

第 3 章

SQL

学习目标

结构化查询语言(Structured Query Language,SQL)是关系数据库的标准语言,本章主要讲授 SQL 查询语言在数据库中的应用。目前,几乎所有的关系型数据库管理系统,如 Oracle、Sybase、Microsoft SQL Server 和 Access 等均采用 SQL 标准。因此,本章的教学目标主要有两个,一是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;二是要求读者熟练掌握 SQL 查询语句,并运用 SQL 查询语句完成对数据库的查询操作。

学习方法

本章重在实验,因此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

本章的重点是 3.2 节、3.3 节和 3.4 节,难点是 3.4 节。

本章导读

- (1) 各种关系代数运算功能在 SQL 查询语句中是如何表达的?
- (2) 连接操作在 SQL 中是如何表达的? 它们分别应用于什么情况下?
- (3) 在使用分组聚合查询时需要注意什么?
- (4) 理解相关子查询与非相关子查询的概念,它们有什么区别、分别应用于什么情况下?
- (5) 理解子查询的概念,IN 子查询和存在量词 EXISTS 子查询有什么区别、分别应用于什么情况下?
- (6) 理解查询表的概念,查询表与子查询有何异同点?

3.1 SQL 概述

SQL 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。

自从 SQL 成为国际标准以来,很多数据库厂商都对 SQL 进行了再开发和扩展,但查询

(SELECT)、插入(INSERT)、修改(UPDATE)、删除(DELETE)、创建(CREATE)以及对对象删除(DROP)等标准 SQL 语句仍然可被用来完成几乎所有的数据库操作。

3.1.1 SQL 发展

SQL 是关系数据库的标准语言,是数据库领域中一种主流语言,它经历了如下几个阶段。

(1) SQL-86: 第一个 SQL 标准,由美国国家标准局(American National Standard Institute, ANSI)公布,1987 年国际标准化组织(International Organization for Standardization, ISO)通过。该标准也称为 SQL-1。

(2) SQL-92: 在 1992 年,由 ISO 和 ANSI 对 SQL-86 进行了重新修订,发布了第二个 SQL 标准 SQL-92,该标准也称为 SQL-2。

(3) SQL-99: 随着信息技术的应用,数据库理论和技术得到了广泛的应用和发展。在 1999 年,ISO 发布了反映最新数据库理论和技术标准 SQL-99,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。

(4) SQL-2003: 该标准是最新的标准,也称为 SQL-4,于 2003 年发布,包括 9 部分:

- ① ISO/IEC 9075-1: Framework (SQL/Framework);
- ② ISO/IEC 9075-2: Foundation (SQL/Foundation);
- ③ ISO/IEC 9075-3: Call Level Interface (SQL/CLI);
- ④ ISO/IEC 9075-4: Persistent Stored Modules (SQL/PSM);
- ⑤ ISO/IEC 9075-9: Management of External Data (SQL/MED);
- ⑥ ISO/IEC 9075-10: Object Language Bindings (SQL/OLB);
- ⑦ ISO/IEC 9075-11: Information and Definition Schemas (SQL/Schemata);
- ⑧ ISO/IEC 9075-13: Java Routines and Types Using the Java Programming Language (SQL/JRT);
- ⑨ ISO/IEC 9075-14: XML-Related Specifications (SQL/XML)。

目前,许多数据库厂商都支持 SQL-92 的绝大多数标准,以及 SQL-99 和 SQL-2003 的部分标准,并对 SQL 进行了扩展。这些扩展的 SQL,不仅遵循标准 SQL 规定的功能,而且还增强了许多功能,并赋予 SQL 不同的名字,如 Oracle 产品将 SQL 称为 PL/SQL, Sybase 和 Microsoft SQL Server 产品将 SQL 称为 Transact-SQL。

SQL 由 4 部分组成,包括数据定义语言 DDL、数据操纵语言 DML、数据控制语言 DCL 和其他,其功能如下:

(1) 数据定义语言(data definition language, DDL): 主要用于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 类语句,即定义、修改和删除。

(2) 数据操纵语言(data manipulation language, DML): 主要用于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。

(3) 数据控制语言(data control language, DCL): 主要用于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。

(4) 其他: 主要是嵌入式 SQL 和动态 SQL 的定义,规定了 SQL 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

SQL 因其简单、灵活、易掌握,受到了广大用户的欢迎,SQL 既可以作为交互式数据库语言使用,也可以作为程序设计语言的子语言使用,它是一个兼有关系代数和元组演算特征的语言,其特点如下所述。

