第 3 章 MySQL 数据定义与操作

关系数据库的标准语言是 SQL(Structured Query Language,结构化查询语言)。虽然 SQL 的字面含义是“查询语言”,但其功能却包括数据定义、查询、更新和保护等许多内容。实际上,各种不同数据库管理系统对 SQL 的实现都存有小的差异。以 MySQL 为例,本章讲述基本数据定义和常用查询及更新;第 4 章讲述数据保护;第 5 章讲述应用环境中的 MySQL。

3.1 SQL 与 MySQL

3.1.1 SQL 发展史

1970 年,美国 IBM 研究中心的 E.F.codd 连续发表多篇论文,提出关系模型。1972 年,IBM 公司开始研制实验型关系数据库管理系统 SYSTEM R,配制的查询语言称为 SQUARE(Specifying Queries As Relational Expression)语言。1974 年,同一实验室的 R. F. Boyce 和 D. D. Chamberlin 对 SQUARE 进行改进,减少了一些数学符号,采用英语单词表示和结构式的语法规则,并重命名为 SEQUEL(Structured English Query Language)语言;后来 SEQUEL 简称 SQL(Structured Query Language),即“结构化查询语言”。

1986 年,美国国家标准化协会(ANSI)发布了 ANSI 文件 X5.135—1986《数据库语言 SQL》,1987 年 6 月,国际标准化组织(ISO)采纳其为国际标准,现在称为“SQL86”。ANSI 在 1989 年 10 月又颁布了增强完整性特征的 SQL89 标准。随后,ISO 对标准进行了大量的修改和扩充,在 1992 年 8 月发布了标准化文件 ISO/IEC9075:1992《数据库语言 SQL》。人们习惯称这个标准为“SQL2”。1999 年,ISO 发布了标准化文件 ISO/IEC9075:1999《数据库语言 SQL》,人们习惯称这个标准为“SQL3”。实践中,各种数据库管理系统在其实现中都对 SQL 规范做了改动。在未来很长一段时间里,SQL 仍将是数据库领域的主流语言。不仅如此,像 SQL 一样简单是数据管理系统的普遍追求,万维网联盟(W3C)发布的 XML 数据查询语言标准 XQuery 就模仿了 SQL,RDF 数据查询语言标准 SPARQL 被称为 RDF 上的 SQL,大数据领域也已将提供类 SQL 的查询语言作为大数据管理的重要技术目标之一,比如微软的 DryadLINQ 就使用类似 SQL 的非过程化声明式语言。

3.1.2 MySQL

MySQL是最流行的关系型数据库管理系统之一，也是Web应用方面最好的RDBMS(Relational Database Management System，关系数据库管理系统)软件之一。MySQL分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择MySQL作为网站数据库。与PostgreSQL、Oracle、DB2、SQL Server等相比，MySQL自有它的不足之处，但是这丝毫也没有减少它受欢迎的程度。对于一般的个人使用者和中小型企业来说，MySQL提供的功能已经绰绰有余，而且由于MySQL是开放源码软件，因此可以大大降低总体拥有成本。

Linux作为操作系统、Apache或Nginx作为Web服务器，MySQL作为数据库，PHP/Perl/Python作为服务器端脚本解释器。由于这四个软件都是免费或开放源码软件(FLOSS)，因此使用这种方式不用花一分钱(除人工成本)就可以建立起一个稳定、免费的网站系统，被业界称为LAMP或LNMP组合。

3.1.3 数据库语言组成

数据库语言基本部分主要有：①数据定义语言，即SQL DDL，用来创建数据库中的各类对象，如表的创建、撤销与更改；②数据操作语言，即SQL DML，分为数据查询和数据更新，用于实现投影、选择、笛卡儿积和联接、聚集查询和数据插入、删除和修改操作；③数据保护语言，即SQL DPL，可用来授予或收回访问数据库的权限，协调事务间的动作等，由DBMS提供统一数据保护功能，维护数据的保密性、完整性和可用性，是数据库系统的主要特征之一。

过去，SQL是面向非过程编程与操作，需要和其他语言结合使用。MySQL能和其他语言如C语言自然融合。另外，提供ODBC和JDBC的原生接口，同时为大多数编程语言提供绑定接口，包括C、C++、PHP、Perl、Tcl/tk、Python、Ruby、VB、VC、Delphi、ASP等高级语言。

3.1.4 数据库语言的特点

(1) SQL是非过程化的第四代编程语言。用户无须指定对数据的存放方法。因此，其他语言需要一大段程序才能实现的功能，SQL只需要一个语句就可以实现。SQL具有十分灵活和强大的查询功能，其SELECT语句能完成相当复杂的查询操作。有人把非过程化的编程语言称为第四代语言，特点是只要提出“做什么”，而无须说明“怎么做”。

(2) 一种语言多种使用方式。SQL是“自含式”语言，也是“嵌入式”语言。作为自含式语言，SQL不仅可以作为程序语言进行编程使用，而且可以作为交互命令使用，用户可以在终端键盘上直接输入SQL命令对数据库进行操作。作为嵌入式语言，SQL还可以嵌入到高级语言的程序中。在两种不同的使用方式下，SQL的语法结构基本一致。

