第 3 章　关系数据库标准语言 SQL

SQL 语言也叫结构化查询语言（Structured Query Language），是一种介于关系代数与关系演算之间的语言。其功能包括：数据定义、数据查询、数据操作和数据控制四个方面，是一个通用的、功能很强的关系数据库语言。目前已成为关系数据库的标准语言。

3.1　SQL 概述

1. SQL 语言

SQL 语言是 1974 年由 Boyce 和 Chamberlin 提出的。1975 年至 1979 年 IBM 公司 Sanjose Research Laboratory 研究的关系数据库管理系统原型系统 System R 实现了这种语言，由于它功能丰富、语言简洁、使用方便，被众多计算机公司和软件公司所采用，并经各公司不断修改、扩充和完善，SQL 语言最终发展为关系数据库的标准语言。

第一个 SQL 标准是 1986 年 10 月由美国国家标准局（American National Standards Institute，ANSI）公布的，所以该标准也称 SQL-86。1987 年国际标准化组织（International Organization for Standardization，ISO）也通过了这一标准，此后 ANSI 不断修改和完善 SQL 标准，并于 1989 年第二次公布 SQL 标准（SQL-89），1992 年又公布了 SQL-92 标准。目前常用的数据库管理软件都支持标准的 SQL 语句。

2. 扩展 SQL 语言

尽管 ASNI 和 ISO 已经针对 SQL 制定了一些标准，但标准 SQL 语言只能完成数据库的大部分操作，不适合为关系数据库编写各种类型的程序，各家厂商针对其各自的数据库软件版本又做了某些扩充和修改，一般都根据需要增加了一些非标准的 SQL 语言。经扩充后的 SQL 语言称为扩展 SQL 语言。微软的 SQL Server 数据库服务器支持的 SQL 语言为 Transact-SQL，Oracle 支持的 SQL 语言称为 PL/SQL。

3.2　数据定义

SQL 数据定义功能包括定义基本表、定义视图和定义索引等，如表 3-1 所示。由于视图是基于基本表的虚表，索引是基于基本表的，因此 SQL 通常不提供修改视图和索引语句，用户如果要修改视图或索引，只能先将它们删除，然后重新创建。

表 3-1　SQL 数据定义语句

操作对象	操作方式		
	创建	删除	修改
基本表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

本节只介绍如何定义基本表和索引，视图的概念和定义在 3.5 节讨论。

1. 基本表定义

CREATE TABLE 〈表名〉
(列名 1 数据类型 [列级完整性约束条件],
列名 2 数据类型 [列级完整性约束条件],
……
列名 N 数据类型 [列级完整性约束条件]
[表级完整性约束条件])

说明：

（1）其中表名是要定义的基本表的名称。一个表可以由一个或多个属性列组成。

（2）创建表时通常还可以定义与该表有关的完整性约束条件。完整性约束条件被存入系统的数据字典中。当用户对表中的数据进行更新操作（插入和修改）时，DBMS 会自动检查该操作是否违背这些约束条件。如果完整性约束条件涉及表的多个属性列，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。

1）实体完整性定义语法：

在具体数据库系统中，实体完整性通过定义主键来实现，定义主键的语法为：

[CONSTRAINT 约束名] PRIMARY KEY[(属性列表)]

注意：如果定义在表级，[(属性列表)]不可省略。

2）参照完整性定义语法：

[CONSTRAINT 约束名] FOREIGN KEY(列名)REFERENCES <被参照表表名>(被参照表列名)。

3）自定义完整性定义语法：

- 列值非空：[CONSTRAINT 约束名] NOT NULL。
- 列值唯一：[CONSTRAINT 约束名] UNIQUE[(属性列表)]。
- 逻辑表达式：[CONSTRAINT 约束名] CHECK(表达式)。

（3）数据类型：SQL Server 2012 常用数据类型。

- Int 或 Smallint：整型。
- Bit：整型，只能存储 0 或 1。通常用于存储逻辑型数据。
- Float：浮点型。
- Real：实型。
- decimal [(p[,s])]和 numeric[(p[,s])]：固定精度数值型，使用最大精度时，有效值的范围为-10^38+1 到 10^38-1。p（精度）：最多可以存储的十进制数字的总位数，包括小数点左边和右边的位数，该精度必须是从 1 到最大精度 38 之间的值。默认精度为 18。s（小数位数）：小数点右边可以存储的十进制数字的位数。从 p 中减去此数字可确定小数点左边的最大位数，小数位数必须是从 0 到 p 之间的值。仅在指定精度后才可以指定小数位数。默认的小数位数为 0；因此，0≤s≤p。最大存储大小基于精度而变化。
- Text 或 Ntext：文本。Ntext 采用的是 Unicode 编码，Text 采用的是非 Unicode 编码，最大可以存储 2GB。
- Image：图形和图像，最大可以存储 2GB。
- Binary(n)：长度为 n 字节的固定长度二进制数据，其中 n 是从 1 到 8000 的值。存储

大小为 n 字节。

- Varbinary(n|max)：可变长度二进制数据。n 的取值范围为 1 至 8000。max 指示最大存储大小是 2^31-1 个字节。
- Char(n)或 Nchar(n)：固定长度字符型。Nchar 采用的是 Unicode 编码，Char 采用的是非 Unicode 编码，n 的取值范围为 1 至 8000。
- Varchar(n)或 Nvarchar(n)：可变长字符型。Nvarchar 采用的是 Unicode 编码，Varchar 采用的是非 Unicode 编码，n 的取值范围为 1 至 8000。
- Datetime 和 Smalldatetime：日期时间。
- Date 和 Time：Date 只存日期，Time 只存时间（hh:mm:ss.nnnnn）。

例 3-1　创建图书信息表、读者信息表和借阅表（表中列定义请参考表 2-3 至表 2-5）。

（1）创建图书信息表。

```
CREATE TABLE BOOK
(BOOKID CHAR(20) PRIMARY KEY,
BOOKNAME VARCHAR(60) NOT NULL,
EDITOR VARCHAR(50),
PRICE NUMERIC(6,2),
PUBLISHER    VARCHAR(50),
PUBDATE DATETIME,
QTY INT)
```

（2）创建读者信息表。

```
CREATE TABLE READER
(CARDID CHAR(10) PRIMARY KEY,
NAME VARCHAR(50),
SEX CHAR(2),
DEPT VARCHAR(50),
CLASS INT)
--读者类型：1 代表教师，2 代表学生，3 代表临时读者
```

（3）创建借阅表。

```
CREATE TABLE BORROW
(BOOKID CHAR(20),
CARDID CHAR(10),
BDATE DATETIME NOT NULL,
SDATE DATETIME,
PRIMARY KEY(BOOKID,CARDID,BDATE),
CONSTRAINT FK_BOOKID FOREIGN KEY(BOOKID) REFERENCES BOOK(BOOKID),
CONSTRAINT FK_CARDID FOREIGN KEY(CARDID) REFERENCES READER (CARDID))
```

2. 修改基本表

随着应用环境和应用需求的变化，有时需要修改已建立好的基本表，包括增加新列、增加新的完整性约束条件、修改原有的列定义或删除已有的完整性约束条件等。SQL 语言用 ALTER TABLE 语句修改基本表，其一般格式为：

```
ALTER TABLE <表名>
ALTER COLUMN <列名> <新的类型>[NULL| NOT NULL]
```

```
ADD <新列名> <数据类型> [完整性约束]
ADD <表级完整性定义>
DROP CONSTRAINT <完整性约束名>
DROP COLUMN <列名>
```