1. 综合统一

(1) SQL 集数据定义语言 DDL、数据操纵语言 DML 和数据控制语言 DCL 的功能于一体,语言风格统一,可以独立完成数据库生命周期中的全部活动,包括建立数据库、定义关系模式、创建视图和索引、数据的查询和更新(插入、删除和修改)、数据库重构、数据库安全性控制等一系列操作,这就为数据库应用系统开发提供了良好的环境。例如,用户在数据库投入运行后,还可根据需求随时地逐步地修改模式,并不影响数据库的运行,从而使系统具有良好的可扩充性。

(2) 在关系模型中实体(集)和实体(集)间的联系(集)均用关系表示,这种数据结构的单一性带来了数据操作符的统一,即对实体(集)及实体(集)间联系(集)的每一种操作(如查找、插入、删除和修改)都只需要一种操作符。

2. 高度非过程化

非关系数据模型的数据操纵语言是面向过程的语言,在完成某项操作请求时必须指定存取路径。而用 SQL 进行数据操作时,用户只需提出“做什么”,不必指明“怎么做”,因此用户无需了解存取路径,存取路径的选择以及 SQL 语句的操作过程由系统自动完成。这不仅大大减轻了用户负担,而且有利于提高数据独立性。

3. 面向集合的操作方式

SQL 采用集合操作方式,其操作对象、操作结果都是元组的集合。而非关系数据模型采用的是面向记录的操作方式,其操作对象是一条记录。

(1) 非关系数据模型采用的是面向记录的操作方式,任何一个操作的对象都是一条记录。例如,查询所有平均成绩在 80 分以上的学生姓名,用户必须说明完成该操作请求的具体处理过程,即如何用循环结构按照某条路径一条一条地把满足条件的学生记录读出来。

(2) SQL 采用集合操作方式,不仅查询操作的对象是元组的集合,而且一次更新(即插入、删除和修改)操作的对象也可以是元组的集合。

4. 同一种语法结构提供两种使用方式

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

(2) 作为自含式语言,它能够独立地用于联机交互的使用方式,用户可以在终端键盘上直接输入 SQL 命令来对数据库进行操作。

(3) 作为嵌入式语言,SQL 语句能够嵌入到高级语言(如 Java、VC、VB、Delphi 等)程序中,供程序员设计应用程序时使用。

5. 语言简洁,易学易用

SQL 功能强大,但十分简洁,易学易用。SQL 的动词非常少,主要包括:

(1) 数据查询: SELECT。

- (2) 数据定义: CREATE、DROP、ALTER。
- (3) 数据更新: INSERT、DELETE、UPDATE。
- (4) 数据控制: GRANT、REVOKE。

3.1.3 SQL 查询基本概念

SQL 支持关系数据库管理系统的三级模式结构,其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,如图 3-1 所示。

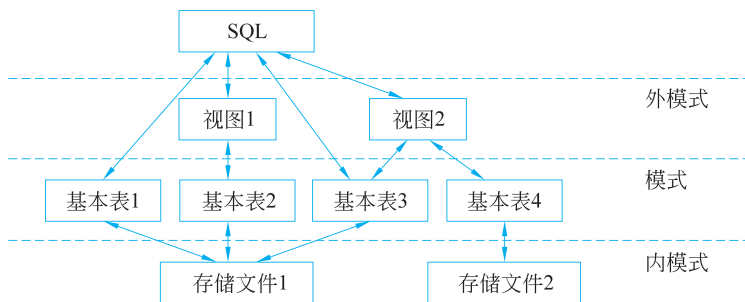


图 3-1 SQL 对关系数据库模式的支持

用户使用 SQL 可对基本表、视图和查询表进行操作。

(1) 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,索引也存放在存储文件中。

(2) 视图。**视图**是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,不存放对应的数据。

(3) 查询表。**查询表**是指一个查询的查询结果所对应的表。

(4) 存储文件。**存储文件**是指数据库中存放关系的物理文件,其逻辑结构组成了关系数据库的内模式,其物理结构对用户是透明的。

本章只介绍 SQL 查询语言,SQL 数据定义语言、数据更新语言及数据库编程将在第 7 章介绍,SQL 数据控制语言以及 SQL 数据定义中的完整性控制功能将在第 9 章介绍。

3.2 单表查询

一条 SQL 查询语句可以完成宿主语言中用若干条语句才能完成的功能。SQL 中最重要的部分是查询语句,查询语句的执行离不开数据库模式。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,其中带下划线的属性或属性集合为主码,斜体的属性为外码,其数据库模式导航图如图 3-7 所示;其实例数据如图 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

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(1)	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	成绩
<u>studentNo</u>	<u>courseNo</u>	<u>termNo</u>	score
char(7)	char(3)	char(3)	numeric(5,1)