(3) SQL是关系数据库的标准语言，作为数据存取共同标准接口，可使不同数据库连接成一个统一的整体，有利于各种数据库之间交换数据，有利于程序的移植，有利于实现高度的数据独立性。SQL的影响已经超出了数据库领域本身，不少软件产品将SQL的

数据查询功能与多媒体图形功能、软件工程工具、软件开发工具和人工智能程序相结合，显示出了相当大的潜力，应用领域前途无量。近年来，随着 Internet 技术的迅猛发展和快速普及，人们在 HTML 和 XML 中加入 SQL 语句，通过 WWW 来访问数据库，使得 SQL 的应用更加广泛和深入。

(4) 结构简洁、易学易用。SQL 是面向数据的语言，它集数据定义、数据操作和数据保护等数据库必需的基本功能为一身，充分体现了关系数据库的本质特点和巨大优势。SQL 不但功能极强，而且设计构思非常巧妙，语言结构简洁明快，语句非常接近英语语句，特别易学易用。

3.1.5　考试系统数据库

本节以考试系统数据库为例。考试系统包括一个题目数据库，里面有大量各个专业、各门课程、各类考试(包括考试、测验和练习)可用的题目。各个院系的考官每次可以从中选择合适的题目组成试卷，称为组卷。考生可以根据需要报考某份试卷，答卷后获得一个分数，称为答卷。考试系统数据库 T-E-S 如下。

考生表：examinee(eeid，eename，eesex，eeage，eedepa)，如表 3-1 所示。

考官表：examiner(erid，ername，ersex，erage，ersalary，erdepa)，如表 3-2 所示。

试卷表：exampaper (eid，ename，etype，eduration)，如表 3-3 所示。

考生答卷表：eeexam(eeid，eid，achieve)，如表 3-4 所示。

考官组卷表：erexam(erid，eid)，如表 3-5 所示。

院系表：department(dname，dloca，dtele)，如表 3-6 所示。

表 3-1　考试系统数据库考生表 examinee

eeid	eename	eesex	eeage	eedepa
278811011013	刘诗诗	男	20	历史学院
278811011014	刘诗诗	男	21	历史学院
278811011219	王琳懿	女	18	文学院
278811011220	王琳懿	女	19	文学院
278811011221	刘慧杰	女	19	文学院
278811011117	刘慧杰	女	19	教育学部
278811011025	张立帆	男	20	心理学院
278811011027	张立帆	男	19	心理学院
278811011028	刘慧杰	男	20	心理学院

表 3-2　考试系统数据库考官表 examiner

erid	ername	ersex	erage	ersalary	erdepa
2009040	成志云	女	35	5000	历史学院
1990122	戴小刚	男	53	8000	教育学部

续表

erid	ername	ersex	erage	ersalary	erdepa
1998039	丁向军	女	42	6500	文学院
2011049	郑博宇	男	32	5000	物理系
2007033	李晓燕	女	38	5000	心理学院
1995057	林永强	男	49	6500	历史学院
2013069	王瑞芬	女	30	5000	心理学院
2010022	姚翠红	女	36	5000	物理系

表 3-3　考试系统数据库试卷表 exampaper

eid	ename	etype	eduration
0205000002	中国近现代史纲要	4	100
0210000001	大学外语	2	180
0201020001	计算机应用基础	4	120
0211000001	大学美育	3	120
0219001014	普通物理学	4	100
0110001001	教育学	1	180
0110001002	心理学	1	180

表 3-4　考试系统数据库考生答卷表 eeexam

eeid	eid	achieve
278811011013	0205000002	92
278811011013	0210000001	85
278811011013	0201020001	88
278811011116	0210000001	90
278811011116	0201020001	80

表 3-5　考试系统数据库考官组卷表 erexam

erid	eid
2009040	0205000002
1998039	0211000001
2007033	0110001002
2010022	0219001014
1990122	0110001001

表 3-6　考试系统数据库院系表 department

dname	dloca	dtele
历史学院	主楼 B2	58809289
教育学部	英东教育楼	58808855
文学院	主楼 B7	58807998
物理系	物理楼	58808135
心理学院	后主楼 12	58807832

3.2　数 据 定 义

数据定义包括数据库对象的创建、删除和修改三个部分。MySQL 中对象标识符和保留字必须以一个字母(a～z 以及带变音符的字母和非拉丁字母)或下画线(_)开头,随后的字符可以是字母、下画线、数字(0～9)、美元符号($)。MySQL 中大小写不敏感,只有引号里面的字符才区分大小写。MySQL 建议 SQL 的保留字用大写字母;表名、属性名等所有数据库对象名全部使用小写字母。

表是数据库中最重要、最基本的操作对象,是数据存储的基本单位。数据在表中是按照行和列的格式来存储的。每行代表唯一一个元组,每列代表一个域。创建表就是约定表中各个属性的属性名及其数据类型(实际上还包括约束定义,将在第 4 章中讲述)。

3.2.1　MySQL 的基本数据类型

MySQL 支持的数据类型主要分为 3 类:数值类型、字符串(字符)类型、日期/时间类型。其中,数值类型包含 TINYINT、SMALLINT、MEDIUMINT、INT(INTEGER)、BIGINT、FLOAT、DOUBLE 和 DECIMAL。字符串(字符)类型主要有 CHAR、VARCHAR、TINYTEXT、TEXT、MEDIUMTEXT、LONGTEXT、TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。日期/时间类型有 DATETIME、DATE、TIMESTAMP、TIME 和 YEAR。

