]

■ 3.2.4 查询表

3.2.4 查询表

■ FROM子句后面可以是基本关系、视图，还可以是查询表

- [例3.26] 查询1999年出生的“女”同学基本信息。
- 分析：可以先将学生表中的女生记录查询出来，然后再对查询表进行选择、投影操作。

```
SELECT studentNo, studentName, birthday
FROM (SELECT * FROM Student WHERE sex='女') AS a
WHERE year(birthday)=1999
```

- 在FROM子句后是一个子查询，表示对子查询的查询结果——查询表进行查询
- 必须为查询表取一个名称(称为元组变量)，如使用AS *a*取名为*a*

```
FROM (SELECT * FROM Student WHERE sex='女') a
```

- 该查询等价于下面的查询：

```
SELECT studentNo, studentName, birthday
FROM student
WHERE year(birthday)=1999 AND sex='女'
```

3.2.5 聚合查询

■ SQL查询提供了丰富的数据分类、统计和计算的功能

- 统计功能通过聚合函数来实现
- 分类功能通过分组子句来实现
- 统计和分组结合在一起实现丰富的查询功能

1 聚合函数

2 分组聚合

3.2.5 聚合查询

■ SQL提供的**聚合函数**(aggregate function)包括:

- **count**([**DISTINCT** | **ALL**] { * | <列名> }): 统计关系的元组个数或一列中值的个数;
- **sum**([**DISTINCT** | **ALL**] <列名>): 统计一列中值的总和(此列必须为数值型);
- **avg**([**DISTINCT** | **ALL**] <列名>): 统计一列中值的平均值(此列必须为数值型);
- **max**([**DISTINCT** | **ALL**] <列名>): 统计一列中值的最大值;
- **min**([**DISTINCT** | **ALL**] <列名>): 统计一列中值的最小值。

■ 指定**DISTINCT**谓词, 表示在计算时首先消除<列名>取重复值的元组, 然后再进行统计

■ 指定**ALL**谓词或没有**DISTINCT**谓词, 表示不消除<列名>取重复值的元组

3.2.5 聚合查询

■ [例3.27] 查询学生总人数。

```
SELECT count(*)
```

```
FROM Student
```

或 SELECT count(*) 学生人数

```
FROM Student
```

■ [例3.28] 查询所有选课学生的人数。

```
SELECT count(studentNo) 学生人数
```

```
FROM Score
```

➤ 查询结果是80

➤ 由于一个学生可以选修多门课程，学号存入元组，使用DISTINCT短语，将查询修改为

```
SELECT count(DISTINCT studentNo) 学
```

```
FROM Score
```

✓ 查询结果为10

SQLQuery2. s... (SA (54))* WZQ. Sec

```
SELECT *
FROM score
```

	studentNo	courseNo	termNo	score
62	1600005	004	161	47.0
63	1600005	010	161	90.0
64	1600005	011	162	82.0
65	1600012	001	161	68.0
66	1600012	002	162	78.0
67	1600012	003	161	76.0
68	1600012	004	161	70.0
69	1600012	005	161	88.0
70	1600012	006	162	82.0
71	1600012	007	162	90.0
72	1600012	010	162	84.0
73	1600014	001	161	60.0
74	1600014	002	162	69.0
75	1600014	003	161	87.0
76	1600014	004	161	45.0
77	1600014	004	162	88.0
78	1600014	005	162	56.0
79	1600014	010	161	90.0
80	1600014	011	162	70.0


```
SELECT *
FROM stu
```

studentNo	studentName
1500002	刘方晨
1500003	王红敏
1500004	张可立
1500005	王红
1600001	李勇
1600002	刘晨
1600003	王敏
1600004	张立
1600005	王红
1600006	李志强
1600007	李立
1600008	黄小红
1600009	黄勇
1600010	李宏冰
1600011	江宏吕
1600012	王立红
1600013	刘小华
1600014	刘宏昊
1600015	吴敏

3.2.5 聚合查询

- [例3.29] 查询学号为“1500003”同学所选修课程的平均分。

```
SELECT avg(score) 平均分
```

```
FROM Score
```

```
WHERE studentNo='1500003'
```

- 在聚合函数遇到空值时，除count(*)外所有的函数皆跳过空值，只处理非空值。

3.2.5 聚合查询

■ 在SQL查询中，往往需要对数据进行**分类运算**（即**分组运算**）

- 分组运算的目的是为了**细化聚合函数的作用对象**
- 如不对查询结果分组，则聚合函数作用于整个查询结果
- 如对查询结果进行分组，则**聚合函数分别作用于每个组**，查询结果按组聚合输出
- SQL通过**GROUP BY**和**HAVING**子句实现分组运算
 - **GROUP BY**对查询结果按某一系列或某几列进行分组，值相等的分为一组；
 - **HAVING**对分组的结果进行选择，仅输出满足条件的组。
该子句必须与**GROUP BY**子句配合使用

分组和聚集函数

■ 分组命令

group by 列名 [**having** 条件表达式]

group by将表中的元组按指定列上值相等的原则分组，然后在每一分组上使用聚集函数，得到单一值

having则对分组进行选择，只将聚集函数作用到满足条件的分组上

■ Group By子句起作用的时机：

- 在Where子句筛选出元组后，才对它们分组

■ 同时Select子句的作用发生变化：对分组进行统计

- 每个分组在结果中被统计为一个元组
- 在Select子句出现的属性只能是：
 - ① 分组属性
 - ② 聚集函数(任意属性)
 - ③ 由①和②组成的表达式

分组和聚集函数

■ Having子句的作用：在分组后，筛选满足指定条件的分组

- 在分组限定条件中出现的属性只能是以下形式：

- ① 分组属性

- ② 聚集函数(任意属性)

■ Having子句只能配合Group By子句使用，而不能单独出现

■ Having子句中的条件和Where子句中条件的不同

- Where子句中的条件用于限定元组，施加在单个元组上

- Having子句中的条件用于限定Group By子句产生的各个分组，施加在整个分组上

Select 语句的运算次序

■ Select语句结构

Select A_1, A_2, \dots, A_n
From R_1, R_2, \dots, R_m
[*Where* 元组限定条件 P]
[*Group By* 属性1, 属性2, ...
 [*Having* 分组限定条件 Q]]
[*Order By* 属性1 [*asc* | *desc*],
 属性2 [*asc* | *desc*], ...]

■ 运算次序

- From (笛卡儿积) \rightarrow [Where (选择)] \rightarrow [Group By (分组)] \rightarrow [Having (限定分组)] \rightarrow [Select (投影, 或统计)] \rightarrow [Order By (结果排序)]

3.2.5 聚合查询

■ [例3.30] 查询每个同学的选课门数、平均分和最高分。

SELECT studentNo, count(*) 门数, avg(score) 平均分,
max(score) 最高分

FROM Score

GROUP BY studentNo

查询每个同学及格的
选课门数、平均分、
最低分和最高分？

	studentNo	门数	平均分	最高分
1	1500001	11	79.909090	98.0
2	1500003	10	63.800000	90.0
3	1500004	9	69.444444	88.0
4	1500005	11	78.636363	90.0
5	1600002	6	76.333333	98.0
6	1600003	4	73.500000	87.0
7	1600004	7	76.142857	90.0
8	1600005	6	77.166666	90.0
9	1600012	8	79.500000	90.0
10	1600014	8	70.625000	90.0

- 结果按学号StudentNo分组，将具有相同StudentNo值的元组作为一组
- 然后对每组进行相应的计数、求平均值和求最大值

3.2.5 聚合查询

- [例3.31] 查询平均分在75分以上的每个同学的选课门数、平均分和最高分。

```
SELECT StudentNo, count(*) 门数, avg(score) 平均分,  
       max(score) 最高分  
FROM Score  
GROUP BY StudentNo  
HAVING avg(score)>=75
```

- 按学号StudentNo分组，将StudentNo值相同的元组作为一组
- 然后对每组进行计数、求平均值和求最大值
- 并判断平均值是否大于等于75，如果是则输出该组，否则丢弃该组，不作为输出结果

分组和聚集函数

- 列出所有课程都及格的学生
的平均成绩

```
select S#, avg(SCORE)  
from SC  
group by S#  
having min(SCORE) >=  
60
```

```
select S#, avg(SCORE)  
from SC  
where SCORE >=60  
group by S#
```

```
DEPT(D# , DNAME , DEAN)  
S(S# , SNAME , SEX , AGE , D#)  
C(C# , CN , PC# , CREDIT)  
SC(S# , C# , SCORE)  
PROF(P# , PNAME, AGE, D# , SAL)  
PC(P# , C#)
```



哪个正确?

分组和聚集函数

- 列出每一年龄组中男学生（超过50人）的人数

写在作业本上 并注明：

2019.9.17:

```
SQLQuery2. s... (SA (54)) * WZQ.ScoreDB - Diagram_0* SQLQuery1. s... (SA  
select * from student
```

结果 消息

	studentNo	studentName	sex	birthday	native	nation	classNo
1	1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
2	1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
3	1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
4	1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
5	1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
6	1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
7	1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
8	1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601
9	1600004	张立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1601
10	1600005	王红	男	1999-04-26 00:00:00.000	南昌	蒙古族	CP1602
11	1600006	李志强	男	1999-12-21 00:00:00.000	北京	汉族	CP1602
12	1600007	李立	女	1999-08-21 00:00:00.000	福建	畲族	IS1601
13	1600008	黄小红	女	1999-08-09 00:00:00.000	云南	傣族	CS1601
14	1600009	黄勇	男	1999-11-21 00:00:00.000	九江	汉族	CP1602
15	1600010	李宏冰	女	1998-03-09 00:00:00.000	上海	汉族	CP1602

目 录



SQL 概述



单表查询



连接查询



嵌套子查询



集合查询



SQL 查询一般格式

3.3 连接查询

- 在实际应用中，往往会涉及到多个关系的查询，需用到**连接运算**或**子查询**
- 连接运算是关系数据库中使用最广泛的一种运算，包括**等值连接**、**自然连接**、**非等值连接**、**自表连接**和**外连接**等
- 3.3.1 等值与非等值连接
- 3.3.2 自表连接
- 3.3.3 外连接

3.3.1 等值与非等值连接

■ 该运算在WHERE子句中加入连接多个关系的连接条件

■ 格式为:

WHERE [<表1>.<属性名1> <比较运算符> [<表2>.<属性名2>

[<逻辑运算符>

[<表3>.<属性名3> <比较运算符> [<表4>.<属性名4> ...]

■ 比较运算符包括:

- >、>=、<、<=、=、<>(或!=)
- 当比较运算符为=时，表示等值连接
- 其他运算为非等值连接
- WHERE子句的连接谓词中的属性称为连接属性
- 连接属性之间必须具有可比性

classNo	className	institute	grade	classNum
CP1601	注册会计师16_01班	会计学院	2016	NULL
CP1602	注册会计师16_02班	会计学院	2016	NULL
CP1603	注册会计师16_03班	会计学院	2016	NULL
CS1501	计算机科学与技术15-01班	信息管理学院	2015	NULL

studentNo	studentName	sex	birthday	native	nation	classNo
1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601

■ 等值连接

- [例3.32] 查找会计学院全体同学的学号、姓名、籍贯、班级编号和所在班级名称。
 - 该查询的结果为学号、姓名、籍贯、班级编号和班级名称，在SELECT子句中必须包含这些属性
 - 由于班级名称和所属学院在班级表Class中，学号、姓名、籍贯、班级编号在学生表Student中，FROM子句必须包含Class表和Student表
 - 由于班级编号classNo既是班级表的主码，也是学生表的外码，这2个表的连接条件是claaaNo相等，在WHERE子句中必须包含连接条件Student.classNo=Class.classNo
 - 本查询要查询出会计学院的学生记录，在WHERE子句中还必须包括选择条件institute='会计学院'

3.3.1 等值与非等值连接

➤ 本查询语句为：

```
SELECT studentNo, studentName, native, Student.classNo, className  
FROM Student, Class
```

```
WHERE Student.classNo=Class.classNo AND institute='会计学院'
```