图 3-6 成绩表 Score

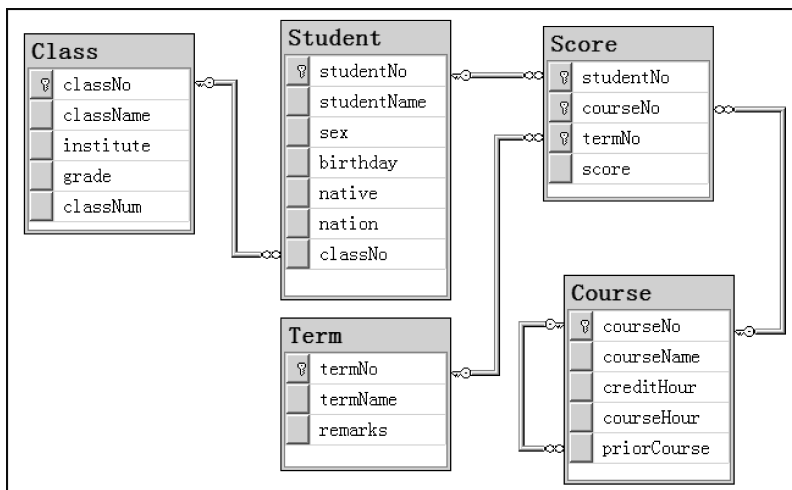


图 3-7 学生成绩管理数据库模式导航图

序号	<u>classNo</u>	className	institute	grade	classNum
1	CP2201	注册会计师 22_01 班	会计学院	2022	NULL
2	CP2202	注册会计师 22_02 班	会计学院	2022	NULL
3	CP2203	注册会计师 22_03 班	会计学院	2022	NULL
4	CS2101	计算机科学与技术 21-01 班	信息管理学院	2021	NULL
5	CS2102	计算机科学与技术 21-02 班	信息管理学院	2021	NULL
6	CS2201	计算机科学与技术 22-01 班	信息管理学院	2022	NULL
7	ER2101	金融管理 21-01 班	金融学院	2021	NULL
8	IS2101	信息管理与信息系统 21-01 班	信息管理学院	2021	NULL
9	IS2201	信息管理与信息系统 22-01 班	信息管理学院	2022	NULL

图 3-8 班级表 Class 的数据

序号	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
5	005	C 语言程序设计	4	80	004
6	006	计算机原理	4	64	005
7	007	数据结构	5	96	005
8	008	操作系统	4	64	007
9	009	数据库系统原理	4	80	008
10	010	会计学原理	4	64	004
11	011	中级财务会计	5	80	010

图 3-9 课程表 Course 的数据

本章给出的所有例题的操作执行结果都是在 SQL Server 2019 数据库中执行得到的。

3.2.1 投影运算

SQL 查询语句的基本结构包括 3 条子句,即 SELECT、FROM 和 WHERE,其中:

- SELECT 子句对应于关系代数中的投影运算,用来指定查询结果中所需要的属性或表达式。
- FROM 子句对应于关系代数中的笛卡儿积,用来给出查询所涉及的表,这些表可以是基本表、视图或查询表。
- WHERE 子句对应于关系代数中的选择运算,用来指定查询结果元组所需要满足的选择条件。

对于 SQL 查询语句,SELECT 和 FROM 子句是必需的,其他是可选的,其基本语法为

```
SELECT A1, A2, ..., An
FROM R1, R2, ..., Rm
WHERE P
```

其中, $A_1, A_2, \dots, A_n$ 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,表示不需要对元组进行选择。SQL 的查询结果中允许包含重复元组,这与关系代数中的投影运算的语义不一样。

SQL 查询的执行过程为:首先对 $R_1, R_2, \dots, R_m$ 执行笛卡儿积,然后在笛卡儿积中选择使得谓词 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。如果需要消除重复元组,则必须使用专门的关键词 DISTINCT。