MySQL 支持的基本数据类型如表 3-7 所示。

表 3-7　MySQL 中的基本数据类型

类 型 名 称	存 储 空 间	描　　述
TINYINT	1 字节	小整数值
SMALLINT	2 字节	大整数值
MEDIUMINT	3 字节	大整数值
INT(INTEGER)	4 字节	大整数值
BIGINT	8 字节	极大整数值
FLOAT	4 字节	单精度浮点数值
DOUBLE	8 字节	双精度浮点数值

续表

类型名称	存储空间	描述
DECIMAL	对 DECIMAL(M,D)，如果 M>D，为 M+2 字节，否则为 D+2 字节	小数值
CHAR(M)	M 字节，其中，0≤M≤255	定长字符串
VARCHAR(M)	L+1 字节，其中，L≤M 和 0≤M≤255	变长字符串
TINYTEXT	L+1 字节，其中，L<2^8	短文本字符串
TEXT	L+2 字节，其中，L<2^16	长文本数据
MEDIUMTEXT	L+3 字节，其中，L<2^24	中等长度文本数据
LONGTEXT	L+4 字节，其中，L<2^32	极大文本数据
TINYBLOB	L+1 字节，其中，L<2^8	非常小的二进制字符串
BLOB	L+2 字节，其中，L<2^16	二进制形式的长文本数据
MEDIUMBLOB	L+3 字节，其中，L<2^24	二进制形式的中等长度文本数据
LONGBLOB	L+4 字节，其中，L<2^32	二进制形式的极大文本数据
YEAR	1 字节	年份值，格式 YYYY
TIME	3 字节	时间值或持续时间，格式 HH:MM:SS
DATE	3 字节	日期值，格式 YYYY-MM-DD
DATETIME	8 字节	混合日期和时间值，格式 YYYY-MM-DD HH:MM:SS
TIMESTAMP	4 字节	混合日期和时间值，时间戳，格式 YYYY-MM-DD HH:MM:SS

在字符串类型数据中，CHAR(M)指长度固定的字符串，M 表示列长度。VARCHAR(M)指长度可变的字符串，M 表示最大列长度，L 表示实际长度。

3.2.2 表的创建、修改和删除

对表结构的操作有创建、修改和删除三种操作。

1. 表的创建

创建表使用 CREATE TABLE 语句，该语句的一般格式为：

```
CREATE TABLE <表名>
(<列名><数据类型>[<列级别约束条件><默认值>]
 [,<列名><数据类型>[<列级别约束条件><默认值>] ]…
);
```

其中，[]内的是可选项，<表名>是所要定义的表名称。

表中每个列的类型可以是基本数据类型，也可以是用户预先定义的域名。主键是一种最基本的完整性约束，完整性约束将在第 4 章中详细介绍。下面举例说明。

【例 1】 对于考试系统数据库中的三个关系表：

试卷表：exampaper(eid,ename,etype)

考官表：examiner(erid,ername, ersex,erage, ersalary,erdepa)
考官制卷表：erexam(erid,eid)
表 examiner 可以用以下语句创建。

```
CREATE TABLE examiner
(
 erid VARCHAR(12),
 ername VARCHAR(20),
 ersex CHAR(2),
 erage SMALLINT,
 ersalary REAL,
 erdepa VARCHAR(20)
);
```

表 erexam 可以用以下语句创建。

```
CREATE TABLE erexam
(
 erid VARCHAR(12),
 eid VARCHAR(10)
);
```

表 exampaper 可以用以下语句创建。

```
CREATE TABLE exampaper
(
 eid VARCHAR(10),
 ename VARCHAR(20),
 etype SMALLINT
);
```

由上述语句可以看到：
(1) 表的定义就是逐列说明属性名称和属性类型。
(2) 每个语句末尾用分号“;”表示语句结束。
(3) 用 CREATE 语句创建的表，最初只是一个空的框架，接下来，可以用 INSERT 命令把数据插入表中(见 3.12 节)。另外，关系数据库产品都有数据装载程序，可以把大量原始数据导入表中。

2. 表结构的修改

在表建立并使用一段时期后，可能需要对表的结构进行修改，比如增加新的属性、删除原有的属性或修改数据类型、宽度等。SQL 使用 ALTER TABLE 语句进行表结构修改。在进行表更新时，如果是新增属性，需要新属性一律为空值；如果是修改原有属性，则要注意是否可能破坏已有数据。

(1) 增加新的属性用“ALTER…ADD…”语句，其句法如下。

```
ALTER TABLE <表名>ADD <列名><类型>
```

【例 2】 在表 examiner 中增加一个地址(address)列，可用下列语句。

```
ALTER TABLE examiner ADD address VARCHAR(30);
```

表在增加一列后，原有元组在新增加的列上的值都被定义为空值(NULL)。

(2) 删除原有的属性用“ALTER…DROP…”语句，其句法如下。

```
ALTERTABLE <表名>DROP  <列名>
```

【例 3】 在表 examiner 中删除地址(address)列，可用下列语句。

```
ALTER TABLE examiner DROP address;
```