其中<表名>指定需要修改的基本表，ADD 子句用于增加新列或新的完整性约束条件，DROP 子句用于删除指定的完整性约束条件或列定义，ALTER 子句用于修改原有列的定义。

注意：ALTER TABLE 每次只能使用一个子句。

例 3-2 在图书信息表中增加一列出版时间（PUBDATE），并将 BOOKID 列宽改为 15。

```
ALTER TABLE BOOK
  ADD PUBDATE DATETIME
GO
ALTER TABLE BOOK
 ALTER COLUMN BOOKID CHAR(15)
```

例 3-3 删除借阅表中的参照完整性。

```
ALTER TABLE BORROW
 DROP CONSTRAINT FK_BOOKID
GO
ALTER TABLE BORROW
 DROP CONSTRAINT FK_CARDID
```

例 3-4 如果例 3-1 中没为借阅表创建参照完整性，或者已按例 3-3 将参照完整性删除，则可按下面方法创建参照完整性。

```
ALTER TABLE BORROW
 ADD CONSTRAINT FK_BOOKID FOREIGN KEY(BOOKID) REFERENCES BOOK(BOOKID)
GO
ALTER TABLE BORROW
 ADD CONSTRAINT FK_CARDID FOREIGN KEY(CARDID) REFERENCES READER(CARDID)
```

3. 删除基本表

当某个基本表不再需要时，可以使用 SQL 语句 DROP TABLE 进行删除。其一般格式为：

```
DROP TABLE <表名>
```

基本表一旦删除，表中的数据和在此表上建立的索引都将自动被删除，而建立在此表上的视图虽仍然保留，但已无法引用。因此执行删除操作一定要格外小心。

4. 建立索引

在 SQL 语言中，建立索引使用 CREATE INDEX 语句，其一般格式为：

```
CREATE [UNIQUE] [CLUSTER] INDEX <索引名>
ON <表名> (<列名> [<ASC|DESC> ] [,<列名> [<ASC|DESC>] ]…);
```

其中，<表名>指定要建索引的基本表的名称。索引可以建在该表的一列或多列上，各列之间用逗号分隔。每个<列名>后面可以用 ASC（升序）或 DESC（降序）指定索引值的排列次序，缺省值为 ASC。

- UNIQUE 表示如果基本表中对应索引列有多条记录具有相同的值，在索引中只出现一次，即只有一条记录参与索引。
- CLUSTER 表示要建立的索引是聚簇索引。所谓聚簇索引是指索引项的顺序与表中记录的物理顺序一致的索引组织。

用户可以在最常查询的列上建立聚簇索引以提高查询效率。显然在一个基本表上最多只能建立一个聚簇索引。建立聚簇索引后，更新索引列数据时，往往导致表中记录的物理顺序的变更，代价较大，因此对于经常更新的列不宜建立聚簇索引。

5. 删除索引

索引一经建立，就由系统使用和维护它，不需要用户干预。建立索引是为了减少查询操作的时间，如果数据修改频繁，系统会花费许多时间来维护索引。这时，可以删除一些不必要的索引。

在 SQL 语言中，删除索引使用 DROP INDEX 语句，其一般格式为：

```
DROP INDEX <表名>.<索引名>
```

例 3-5　对图书信息表按书名创建索引（升序）。

```
CREATE INDEX B_NAME
ON BOOK(BOOKNAME)
```

注意：SQL Server 中，系统会自动为主键创建索引，所以用户不必再为主键创建索引。

3.3　数据查询

建立数据库的目的是查询数据，因此，可以说数据库查询是数据库的核心操作。SQL 语言提供了 SELECT 语句来进行数据库的查询，该语句具有灵活的使用方式和丰富的功能。其一般格式为：

```
SELECT [ALL | DISTINCT ] <目标列表达式> [, <目标列表达式>…]
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 <分组表达式> [HAVING <条件表达式> ] ]
[ORDER BY <排序列名> [ASC | DESC ] ];
```

整个 SELECT 语句的含义是，根据 WHERE 子句的条件表达式，从 FROM 子句指定的基本表或视图中找出满足条件的元组，再按 SELECT 子句中的目标列表达式，选出元组中的属性值形成结果集合。如果有 GROUP 子句，则将结果按<分组表达式>的值进行分组，该属性列值相等的元组为一个组，每个组产生结果表中的一条记录。如果是 GROUP 子句带 HAVING 短语，则只有满足指定条件的组才输出。如果有 ORDER BY 子句，则结果集按<排序列名>的值的升序或降序排序。

ALL | DISTINCT：选择 DISTINCT 表示去掉结果中相同的记录；选择 ALL 表示不去掉相同的记录，默认为 ALL。

SELECT 语句既可以完成简单的单表查询，也可以完成复杂的连接查询和嵌套查询。

本节通过大量的实例介绍了 SELECT 语句的用法，涉及的表结构请参考表 2-3、表 2-4、表 2-5，表中数据请参照第 1 章的表 1-3、表 1-4 和表 1-5。

3.3.1　单表查询

单表查询是指仅涉及一个数据库表的查询，比如选择一个表中的某些列值、选择一个表中的满足条件的行等。单表查询是一种最简单的查询操作。

1. 选择表中的若干列

选择表中的全部列或部分列，这类运算又称为投影。其变化方式主要表现在 SELECT 子

句的<目标表达式>上。

例 3-6 查询所有读者的卡号和姓名。

```
SELECT CARDID, NAME
FROM READER
```

例 3-7 查询所有图书信息。

```
SELECT  *
FROM BOOK
```

说明：*代表所有列

查询结果如下：

Bookid	Bookname	Editor	Price	Publisher	PubDate	Qty
TP2001--001	数据结构	李国庆	22	清华大学出版社	2001-01-08	20
TP2003--002	数据结构	刘娇丽	19	中国水利水电出版社	2003-10-15	50
TP2002--001	高等数学	刘自强	12	中国水利水电出版社	2002-01-08	60
TP2003--001	数据库系统	汪　洋	14	人民邮电出版社	2003-05-18	26
TP2004--005	数据库原理与应用	刘　淳	24	中国水利水电出版社	2004-07-25	100

例 3-8 使用别名，查询所有读者的卡号和姓名。

```
SELECT CARDID 卡号, NAME 姓名
FROM READER
```

如果 READER 表中的数据如下：

CARDID	NAME	SEX	DEPT	CLASS
T0001	刘勇	男	计算机系	1
S0101	丁钰	女	人事处	2
S0111	张清蜂	男	培训部	3
T0002	张伟	女	计算机系	1

则查询结果为：

卡号	姓名
T0001	刘勇
S0101	丁钰
S0111	张清蜂
T0002	张伟

2. 选择表中满足条件的记录

查询满足指定条件的元组可以通过 WHERE<条件表达式>子句实现。条件表达式是操作数与运算符的组合，操作数可以包括常数、变量和字段等。常用运算符如表 3-2 所示。

表 3-2 常用运算符

查询条件	运算符
比较	=，>，>=，<，<=，!=，<>
确定范围	BETWEEN…..AND…... NOT BETWEEN….AND…

续表

查询条件	运算符
集合运算	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值判断	IS NULL，IS NOT NULL
逻辑运算	AND，OR，NOT