上面描述的 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 8 章。

1. 查询指定列

选取表中的全部列或指定列,称为关系代数的投影运算。通过 SELECT 子句确定要查询的属性或表达式。

序号	<u>studentNo</u>	studentName	sex	birthday	native	nation	classNo
1	2100001	李小勇	男	2003-12-21 00:00:00.000	南昌	汉族	CS2101
2	2100002	刘方晨	女	2003-11-11 00:00:00.000	九江	汉族	IS2101
3	2100003	王红敏	女	2002-10-01 00:00:00.000	上海	汉族	IS2101
4	2100004	张可立	男	2004-05-20 00:00:00.000	南昌	蒙古族	CS2101
5	2100005	王红	男	2005-04-26 00:00:00.000	南昌	蒙古族	CS2102
6	2200001	李勇	男	2003-12-21 00:00:00.000	南昌	汉族	CS2201
7	2200002	刘晨	女	2003-11-11 00:00:00.000	九江	汉族	IS2201
8	2200003	王敏	女	2003-10-01 00:00:00.000	上海	汉族	IS2201
9	2200004	张立	男	2004-05-20 00:00:00.000	南昌	蒙古族	CS2201
10	2200005	王红	男	2004-04-26 00:00:00.000	南昌	蒙古族	CP2202
11	2200006	李志强	男	2004-12-21 00:00:00.000	北京	汉族	CP2202
12	2200007	李立	女	2004-08-21 00:00:00.000	福建	畲族	IS2201
13	2200008	黄小红	女	2004-08-09 00:00:00.000	云南	傣族	CS2201
14	2200009	黄勇	男	2004-11-21 00:00:00.000	九江	汉族	CP2202
15	2200010	李宏冰	女	2003-03-09 00:00:00.000	上海	汉族	CP2202
16	2200011	江宏吕	男	2003-12-20 00:00:00.000	上海	汉族	CP2202
17	2200012	王立红	男	2003-11-18 00:00:00.000	北京	汉族	CS2201
18	2200013	刘小华	女	2004-07-16 00:00:00.000	云南	哈尼族	IS2201
19	2200014	刘宏昊	男	2004-09-16 00:00:00.000	福建	汉族	IS2201
20	2200015	吴敏	女	2002-01-20 00:00:00.000	福建	畲族	CP2202

图 3-10 学生表 Student 的数据

序号	<u>termNo</u>	termName	remarks
1	211	2021—2022 学年第一学期	NULL
2	212	2021—2022 学年第二学期	NULL
3	213	2021—2022 学年第三学期	小学期
4	221	2022—2023 学年第一学期	NULL
5	222	2022—2023 学年第二学期	NULL
6	223	2022—2023 学年第三学期	小学期

图 3-11 学期表 Term 的数据

序号	<u>studentNo</u>	<u>courseNo</u>	<u>termNo</u>	score	序号	<u>studentNo</u>	<u>courseNo</u>	<u>termNo</u>	score
1	2100001	001	211	98.0	41	2100005	011	222	68.0
2	2100001	002	211	82.0	42	2200002	001	221	98.0
3	2100001	003	221	82.0	43	2200002	002	221	46.0
4	2100001	004	211	56.0	44	2200002	003	222	98.0
5	2100001	004	221	86.0	45	2200002	004	221	60.0
6	2100001	005	212	77.0	46	2200002	005	222	86.0
7	2100001	006	212	76.0	47	2200002	010	222	70.0
8	2100001	007	212	77.0	48	2200003	001	221	70.0
9	2100001	008	221	82.0	49	2200003	002	221	60.0
10	2100001	009	222	77.0	50	2200003	004	221	77.0
11	2100001	010	211	86.0	51	2200003	005	222	87.0
12	2100003	001	211	46.0	52	2200004	001	221	50.0
13	2100003	002	211	38.0	53	2200004	001	222	68.0
14	2100003	002	212	58.0	54	2200004	002	221	70.0
15	2100003	005	211	60.0	55	2200004	003	222	88.0
16	2100003	006	221	70.0	56	2200004	004	221	78.0
17	2100003	007	212	50.0	57	2200004	010	221	89.0
18	2100003	007	222	66.0	58	2200004	011	222	90.0
19	2100003	008	222	82.0	59	2200005	001	221	82.0
20	2100003	009	222	78.0	60	2200005	002	221	80.0
21	2100003	010	221	90.0	61	2200005	003	222	82.0
22	2100004	001	211	48.0	62	2200005	004	221	47.0
23	2100004	001	222	70.0	63	2200005	010	221	90.0
24	2100004	002	221	68.0	64	2200005	011	222	82.0
25	2100004	003	212	70.0	65	2200012	001	221	68.0
26	2100004	004	211	58.0	66	2200012	002	222	78.0
27	2100004	005	222	88.0	67	2200012	003	221	76.0
28	2100004	006	222	72.0	68	2200012	004	221	70.0
29	2100004	007	221	71.0	69	2200012	005	221	88.0
30	2100004	008	221	80.0	70	2200012	006	222	82.0
31	2100005	001	212	79.0	71	2200012	007	222	90.0
32	2100005	002	211	80.0	72	2200012	010	222	84.0
33	2100005	003	211	69.0	73	2200014	001	221	60.0
34	2100005	004	211	87.0	74	2200014	002	222	69.0
35	2100005	005	211	77.0	75	2200014	003	221	87.0
36	2100005	006	212	69.0	76	2200014	004	221	45.0
37	2100005	007	221	90.0	77	2200014	004	222	88.0
38	2100005	008	221	87.0	78	2200014	005	222	56.0
39	2100005	009	222	90.0	79	2200014	010	221	90.0
40	2100005	010	212	69.0	80	2200014	011	222	70.0