- 在连接操作中，如果涉及到多个表的相同属性名，必须在相同的属性名前加上表名加以区分

➤ 如Student.classNo、Class.classNo

➤ WHERE子句中

➤ Student.classNo=Class.classNo为连接条件

➤ institute='会计学院' 为选择条件

3.3.1 等值与非等值连接

- 可为参与连接的表取别名(称为元组变量), 在相同的属性名前加上表的别名。
- 将Student表取别名为 a , Class表取别名为 b , 班级编号分别用 $a.classNo$ 和 $b.classNo$ 表示。本例可以改写为:

```
SELECT studentNo, studentName, native,  $b.classNo$ , className  
FROM Student AS  $a$ , Class AS  $b$   
WHERE  $a.classNo=b.classNo$  AND institute='会计学院'
```

➤ 或者

```
SELECT studentNo, studentName, native,  $b.classNo$ , className  
FROM Student  $a$ , Class  $b$   
WHERE  $a.classNo=b.classNo$  AND institute='会计学院'
```

- 对于不同的属性名, 可以不在属性名前加上表名(别名)。

3.3.1 等值与非等值连接

● [例3.33] 查找选修了课程名称为“计算机原理”的同學学号、姓名。

- 查询结果为学号、姓名，在SELECT子句中必须包含这些属性
- 学号和姓名在学生表中，课程名称在课程表中，FROM子句必须包含学生表Student、课程表Course

studentNo	courseNo	termNo	score
1500001	001	151	98.0
1500001	002	151	82.0
1500001	003	161	82.0
1500001	004	151	56.0
1500001	004	161	86.0
1500001	005	152	77.0

课程表之间是多对多联系，需通过成绩表转换为两两联系，FROM子句必须包含成绩表Score

学生表的主码，也是成绩表的外码，这2个表的连接条件是学号相等。在WHERE子句中涉及三个关系表的条件为：

courseNo	courseName	creditHour	courseHour	priorCourse
001	大学语文	2	32	NULL
002	体育	2	32	NULL
003	大学英语	3	48	NULL
004	高等数学	6	96	NULL
005	C语言程序设计	4	80	004
006	计算机原理	4	64	005

studentNo=Score
courseNo=Student
courseName='计

studentNo	studentName	sex	birthday	native	nation	classNo
1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601

3.3.1 等值与非等值连接

➤ 本查询语句为：

SELECT *a*.studentNo, studentName

FROM Student *a*, Course *b*, Score *c*

WHERE *b*.courseNo=*c*.courseNo // 表*b*与表*c*的连接条件

AND *c*.studentNo=*a*.studentNo // 表*c*与表*a*的连接条件

AND *b*.courseName='计算机原理'

● 本例使用了元组变量，其连接条件为：

b.courseNo=*c*.courseNo **AND** *c*.studentNo=*a*.studentNo

3.3.1 等值与非等值连接

- [例3.34] 查找同时选修了编号为“001”和“002”课程的同学学号、姓名、课程号和相应成绩，并按学号排序输出。

➤ 查询结果为学号、姓名、课程号和相应成绩，在SELECT子句中必须包含这些属性

学号和姓名在学生表中，课程号和成绩在成绩表中，FROM

studentNo	courseNo	termNo	score
1500001	001	151	98.0
1500001	002	151	82.0
1500001	003	161	82.0
1500001	004	151	56.0
1500001	004	161	86.0
1500001	005	152	77.0
1500001	006	152	76.0
1500001	007	152	77.0
1500001	008	161	82.0
1500001	009	162	77.0
1500001	010	151	86.0
1500003	001	151	46.0
1500003	002	151	38.0
1500003	002	152	58.0
1500003	005	151	60.0
1500003	006	161	70.0
1500003	007	152	50.0

成绩表Score

成绩表的外码，这2个表的连接

必须包含这个连接条件

Name, b.courseNo, b.score

stNo // 表a与表b的连接条件

3.3.1 等值与非等值连接

➤ 为表示同时选修“001”和“002”课程的选择条件

✓ 首先在WHERE子句中直接包含选择条件 `courseNo='001'` 以查找出所有选修了“001”课程的同学

```
SELECT a.studentNo, studentName, b.courseNo, b.score
```

```
FROM Student a, Score b
```

```
WHERE a.studentNo=b.studentNo // 表a与表b的连接条件
```

```
AND b.courseNo='001'
```

➤ 注意：不能直接表示同时选修“001”和“002”课程的选择条件

```
AND b.courseNo='001'
```

```
AND b.courseNo='002'
```

✗ !

3.3.1 等值与非等值连接

➤为表示同时选修“001”和“002”课程的选择条件

✓其次，基于成绩表Score构造一个查询表 c ，查找出选修了编号为“002”课程的所有同学

(SELECT * FROM Score WHERE courseNo='002') c

✓最后，将选修了编号为“001”课程的元组与查询表 c 的元组关于学号进行等值连接（连接条件是什么？）

SELECT a .studentNo, studentName, b .courseNo, b .score, c .courseNo, c .score
FROM Student a , Score b ,

(SELECT * FROM Score WHERE courseNo='002') c

WHERE b .courseNo='001'

AND a .studentNo= b .studentNo // 表 a 与表 b 的连接条件


AND a .studentNo= c .studentNo // 表 a 与表 c 的连接条件

➤如果连接成功，表示该同学同时选修了这两门课程

3.3.1 等值与非等值连接

- 要求按学号排序输出，需要排序语句ORDER BY
- 本查询语句为：

```
SELECT a.studentNo, studentName, b.courseNo, b.score, c.courseNo, c.score
FROM Student a, Score b,
      (SELECT * FROM Score WHERE courseNo='002') c
WHERE b.courseNo='001'
      AND a.studentNo=b.studentNo
      AND a.studentNo=c.studentNo
ORDER BY a.studentNo
```



	studentNo	studentName	courseNo	score	courseNo	score
1	1500001	李小勇	001	98.0	002	82.0
2	1500003	王红敏	001	46.0	002	38.0
3	1500003	王红敏	001	46.0	002	58.0
4	1500004	张可立	001	48.0	002	68.0
5	1500004	张可立	001	70.0	002	68.0
6	1500005	王红	001	79.0	002	80.0
7	1600002	刘晨	001	98.0	002	46.0
8	1600003	王敏	001	70.0	002	60.0
9	1600004	张立	001	50.0	002	70.0
10	1600004	张立	001	68.0	002	70.0
11	1600005	王红	001	82.0	002	80.0
12	1600012	王立红	001	68.0	002	78.0
13	1600014	刘宏昊	001	60.0	002	69.0

3.3.1 等值与非等值连接

➤该查询也可以表示为:

```
SELECT a.studentNo, studentName, b.courseNo, b.score, c.courseNo, c.score
FROM Student a,
      (SELECT * FROM Score WHERE courseNo='001') b,
      (SELECT * FROM Score WHERE courseNo='002') c
WHERE a.studentNo=b.studentNo    // 表a与表b的连接条件
      AND a.studentNo=c.studentNo // 表a与表c的连接条件
ORDER BY a.studentNo
```

3.3.1 等值与非等值连接

➤该查询还可以表示为：

```
SELECT a.studentNo, studentName, b.courseNo, b.score, c.courseNo, c.score
FROM Student a, Score b, Score c
WHERE a.studentNo=b.studentNo    // 表a与表b的连接条件
      AND a.studentNo=c.studentNo // 表a与表c的连接条件
      AND b.courseNo='001'        // 表b上的选择条件
      AND c.courseNo='002'        // 表c上的选择条件
ORDER BY a.studentNo
```

➤注意：不能在同一个表**b**上同时表示选修“001”和“002”课程

```
AND b.courseNo='001'
```

```
AND b.courseNo='002'    × !
```

3.3.1 等值与非等值连接

■ [例3.35] 查询获得的总学分(注：只有成绩合格才能获得该课程的学分)大于或等于28的同学们的学号、姓名和总学分，并按学号排序输出。

studentNo	courseNo	termNo	score	courseNo	courseName	creditHour	courseHour	priorCourse	studentNo	studentName	sex	birthday	native	nation	classNo
1500001	001	151	98.0	001	大学语文	2	32	NULL	1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
1500001	002	151	82.0	002	体育	2	32	NULL	1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
1500001	003	161	82.0	003	大学英语	3	48	NULL	1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
1500001	004	151	56.0	004	高等数学	6	96	NULL	1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
1500001	004	161	86.0	005	C语言程序设计	4	80	004	1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
1500001	005	152	77.0	006	计算机原理	4	64	005	1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
1500001	006	152	76.0	007	数据结构	5	96	005	1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
1500001	007	152	77.0	008	操作系统	4	64	007	1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601
1500001	008	161	82.0	009	数据库系统原理	4	80	008	1600004	张立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1601
1500001	009	162	77.0	010	会计学原理	4	64	004	1600005	王红	男	1999-04-26 00:00:00.000	南昌	蒙古族	CP1602
1500001	010	151	86.0	011	中级财务会计	5	80	010	1600006	李志强	男	1999-12-21 00:00:00.000	北京	汉族	CP1602
1500001	011	151	86.0												
1500003	001	151	46.0												
1500003	002	151	38.0												
1500003	002	152	58.0												
1500003	005	151	60.0												
1500003	006	161	70.0												
1500003	007	152	50.0												

$\text{sum}(\text{creditHour}) \geq 28$

$a.\text{studentNo}$

结果中需要同时包含学号和姓名

GROUP BY子句需要按“ $a.\text{studentNo}, \text{studentName}$ ”进行聚合

合，不能仅按“ $a.\text{studentNo}$ ”进行聚合，否则无法输出 studentName

60

time -- 删除结束而安

3.4.2 分组聚合

- 本查询既使用了**WHERE**子句，也使用了**HAVING**子句，都是选择满足条件的元组，但其选择的范围是不一样的：
 - (1) **WHERE**子句：作用于整个查询对象，对元组进行过滤。
 - (2) **HAVING**子句：仅作用于分组，对分组进行过滤。
- 本例的查询过程是：
 - ① 首先在Score 表中选择课程成绩大于等于60分的元组(只有60分及以上才能获得学分)，将这些元组与Student和Score表进行连接，形成一个新关系；
 - ② 在新关系中按学号进行分组，统计每组的总学分；
 - ③ 将总学分大于等于28的组选择出来形成一个结果关系；
 - ④ 将结果关系输出。

3.3.1 等值与非等值连接

■ 自然连接

- SQL不直接支持自然连接，完成自然连接的方法是在等值连接的基础上消除重复列
- [例3.36] 实现成绩表Score和课程表Course的自然连接。

```
SELECT studentNo, a.courseNo, score, courseName,  
        creditHour, courseHour, priorCourse
```

```
FROM Score a, Course b
```

```
WHERE a.courseNo=b.courseNo    // 表a与表b的连接条件
```

- 本例课程编号在两个关系中同时出现，但在SELECT子句中仅需出现1次，因此使用*a.courseNo*，也可以使用*b.courseNo*。其他列名是唯一的，不需要加上元组变量

3.3.1 等值与非等值连接

■ 非等值连接

- 非等值连接使用的比较少。
- 在关系代数部分已经举过了一个非等值连接的例子（P53-54，例2.16），这里就不再举例了。
 - 在数据库ScoreDB中，查找课程号为AC001课程的考试中比学号为1503045的学生考得更好的所有学生的姓名和成绩。

3.3.2 自表连接

■ 若某个表与自己进行连接，称为自表连接

- [例3.37] 查找同时选修了编号为“001”和“002”课程的同学学号、姓名、课程号和相应成绩，并按学号排序输出。

- 学生姓名在学生表中，FROM子句必须包含学生表(取别名为 a)

- 可以考虑两个成绩表，分别记为 b 和 c

- ✓ b 表用于查询选修了编号为“001”课程的同学

- ✓ c 表用于查询选修了编号为“002”课程的同学

- FROM子句还必须包含两个成绩表 b 和 c ，且在WHERE子句中包含两个选择条件：

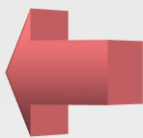
$b.courseNo='001' \text{ AND } c.courseNo='002'$

3.3.2 自表连接

- 一方面，成绩表**b**与成绩表**c**在学号上做等值连接(自表连接)，如果连接成功，表示学生同时选修了编号为“001”和“002”的课程
- 另一方面，学生表与成绩表**b** (或成绩表**c**)在学号上做等值连接。
WHERE子句包含两个连接条件：

b.studentNo=c.studentNo AND a.studentNo=b.studentNo

- 本查询语句为：



```
SELECT a.studentNo, studentName, b.courseNo, b.score, c.courseNo, c.score
FROM Student a, Score b, Score c
WHERE b.courseNo='001' AND c.courseNo='002'
      AND a.studentNo=b.studentNo AND b.studentNo=c.studentNo
ORDER BY a.studentNo
```

- 本查询结果与例3.34相同
- 在该查询中，FROM子句后面包含了两个参与自表连接的成绩表Score，必须定义元组变量加以区分
- 自表连接的条件是**b.studentNo=c.studentNo**

3.3.2 自表连接

- [例3.38] 在学生表Student中查找与“李小勇”同学在同一个班的同学姓名、班级编号和出生日期。

SELECT *a*.studentName, *a*.classNo, *a*.birthday

FROM Student *a*, Student *b*

studentNo	studentName	sex	birthday	native	nation	classNo
1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601
1600004	张立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1601
1600005	王红	男	1999-04-26 00:00:00.000	南昌	蒙古族	CP1602
1600006	李志强	男	1999-12-21 00:00:00.000	北京	汉族	CP1602

studentNo	studentName	sex	birthday	native	nation	classNo
1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601
1600004	张立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1601
1600005	王红	男	1999-04-26 00:00:00.000	南昌	蒙古族	CP1602
1600006	李志强	男	1999-12-21 00:00:00.000	北京	汉族	CP1602

(**SELECT** * **FROM** Student **WHERE** studentName='李小勇') *b*

WHERE *a*.classNo=*b*.classNo

3.3.3 外连接

- 在一般的连接中，**只有满足连接条件的元组**才被检索出来，对于没有满足连接条件的元组是不作为结果被检索出来的。

- [例3.39] 查询**2015级**每个班级的班级名称、所属学院、学生学号、学生姓名，按班级名称排序输出。

```
SELECT className, institute, studentNo, studentName
FROM Class a, Student b
WHERE a.classNo=b.classNo AND grade=2015
ORDER BY className
```

classNo	className	institute	grade	classNum
CP1601	注册会计16_01班	会计学院	2016	NULL
CP1602	注册会计16_02班	会计学院	2016	NULL
CP1603	注册会计16_03班	会计学院	2016	NULL
CS1501	计算机科学与技术15-01班	信息管理学院	2015	NULL
CS1502	计算机科学与技术15-02班	信息管理学院	2015	NULL
CS1601	计算机科学与技术16-01班	信息管理学院	2016	NULL
ER1501	金融管理15-01班	金融学院	2015	NULL

3 外
有满

studentNo	studentName	sex	birthday	native	nation	classNo
1500001	李小勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1501
1500002	刘方晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1501
1500003	王红敏	女	1997-10-01 00:00:00.000	上海	汉族	IS1501
1500004	张可立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1501
1500005	王红	男	2000-04-26 00:00:00.000	南昌	蒙古族	CS1502
1600001	李勇	男	1998-12-21 00:00:00.000	南昌	汉族	CS1601
1600002	刘晨	女	1998-11-11 00:00:00.000	九江	汉族	IS1601
1600003	王敏	女	1998-10-01 00:00:00.000	上海	汉族	IS1601
1600004	张立	男	1999-05-20 00:00:00.000	南昌	蒙古族	CS1601

索引出来，对于没有满足连接条件的元组是不作为结果被检索出来的。

- [例3.39] 查询2015级每个班级的班级名称、所属学院、学生学号、学生姓名，按班级名称排序输出。

SELECT className, institute, studentNo, studentName

	className	institute	studentNo	studentName
1	计算机科学与技术15-01班	信息管理学院	1500001	李小勇
2	计算机科学与技术15-01班	信息管理学院	1500004	张可立
3	计算机科学与技术15-02班	信息管理学院	1500005	王红
4	信息管理与信息系统15-01班	信息管理学院	1500002	刘方晨
5	信息管理与信息系统15-01班	信息管理学院	1500003	王红敏

3.3.3 外连接

■ 从查询结果中可以看出：

- 班级表中的“金融管理15-01班”没有出现在查询结果中，原因是该班没有学生

	<u>classNo</u>	className	institute	grade	classNum
1	CP1601	注册会计 16_01 班	会计学院	2016	NULL
2	CP1602	注册会计 16_02 班	会计学院	2016	NULL
3	CP1603	注册会计 16_03 班	会计学院	2016	NULL
4	CS1501	计算机科学与技术 15-01 班	信息管理学院	2015	NULL
5	CS1502	计算机科学与技术 15-02 班	信息管理学院	2015	NULL
6	CS1601	计算机科学与技术 16-01 班	信息管理学院	2016	NULL
7	ER1501	金融管理 15-01 班	金融学院	2015	NULL
8	IS1501	信息管理与信息系统 15-01 班	信息管理学院	2015	NULL
9	IS1601	信息管理与信息系统 16-01 班	信息管理学院	2016	NULL

图 3-8 班级表 Class 的数据

3.3.3 外连接

■ 从查询结果中可以看出：

- 班级表中的“金融管理15-01班”没有出现在查询结果中，原因是该班没有学生

■ 在实际应用中，往往需要将不满足连接条件的元组也检索出来，只是在相应的位置用空值替代，这种查询称为外连接查询

■ 外连接分为左外连接、右外连接和全外连接

■ 在FROM子句中，写在左边的表称为左关系，写在右边的表称为右关系

3.3.3 外连接

■ 左外连接

- 连接结果中包含左关系中的所有元组，对于左关系中没有连接上的元组，其右关系中的相应属性用空值替代
- [例3.40] 使用左外连接查询2015级每个班级的班级名称、所属学院、学生学号、学生姓名，按班级名称和学号排序输出。

```
SELECT className, institute, studentNo, studentName
FROM Class a LEFT OUTER JOIN Student b
      ON a.classNo=b.classNo
WHERE grade=2015
ORDER BY className, studentNo
```

■ 左外连

● 连接 有连

	className	institute	studentNo	studentName
1	计算机科学与技术15-01班	信息管理学院	1500001	李小勇
2	计算机科学与技术15-01班	信息管理学院	1500004	张可立
3	计算机科学与技术15-02班	信息管理学院	1500005	王红
4	金融管理15-01班	金融学院	NULL	NULL
5	信息管理与信息系统15-01班	信息管理学院	1500002	刘方晨
6	信息管理与信息系统15-01班	信息管理学院	1500003	王红敏

- [例3.40] 使用左外连接查询**2015级**每个班级的班级名称、所属学院、学生学号、学生姓名，按班级名称和学号排序输出。

```

SELECT className, institute, studentNo, studentName
FROM Class a LEFT OUTER JOIN Student b
      ON a.classNo=b.classNo
WHERE grade=2015
ORDER BY className, studentNo

```

3.3.3 外连接

■ 右外连接

- 连接结果中包含右关系中的所有元组，对于右关系中没有连接上的元组，其左关系中的相应属性用空值替代
- [例3.41] 使用右外连接查询2015级每个班级的班级名称、所属学院、学生学号、学生姓名，按班级名称和学号排序输出。

```
SELECT className, institute, studentNo, studentName
FROM Class a RIGHT OUTER JOIN Student b
      ON a.classNo=b.classNo
WHERE grade=2015
ORDER BY className, studentNo
```

3.3.3 外连接

■ 全外连接：连接结果中包含左、右关系中的所有元组

- 对左关系中没有连接上的元组，其右关系中的相应属性用空值替代
- 对右关系中没有连接上的元组，其左关系中的相应属性用空值替代

- [例3.42] 使用全外连接查询2015级每个班级的班级名称、所属学院、学生学号、学生姓名，按班级名称和学号排序输出。

```
SELECT className, institute, studentNo, studentName
FROM Class a FULL OUTER JOIN Student b
      ON a.classNo=b.classNo
WHERE grade=2015
ORDER BY className, studentNo
```