（1）比较运算：比较运算的操作数一般为数值型、字符型和日期时间型。英文字母按 ASCII 码比较，中文按拼音比较，日期时间按先后顺序比较。

比较运算符一般包括：

=　　等于

>　　大于

>=　　大于或等于

<　　小于

<=　　小于或等于

!=或<>　不等于

例 3-9　查询价格在 20 元以上的所有图书信息。

```
SELECT *
FROM BOOK
WHERE PRICE>=20
```

例 3-10　查询在 2003 年后的借书记录。

```
SELECT *
FROM BORROW
WHERE BDATE>'2003-01-01'
```

（2）确定范围：判断查找属性值在（BETWEEN AND）或不在（NOT BETWEEN AND）指定范围。

例 3-11　查询价格在 20～30 元之间的所有图书信息。

```
SELECT *
FROM BOOK
WHERE PRICE BETWEEN 20 AND 30
```

（3）集合运算：关键字 IN 可用于查找属性值是否属于指定集合中的元素。

例 3-12　查询电子工业出版社、清华大学出版社和高等教育出版社出版的所有图书的书名。

```
SELECT BOOKNAME
FROM BOOK
WHERE PUBLISHER IN ('电子工业出版社', '清华大学出版社', '高等教育出版社')
```

（4）字符匹配：LIKE 可以用来进行字符串匹配。其语法格式如下：

```
[NOT] LIKE "<匹配串>" [ESCAPE] "<换码字符>"
```

含义是查找指定的属性列与<匹配串>相匹配的元组。

<匹配串>可以是一个完整的字符串，也可以含有通配符%和_。

- %：代表任意长度（长度可以为 0）的字符串。
- _：代表任意单个字符。

如果<匹配串>是不含通配符的完整字符串，则 LIKE 和“=”功能相同。

例 3-13 查询以“数据库”开头的所有图书的书名和出版社。

```
SELECT BOOKNAME, PUBLISHER
FROM BOOK
WHERE BOOKNAME LIKE '数据库%'
```

查询结果如下：

```
BOOKNAME                PUBLISHER
--------------------------------------- ----------
数据库系统                人民邮电出版社
数据库原理与应用          中国水利水电出版社
-------------------------------------------------
```

如果用户要查询的字符串本身就含有%或_，这时可以使用[ESCAPE] "<换码字符>"选项进行转义。

例 3-14 查询书名含有 DELPHI_6 的所有图书信息。

```
SELECT *
FROM BOOK
WHERE BOOKNAME LIKE '%DELPHI\_6%' ESCAPE '\'
```

“ESCAPE "\"”短语表示“\”为转义字符。这样，匹配的串中紧跟在“\”后面的字符“_”就不再具有通配符的含义，而被转义为普通的“_”字符。

（5）空值判断：如果某字段允许取空值，并且没有指定默认值，那么如果在该字段上没有赋值，系统将为其赋空值（NULL），空值不是 0，也不是空字符，而是表示不确定。判断一个字段是否为空值，只能使用 IS NULL 或者 IS NOT NULL。

例 3-15 读者借书后还未还书时，借阅表中的还书日期为空值。查询所有未还书籍的读者号和借书时间。

```
SELECT CARDID , BDATE
FROM BORROW
WHERE SDATE IS NULL
```

如果 BORROW 表的数据如下：

```
BOOKID              CARDID          BDATE                SDATE
------------------- --------------- -------------------- ----------
TP2003--002         T0001           2003-11-18           2003-12-09
TP2001--001         S0101           2003-02-28           2003-05-20
TP2003--001         S0111           2004-05-06           NULL
TP2003--002         S0101           2004-02-08           NULL
```

则查询结果如下：

```
CARDID              BDATE
------------------- --------------------
S0111               2004-05-06
S0101               2004-02-08
-------------------------------------------------
```

（6）逻辑运算：利用逻辑运算符可以将上述条件进行组合，实现多重条件查询。

例 3-16 查询单位为“计算机系”且类别为教师的所有读者信息。

```
SELECT *
FROM READER
WHERE DEPT='计算机系' AND CLASS=2
```

3. 对查询结果排序

如果没有指定查询结果的显示顺序，DBMS 将按其最方便的顺序（通常是元组在表中的先后顺序）输出查询结果。用户也可以用 ORDER BY 子句指定按照一个或多个属性列的升序（ASC）或降序（DESC）重新排列查询结果，其中升序 ASC 为缺省值。

注意：对查询结果排序只作用于显示的结果，不会影响表的物理顺序。

例 3-17　查询 2003 年后出版的所有图书并按出版先后顺序排序。

```
SELECT *
FROM BOOK
WHERE PUBDATE>='2003-01-01'
ORDER BY PUBDATE
```

4. 使用集函数

为了进一步方便用户，增强检索功能，SQL 提供了许多集函数，主要包括：

- COUNT ([DISTINCT |ALL] *)　统计元组个数。
- COUNT ([DISTINCT | ALL] <列名>)　统计一列中值的个数。
- SUM ([DISTINCT | ALL] <列名>)　计算一列值的总和（此列必须是数值型）。
- AVG([DISTINCT | ALL] <列名>)　计算一列值的平均值（此列必须是数值型）。
- MAX([DISTINCT | ALL] <列名>)　计算一列值的最大值。
- MIN([DISTINCT | ALL] <列名>)　计算一列值的最小值。

如果指定 DISTINCT，则表示在计算时要取消指定列中的重复值。如果不指定 DISTINCT 或指定 ALL（为默认），则表示不取消重复值。

例 3-18　查询读者总数。

在 READER 表中，每个读者有一条记录，所以读者数等于记录数。

```
SELECT COUNT(*)
FROM READER
```

例 3-19　查询有未还书的读者数。

```
SELECT COUNT(DISTINCT CARDID)
FROM BORROW
WHERE SDATE IS NULL
```

这里一定要使用 DISTINCT 选项，因为有些读者会有多本未还图书。

例 3-20　查询库存书总数。

```
SELECT SUM(QTY)
FROM BOOK
```

5. 分组统计

GROUP BY<分组表达式>子句可以将查询结果中的各行按一列或多列取值相等的原则进行分组。分组一般与集函数一起使用。

对查询结果分组的目的是为了细化集函数的作用范围。如果未对查询结果分组，集函数将作用于整个查询结果，即对查询结果中的所有记录进行计算。如果有分组，集函数将作用于每一个分组，即集函数对每个组分别进行计算。

例 3-21　统计不同类型的读者数。

```
SELECT CLASS, COUNT(CARDID)
FROM READER
GROUP BY CLASS
```

例 3-22 按出版年份统计库存量。

```
SELECT DATEPART(YY,PUBDATE), SUM(QTY)
FROM BOOK
GROUP BY DATEPART(YY,PUBDATE)
```

注：DATEPART(YY,PUBDATE)是 SQL Server 2012 中的函数，其作用是从一个日期中分离出年份。DATEPART 函数的用法请读者查阅在线帮助。

注意：如果要对分组进行条件筛选，可以使用 HAVING <条件表达式>。WHERE 中的条件作用于每一个元组，HAVING 中的条件作用于每一个分组，其用法和作用都不相同，不能相互替代。

例 3-23 查询借书数量大于 10 本的读者卡号。

```
SELECT CARDID
FROM BORROW
WHERE SDATE IS NULL
GROUP BY CARDID HAVING COUNT(BOOKID)>10
```

3.3.2 多表查询

一个数据库中的多个表之间一般都存在某种内在联系，它们共同提供有用的信息。前面的查询都是针对一个表进行的。若一个查询同时涉及两个以上的表，则称之为多表查询或连接查询。连接查询实际上是关系数据库中最主要的查询，主要包括等值查询、非等值连接查询、自身连接查询、外连接查询和复合条件连接查询。

1. 等值与非等值连接查询

当用户的一个查询请求涉及到数据库的多个表时，必须按照一定的条件把这些表连接在一起，以便能够共同提供用户需要的信息。用来连接两个表的条件称为连接条件或连接谓词，其一般格式为：

[<表名 1> .]<列名 1> <比较运算符>[<表名 2> .]<列名 2>

当连接运算符为“＝”时，称为等值连接。使用其他运算符称为非等值连接。

连接谓词中的列名称为连接字段。连接条件中的各连接字段类型必须是可比的，但不必是相同的。例如，可以都是字符型或日期型；也可以一个是整型，另一个是实型，整型和实型都是数值型，因此是可比的。但若一个是字符型，另一个是整型就不允许了，因为它们是不可比的类型。

从概念上讲，DBMS 执行连接操作的过程为：首先在表 1 中定位到第一个元组，然后从表 2 的第一个元组开始顺序扫描或按索引扫描表 2 的所有元组，查找满足连接条件的元组，每找到一个元组，就将表 1 中的第一个元组与该元组拼接起来，形成结果表中的一个元组。表 2 全部扫描完成后，再到表 1 中找到第二个元组，然后从头开始顺序扫描或按索引扫描表 2，查找满足连接条件的元组，每找到一个元组，就将表 1 中的第二个元组与该元组拼接起来，形成结果表中的一个元组。重复上述操作，直到表 1 全部处理完成为止。

例 3-24 查询所有借书未还的读者的姓名。

```
SELECT NAME
FROM READER,BORROW
WHERE READER.CARDID=BORROW.CARDID AND SDATE IS NULL
```

结果如下：

```
NAME
----------
丁钰
张清蜂
-----------
```

读者可根据连接的操作过程，参考第 2 章的表 2-3 和表 2-4 分析查询的结果，并与计算机的查询结果进行比较。

如果属性名在参与连接的表中不是唯一的，则必须在属性名前加表名，如例 3-24 中的 CARDID。如果属性名在参与连接的表中是唯一的，则属性名前可以加表名也可以不加表名。

例 3-25　查询所有读者信息及借阅情况。

```
SELECT READER.* , BORROW.*
FROM READER , BORROW
WHERE READER.CARDID=BORROW.CARDID
```

该查询结果中将包含 READER 和 BORROW 表中的所有列。

2. 自然连接

如果按照两个表中的相同属性进行等值连接，且目标列中去掉了重复的属性列，但保留所有不重复的属性列，则称为自然连接。

例 3-26　自然连接 READER 和 BORROW 表。

```
SELECT READER.CARDID, NAME, SEX, DEPT, CLASS , BOOKID, BDATE, SDATE
FROM READER,BORROW
WHERE READER.CARDID=BORROW.CARDID
```

查询结果如下：

CARDID	NAME	SEX	DEPT	CLASS	BOOKID	BDATE	SDATE
T0001	刘勇	男	计算机系	1	TP2003--002	2003-11-18	2003-12-09
S0101	丁钰	女	人事处	2	TP2001--001	2003-02-28	2003-05-20
S0111	张清蜂	男	培训部	3	TP2003--001	2004-05-06	NULL
S0101	丁钰	女	人事处	2	TP2003--002	2004-02-08	NULL

3. 自身连接

连接操作不仅可以在两个表之间操作，也可以是一个表与其自己进行连接，这种操作称为自身连接。

例 3-27　查询书名相同而出版社不同的所有图书的书名。

```
SELECT DISTINCT B1.BOOKNAME
FROM BOOK B1,BOOK B2
WHERE B1.BOOKNAME=B2.BOOKNAME
        AND B1.PUBLISHER<>B2.PUBLISHER
```

查询结果为：

```
BOOKNAME
-------------
数据结构
--------------
```

如果连接查询涉及同一个表中的不同记录，一般要使用自身连接。

其中 B1、B2 为表的别名。如果要打开两个相同的表，一定要使用别名。

4. 外连接

在通常的连接操作中（一般也叫内连接），只有满足连接条件的元组才能作为结果输出。如例 3-24，如果某读者还没有借书记录，在结果集中就看不到该读者的信息。如果希望没有借书的读者也出现在结果集合中，只能使用外连接语法实现。

外连接又分为左外连、右外连和全外连。

- 左外连：查询结果中不仅包含符合连接条件的行，而且包含左表中所有数据行。
- 右外连：查询结果中不仅包含符合连接条件的行，而且包含右表中所有数据行。
- 全外连：查询结果中不仅包含符合连接条件的行，而且包含两个连接表中所有数据行。

外连接的实现：外连接可以使用 SELECT 语句中的 FORM 子句来实现。SQL-92 标准所定义的 FROM 子句的连接语法格式为：

FROM JOIN_TABLE JOIN_TYPE JOIN_TABLE [ON (JOIN_CONDITION)]

其中：

JOIN_TABLE 指出参与连接操作的表名。

JOIN_CONDITION 为连接条件。

JOIN_TYPE 为连接类型，可以是：INNER 内连接；LEFT OUTER 左外连；RIGHT OUTER 右外连；FULL OUTER 全外连。

例 3-28 在例 3-24 中要求将没有借书记录的读者也显示出来。

```
SELECT READER.CARDID, NAME, SEX, DEPT, CLASS, BOOKID, BDATE, SDATE
FROM READER LEFT OUTER JOIN  BORROW
ON READER.CARDID=BORROW.CARDID
```

查询结果如下：

CARDID	NAME	SEX	DEPT	CLASS	BOOKID	BDATE	SDATE
T0001	刘勇	男	计算机系	1	TP2003--002	2003-11-18	2003-12-09
S0101	丁钰	女	人事处	2	TP2001--001	2003-02-28	2003-05-20
S0101	丁钰	女	人事处	2	TP2003--002	2004-02-08	NULL
S0111	张清蜂	男	培训部	3	TP2003--001	2004-05-06	NULL
T0002	张伟	女	计算机系	1	NULL	NULL	NULL

读者 T0002 没有借书记录，外连接时相当于 BORROW 用一条空记录与之匹配。

5. 连接查询综合实例

例 3-29 查询借书期限超过 2 个月的所有读者的姓名、所借书籍名和借书日期。

```
SELECT NAME,BOOKNAME,BDATE
FROM BOOK,READER,BORROW
WHERE BOOK.BOOKID=BORROW.BOOKID
AND READER.CARDID=BORROW.CARDID
AND DATEDIFF(MM, BDATE,GETDATE()) >2
AND SDATE IS NULL
```

注：DATEDIFF 是 SQL Server 提供的函数，其作用是计算两个日期相差的年、月或日等，第一个参数用 MM 代表计算两个日期相差的月数；GETDATE()是获取系统当前日期，有关日期函数的具体用法请参照 SQL Server 的在线帮助。

例 3-30　按读者姓名查询指定读者的借还书历史记录。假设读者姓名为“刘勇”。

```
SELECT BOOKNAME,BDATE,SDATE
FROM BORROW,BOOK,READER
WHERE BOOK.BOOKID=BORROW.BOOKID
AND READER.CARDID=BORROW.CARDID
AND READER.NAME='刘勇'
```

如果按卡号查询，只须将 READER.NAME='刘勇'改为：

```
READER.CARDID='指定的卡号'。
```

例 3-31　查询指定图书的去向。如指定书名为“数据结构”。

```
SELECT READER.NAME,DEPT
FROM BORROW,READER,BOOK
WHERE BORROW.CARDID=READER.CARDID
AND BORROW.BOOKID=BOOK.BOOKID
AND BOOK.BOOKNAME='数据结构'
AND SDATE IS NULL
```

该例稍做修改即可完成按书号查询。

3.3.3　嵌套查询

在 SQL 语言中，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 短语条件中的查询称为嵌套查询或子查询。嵌套查询的求解方法是由里向外处理。即每个子查询在其上一级查询处理之前求解，子查询的结果将作为其父查询的查找条件。嵌套查询使得可以用一系列简单查询构成复杂的查询，从而明显地增强了 SQL 的查询能力。

1. 带 IN 谓词的子查询

带有 IN 谓词的子查询是指父查询与子查询之间用 IN 进行连接，判断某个属性列值是否在子查询的结果中。由于在嵌套查询中，子查询的结果往往是一个集合，所以谓词 IN 是嵌套查询中最经常使用的谓词。

例 3-32　查询借了“数据库系统”书籍的所有读者的姓名。

```
SELECT NAME
FROM READER
WHERE CARDID IN
  (SELECT CARDID
     FROM BORROW
    WHERE BOOKID IN
      (SELECT BOOKID
         FROM BOOK
        WHERE BOOKNAME='数据库系统'))
```

上例中各个子查询都只执行一次，其结果作为父查询的条件，操作过程可以如下描述：

第一步：根据书籍名称查出书籍 ID 号。

第二步：根据书籍 ID 号查询读者卡号。

第三步：根据读者卡号查询读者姓名。

上例有一个特点：子查询的查询条件不依赖于父查询，这类子查询称为不相关子查询（Uncorrelated Subquery）。

上例的嵌套查询也可以用连接查询实现（读者自己完成），但其执行速度远远快于连接查询。假设 READER 表、BORROW 表和 BOOK 表的记录数分别为 M、N 和 P，则上例的嵌套查询扫描记录的次数为 M+N+P，而连接查询扫描记录的次数为 M*N*P。

2. 带有比较运算符的子查询

带有比较运算符的子查询是指父查询与子查询之间用比较运算符进行连接。当用户能确切知道内层查询返回的是单值时，可以用>、<、=、>=、<=、!=或<>等比较运算符。

例 3-33　查询与“刘勇”在同一个部门的所有读者的信息。

```
SELECT *
FROM READER
WHERE DEPT =(
  SELECT DEPT
  FROM READER
  WHERE NAME='刘勇')
```

3. 带有 ANY 或 ALL 谓词的子查询

如果用户不能确切知道子查询的返回结果为单值时，可以使用带有 ANY 或 ALL 谓词的子查询，但 ANY 或 ALL 谓词必须与比较运算符一起使用。其语义为：

- >ANY：大于子查询结果中的某个值。
- <ANY：小于子查询结果中的某个值。
- >=ANY：大于等于子查询结果中的某个值。
- <=ANY：小于等于子查询结果中的某个值。
- =ANY：等于子查询结果中的某个值。
- !=ANY 或<>ANY：不等于子查询结果中的某个值。
- >ALL：大于子查询结果中的所有值。
- <ALL：小于子查询结果中的所有值。
- >=ALL：大于等于子查询结果中的所有值。
- <=ALL：小于等于子查询结果中的所有值。
- !=ALL 或<>ANY：不等于子查询结果中的任何一个值。

例 3-34　查询所有正借阅“中国水利水电出版社”出版的书籍的读者姓名。

```
SELECT NAME
FROM READER
WHERE CARDID =ANY(
  SELECT CARDID
   FROM BORROW
   WHERE SDATE IS NULL AND BOOKID =ANY(
        SELECT BOOKID
        FROM BOOK
        WHERE Publisher='中国水利水电出版社'))
```

事实上，用集函数实现子查询通常比直接用 ANY 或 ALL 查询效率要高。ALL 和 ANY 与集函数的对应关系如表 3-3 所示。

表 3-3　ANY，ALL 谓词与集函数及 IN 谓词的等价转换关系

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

4. 带有 EXISTS 谓词的子查询

EXISTS 代表存在。带有 EXISTS 谓词的子查询不返回任何实际数据，它只产生逻辑真值 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，否则返回假值。

例 3-35　查询借阅了书号为 TP2004--005 图书的所有读者姓名。

```
SELECT NAME
FROM READER
WHERE EXISTS (
    SELECT *
    FROM BORROW
    WHERE BORROW.CARDID=READER.CARDIDID AND BOOKID='TP2004--005')
```

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，给出列名也无实际意义。

这类查询与前面的不相关子查询有一个明显区别，即子查询的查询条件依赖于外层父查询的某个属性值（在本例中是依赖于 READER 表的 CARDID 值），我们称这类查询为相关子查询（Correlated Subquery）。求解相关子查询不能像求解不相关子查询那样，一次将子查询求解出来，然后求解父查询。相关子查询的内层查询由于与外层查询有关，因此必须反复求值。从概念上讲，相关子查询的一般处理过程如下：

首先取外层查询表中的第一个元组，根据它与内层查询相关的属性值处理内层查询，如果内查询找到一条记录，则停止进一步的查询，并返回真值（即内层查询结果非空），取此元组放入结果表；然后再检查外层查询表的下一个元组；重复这一过程，直至外层查询表全部检查完为止。

与 EXISTS 量词相对应的是 NOT EXISTS 谓词。使用不存在量词 NOT EXISTS 后，若内层查询结果为空，则外层的 WHERE 子句返回真值，否则返回假值。

3.3.4　SQL 集合运算——差集、并集、交集

SQL-3 标准中提供了三种对检索结果进行集合运算的命令：并集 UNION、交集 INTERSECT、差集 EXCEPT（在 Oracle 中叫做 MINUS）。在有些数据库中对集合运算的支持不够充分，如 MySQL 中只有 UNION，没有其他两种。实际上这些运算都可以通过普通的 SQL 语句来实现，但读者掌握这些方法有时会使查询语句写起来更简洁。

SQL Server 2008 之后已经完全支持 UNION、EXCEPT 以及 INTERSECT 集合运算符。其语法格式为：

查询 1

UNION 或 EXCEPT 或 INTERSECT

查询 2

（1）UNION 运算符：UNION 运算符是合并查询 1 和查询 2 的结果，并消去结果集合中

任何重复行。如果不想消除重复行，可以使用 UNION ALL 运算符。

（2）EXCEPT 运算符：EXCEPT 运算符是将查询 1 的结果集合减去也在查询 2 的结果中的行，即结果集合是包含在查询 1 中但不包含在查询 2 中的行，并消除所有重复行而派生出一个结果集合。如果不想消除重复行，可以使用 EXCEPT ALL 运算符。

（3）INTERSECT 运算符：取两个查询结果中的交集，即最终结果中包含既在查询 1 的结果中又在查询 2 的结果中的行，并消除所有重复行。如果不想消除重复行，可以使用 INTERSECT ALL 运算符。

注意：使用 UNION、INTERSECT 或 EXCEPT 运算符合并的所有查询必须在其目标列表中有相同数目的表达式。

例 3-36 查询所有图书的借出数量，并要求没有被借出的图书也要显示借出数量为 0。

```
SELECT BOOKID,COUNT(CARDID) NUM
FROM BORROW
WHERE SDATE IS NULL
GROUP BY BOOKID
UNION
SELECT BOOKID,0 NUM
FROM BOOK
WHERE BOOKID NOT IN (SELECT BOOKID FROM BORROW WHERE SDATE IS NULL)
```

查询结果如下：

bookid	num
TP2001--001	0
TP2002--001	0
TP2003--001	1
TP2003--002	1
TP2004--005	0

没有被借出的图书也要显示借出数量为 0 的查询语句也可以用 EXCEPT 实现，所以上面的代码也可以写成如下形式：

```
SELECT BOOKID,COUNT(CARDID) NUM
FROM BORROW
WHERE SDATE IS NULL
GROUP BY BOOKID
UNION
(SELECT BOOKID,0 NUM
FROM BOOK
EXCEPT
SELECT BOOKID,0 NUM
FROM BORROW
WHERE SDATE IS NULL)
```

3.4 数据更新

SQL 中数据更新包括插入数据、修改数据和删除数据三条语句。

3.4.1　插入数据

SQL 的数据插入语句 INSERT 通常有两种形式：一种是插入一个元组，另一种是插入子查询结果。后者可以一次插入多个元组。

1. 插入单个元组

插入单个元组的 INSERT 语句的语法格式为：

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

其功能是将新元组插入指定表中。如果有属性列表，VALUES 中的参数将按次序分别赋予各属性列，即新记录属性列 1 的值为常量 1，属性 2 的值为常量 2……如果某些属性列在 INTO 子句中没有出现，则新记录在这些列上将取默认值或空值。

注意：在表定义时说明了 NOT NULL 的属性列不能取空值，否则会出错。

如果 INTO 子句中没有指明任何列名，则新插入的记录必须在每个属性列上均有值。其赋值顺序与表中字段顺序相同。

例 3-37　在读者表中插入一条新的记录（'T0031', '刘伟', '男', '计算机系',1）

```
INSERT INTO READER
VALUES('T0031', '刘伟', '男', '计算机系',1)
```

2. 插入子查询结果

插入子查询结果的 INSERT 语句的格式为：

```
INSERT INTO<表名>[(<属性列 1>[,<属性列 2>…])]
子查询;
```

其功能是以批量插入，一次将子查询的结果全部插入指定表中。子查询结果中列数应与 INTO 子句中的属性列数相同，否则会出现语法错误。

例 3-38　按书号统计每种图书的借出数量并保存到另一个表中。

```
CREATE TABLE BOOKQTY
(BOOKID CHAR(20),
  QTY INT)
GO
INSERT INTO BOOKQTY
SELECT BOOKID, COUNT(*)
FROM BORROW
WHERE SDATE IS NULL
GROUP BY BOOKID
```

3.4.2　修改数据

修改操作又称为更新操作，其语句格式为：

```
UPDATE <表名>
SET<列名>=<表达式>[，<列名>=<表达式>]…
[WHERE<条件>];
```

其功能是修改指定表中满足 WHERE 条件的元组。其中 SET 子句用于指定修改方法，即用<表达式>的值取代相应的属性列值。如果省略 WHERE 子句，则表示要修改表中的所有元组。

例 3-39　读者还书操作。设读者卡号为 T0001，书号为 TP2003--002。

```
UPDATE BORROW
SET SDATE=GETDATE()
WHERE BOOKID='TP2003--002' AND CARDID='T0001'
GO
UPDATE BOOK
SET QTY=QTY+1
WHERE BOOKID='TP2003--002'
```

例 3-40　读者借书操作。设读者卡号为 T0001，书号为 TP2004--005。

```
UPDATE BOOK
SET QTY=QTY-1
WHERE BOOKID='TP2004--005'
GO
INSERT INTO   BORROW(BOOKID,CARDID,BDATE)
VALUES('TP2004--005', 'T0001',GETDATE())
```

修改操作要注意数据库的一致性，UPDATEA 语句一次只能操作一个表。但如果执行完一条语句之后，机器突然出现故障，无法再继续执行第二条 UPDATE 语句，则数据库中的数据就会处于不一致状态。如例 3-39，如果第二个更新语句没有执行，就会出现实际图书数量与数据库不符。因此必须保证这两条 UPDATE 语句要么都做，要么都不做。为解决这个问题，数据库系统通常都引入了事务（Transaction）的概念（参考第 6 章）。

3.4.3　删除数据

删除数据指删除表中的某些记录，删除语句的一般格式为：

```
DELETE
FROM<表名>
[WHERE<条件>];
```

DELETE 语句的功能是从指定表中删除满足 WHERE 子句条件的所有元组。如果省略 WHERE 子句，表示删除表中的全部元组，但表的结构仍在。也就是说，DELETE 语句删除的是表中的数据，而不是表的结构。

例 3-41　删除卡号为 T0035 的读者的所有借书记录，然后删除该读者信息。

```
DELETE
FROM BORROW
WHERE CARDID='T0035'
GO
DELETE
FROM READER
WHERE CARDID ='T0035'
```

本例要先删除借书记录，是因为借阅表引用了读者表中的 CARDID 字段。

DELETE 操作也是一次只能操作一个表，因此同样会遇到 UPDATE 操作中提到的数据不一致问题。

例 3-42　清空借阅表。

```
DELETE
FROM BORROW
```

例 3-43　带子查询的删除操作。删除没有借书记录的所有读者。

```
DELETE
FROM READER
WHERE NOT EXISTS(
SELECT *
FROM BORROW
WHERE BORROW.CARDID=READER.CARDID)
```

3.5　视图

视图是关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制。视图是从一个或几个基本表（或视图）导出的表，它与基本表不同，是一个虚表。换句话说，数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。基本表中的数据发生变化，从视图中查询出的数据也就随之变化了。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及其变化。

视图一经定义，就可以和基本表一样被查询和删除，也可以在一个视图之上再定义新的视图，但对视图的更新（增、删、改）操作则有一定的限制。

1. 建立视图

SQL 语言用 CREATE VIEW 命令建立视图，其一般格式为：

```
CREATE VIEW<视图名>[(<列名 1>[，<列名 2>]…)]
AS<子查询>
[WITH CHECK OPTION];
```

其中子查询可以是任意复杂的 SELECT 语句，但通常不允许含有 ORDER BY 子句和 DISTINCT 短语。

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

如果 CREATE VIEW 语句仅指定了视图名，省略了组成视图的各个属性列名，则隐含该视图由子查询中 SELECT 子句目标列中的字段组成。但在下列三种情况下必须明确指定组成视图的所有列名：

（1）其中某个目标列不是单纯的属性名，而是集函数或列表达式。

（2）多表连接并选出了几个同名列作为视图的字段。

（3）需要在视图中为某个列启用新的更合适的名字。

需要说明的是，组成视图的属性列名必须依照上面的原则，或者全部省略或者全部指定，没有第三种选择。

若一个视图是从单个基本表导出的，并且只是去掉了基本表的某些行和某些列，但保留了码，称这类视图为行列子集视图。

视图不仅可以建立在一个或多个基本表上，也可以建立在一个或多个已定义好的视图上，或同时建立在基本表与视图上。

定义基本表时，为了减少数据库中的冗余数据，表中只存放基本数据，由基本数据经过各种计算派生出的数据一般是不存储的。由于视图中的数据并不实际存储，所以定义视图时可以

根据应用的需要，设置一些派生属性列。这些派生属性列由于在基本表中并不实际存在，所以有时也称它们为虚拟列。

例 3-44 建立读者类别为学生（CLASS=1）的读者视图。

```
CREATE VIEW S_READER
AS
SELECT *
FROM READER
WHERE CLASS=1
```

本例中省略了视图的列名，隐含了该视图由子查询中 SELECT 子句中的目标列组成，由于 SELECT 的目标列为*，即为基本表 READER 中的全部列，所以视图 S_READER 中包含 READER 表的全部列。

例 3-45 创建教师读者视图，并要求进行修改和插入操作时仍保证视图只有教师记录。

```
CREATE VIEW T_READER
AS
SELECT CARDID, NAME, SEX, DEPT
FROM READER
WHERE CLASS=2
WITH CHECK OPTION
```

由于在定义视图时加上了 WITH CHECK OPTION 子句，以后对该视图行进行插入、修改和删除操作时，DBMS 会自动加上 WHERE CLASS=2 条件。

例 3-46 创建教师借阅视图。

```
CREATE VIEW T_BORROW
AS
SELECT CARDID, BOOKID, BDATE, SDATE
FROM BORROW
WHERE CARDID IN(
SELECT CARDID
FROM READER
WHERE CLASS=2)
```

本例虽然涉及两个表，但结果仍是从一个表中导出，所以仍是行列子集视图。

例 3-47 创建尚有未归还图书的读者借阅视图，包括读者姓名、所借书名和借书时间。

```
CREATE VIEW V_BORROW
AS
SELECT NAME,BOOKNAME,BDATE
FROM BORROW,READER,BOOK
WHERE BORROW.BOOKID=BOOK.BOOKID
AND BORROW.CARDID=READER.CARDID
AND SDATE IS NULL
```

例 3-48 统计每种图书的藏书数量（=在库数量+借出数量）。在库数量即 BOOK 表中的 QTY 字段。

```
CREATE VIEW B_QTY(BOOKID,NUM)
AS
SELECT BOOKID,COUNT(*)
FROM BORROW
```

```
WHERE SDATE IS NULL
GROUP BY BOOKID
UNION
SELECT BOOKID,0
FROM BOOK
WHERE BOOKID NOT IN (SELECT BOOKID FROM BORROW WHERE SDATE IS NULL)
GO
CREATE VIEW BOOK_QTY(BOOKID,SUMQTY)
AS
SELECT BOOK.BOOKID, QTY+NUM
FROM BOOK, B_QTY
WHERE BOOK.BOOKID=B_QTY.BOOKID
```

可能有些读者认为上面的代码也可以用如下代码实现：

```
CREATE    VIEW B_QTY(BOOKID,NUM)
AS
SELECT BOOKID,COUNT(*)
FROM BORROW
WHERE SDATE IS NULL
GROUP BY BOOKID
GO
CREATE    VIEW BOOK_QTY(BOOKID,SUMQTY)
AS
SELECT BOOK.BOOKID, QTY+NUM
FROM BOOK LEFT OUTER JOIN    B_QTY
ON    BOOK.BOOKID=B_QTY.BOOKID
```

查询视图 BOOK_QTY(SELECT * FROM BOOK_QTY)，发现结果如下：

```
BOOKID              SUMQTY
TP2001--001         NULL
TP2002--001         NULL
TP2003--001         27
TP2003--002         51
TP2004--005         NULL
```

为什么 SUMQTY 列会出现 NULL 呢？原因是：左外连时，如果左表中记录在右表中找不到匹配的记录时，右表用一条空记录与之匹配，这样，视图中 QTY+NUM 表达式就会出现数字与 NULL 相加，而在 SQL Server 中，当整数与空值相加时，结果为空值。

2. 删除视图

视图建好后，若导出此视图的基本表被删除了，该视图将失效，但一般不会被自动删除。删除视图通常需要显式地使用 DROP VIEW 语句进行。该语句的格式为：

```
DROP VIEW<视图名>;
```

一个视图被删除后，由该视图导出的其他视图也将失效，用户应该使用 DROP VIEW 语句将它们一一删除。

例 3-49　删除例 3-46 创建的视图 T_BORROW。

```
DROP VIEW T_BORROW
```

3. 查询视图

视图定义后，用户就可以像对基本表进行查询一样对视图进行查询了。也就是说，在 3.3

节中介绍的对基本表的各种查询操作一般都可以用于查询视图。

例 3-50 查询读者“刘伟”的借书信息。

```
SELECT NAME, BOOKNAME, BDATE
FROM V-BORROW
WHERE NAME='刘伟'
```

注：V-BORROW 是例 3-47 创建的视图。

例 3-51 统计图书馆藏书总量。

```
SELECT SUM(SUMQTY)
FROM B-QTY
```

注：B-QTY 是例 3-48 创建的视图。

4. 更新视图

更新视图包括插入（INSERT）、删除（DELETE）和修改（UPDATE）三类操作。

由于视图是不实际存储数据的虚表，因此对视图的更新，最终要转换为对基本表的更新。

为防止用户通过视图对数据进行增、删、改操作时，无意或故意操作不属于视图范围内的基本表数据，可在定义视图时加上 WITH CHECK OPTION 子句，这样在视图上增、删、改数据时，DBMS 会进一步检查视图定义中的条件，若不满足条件，则拒绝执行该操作。

例 3-52 通过视图 T_READER 修改读者（T0001）为“电子系”。

```
UPDATE T_READER
SET DEPT='电子系'
WHERE CARDID='T0001'
```

由于 T_READER 定义时带了 WITH CHECK OPTION 子句，所以 DBMS 实际执行的操作相当于：

```
UPDATE READER
SET DEPT='电子系'
WHERE CARDID='T0001' AND CLASS=2
```

一般对所有行列子集视图都可以执行修改和删除元组的操作，如果基本表中所有不允许空值的列出现在视图中，并不带 WITH CHECK OPTION 子句，则也可以对其执行插入操作。除行列子集视图外，视图理论上是可更新的，但它们的确切特征还是尚待研究的课题。目前各个关系数据库系统一般都只允许对行列子集的更新，而且各个系统对视图的更新还有更进一步的规定，由于各系统实现方法上的差异，这些规定也不尽相同。

应该指出的是，不可更新的视图与不允许更新的视图是两个不同的概念。前者指理论上已证明其是不可更新的视图。后者指实际系统中不支持其更新，但它本身有可能是可更新的视图。

5. 视图的用途

视图最终是定义在基本表之上的，对视图的一切操作最终也要转换为对基本表的操作。既然如此，为什么还要定义视图呢？这是因为合理使用视图能够带来许多好处。

（1）视图能够简化用户的操作。视图机制使用户可以将注意力集中在他所关心的数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使用户眼中的数据库结构简单、清晰，并且可以简化用户的数据查询操作。

（2）视图使用户能以多种角度看待同一数据。视图机制能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户使用同一个数据库时，这种灵活性是非常重要的。

（3）视图对重构数据库提供了一定程度的逻辑独立性。数据的物理独立性是指用户和用

户程序不依赖于数据库的物理结构。数据的逻辑独立性是指当数据库重构时，如增加新的关系或对原有关系增加新的字段等，用户和用户程序不会受影响。层次数据库和网状数据库一般能较好地支持数据的物理独立性，而对于逻辑独立性则不能完全地支持。

（4）视图能够对机密数据提供安全保护。有了视图机制，就可以在设计数据库应用系统时，对不同的用户定义不同的视图，使机密数据不出现在不应看到这些数据的用户视图上，这样就由视图机制自动提供了对机密数据的安全保护功能。视图还可以实现对记录行的授权。

3.6　数据控制

由 DBMS 提供统一的数据控制功能是数据库系统的特点之一。数据控制也称为数据保护，包括数据的安全性控制、完整性控制、并发控制和恢复。

SQL 语言提供了数据控制功能，能够在一定程度上保证数据库中数据的安全性和完整性，并提供了一定的并发控制及恢复能力。

数据库的完整性是指数据库中数据的正确性与相容性。SQL 语言定义完整性约束条件的功能主要体现在 CREATE TABLE 语句中，可以在该语句中定义码、取值唯一的列、参照完整性及其他一些约束条件。

并发控制指的是当多个用户并发地对数据库进行操作时，对他们加以控制、协调，以保证并发操作正确执行，并保持数据库的一致性。恢复指的是当发生各种类型的故障，使数据库处于不一致状态时，将数据库恢复到一致状态的功能。SQL 语言也提供了并发控制及恢复的功能，支持事务、提交和回滚等概念。

数据库的安全性是指保护数据库，防止不合法的使用所造成的数据泄露和破坏。数据库系统中保证数据安全性的主要措施是进行存取控制，即规定不同用户对于不同数据对象所允许执行的操作，并控制各用户只能存取他有权存取的数据。不同的用户对不同的数据应具有何种操作权力，是由 DBA 和表的建立者（即表的属主）根据具体情况决定的，SQL 语言则为 DBA 和表的属主定义与回收这种权力提供了手段。本节只介绍数据库对象授权机制。

1．授权

SQL 语言用 GRANT 语句向用户授予操作权限，GRANT 语句的一般格式为：

```
GRANT <权限 1> [,<权限>2]…
[ON <对象名> ]
TO <用户 1> [,<用户 2> ]…
[WITH GRANT OPTION]
```

其功能是将指定对象的指定操作权限授予指定的用户。

对象主要包括数据库、基本表、属性列、存储过程和视图。不同对象类型具有不同的操作权限（见表 3-4）。

表 3-4　不同对象类型允许的操作权限

对象	对象类型	操作权限
属性列	TABLE	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGES
基本表	TABLE	SELECT，INSERT，UPDATE，DELETE，ALTER，INDEX，ALL PRIVILEGES

续表

对象	对象类型	操作权限
视图	TABLE	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGES
数据库	DATABASE	CREATE TABLE，CREATE VIEW
存储过程	PROCEDURE	EXEC

接受权限的用户可以是一个或多个具体用户，也可以是角色，如 PUBLIC，即全体用户。

如果指定了 WITH GRANT OPTION 子句，则获得某种权限的用户还可以把这种权限再授予别的用户。如果没有指定 WITH GRANT OPTION 子句，则获得某种权限的用户只能使用该权限，但不能传播该权限。

例 3-53 假若 USER1 是数据库 BOOKSYS 的用户，把对 BOOK 表的查询和修改权限授给用户 USER1。

```
USE BOOKSYS
GRANT SELECT,UPDATE ON BOOK TO USER1
```

例 3-54 授予用户 USER1 创建表的权限，并允许他将此权限授予其他用户。

```
USE BOOKSYS
GRANT CREATE TABLE TO USER1
WITH GRANT OPTION
```

例 3-55 授予用户 USER1 有更新读者表（READER）中读者姓名字段的权限。

```
GRANT UPDATE ON READER(NAME)TO USER1
```

例 3-56 将 BOOK 表的 SELECT 权限授予所有用户。

```
GRANT SELECT ON BOOK TO PUBLIC
```

例 3-57 让用户 USER1 有操作 BOOK 表的所有权限。

```
GRANT ALL PRIVILEGES ON BOOK TO USER1
```

或者：

```
GRANT ALL ON BOOK TO USER1
```

ALL PRIVILEGES 或 ALL 代表所有操作权限，但不包括 CREATE TABLE、CREATE VIEW 等权限。

2. 权限收回

授予的权限可以由 DBA 或其授权者用 REVOKE 语句收回，REVOKE 语句的一般格式为：

```
REVOKE <权限 1> [,<权限 2>]…
[ON <对象名>]
FROM <用户 1> [,<用户 2>]…
```

其功能是从指定的用户中收回指定的权限。

可见，SQL 提供了非常灵活的授权机制。用户对自己建立的基本表和视图拥有全部的操作权限，并且可以用 GRANT 语句把其中某些权限授予其他用户。被授权的用户如果有“继承授权”的许可，还可以把获得的权限再授予其他用户。DBA 拥有对数据库中所有对象的所有权限，并可以根据应用的需要将不同的权限授予不同的用户。而所有授予出去的权限在必要时又都可以用 REVOKE 语句收回。

例 3-58 回收 USER1 对 BOOK 表的查询与修改权限。

```
REVOKE SELECT, UPDATE ON BOOK FROM USER1
```

如果要收回 USER1 的所有权限可以使用下面的命令：

REVOKE ALL PRIVILEGES ON BOOK FROM USER1

3．拒绝语句

有时要对用户在某对象上的操作权限暂时禁用，或取消某用户从角色中继承下来的权限，可以使用拒绝语句，其语法格式如下：

DENY〈ALL | 权限〉ON<对象> TO〈用户〉

例 3-59　如果按例 3-56 将 BOOK 表的 SELECT 权限授予了 PUBLIC，因为所有用户都是 PUBLIC 的成员，所以 USER1 也获得了对 BOOK 表的 SELECT 操作权限，如果要收回这个权限，必须用 DENY 语句。

DENY SELECT ON BOOK TO USER1

习题三

一、用 SQL 语言完成下列操作：

1．在数据库 ST 中创建如下表：

STUDENT(SNO, SNAME, SEX, AGE)

其中：SNO 为学号；SNAME 为学生姓名；SEX 为性别；AGE 为年龄。

COURSE(CNO, CNAME, CREDIT)

其中：CNO 为课程号；CNAME 为课程名；CREDIT 为学分。

SC(SNO, CNO, GRADE)

其中：SNO、CNO 的含义同上；GRADE 为成绩。

要求：

（1）各字段数据类型请读者按语义分析自行决定。

（2）每个表要定义主键。

（3）为 SEX 字段定义约束条件：只能为“男”或者“女”。

2．给用户 USER10 授权：USER10 对以上三个表有查询和插入权限。

3．创建一个视图 TEST。视图的功能为：查询所有学生的姓名、所选课程名称和成绩。

二、假设某数据库中有如下 4 个表，请用 SQL 语言完成下列操作。

STUDENT

学号	姓名	性别	班级名	系别代号	地址	出生日期
011110	李建国	男	计 0121	01	湖北武汉	1984-9-28
011103	李宁	女	电 0134	02	江西九江	1985-5-6
011202	赵娜	女	英 0112	03	广西南宁	1984-2-21
021204	孙亮	男	电 0134	02	湖南长沙	1986-9-8
011110	赵琳	女	计 0121	01	江苏南京	1985-11-18
021405	罗宇波	男	英 0112	03	江苏南通	1985-12-12

SC

学号	课程号	成绩
011110	01	50
021204	02	70
011103	03	90
011202	04	98
021405	02	67
021204	03	45
011110	02	80
021405	04	75
011202	03	89
011110	04	59
011103	01	80

COURSE

课程号	课程名	教师
01	英语	刘江虎
02	数学	李小则
03	C 语言	何晓敏
04	数据库	张　超

DEP

代号	系别名
01	计算机系
02	机电系
03	英语系

1．查询计算机系所有学生的信息，并按学号排序。

2．查询所有学生的学号、姓名、所选课程的课程名、所选课程的成绩。

3．删除学生“李宁”的信息及选课记录。

4．插入新的学生信息（031259，张明，01）。

5．查询“李宁”所在系的所有学生信息。

6．更新“李建国”的学号为 021110。

7．创建“计算机系”的学生信息视图。

8．按学生姓名创建索引。

9．计算 03 号课程的最高分。

10．统计选修 03 课程的学生人数。