图 3-12 成绩表 Score 的数据

【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 共 3 个属性的值，形成一个新元组，最后将这些新元组组织成一个结果关系输出。该查询的结果如图 3-13 所示。

	classNo	className	institute
1	CP2201	注册会计师22_01班	会计学院
2	CP2202	注册会计师22_02班	会计学院
3	CP2203	注册会计师22_03班	会计学院
4	CS2101	计算机科学与技术21-01班	信息管理学院
5	CS2102	计算机科学与技术21-02班	信息管理学院
6	CS2201	计算机科学与技术22-01班	信息管理学院
7	ER2101	金融管理21-01班	金融学院
8	IS2101	信息管理与信息系统21-01班	信息管理学院
9	IS2201	信息管理与信息系统22-01班	信息管理学院

图 3-13 例 3.1 的查询结果

2. 消除重复元组

SQL 查询默认是不消除重复元组，因为消除重复元组要消耗系统资源。如果需要消除重复元组，则可以使用 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。如果需要消除重复元组，则可使用如下查询，其查询结果如图 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 例 3.2 的查询结果一

	institute
1	会计学院
2	金融学院
3	信息管理学院

图 3-15 例 3.2 的查询结果二

3. 查询所有列

查询所有的属性列，SQL 可以使用两种方法：一是将所有的列在 SELECT 子句中列出

(可以改变列的显示顺序);二是使用 * 符号,* 表示所有属性,此时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
```

4. 给属性列取别名

可以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那些经过计算的列。

【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
```

该查询的结果如图 3-16 所示。该查询也可以使用 AS 关键字来取别名,如

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

5. 查询经过计算的列

SELECT 子句中可以使用属性、常量、函数和表达式,如果是函数或表达式,则先计算函数或表达式的值,然后将计算的结果显示出来。

【例 3.5】 查询每门课程的课程号、课程名以及周课时(周课时等于课时数除以 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) 课程名, courseHour/16 AS 周课时
FROM Course
```

其中,函数 lower()表示将大写字母改为小写字母,其查询结果如图 3-17 所示。

	所属学院	班级编号	班级名称
1	会计学院	CP2201	注册会计22_01班
2	会计学院	CP2202	注册会计22_02班
3	会计学院	CP2203	注册会计22_03班
4	信息管理学院	CS2101	计算机科学与技术21-01班
5	信息管理学院	CS2102	计算机科学与技术21-02班
6	信息管理学院	CS2201	计算机科学与技术22-01班
7	金融学院	ER2101	金融管理21-01班
8	信息管理学院	IS2101	信息管理与信息系统21-01班
9	信息管理学院	IS2201	信息管理与信息系统22-01班

图 3-16 例 3.4 的查询结果

	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

图 3-17 例 3.5 的查询结果

3.2.2 选择运算

WHERE 子句可以实现关系代数中的选择运算,用于查询满足选择条件的元组,这是查询中涉及最多的一类查询。WHERE 子句中常用的查询条件运算符如下所示。

- 比较运算: $>$ 、 $>=$ 、 $<$ 、 $<=$ 、 $=$ 、 $<>$ (或 $\neq$)。
- 范围查询: $[NOT] BETWEEN \cdots AND$ 。
- 集合查询: $[NOT] IN$ 。
- 空值查询: $[NOT] IS NULL$ 。
- 字符匹配查询: $[NOT] LIKE$ 。
- 逻辑查询: AND 、 OR 、 NOT 。

1. 比较运算

使用比较运算符 $>$ 、 $>=$ 、 $<$ 、 $<=$ 、 $=$ 、 $<>$ (或 $\neq$)实现相应的比较运算。

【例 3.6】 查询 2021 级的班级编号、班级名称和所属学院。

```
SELECT classNo, className, institute
FROM Class
WHERE grade=2021
```

其查询结果如图 3-18 所示。

	classNo	className	institute
1	CS2101	计算机科学与技术21-01班	信息管理学院
2	CS2102	计算机科学与技术21-02班	信息管理学院
3	ER2101	金融管理21-01班	金融学院
4	IS2101	信息管理与信息系统21-01班	信息管理学院

图 3-18 例 3.6 的查询结果

该查询的执行过程可能有多种方法:一种是全表扫描法,即依次取出 Class 表中的每个元组,判断该元组的 grade 属性值是否等于 2021,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 属性列上创建索引。另一种是索引搜索法,如果该表在 grade 属性列上建有索引,且满足条件的记录不多,则可以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。

【例 3.7】 在学生 Student 表中查询年龄大于或等于 20 岁的同学学号、姓名和出生日期。