(3) 修改属性的数据类型，使用下面的命令。

```
ALTER TABLE <表名>MODIFY <属性名><数据类型>;
```

【例 4】 把表 exampaper 中类型(etype)属性改为普通整型，可用下列语句。

```
ALTER TABLE exampaper MODIFYetype INT;
```

(4) 修改属性的属性名，使用下面的命令。

```
ALTER TABLE <表名> CHANGE <旧属性名><新属性名><新数据类型>;
```

【例 5】 把表 exampaper 中类型(etype)属性名改为 examtype，可用下列语句。

```
ALTER TABLE exampaper CHANGE etype examtype SMALLINT;
```

3. 表的改名

修改表名的句法如下。

```
ALTER TABLE <旧表名>RENAME [TO] <新表名>;
```

【例 6】 把表 exampaper 的名字改为 epaper，可用下列语句。

```
ALTER TABLE exampaper RENAME epaper;
```

4. 表的撤销

在表不再需要时，可以用 DROP TABLE 语句撤销。在一个表撤销后，其所有数据也就丢失了。

撤销语句的句法如下。

```
DROP TABLE [IF EXISTS] <表名>
```

可以同时删除多张数据表，多个表名之间用逗号分隔。

【例 7】 撤销表 examiner，可用下列语句实现。

```
DROP TABLE examiner;
```

3.3 投影与广义投影

投影是指选取表中的某些属性的属性值。广义投影是对投影的扩展，指在选取属性列时，允许进行算术运算，即允许涉及属性和常量的算术表达式。语法如下。

```
SELECT [ALL|DISTINCT] <目标列表达式>[,<目标列表达式>] …
FROM <表名>[, <表名>] …
```

说明：

(1) 查询输入是 FROM 子句中列出的关系。

(2) 最简单的(<目标列表达式> [, <目标列表达式>] …)是 *，它输出 FROM 子句中表的所有属性；否则，它是一个逗号分隔的表达式列表。目标列表达式可能是一个属性名，也可以是任意常量算术表达式。如果是一般表达式，那么原理上向返回的表中增加一个新的虚拟属性。目标列表达式为结果中的每一行进行一次计算，计算之前用该行的数值替换任何目标列表达式里引用的属性。

(3) 投影结果中可能出现各个属性值均相等的元组对，但从数据库管理系统实现的角度看，投影过程会对每个新产生的结果元组进行标识，即能把每个元组视为不同。也就是说，由于去重是一项耗时的工作，DBMS 采取惰性原则：除非用户明确指出要求去重(即在 SELECT 保留字后跟 DISTINCT)，否则认为每个元组皆不同。SELECT 保留字后跟 ALL 指要求保留重复。

(4) 可以对结果表中行的显示次序进行控制。ORDER BY 子句让查询结果中的行按一个或多个属性(表达式)进行排序，升序时用 ASC，排序列为空值的行最先显示；降序时用 DESC，排序列为空值的行最后显示；默认为升序。

(5) 投影和广义投影都不会对原表产生任何改变，其他查询类似。

【例 8】 查询全部试卷的试卷号与试卷名。

试卷号和试卷名全部数据都存在表 exampaper 中，所以 FROM exampaper。需要列出每一个 eid 和每一个 ename 的值，所以 SELECT eid,ename。整个查询语句就是：

```
SELECT eid,ename
FROM exampaper;
```

这个语句的执行过程就是对 exampaper 表输出其每一行的试卷号 eid 和试卷名 ename 值。

【例 9】 查询全部试卷号 eid 对应的组卷考官号，即输出所有试卷号、考官号。

试卷号和考官号的对应关系全部都存在表 erexam 中，所以 FROM erexam。需要列出每一个 eid 和每一个 erid 的值，所以 SELECT eid,erid。整个查询语句就是：

```
SELECT eid, erid
FROM erexam;
```

【例 10】 查询全部试卷本身的属性信息，也就是输出 exampaper 表中全部行的所有列。

试卷本身的属性信息都存在 exampaper 表中(注意：试卷报考信息存在 eeexam 表中、试卷组卷信息存在 erexam 表中，但是这些数据并不是查询要求的)，所以 FROM exampaper。需要输出 exampaper 表每一行的所有属性列，可以用 SELECT * 或在 SELECT 后面列出该表的所有属性列名。完整的查询语句是：

```
SELECT eid, ename, etype, eduration
FROM exampaper;
```

或

```
SELECT  *
FROM exampaper;
```

【例 11】 查询每位考生每门考试的扣分情况，即成绩与满分(100)之差。

```
SELECT eeid, 100-achieve
FROM eeexam;
```

该查询结果如表 3-8 所示。

表 3-8 查询结果

eeid	100-achieve	eeid	100-achieve
278811011013	8	278811011116	10
278811011013	15	278811011116	20
278811011013	12		

【例 12】 查询全体考官情况，查询结果按所在院系名升序排列，同一学院中的考官按年龄降序排列。

```
SELECT  *
FROM  examiner
ORDER BY erdepa ASC, erage DESC;
```

3.4 选　　择

选择操作从关系表中选择一些满足特定条件的元组行，比如选择属性满足一些条件(谓词表达式)的元组行，或者无条件选择关系表中的所有元组行。

【例 13】 查询 examiner 表中所有考官本身的属性信息。

```
SELECT *
FROM examiner;
```

【例 14】 查询工资大于 5000 元的所有考官信息。

```
SELECT *
FROM examiner
WHERE ersalary>5000;
```

通常，保留字如 FROM、WHERE 等另起一行与 SELECT 左对齐，这种风格显得直观易读；FROM 子句给出查询所涉及的关系表；WHERE 子句给出查询条件，像关系代数中的选择条件，只有满足这个条件的元组行才出现在查询结果中；SELECT 子句给出满足查询条件

元组行的哪些属性(或目标列表达式)出现在查询结果中。SELECT…FROM…WHERE 查询的一种解释是,首先考虑 FROM 子句提及关系表中的每个元组行,选择出其中使 WHERE 子句条件为真的,按 SELECT 子句出现的属性序列构造查询结果中的元组行。WHERE 子句中的条件表达式可以用各种运算符组合而成,常用的运算符如表 3-9 所示。

表 3-9　常用的运算符

运算符名称	符号及格式	说　明
比较	=,>,<,>=,<=,! =,<>	比较两个表达式的值
确定范围	<表达式 1> [NOT] BETWEEN <表达式 2> AND <表达式 3>	(不)在给定范围
是否空值	<表达式 1> IS [NOT] NULL	判断是否为空
是否属于集合	<元组行> [NOT] IN <集合>	判断某元组行是否在某集合内
限定比较判断	<元组行> ALL|SOME|ANY (<集合>)	元组行与集合中每一个或某一个元组行满足比较
存在判断	[NOT] EXISTS(<集合>)	判断集合中是否存在一个元组行
串匹配	[NOT] LIKE	%:与零个或多个字符组成的字符串匹配; _:与单个字符匹配;ESCAP:定义转义符
逻辑表达式	AND,OR,NOT	复合多个查询条件

下面给出一些典型选择查询的示例,其中有些也涉及了投影。

1. 比较选择

【例 15】 查询历史学院全体考官。

```
SELECT *
FROM examiner
WHERE erdepa='历史学院';
```

【例 16】 查询所有工资在 5800 元以上的考官,按工资数升序排列。

```
SELECT *
FROM  examiner
WHERE ersalary>5800
ORDER BY ersalary;
```

【例 17】 查询工资不到 10000 元的考官。

```
SELECT *
FROM  examiner
WHERE ersalary<=10000;
```

2. 范围选择

【例 18】 查询工资为 6000～9000 元(包括 6000 元和 9000 元)的考官,按工资降序

排列。

```
SELECT *
FROM  examiner
WHERE ersalary BETWEEN 6000 AND 9000
ORDER BY ersalary DESC;
```

【例 19】 查询工资不为 16000～19000 元(包括 16000 元和 19000 元)的考官。

```
SELECT *
FROM  examiner
WHERE ersalary NOT BETWEEN 16000 AND 19000;
```

3. 空值选择

【例 20】 查询尚未登记联系电话的院系。

```
SELECT  *
FROM  department
WHERE  dtele IS NULL;
```

【例 21】 查询已经登记联系电话的院系。

```
SELECT  *
FROM  department
WHERE  dtele IS NOT NULL;
```

4. 集合归属选择

【例 22】 查询历史学院和心理学院的考官。

```
SELECT *
FROM  examiner
WHERE erdepa  IN ('历史学院', '心理学院');
```

【例 23】 查询既不是历史学院也不是心理学院的考官。

```
SELECT *
FROM  examiner
WHERE erdepa NOT IN ('历史学院', '心理学院' );
```

5. 串匹配选择

字符串的匹配可以使用“＝”“LIKE”及正则表达式运算符。“＝”要求两边字符串严格相同;“LIKE”允许使用通配符“_”(下画线匹配任何单个字符)和“%”(一个百分号匹配零个或多个任何字符)。如果不使用通配符,“LIKE”和“＝”等价。通过正则表达式进行字符匹配时,使用 REGEXP 操作符。正则表达式中,“^”表示以该符号后紧跟的字符或字符串开头的字符串;“$”表示以该符号紧跟的字符或字符串结尾的字符串;“.”表示任意一个字符;“*”表示前面的字符出现任意多次或 0 次;“+”表示前面的字符至少出现一次;“[]”表示一个字符集合中任意一个字符;[^]表示不在括号中的任何字符;“{n}”表示前面的字符出现 n 次;“{m,n}”表示前面的字符出现至少 m 次,最多 n 次。正则表达式

运算符和 LIKE 一样，默认转义符为“\”，也可以用 ESCAPE 定义别的字符作为转义符。转义符后的元字符及所定义的转义符都代表作为普通字符的自己。

1）匹配串为固定字符串

【例 24】 查询联系电话为 58807998 的院系。

```
SELECT *
FROM  department
WHERE  dtele LIKE '58807998';
```

等价于：

```
SELECT *
FROM  department
WHERE  dtele='58807998';
```

2）匹配串为含通配符的字符串

【例 25】 查询所有以“学院”命名的院系，如文学院、数学科学学院等。

```
SELECT  *
FROM department
WHERE dname LIKE '%学院';
```

等价于：

```
SELECT  *
FROM department
WHERE dname REGEXP '学院$';
```

【例 26】 查询所有以“系”命名，且全名为三个汉字的院系，如物理系、天文系等。

```
SELECT *
FROM  department
WHERE  dname LIKE '__系';
```

【例 27】 查询院系名中第二个字为“学”字的院系，如文学院、化学系等。

```
SELECT *
FROM department
WHERE dname LIKE '_学%';
```

【例 28】 查询所有院系名不以“教育”开头的院系。

```
SELECT *
FROM department
WHERE dname NOT LIKE '教育%';
```

3）使用转义字符将通配符转义为普通字符

【例 29】 查询“大学外语_”试卷的试卷号和类别。

```
SELECT eid, etype
```

```
FROM exampaper
WHERE ename LIKE '大学外语\_ ';
```

【例 30】 查询以“数据库_”开头的试卷。

```
SELECT  *
FROM  exampaper
WHERE  ename LIKE  '数据库*_%'  ESCAPE  '*';
```

其中，ESCAPE '*'表示“*”为转义字符。

6. 逻辑表达式选择

【例 31】 查询历史学院年龄在 58 岁以上、工资尚不足 15000 元的考官。

```
SELECT  *
FROM  examiner
WHERE erdepa='历史学院' AND erage>58 AND ersalary<=15000;
```

【例 32】 查询历史学院和心理学院的女考官。

```
SELECT  *
FROM  examiner
WHERE  ersex='女' AND (erdepa='历史学院' OR erdepa='心理学院');
```

基于 ALL、SOME、ANY 的限定比较选择，基于 EXISTS、NOT EXISTS 的存在性选择和基于 UNIQUE、NOT UNIQUE 的唯一性选择，大多出现在 3.10 节所述的嵌套查询中。

3.5 集 合 操 作

并、交、差查询对应于数学集合论中的∪、∩和－运算。当两个子查询结果的结构完全一致时，可以让这两个子查询执行并、交、差操作。多数数据库软件支持并、交、差操作，对应的运算符一般为 UNION、INTERSECT 和 EXCEPT/MINUS，但遗憾的是 MySQL 只支持并操作，不支持交、差操作，只能通过其他方法间接实现交、差操作。以下为并操作的实现语句。

```
(SELECT 查询语句 1)
UNION [ALL]
(SELECT 查询语句 2)
```

为了能够计算两个查询的并，这两个查询必须是“兼容的”，也就是它们有同样数量的列，并且对应列的数据类型是兼容的。集合操作中不带保留字 ALL 时，默认像 DISTINCT 那样删除结果中所有重复的行，返回结果表自动消除重复元组；而声明了 ALL 时，返回结果保留重复元组。如果表 t1 中行 r 出现 n 次，而表 t2 中行 r 出现 m 次，那么在 TABLE t1 UNION ALL TABLE t2 中行 r 将出现 n＋m 次。集合操作也可以嵌套和级联。

【例 33】 查询历史学院的考官和工资不到 6000 元的考官。

方法一：

```
SELECT *
```

```
FROM examiner
WHERE erdepa='历史学院'
UNION
SELECT *
FROM examiner
WHERE ersalary<=6000;
```

方法二：

```
SELECT DISTINCT  *
FROM examiner
WHERE erdepa='历史学院' OR ersalary<=6000;
```

3.6　联 接 查 询

可以通过笛卡儿积(交叉联接)、自然联接、属性联接、条件联接和外联接等实现多表查询，即从多个表中进行查询，最常使用的联接形式是自然联接。

对于两个关系表的联接操作，联接操作符分成联接类型和联接条件两部分。联接类型决定了如何处理联接条件中不匹配的元组，即分为内联接和(左/右/全)外联接。联接条件决定了两个关系表中哪些元组应该匹配，以及联接结果中出现哪些属性，其中，如果条件为永真则等价于交叉联接；如果条件为全部共同属性值相等则等价于自然联接。

MySQL 中联接的一般形式如下。

```
t1{ [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN t2 ON <逻辑表达式>
t1{ [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN t2 USING (列 1,…,列 n)
t1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } JOIN t2
```

INNER 和 OUTER 对所有联接类型都是可选的。默认为 INNER；LEFT、RIGHT、FULL 均隐含外联接。联接条件在 ON、USING 子句中声明，或用保留字 NATURAL 隐式声明，联接条件用来判断来自两个源表中的哪些行是“匹配”的。

ON 子句接收一个和 WHERE 子句相同的布尔表达式，如果两个分别来自 R_1 和 R_2 的元组在 ON 表达式上运算的结果为真，那么它们就算是匹配的行。

USING 接收一个用逗号分隔的属性名列表，这些属性必须是联接表共有的，其值相同时则匹配。JOIN USING 将每一对同名的输入属性输出为一个属性。

NATURAL 是 USING 的缩写形式：自动形成一个由两个表中同名的属性组成的 USING 列表(同名属性在结果中只出现一次)。

3.6.1　笛卡儿积

基本查询包括三个子句：SELECT 子句、FROM 子句和 WHERE 子句。在写多表联接查询语句时，查询涉及多个表，最简单直接的方法就是在 FROM 后面依次写上这些表名，并以逗号或 CROSS JOIN 分隔，FROM 子句的结果表就是这些表的笛卡儿积，结果表包含所有这些表的所有列，如果两个表中有同名列，MySQL 客户端并不会对其做出显

式区分，可根据查询表的顺序做列名表名一一对应。根据查询需要，还可通过 WHERE 子句对笛卡儿积结果表施加选择操作，以撷取那些符合查询条件的行。

【例 34】 查询每个考生及其报考试卷的情况

```
SELECT *
FROM  examinee,eeexam  /* 等价于 FROM examinee CROSS JOIN eeexam */
WHERE  examinee.eeid =eeexam.eeid;
```

通过笛卡儿积把来自 examinee 和 eeexam 中具有相同 eeid 的元组行进行匹配。

【例 35】 查询报考 0205000002 试卷且成绩在 90 分以上的所有考生的考试号和姓名。

```
SELECT examinee.eeid, eename
FROM  examinee, eeexam
WHERE examinee.eeid=eeexam.eeid AND
      eeexam.eid='0205000002'AND
      eeexam.achieve >=90;
```

3.6.2 内联接

1. 自然联接

自然联接(NATURAL JOIN)运算作用于两个关系表，并产生一个关系表作为结果。不同于笛卡儿积运算将第一个关系表的每个元组与第二个关系表的所有元组都进行联接，自然联接只考虑那些在两个关系表模式中都出现的属性上取值相同的元组对，即在两个关系表的公共属性上做等值联接，运算结果中公共属性只出现一次。

【例 36】 表 eeexam 与 examiner 的自然联接。

```
SELECT  *
FROM   examiner NATURAL JOIN erexam
```

或

```
SELECT  *
FROM   examiner NATURAL INNER JOIN erexam
```

【例 37】 查询报考 0205000002 试卷且成绩在 90 分以上的所有考生。

```
SELECT  eeid, eename
FROM    examinee NATURAL JOIN eeexam
WHERE   eeexam.eid='0205000002' AND eeexam.achieve >=90;
```

等价于：

```
SELECT examinee.eeid, eename
FROM  examinee, eeexam
WHERE  examinee.eeid =eeexam.eeid AND
       eeexam.eid='0205000002' AND
       eeexam.achieve >90;
```

在一个查询的 FROM 子句中，可以用自然联接将多个关系表结合在一起。

【例 38】　查询考官组卷的考官姓名及试卷名。

```
SELECT ername ename
FROM examiner NATURAL JOIN erexam , exampaper
WHERE erexam.eid=exampaper.eid;
```

或

```
SELECT ername ename
FROM examiner NATURAL JOIN erexam NATURAL JOIN exampaper;
```

2. 属性联接

属性联接，即属性内联接，是在笛卡儿积的基础上选取指定同名属性上取值相等的行，结果表中这些指定同名属性只出现一次。

【例 39】　查询各位考官所组试卷。

```
SELECT  *
FROM  examiner  JOIN  erexam USING(erid);
```

或

```
SELECT  *
FROM  examiner  INNER JOIN  erexam USING(erid);
```

属性联接与自然联接的区别在于当参与联接运算的两个表有多个同名属性时，自然联接的匹配条件是所有同名属性全部取值相等；而属性联接的匹配条件是指定其中若干同名属性取值相等，如果属性联接指定全部同名列来匹配则等价于自然联接。

3. 条件联接

条件联接，即条件内联接，是在笛卡儿积运算的基础上选取满足给定条件的行。条件联接把一个选择运算和一个笛卡儿积运算合并为单独的一个运算，用 JOIN…ON…实现。

【例 40】　查询考官姓名及所在学院办公电话。

```
SELECT ername, dtele
FROM examiner JOIN department ON examiner.erdepa=department.dname;
```

或

```
SELECT ername, dtele
FROM examiner INNER JOIN department ON examiner.erdepa=department.dname;
```

3.6.3　外联接

内联接可能会出现左表当中的一些行在右表中没有相匹配的行，或右表当中的一些行在左表中没有相匹配的行，这些没有找到匹配的行称为悬浮行。

内联接和外联接的区别就在于对悬浮行的处理不同。

内联接抛弃所有悬浮行；外联接对悬浮行的处理有三种方式：左外联接，右外联接，全外联接。三种方式都要首先计算内联接，然后再在内联接的结果中加上相应的左（右、

左和右）表中的悬浮行。

例如，左外联接是这样计算的：首先，计算内联接的结果；然后，把左侧表中的悬浮行加入结果表，这些行中来自右侧表的属性赋为空值 null。

与左外联接类似，右外联接就是把右侧表中的悬浮行补上空值后加入结果表。

全外联接是左外联接和右外联接的组合，左侧表中的悬浮行补上空值后加到结果表中，同时，右侧表中的悬浮行补上空值后也加到结果表中。

1. 基于自然联接的外联接

外联接运算可以避免悬浮元组信息的丢失。外联接运算有三种形式：左外联接，右外联接，全外联接。基于自然联接的外联接运算分别用 NATURAL LEFT OUTER JOIN、NATURAL RIGHT OUTER JOIN、NATURAL FULL OUTER JOIN 等来实现。

【例 41】 关系表 erexam 和 exampaper 的左外联接用 SQL 语句实现为：

```
SELECT *
FROM erexam NATURAL LEFT OUTER JOIN exampaper;
```

或

```
SELECT *
FROM erexam NATURAL LEFT JOIN exampaper;
```

2. 基于属性联接的外联接

基于属性联接的外联接运算分别用 LEFT OUTER JOIN… USING(…)、RIGHT OUTER JOIN… USING(…)、FULL OUTER JOIN… USING(…)等来实现。

【例 42】 查询各位考官情况及其组卷信息。

```
SELECT  *
FROM  examiner  LEFT JOIN  erexam USING(erid);
```

或

```
SELECT  *
FROM  examiner  LEFT OUTER JOIN  erexam USING(erid);
```

3. 基于条件联接的外联接

基于条件联接的外联接运算分别用 LEFT OUTER JOIN…ON…、RIGHT OUTER JOIN…ON…、FULL OUTER JOIN…ON…等来实现。

【例 43】 关系 examiner 和 department 基于学院名相等的左外联接。

```
SELECT ername, dtele
FROM examiner LEFT OUTER JOIN department ON examiner.erdepa=department.dname;
```

或

```
SELECT ername, dtele
FROM examiner LEFT JOIN department ON examiner.erdepa=department.dname;
```

表 3-10 给出了 MySQL 中各种联接查询的总结对比。

表 3-10　MySQL 各种联接查询的总结对比

运　　算	等价称谓	联接条件	考虑悬浮元组	运 算 符	结果列
笛卡儿积	交叉联接	无条件		“,”、CROSS JOIN	全部列
自然联接	自然内联接	所有同名属性之值相等 NATURAL	否 INNER	t_1 NATURAL JOIN t_2 t_1 NATURAL INNER JOIN t_2	同名属性在结果中只出现一次
条件联接	条件内联接	给出的布尔表达式为真 ON ＜逻辑表达式＞	否 INNER	t_1 JOIN t_2 ON ＜…＞ t_1 INNER JOIN t_2 ON ＜…＞	全部列
属性联接	属性内联接	给出的同名属性之值相等 USING ＜同名列序列＞	否 INNER	t_1 JOIN t_2 USING a_1,… t_1 INNER JOIN t_2 USING a_1,…	匹配的同名属性在结果中只出现一次
自然(左｜右｜全)外联接	(左｜右｜全)外联接	所有同名属性之值相等 NATURAL	是 OUTER	t_1 NATURAL (LEFT｜R｜GHT｜FULL)JOIN t_2 t_2 NATURAL (LEFT｜RIGHT｜FULL) OUTER JOIN t_2	同名属性在结果中只出现一次
条件(左｜右｜全)外联接		给出的布尔表达式为真 ON ＜逻辑表达式＞	是 OUTER	t_1(LEFT｜RIGHT｜FULL) JOIN t_2 ON＜…＞ t_1(LEFT｜RIGHT｜FULL) OUTER JOIN t_2 ON＜…＞	全部列
属性(左｜右｜全)联接		给出的同名属性之值相等 USING ＜同名列序列＞	是 OUTER	t_1(LEFT｜RIGHT｜FULL) JOIN t_2 USING a_1,… t_1(LEFT｜RIGHT｜FULL) OUTER JOIN t_2 USING a_1,…	匹配的同名属性在结果中只出现一次

3.7 更　　名

有时，一个表在FROM子句中多次出现，即这个表被多次引用。为区别不同的引用，应给表或表引用起一个临时的表别名，语法如下。

```
FROM 表 [AS] 别名(列别名1 [, 列别名2 [, …]] )
```

取了别名之后就不允许再用最初的名字了。如果声明的属性别名比表里实际的属性少，那么后面的属性就没有别名。

【例44】 查询同院系工作的两位考官。

```
SELECT  AAA.ername, BBB.ername
FROM  examiner AAA, examiner BBB
WHERE AAA.erdepa =BBB.erdepa AND AAA.erid<BBB.erid;
```

有时，用户也可以要求输出的列名与表中列名一致或不一致，可在SELECT子句中用“旧名 AS 别名”形式更名。

在实际使用时，AS字样可省略。

【例45】 查询考官院系名(给出前四个字符)和考官名。

```
SELECT SUBSTRING(erdepa,1,4) department, ername name
FROM examiner;
```

【例46】 查询考官号、考官名和考官组卷卷号。

```
SELECT  ER.erid, ER.ername, EE.eid
FROM examiner AS ER, erexam AS EE
WHERE ER.erid=EE.erid;
```

【例47】 表erexam与自身的笛卡儿积。

```
SELECT  *
FROM  erexam ERX1, erexam  ERX2
```

【例48】 表eeexam与自身的自然联接。

```
SELECT  *
FROM  eeexam  EEX1  NATURAL JOIN  eeexam EEX2
```

3.8 聚 集 查 询

3.8.1 基本聚集

MySQL支持聚集函数。一个聚集函数从多个输入行中计算出一个结果。经常使用的聚集函数包括COUNT(数目)、SUM(总和)、AVG(均值)、MAX(最大值)、MIN(最