```
SELECT studentNo, studentName, birthday
FROM Student
WHERE year(getdate())-year(birthday)>=20
```

其中,函数 getdate()可获取当前系统的日期,函数 year()用于提取日期中的年份。

2. 范围查询

BETWEEN...AND 可用于查询属性值在某一个范围内的元组,NOT BETWEEN...AND 可用于查询属性值不在某一个范围内的元组。BETWEEN 后面是属性的下限值,

AND 后面是属性的上限值。

【例 3.8】 在选课 Score 表中查询成绩为 80~90 分的同学学号、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。

【例 3.9】 在选课 Score 表中查询成绩不为 80~90 分的同学学号、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. 集合查询

IN 可用于查询属性值在某个集合内的元组, NOT IN 可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 3.4.1 节)。

【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。

【例 3.11】 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)。

```
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。

4. 空值查询

SQL 支持空值运算, 空值表示未知或不确定的值, 空值表示为 NULL。IS NULL 可用于查询属性值为空值; IS NOT NULL 可用于查询属性值不为空值。这里的 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。

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

查询结果如图 3-19 所示。

	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

图 3-19 例 3.12 的查询结果

【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
```

5. 字符匹配查询

对于字符型数据,LIKE 可用于字符匹配查询。LIKE 的语法格式为

```
[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_。其中,符号%表示任意长度的字符串,如 ab%,表示所有以 ab 开头的任意长度的字符串;再如 zhang%ab,表示以 zhang 开头,以 ab 结束,中间可以包含任意多个字符的字符串。

符号_(下画线)表示任意一个字符,如 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;再如 a__b 表示所有以 a 开头,以 b 结束的 4 个字符的字符串。其中第 2、第 3 个字符为任意字符。

【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%'
```

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 1 条下画线代表 1 个汉字;如果是西文,则 1 条下画线代表 1 个字符。

【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%'
```

【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。因此，该查询等价于下面的查询：

```
SELECT studentNo, studentName
FROM Student
WHERE nation='蒙古族'
```

如果用户的查询字符串中本身要包含通配符 % 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“22_”符号的班级名称。

```
SELECT className
FROM Class
WHERE className LIKE '%22\_%' ESCAPE '\'
```

ESCAPE \ 表示 \ 为换码字符，这样紧跟在 \ 符号后的 _ 不是通配符，而是普通的用户要查询的符号。查询结果如图 3-20 所示。

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22_01班
2	注册会计22_02班
3	注册会计22_03班

图 3-20 例 3.18 的
查询结果

6. 逻辑运算

SQL 提供了复合条件查询，使用 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。

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。例如，如果要在 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进行表示。通过连接运算表示该查询,请参见例 3.34 和例 3.36;通过嵌套子查询表示该查询,请参见例 3.44 和例 3.45。

【例 3.20】 在 Student 表中查询 2003 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)=2003 AND nation='汉族'
```

【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 表中查询成绩为 80~90 分的同学学号、课程号和相应成绩。

```
SELECT studentNo, courseNo, score
FROM Score
WHERE score>= 80 AND score<=90
```

【例 3.23】 在选课 Score 表中查询成绩不为 80~90 分的同学学号、课程号和相应成绩。

```
SELECT studentNo, courseNo, score
FROM Score
WHERE score<80 OR score>90
```

3.2.3 排序运算

SQL 支持排序运算,通过使用 ORDER BY 子句实现,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,默认按升序(ASC)排序;若要按降序排序,必须指明 DESC 选项。

该运算表示的含义是: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,以此类推。

【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
```

查询结果如图 3-21 所示。

【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。

	studentName	native	classNo
1	黄小红	云南	CS2201
2	刘小华	云南	IS2201
3	黄勇	九江	CP2202
4	刘方晨	九江	IS2101
5	刘晨	九江	IS2201
6	刘宏昊	福建	IS2201
7	吴敏	福建	CP2202
8	李立	福建	IS2201
9	李志强	北京	CP2202
10	王立红	北京	CS2201

图 3-21 例 3.24 的查询结果

	studentNo	studentName	classNo	birthday
1	2200010	李宏冰	CP2202	2003-03-09 00:00:00.000
2	2200015	吴敏	CP2202	2002-01-20 00:00:00.000
3	2200008	黄小红	CS2201	2004-08-09 00:00:00.000
4	2100002	刘方晨	IS2101	2003-11-11 00:00:00.000
5	2100003	王红敏	IS2101	2002-10-01 00:00:00.000
6	2200002	刘晨	IS2201	2003-11-11 00:00:00.000
7	2200003	王敏	IS2201	2003-10-01 00:00:00.000
8	2200007	李立	IS2201	2004-08-21 00:00:00.000
9	2200013	刘小华	IS2201	2004-07-16 00:00:00.000

图 3-22 例 3.25 的查询结果

3.2.4 查询表

查询表是指一个查询的查询结果所对应的表。SQL 中的 FROM 子句后面可以是基本表、视图,还可以是查询表。

【例 3.26】 查询 2004 年出生的“女”同学基本信息。

分析: 可以先构造一个查询从学生表中将女生记录查询出来,并将查询结果作为一个查询表,然后再对该查询表进行选择、投影操作。

```
SELECT studentNo, studentName, birthday
FROM (SELECT * FROM Student WHERE sex='女') AS a
WHERE year(birthday)=2004
```

该查询在 FROM 子句的后面是一个查询表,表示对该查询表进行查询,必须为查询表取一个名称(该名称称为**元组变量**),如使用 AS a 取名为 a,也可以省略“AS”,写成

```
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)=2004 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,其统计功能是通过聚合函数来实现,分类功能是通过分组子句来实现,并且统计和分组往往结合在一起实现丰富的查询功能。

1. 聚合函数

SQL 查询提供的聚合函数(aggregate function)主要包括以下几个。

- count([DISTINCT | ALL] { * | <表达式> }): 统计关系的元组个数或关系中指定表达式的值的个数。
- sum([DISTINCT | ALL] <表达式>): 统计关系中指定表达式的值的总和(此表达式必须为数值型)。
- avg([DISTINCT | ALL] <表达式>): 统计关系中指定表达式的值的平均值(此表达式必须为数值型)。
- max([DISTINCT | ALL] <表达式>): 统计关系中指定表达式的值的最大值。
- min([DISTINCT | ALL] <表达式>): 统计关系中指定表达式的值的最小值。

如果使用 DISTINCT 关键字,表示在计算时首先消除关系中<表达式>取重复值的元组,然后再进行统计;如果使用 ALL 关键字(这是默认方式),表示不消除关系中<表达式>取重复值的元组。

【例 3.27】 查询蒙古族的学生总人数。

```
SELECT count( * )
FROM Student
WHERE nation='蒙古族'
```

查询结果如图 3-23(a)所示,可以看出,输出的查询结果没有列名。为了便于理解,可以对计算列取一个列名,上述查询可修改为如下,其查询结果如图 3-23(b)所示。

```
SELECT count( * ) 学生人数
FROM Student
WHERE nation='蒙古族'
```

	(No column name)
1	4

(a) 不带列名的输出结果

	学生人数
1	4

(b) 带列名的输出结果

图 3-23 例 3.27 的查询结果

【例 3.28】 查询所有选课学生的人数。

```
SELECT count(studentNo) 学生人数
FROM Score
```

该查询的结果是 80。由于一个学生可以选修多门课程,学号存在重复,上述查询没有消除重复元组。为了消除重复的元组,必须使用 DISTINCT 关键字,可将查询修改为:


```
SELECT count(DISTINCT studentNo) 学生人数
FROM Score
```

此时的查询结果为 10。

【例 3.29】 查询学号为 2200003 同学所选课程的平均分。

```
SELECT avg(score) 平均分
FROM Score
WHERE studentNo='2200003'
```

在聚合函数遇到空值时,除 count(*)外,其他函数皆跳过空值,只处理非空值。

2. 分组聚合

在 SQL 查询中,往往需要对数据进行分类运算(即分组运算),分组运算的目的是细化聚合函数的作用对象。如果不对查询结果进行分组,则聚合函数作用于满足 WHERE 选择条件的整个查询结果,最后输出聚合计算结果;如果对满足 WHERE 选择条件的查询结果进行分组,则聚合函数分别作用于每个组中的所有元组,最后按组输出聚合计算结果。SQL 语句中通过使用 GROUP BY 和 HAVING 子句来实现分组运算,其中:

- GROUP BY 子句对满足 WHERE 选择条件的查询结果按某一列或某几列进行分组,值相等的分为一组;
- HAVING 子句对分组的聚合计算结果进行选择,仅输出满足 HAVING 条件的组的聚合计算结果。该子句必须与 GROUP BY 子句配合使用。

【例 3.30】 查询至少有 6 门选修课程已获得学分(成绩大于或等于 60 分才能获得学分)的学生情况,要求输出学号以及该同学已获得学分的选课门数、平均分和最高分。

```
SELECT studentNo, count(*) 门数, avg(score) 平均分, max(score) 最高分
FROM Score
WHERE score>=60
GROUP BY studentNo
HAVING count(*)>=6
```

查询结果如图 3-24 所示。

该查询首先按 WHERE 子句中的查询条件 score>=60 筛选出已获得学分的学生选课元组;然后按学号 studentNo 进行分组,将具有相同 studentNo 值的元组作为一组,并对每组中的元组进行相应的聚合计算,包括计数(统计选课门数)、求平均分和最高分;最后按 HAVING 条件(即计数结果大于或等于 6)对分组聚合计算结果元组进行筛选,并输出符合要求的结果元组。

	studentNo	门数	平均分	最高分
1	2100001	10	82.300000	98.0
2	2100003	6	74.333333	90.0
3	2100004	7	74.142857	88.0
4	2100005	11	78.636363	90.0
5	2200004	6	80.500000	90.0
6	2200012	8	79.500000	90.0
7	2200014	6	77.333333	90.0

图 3-24 例 3.30 的查询结果

【例 3.31】 查询学生人数不少于 3 人的每个民族的人数和平均年龄。

```
SELECT nation, count(*) 人数, avg(year(getdate())-year(birthday)) 平均年龄
FROM Student
GROUP BY nation
HAVING count(*)>=3
```

该查询按民族 nation 进行分组,将具有相同 nation 值的元组作为一组,然后对每组中的元组进行相应的计数(即计算学生人数)和求平均值(即对表达式 $\text{year}(\text{getdate}()) - \text{year}(\text{birthday})$ 求平均值),并判断每组的计数结果(即学生人数)是否大于或等于 3,如果是则输出该组的聚合计算结果元组,否则丢弃该组元组,不作为输出结果。

注意: 例 3.29 和例 3.30 的结果中是将重修课程所取得的多个成绩都参与计算的。对于一门课程,不管你重修该课程多少次,应该是只有一个成绩参与计算,通常是将该课程多次选修中最高的一个成绩或最后一次选修的成绩认定为该课程的成绩。

3.3 连接查询

前面的查询实例都是针对一个关系进行操作,而在实际应用中,往往会涉及多个关系的查询,这时需用到连接运算或子查询,本节介绍连接运算。

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

等值与非等值连接运算是在 WHERE 子句中加入连接多个关系的连接条件,其格式为:

```
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 $\neq$)。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。

1. 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。

分析:

(1) 该查询的结果为同学的学号、姓名、籍贯、班级编号和所在班级名称,在 SELECT 子句中必须包含这些属性。

(2) 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,因此 FROM 子句必须包含 Class 表和 Student 表。

(3) 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 classNo 相等,因此在 WHERE 子句中必须包含连接条件 $\text{Student.classNo} = \text{Class.classNo}$ 。

(4) 本查询要查询出会计学院的学生记录,因此在 WHERE 子句中还必须包括选择条件 $\text{institute} = \text{'会计学院'}$ 。

(5) 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 子句中的 Student.classNo = Class.classNo 为连接条件,institute='会计学院'为选择条件。

为了简化,可为参与连接的表取别名(称为**元组变量**),这样在相同的属性名前加上表的别名就可以了,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'
```

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。对于多个表中的不同属性名,可以不要在属性名前加上表名。

注意: 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。本例中班级号 classNo 在学生表 Student、班级表 Class 两个关系中同时出现,但在 SELECT 子句中仅需要出现 1 次,因此使用 *b.classNo*,也可以使用 *a.classNo*。其他列名是唯一的,不需要加上元组变量。

【例 3.33】 查找选修了课程名称为“计算机原理”的同学的学号、姓名。

分析:

(1) 该查询的结果为学号、姓名,在 SELECT 子句中必须包含这些属性。

(2) 由于学生的学号和姓名在学生表中,课程名称在课程表中,因此 FROM 子句必须包含学生表 Student、课程表 Course;由于学生表与课程表之间是多对多联系,需要通过成绩表转换为两个多对一的联系,因此 FROM 子句还必须包含成绩表 Score。

(3) 由于课程号既是课程表的主码,也是成绩表的外码,这 2 个表的连接条件是课程号相等;学号既是学生表的主码,也是成绩表的外码,这 2 个表的连接条件是学号相等。因此在 WHERE 子句中涉及三个关系的连接,其连接条件为:

```
Course.courseNo=Score.courseNo AND Score.studentNo=Student.studentNo
```

(4) 本查询要查找出选修了课程名称为“计算机原理”的课程的同学情况,因此在 WHERE 子句中还必须包括选择条件 courseName='计算机原理'。

(5) 本查询语句为:

```
SELECT a.studentNo, studentName
FROM Student a, Course b, Score c
WHERE b.courseNo=c.courseNo AND c.studentNo=a.studentNo
      AND b.courseName='计算机原理'
```

本例使用了元组变量,其连接条件为: