学习情境3

数据查询技术

【能力要求】

- 能够使用查询语句完成数据查询任务。
- 能够结合查询技术完成修改信息页面的功能。
- 能够根据登录流程，结合查询完成登录页面的功能。
- 能够结合查询技术完成添加课程、修改课程页面的功能。
- 能够完成必要的存储过程、触发器和函数功能设计。
- 能够合理配置 SQL Server 数据库的安全性。

【任务分解】

- 任务 3-1　数据查询
- 任务 3-2　使用视图
- 任务 3-3　设计并实现“修改读者”页面
- 任务 3-4　设计并实现“添加图书”页面
- 任务 3-5　设计并实现“修改图书”页面
- 任务 3-6　设计并实现“管理员登录”页面
- 任务 3-7　存储过程设计
- 任务 3-8　配置数据库安全性

【重难点】

- 多表查询。
- 查询在修改信息页面和登录页面中的应用。
- 存储过程设计。
- 数据库安全性配置。

【自主学习内容】

设计“邮箱应用系统”的“用户登录”页面，编写代码实现登录功能，登录后页面转向主页面。

任务 3-1　数 据 查 询

3.1.1　查询语句格式

查询语句的语法格式如下：

```
SELECT [TOP n [PERCENT]] <字段列表>[INTO 表名] FROM <表名/视图名列表>[WHERE 条件表达式] [ORDER BY 字段名 1[ASC|DESC][,字段名 2 [ASC|DESC] [,...]]][GROUP BY 字段名列表 [HAVING 条件]]
```

其中，TOP n [PERCENT]关键字表示显示查询结果中的前 n 条或前百分之 n 条；[INTO 表名]关键字可以将查询结果存储在数据表中；WHERE 条件表达式关键字可以将符合条件表达式的记录显示在查询结果中；ORDER BY 关键字将查询结果按照指定字段升序(ASC)或降序(DESC)排序；GROUP BY 关键字将查询结果按照指定字段分组。

各关键字的使用方法请参照本学习情境的实际查询案例。

3.1.2　查询数据介绍

本学习情境的查询任务基于 library 数据库中的 booktype、readers、books 和 reader_book 4 张数据表，4 张数据表的记录如图 3-1～图 3-4 所示。

type_id	type_name
1	计算机
2	数学
3	文艺
4	杂志
5	历史

图 3-1　booktype 表结构及数据

reader_id	reader_name	sex	tel	birthday	days
2021001	孙倩	1	1	2011-01-01 00:00:00.000	20
2021002	陈诺	1	2	2011-01-01 00:00:00.000	20
2021003	李欢	0	3	2011-01-01 00:00:00.000	20
2021004	刘艾	0	5	2011-01-01 00:00:00.000	20
2021005	李冉	0	123	2011-04-08 00:00:00.000	5
2021006	李璐	0	124	2011-04-09 00:00:00.000	50
2021007	李龙	1	11	2011-01-01 00:00:00.000	70

图 3-2　readers 表结构及数据

book_id	book_name	type_id	press	ptime
1000001	数据库	1	清华大学出版社	1999-12-31
1000002	面向对象	1	清华大学出版社	1999-12-31
1000003	JAVA程序设计	1	清华大学出版社	1999-12-31
1000004	网页设计	1	清华大学出版社	1999-12-31
1000005	Unity 3D	1	清华大学出版社	1999-12-31
1000006	读者	4	待定	1999-12-31
1000007	明朝那些事儿	5	待定	1999-12-31
1000008	奥数那些事儿	2	待定	1999-12-31

图 3-3　books 表结构及数据

reader_id	book_id	btime
2021001	1000001	2021-02-01
2021001	1000002	2020-05-21
2021002	1000001	2021-05-21
2021003	1000008	2021-02-01
2021007	1000001	2019-05-21
2021007	1000008	2021-02-01

图 3-4　reader_book 表结构及数据

提示：数据查询与表记录无关，与表结构相关。要清楚所查询数据可以从哪些表找到，另外，还要清楚数据表之间的关联，这在设置多表查询的连接条件时非常关键，有关数据表间的联系请参考任务 2-3 中的外键约束。

3.1.3　单表查询

单表查询是指查询数据可以从一张表得到，语法上体现为 FROM 关键字后的表名列表只有一张表，是数据查询中最简单的查询。

1. 查询语句的最简语法格式

```
SELECT <字段列表>FROM <表名/视图名列表>[WHERE 条件表达式]
```

2. 单表查询案例及查询关键字的使用

(1) 查询数据表中部分字段。

将要查询的字段写在“字段名列表”处，字段之间用英文逗号隔开，字段顺序可以调换，如：

```
SELECT reader_id,reader_name FROM readers
```

语句执行结果如图 3-5 所示，因语句没有 where 条件，所以显示 readers 表中所有记录的 reader_id 和 reader_name 字段的值，查询字段顺序可以和字段的物理顺序不一致。

(2) 查询数据表中的全部字段。

查询全部字段时，可以将所有字段一一列出，也可以用 * 代替，例如：

```
SELECT * FROM readers
```

将 readers 表中的所有字段按照物理顺序列出来，查询结果与图 3-2 相同。

(3) 为查询字段设置别名。

设置的别名仅在查询结果中有效，并不改变字段的实际名称，设置别名的方法有：

- 字段名 [AS] 别名

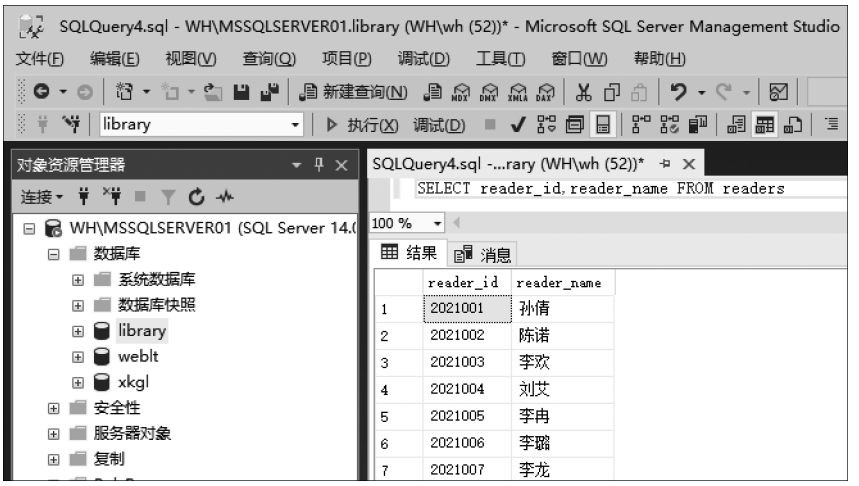


图 3-5　部分字段查询结果

• 别名＝字段名

```
SELECT 姓名=reader_name, reader_id AS 读者编号, sex 性别 FROM readers
```

查询结果如图 3-6 所示，字段名显示为汉字的“姓名”“读者编号”和“性别”。

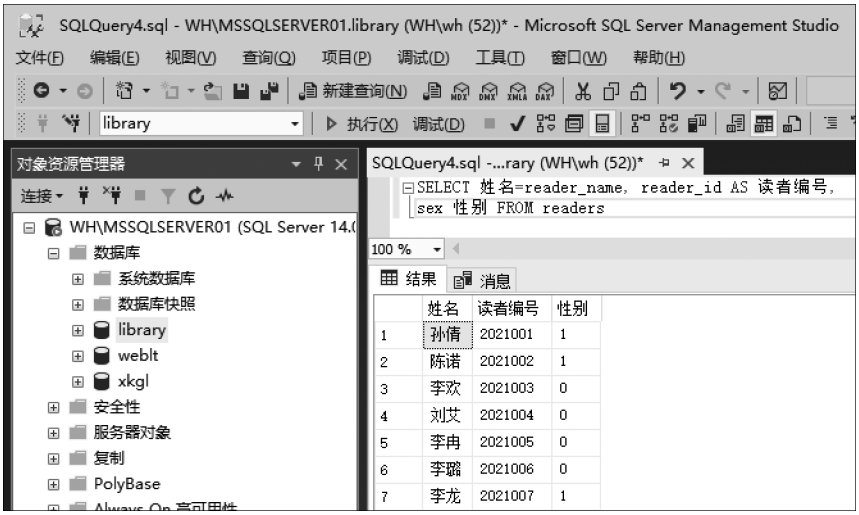


图 3-6　设置别名后的查询结果

(4) 查询经过计算的值。

有些查询结果不一定能够直接从表中得到，可能需要计算，如“查询所有学生的姓名和年龄”，年龄在 readers 数据表中没有直接给出，但有学生的出生日期，年龄可以通过出生日期计算出来，查询语句如下：

```
SELECT reader_name 姓名,YEAR (GETDATE ())-YEAR (birthday) 年龄 FROM readers
```

查询结果如图 3-7 所示。

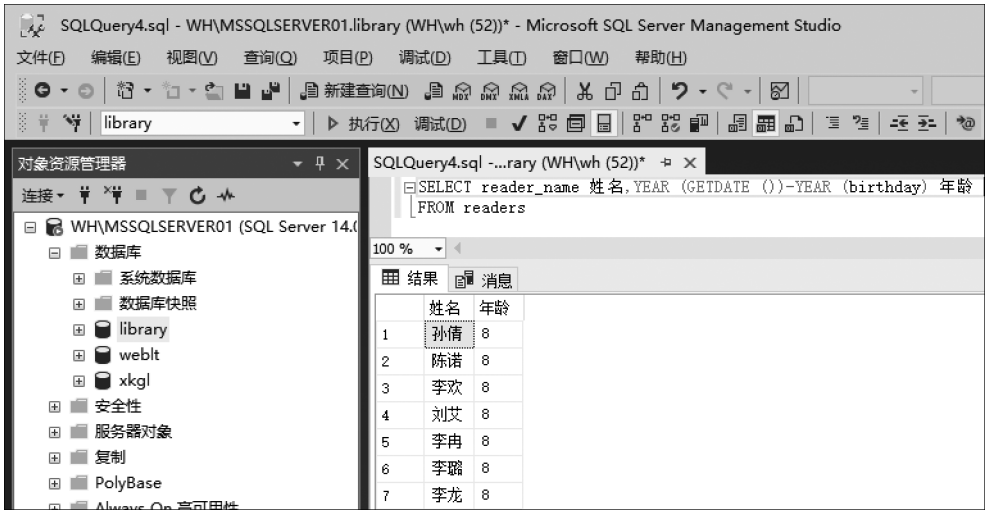


图 3-7　计算值查询结果

(5) 去除查询结果的重复值。

若要去除查询结果中的重复值,可以在字段前加 DISTINCT 关键字,例如:

```
SELECT reader_id FROM reader_book
SELECT distinct reader_id FROM reader_book
```

查询结果如图 3-8 所示。

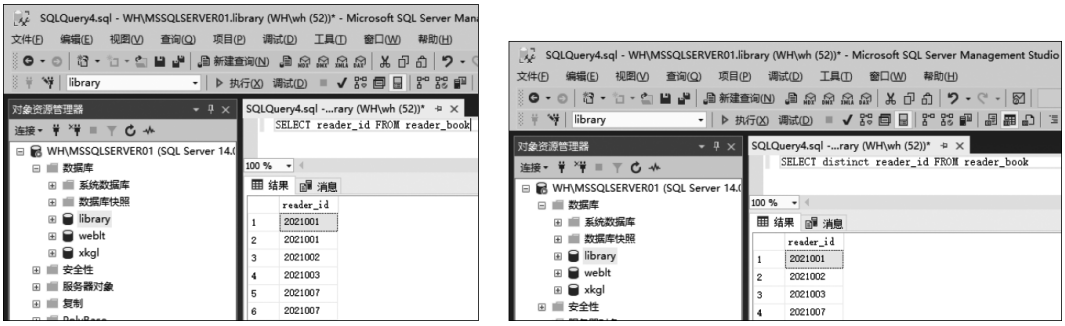


图 3-8　去除重复前后比较

(6) 显示部分查询结果。

可使用 TOP n 或 TOP n PERCENT 关键字返回部分查询结果,SQL Server 中必须将 TOP 关键字放在 SELECT 关键字之后,字段名列表之前,例如:

```
SELECT TOP 2 reader_id,reader_name FROM readers
SELECT TOP 2 PERCENT reader_id,reader_name FROM readers
```

查询结果如图 3-9 所示。

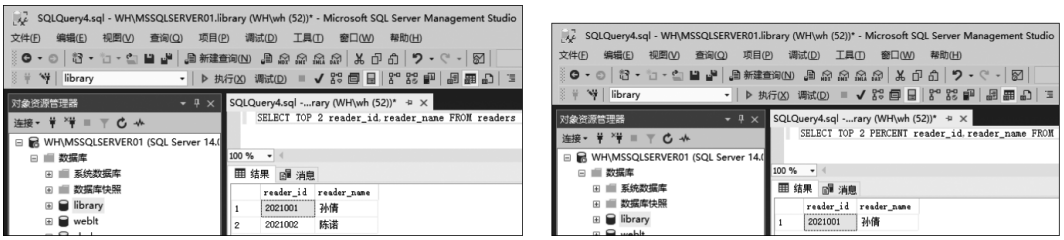


图 3-9　TOP 关键字查询结果

(7) 保存查询结果到数据表。

使用 INTO 关键字将查询结果保存到数据表,SQL Server 中必须将 INTO 关键字放在字段名列表之后,FROM 关键字之前,例如:

```
SELECT * INTO readers_new FROM readers WHERE sex=1
```

功能说明:将 readers 表中 sex 为 1 的记录另存在 readers_new 表中。

```
SELECT reader_id, reader_name INTO # readers1 FROM readers WHERE sex=0
```

功能说明:将 readers 表中 sex 为 0 的记录另存在临时表 # readers1 中,# readers1 表只包含 reader_id 和 reader_name 两个字段。

提示:临时表存储在系统数据库 tempdb 中,当服务器重启后,所有的临时表将被自动清除,临时表的最大特点是可以被所有数据库共享,因此如果要临时共享数据表,可以将数据表保存为临时表。

(8) 排序查询结果。

对查询结果排序的关键字为 ORDER BY,排序字段可以直接用字段名表示,也可以用字段列表中的序号表示。查询语句中如果无 WHERE 关键字,ORDER BY 关键字放在 FROM 关键字之后,有 WHERE 关键字,则放在 WHERE 关键字之后,排序有升序和降序两种方式,默认为升序,也可加关键字 ASC,降序加关键字 DESC。

```
SELECT * FROM readers ORDER BY birthday
```

功能说明:查询 readers 表中的所有记录,按 birthday 升序排序。

```
SELECT * FROM readers ORDER BY birthday DESC
```

功能说明:查询 readers 表中的所有记录,按 birthday 降序排序。

```
SELECT * FROM reader_book ORDER BY score DESC,reader_id
```

功能说明:查询 reader_book 表中的所有记录,先按 score 字段降序排序,score 值相同按 reader_id 字段升序排序。

```
SELECT type_id,type_name FROM booktype ORDER BY 2
```

功能说明:查询 booktype 表的所有记录,只显示 type_id、type_name 字段,按照第 2 个字段(type_name)升序排序查询结果。

(9) 查询条件。

使用 WHERE 关键字指定查询条件，查询语句无 WHERE 关键字会显示所有记录，有 WHERE 关键字，显示符合条件的记录，条件表达式中常用的运算符如表 3-1 所示。

表 3-1　条件表达式中常用的运算符

类　别	运 算 符	类　别	运 算 符
关系运算	=、>、<、>=、<=、<>、!=	模糊运算	LIKE
逻辑运算	AND、OR、NOT	空值运算	IS [NOT] NULL
集合运算	IN、NOT IN、ANY、ALL	范围运算	BETWEEN　AND

```
SELECT * FROM readers WHERE sex<>1
```

功能说明：查询 readers 表中 sex 值不等于 1 的记录，<>可以用!=表示。

```
SELECT * FROM readers WHERE birthday between '2011-1-1' AND '2018-1-1'
```

功能说明：查询 readers 表中 birthday 在 2011-1-1 到 2018-1-1 之间的数据，条件等价于“birthday>='2011-1-1' AND birthday <= '2018-1-1'”。

```
SELECT * FROM reader_book WHERE btime is null
```

功能说明：查询 reader_book 表中 btime 为空的数据。

```
SELECT * FROM reader_book WHERE book_id IN('1000001','1000002','1000003')
```

功能说明：查询 reader_book 表中 book_id 为 1000001、1000002 或 1000003 的数据，条件等价于“book_id='1000001' or book_id='1000002' or book_id='1000003'”。

模糊运算符 LIKE 关键字中所用的通配符如表 3-2 所示。

表 3-2　LIKE 关键字中所用的通配符

通 配 符	含　义
%	表示若干任意字符
_	表示单个任意字符
[]	表示方括号里列出的任意一个字符
[^]	任意一个没有在方括号里列出的字符

```
SELECT reader_id,reader_name FROM readers WHERE reader_name like '赵%'
```

功能说明：查询 readers 表中姓赵的记录，条件等价于“SUBSTRING (reader_name, 1,1)='赵'”。

```
SELECT reader_id,reader_name FROM readers WHERE reader_name like '%赵%'
```

功能说明：查询 readers 表 reader_name 中含有“赵”字的记录。

```
SELECT reader_id,reader_name FROM readers WHERE reader_name like '_赵%'
```

功能说明：查询 readers 表 reader_name 中第二个字是“赵”字的记录。

```
SELECT reader_id,reader_name FROM readers WHERE reader_name like '_赵_'
```

功能说明：查询 readers 表 reader_name 中有三个字且第二个字是“赵”字的记录。

```
SELECT reader_id,reader_name FROM readers WHERE reader_name like '[赵,钱,孙]%'
```

功能说明：查询 readers 表中姓赵、钱、孙的读者的记录。

```
SELECT reader_id,reader_name FROM readers WHERE reader_name like '[^赵,钱,孙]%'
```

功能说明：查询 readers 表中不姓赵、钱、孙的读者的记录。

(10) 集合函数的统计功能。

常用集合函数如表 3-3 所示。

表 3-3　常用集合函数

函　数	功　能
SUM()	求数值型字段的和
AVG()	求数值型字段的平均
COUNT()	统计数量(行数)
MAX()	求最大
MIN()	求最小

```
SELECT sum(days),avg(days),max(days),min(days),count(*) FROM readers
```

功能说明：查询 readers 数据表中的 days 字段的和、平均值、最高值、最低值和记录条数值，结果如图 3-10 所示。

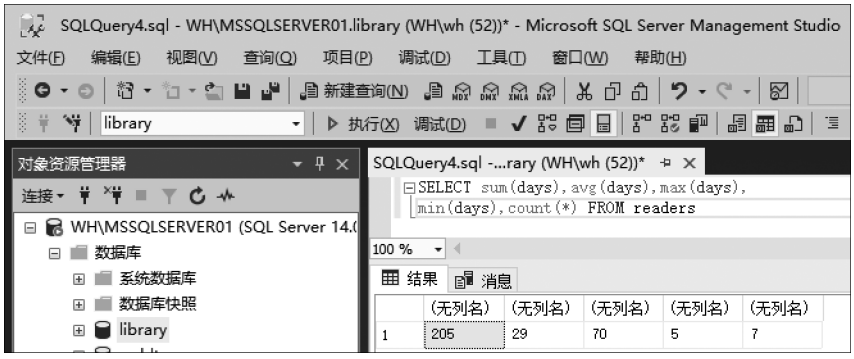


图 3-10　集合函数查询结果

```
SELECT MAX(reader_name),MIN(reader_name) FROM readers
```

功能说明：字符类按照字母顺序排序，结果如图 3-11 所示。

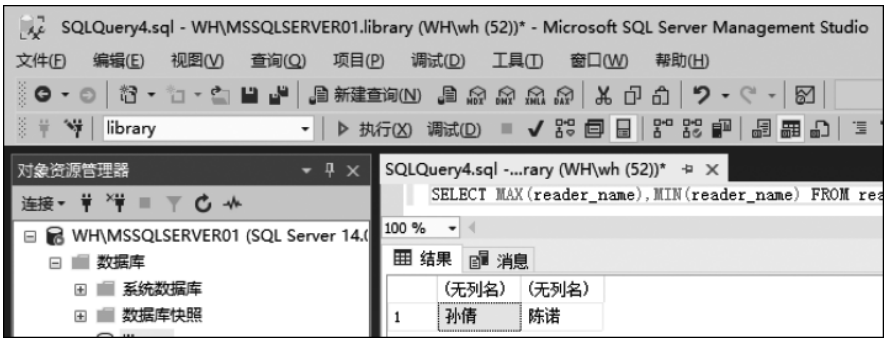


图 3-11　reader_name 字段的 MAX 和 MIN

3.1.4　多表查询

当 FROM 关键字后的表名数量多于 1 时称为多表查询，是数据库中最主要的查询方式，多表查询为了保证查询结果的准确性应带上连接条件。连接条件有内连接、左连接、右连接、完全连接 4 种。本节通过不同案例说明连接条件的使用方法。

1. 最简单的内连接条件表示

(1) 查询“孙倩”读者的借阅时间。

分析：借阅时间(btime)只有 reader_book 表有，条件“孙倩”是 readers 表中 reader_name 字段的值，因此 FROM 关键字后的表名有两个，语句如下：

```
SELECT btime FROM readers,reader_book WHERE reader_name='孙倩'
```

查询结果如图 3-12(a)所示，显然结果比实际多很多，是因为没有带上连接条件。加上简单的连接条件，语句如下：

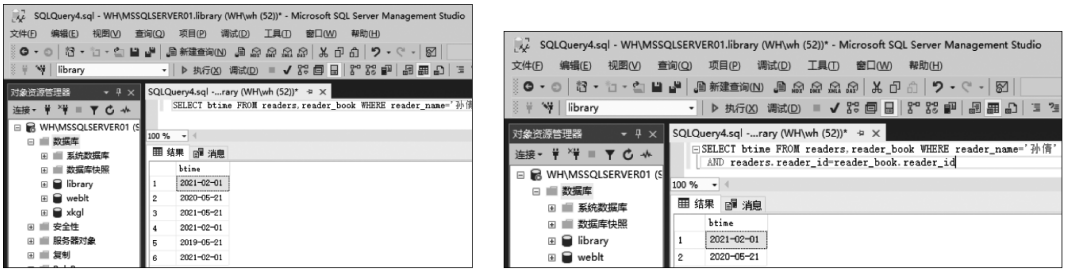
```
SELECT btime FROM readers,reader_book WHERE reader_name='孙倩' AND readers.
reader_id=reader_book.reader_id
```

readers 表和 reader_book 有一个相同意义的字段 reader_id，直接在题目条件后加上“AND readers.reader_id＝reader_book.reader_id”，即可查询到正确的结果，如图 3-12(b)所示。

提示：查询若涉及多张表，为了查询的正确性，一定要带上连接条件，一般情况下，2 张表至少要有 1 个连接条件，3 张表至少要有 2 个连接条件，表数量越多，则连接条件越多。

(2) 查询读者的姓名、所借书名和借阅时间。

```
SELECT reader_name,book_name,btime FROM readers,reader_book,books WHERE
readers .reader_id = reader_book .reader_id AND books .book_id= reader_book .
book_id
```



(a) 不带连接条件的结果　　(b) 带连接条件的结果

图 3-12　查询结果

查询结果如图 3-13 所示。

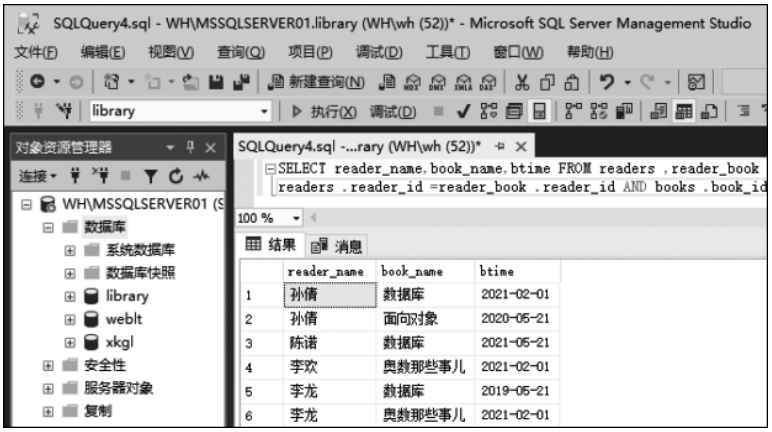


图 3-13　3 张表的查询结果

提示：如果查询数据仅来自 readers 表和 books 表，reader_book 表也一定要带上，因为 readers 表和 books 表没有直接相关联的字段，它们通过 reader_book 表实现了多对多联系，reader_book 表是 readers 表和 books 表的桥梁。

(3) 查询读者的姓名、所借图书的书名、借阅时间和图书类型名。

```
SELECT reader_name,book_name,btime,type_name FROM readers,reader_book,books,
booktype WHERE readers.reader_id = reader_book.reader_id AND books.book_id=
reader_book.book_id AND booktype.type_id=books.type_id
```

查询结果如图 3-14 所示。

2. 内连接

内连接(INNER JOIN)与上述所说的连接条件的查询结果一样，均将表中有等价关系的数据显示出来。内连接使用 INNER JOIN 关键字表示，语法格式如下：

```
表 1 INNER JOIN 表 2 ON 表 1.连接字段=表 2.连接字段
```

内连接查询结果为表 1、表 2 中连接字段等价的数据，上面 3 个查询可分别用如下 3

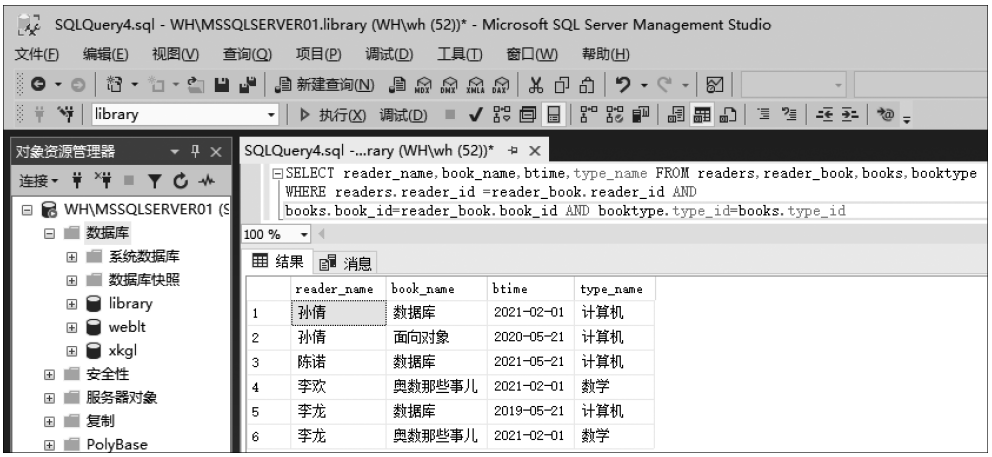


图 3-14　4 张表的查询结果

条内连接的条件表示。

```
SELECT btime FROM readers INNER JOIN reader_book ON readers .reader_id=reader_
book .reader_id WHERE reader_name='孙倩'
SELECT reader_name,book_name,btime FROM readers INNER JOIN reader_book ON
readers.reader_id =reader_book .reader_id INNER JOIN books ON reader_book .book
_id=books .book_id
SELECT reader_name,book_name,btime,type_name FROM readers INNER JOIN reader_
book ON readers .reader_id =reader_book .reader_id INNER JOIN books ON reader_
book.book_id =books.book_id INNER JOIN booktype ON booktype.type_id=books.
type_id
```

3. 左连接

左连接(LEFT JOIN)也叫左外连接,有左表和右表之分,语法格式如下:

```
表 1 LEFT [OUTER] JOIN 表 2 ON 表 1.连接字段=表 2.连接字段
```

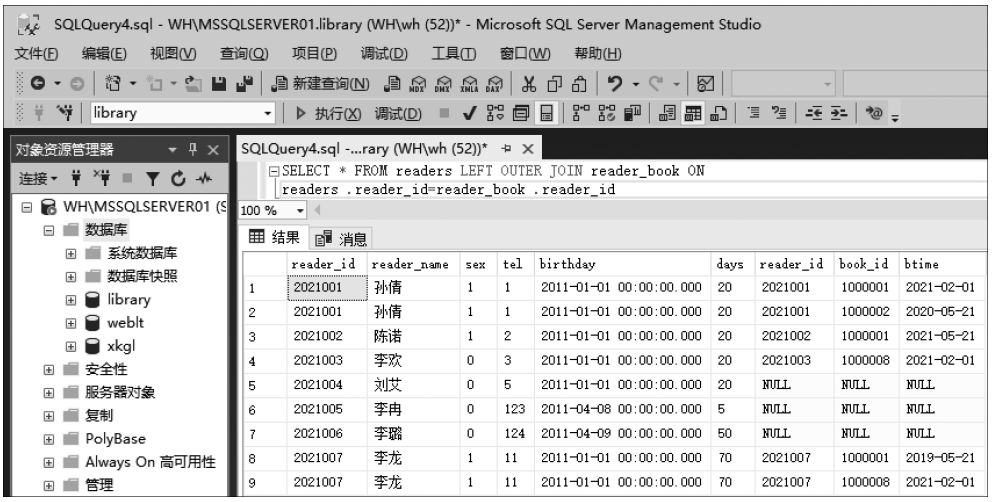
该连接条件会将左表中的数据全部显示出来,右表记录在左表中有对应值就显示出来,没有对应值显示为 NULL,例如:

```
SELECT * FROM readers LEFT OUTER JOIN reader_book ON readers .reader_id=reader_
book .reader_id
```

结果如图 3-15 所示,readers 为左表,reader_book 为右表,OUTER 关键字可以省略。

4. 右连接

右连接(RIGHT JOIN)和左连接相似,只是将右表结果全部显示,左表与右表有对应的值就显示出来,无对应值以 NULL 显示,关键字为 RIGHT [OUTER] JOIN,例如。



	reader_id	reader_name	sex	tel	birthday	days	reader_id	book_id	btime
1	2021001	孙倩	1	1	2011-01-01 00:00:00.000	20	2021001	1000001	2021-02-01
2	2021001	孙倩	1	1	2011-01-01 00:00:00.000	20	2021001	1000002	2020-05-21
3	2021002	陈诺	1	2	2011-01-01 00:00:00.000	20	2021002	1000001	2021-05-21
4	2021003	李欢	0	3	2011-01-01 00:00:00.000	20	2021003	1000008	2021-02-01
5	2021004	刘艾	0	5	2011-01-01 00:00:00.000	20	NULL	NULL	NULL
6	2021005	李冉	0	123	2011-04-08 00:00:00.000	5	NULL	NULL	NULL
7	2021006	李璐	0	124	2011-04-09 00:00:00.000	50	NULL	NULL	NULL
8	2021007	李龙	1	11	2011-01-01 00:00:00.000	70	2021007	1000001	2019-05-21
9	2021007	李龙	1	11	2011-01-01 00:00:00.000	70	2021007	1000008	2021-02-01

图 3-15　readers 表和 reader_book 表左连接查询结果

```
SELECT * FROM reader_book RIGHT JOIN readers ON readers .reader_id= reader_book .
reader_id
```

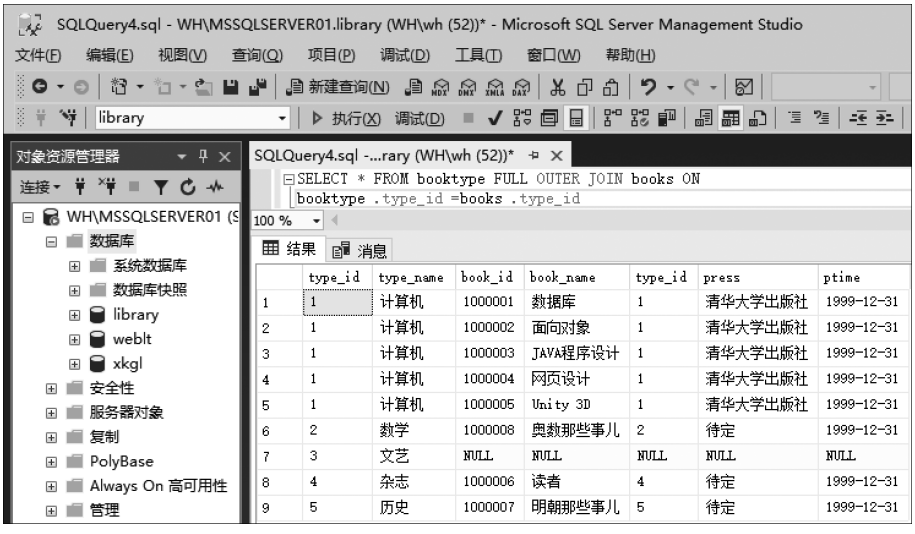
查询结果与上述左连接结果一样。

5. 完全连接

完全连接(FULL JOIN)是左连接与右连接的综合，会将左表、右表的数据全部显示，有对应的值就对应显示，无对应的以 NULL 填充，关键字为 FULL [OUTER] JOIN，例如：

```
SELECT * FROM booktype FULL OUTER JOIN books ON booktype .type_id =books .type_id
```

查询结果如图 3-16 所示。



	type_id	type_name	book_id	book_name	type_id	press	ptime
1	1	计算机	1000001	数据库	1	清华大学出版社	1999-12-31
2	1	计算机	1000002	面向对象	1	清华大学出版社	1999-12-31
3	1	计算机	1000003	JAVA程序设计	1	清华大学出版社	1999-12-31
4	1	计算机	1000004	网页设计	1	清华大学出版社	1999-12-31
5	1	计算机	1000005	Unity 3D	1	清华大学出版社	1999-12-31
6	2	数学	1000008	奥数那些事儿	2	待定	1999-12-31
7	3	文艺	NULL	NULL	NULL	NULL	NULL
8	4	杂志	1000006	读者	4	待定	1999-12-31
9	5	历史	1000007	明朝那些事儿	5	待定	1999-12-31

图 3-16　booktype 表和 books 表完全连接查询结果

因 books 表中的 type_id 全部能在 booktype 中找到对应的值，所以此时的 FULL JOIN 与 LEFT JOIN 的查询结果一样。

6. 交叉连接

交叉连接(CROSS JOIN)也叫无连接，查询结果是表中数据的所有组合可能，例如：

```
SELECT * FROM booktype CROSS JOIN books
```

等价于

```
SELECT * FROM booktype , books
```

因 books 表有 8 条记录，booktype 表有 5 条记录，其所有组合可能记录数为 8×5＝40，故无连接的查询结果有 40 条记录。

7. 自连接

自连接是指同一张表内进行的连接，此时要为表取别名。

例如，查询和“孙倩”性别相同且出生日期在 2000 年之后的读者信息，使用自连接表示的语句如下：

```
SELECT B.* FROM readers A, readers B WHERE A.reader_name ='孙倩' AND B.birthday>
'2000-12-31' AND A.sex =B.sex AND B.reader_name<>'孙倩'
```

提示：此时 readers 表分别取别名 A、B，WHERE 关键字后的 A、B 不能用反了，否则查询结果不正确，查询结果如图 3-17 所示。

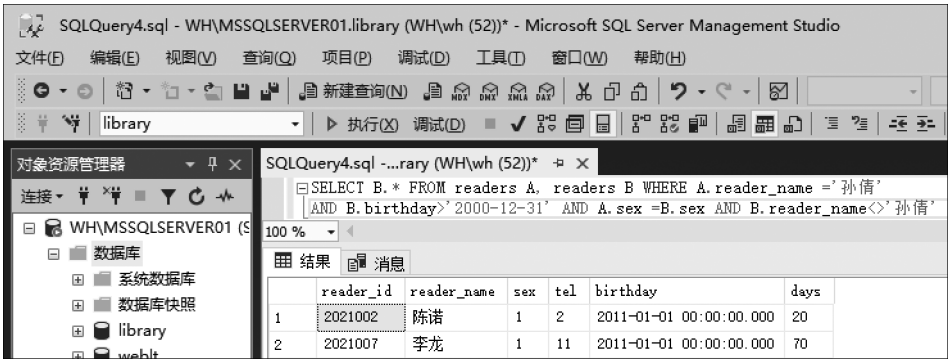


图 3-17　自连接查询结果

上述查询任务也可使用子查询实现，语句如下：

```
SELECT * FROM readers WHERE birthday>'2000-12-31' AND sex=(SELECT sex FROM
readers WHERE reader_name='孙倩') AND reader_name <>'孙倩'
```

子查询将在 3.1.6 节中介绍。

8. 合并多个查询结果

合并多个查询结果也叫联合查询(UNION [ALL]),可以将两个以上的查询结果集合并成一个结果集,例如:

```
SELECT reader_id,reader_name FROM readers
union
SELECT book_id,book_name FROM books
```

查询结果如图 3-18 所示。

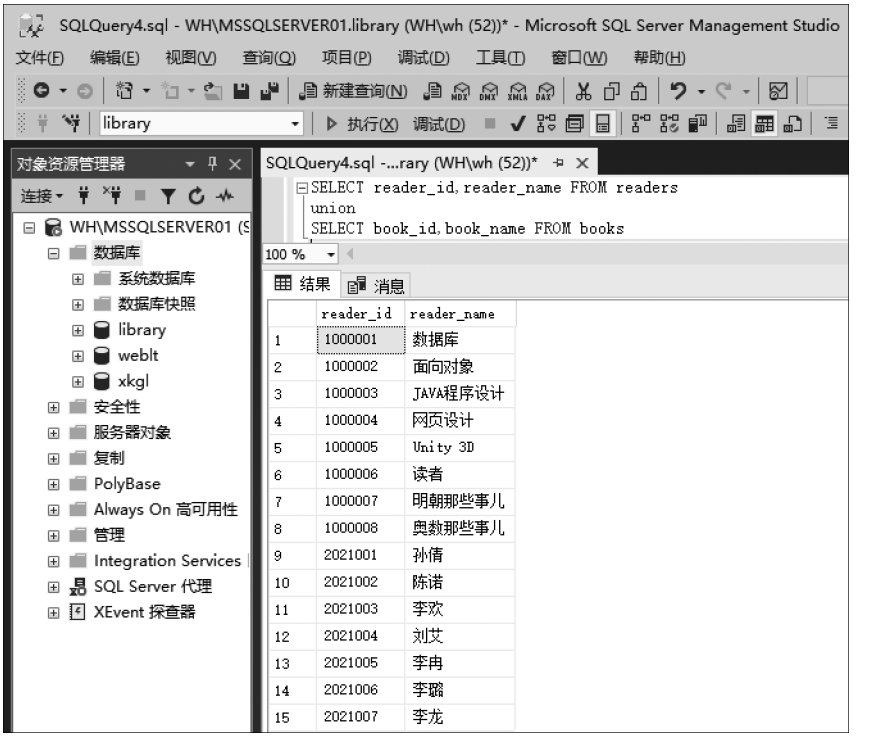


图 3-18　合并查询结果

提示:联合查询是将查询结果集顺序合并,要求每个查询结果集中的字段数、数据类型相同,宽度不同时以最宽的字段宽度输出结果。结果集中的字段名来自第一个 SELECT 语句。最后一个 SELECT 语句可以带 ORDER BY 子句,对整个查询结果起作用,但只可用第一个 SELECT 子句中的字段为排序关键字。不带 ALL 关键字只保存结果集中重复值中的一个,有 ALL 关键字会保留所有结果,例如:

```
SELECT reader_id,reader_name FROM readers
union all
SELECT book_id,book_name FROM books ORDER BY reader_id
```

查询结果如图 3-19 所示。

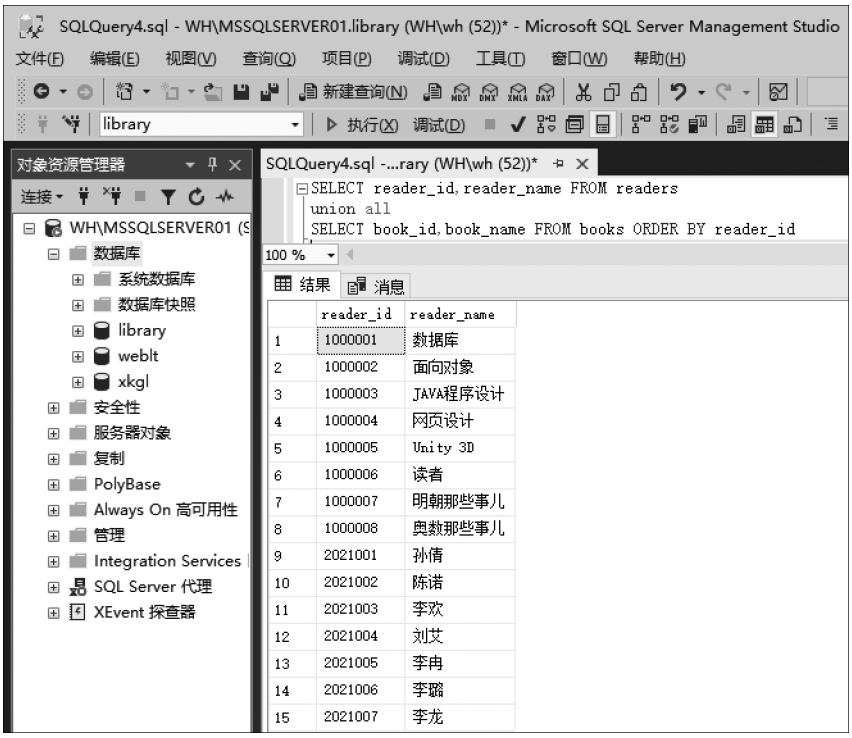


图 3-19　排序合并查询结果

3.1.5　使用数据查询添加记录

1. 语法格式

查询结果可以添加到数据表记录中,语法格式如下:

```
INSERT 表名[(字段名列表)] SELECT 字段名列表 FROM 表名列表 WHERE 条件
```

2. 举例

下列 T-SQL 程序段可将查询结果添加到数据表。

```
--生成一个新表 reader,其结构与 readers 表一样,但记录为空
SELECT * INTO reader FROM readers WHERE 2>3
--将 readers 表中 sex 为 1 的记录添加到 read 表中
INSERT reader SELECT * FROM readers WHERE sex=1
--将 readers 表中 sex 为 0 的记录添加到 reader 表中,只需要 reader_id、sex 和 reader_name 的值
INSERT reader(reader_id, sex, reader_name) SELECT reader_id, sex , reader_name
FROM readers WHERE sex=0
```

执行后 reader 表记录如图 3-20 所示。

reader_id	reader_name	sex	tel	birthday	days
2021001	孙倩	1	1	2011-01-01 00:00:00.000	20
2021002	陈诺	1	2	2011-01-01 00:00:00.000	20
2021007	李龙	1	11	2011-01-01 00:00:00.000	70
2021003	李欢	0	NULL	NULL	NULL
2021004	刘艾	0	NULL	NULL	NULL
2021005	李冉	0	NULL	NULL	NULL
2021006	李璐	0	NULL	NULL	NULL

图 3-20　reader 表记录

3.1.6　子查询

1. 引入问题

查询任务：查询没有借书读者的姓名、性别。

分析：读者没借书的特点是该读者的编号在 readers 表中，而在 reader_book 表中没有。

2. 嵌套子查询基本语法格式

当一个查询作为另一个查询的条件时，称为子查询，常见子查询的语法格式如下：

```
SELECT 字段名列表 FROM 表名列表 WHERE 字段名 IN|NOT IN|关系表达式 ANY|关系表达式 ALL
(SELECT 字段名 FROM 表名列表 WHERE 条件表达式)
```

提示：圆括号内的 SELECT 块可称为内查询或子查询，圆括号外的查询叫父查询或外层查询，通过父查询 WHERE 后的字段名与子查询 SELECT 后的字段名相关联，通常子查询的字段名列表只有一个，应与父查询中的字段名含义相同。

3. 解决问题

查询没有借书读者的姓名、性别的语句如下：

```
SELECT reader_name,sex FROM readers WHERE reader_id NOT IN (SELECT reader_id
FROM reader_book)
```

查询结果如图 3-21 所示。

同类问题：查询没有读者借阅的图书信息，即 book_id 在 books 表而在 reader_book 表中没有，语句如下：

```
SELECT * FROM books WHERE book_id NOT IN(SELECT book_id FROM reader_book )
```

查询结果如图 3-22 所示。

其他使用子查询的案例有：

```
SELECT * FROM readers WHERE birthday>(SELECT birthday FROM readers WHERE reader_
name='孙倩')
```

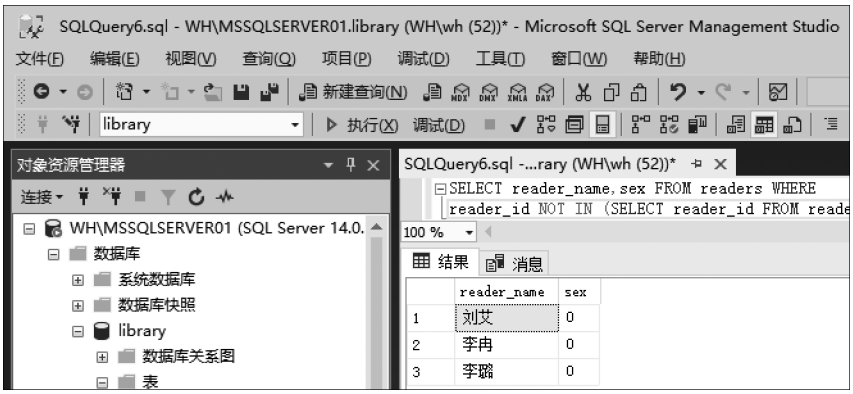


图 3-21　没有借书读者的姓名、性别

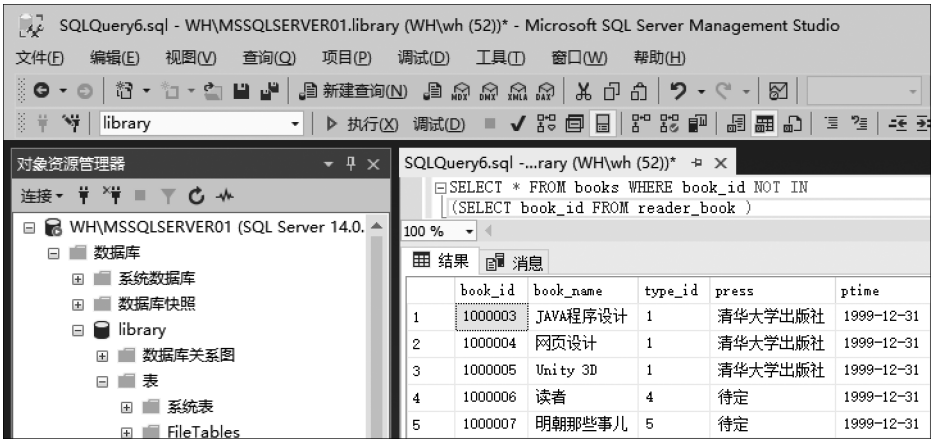


图 3-22　没有读者借阅的图书信息

功能说明：查询 birthday 比孙倩读者 birthday 大的学生的信息，结果集中没有孙倩的数据，如果条件将“>”改成“>=”，则结果集中有孙倩的数据。

```
SELECT * FROM readers WHERE birthday>(SELECT birthday FROM readers WHERE sex=0)
```

功能说明：语句出错，因为 readers 表中 sex 为 0 的记录有 4 条，子查询中的 birthday 结果集有 4 个，即子查询的结果不止一个，此时可加上 ALL 或 ANY 关键字。

```
SELECT * FROM readers WHERE birthday>any (SELECT birthday FROM readers WHERE sex=0)
```

功能说明：>ANY 表示只要比子查询中的任意一个大即可，即比子查询结果中最小的大即可，查询结果如图 3-23 所示；>ALL 关键字表示要比结果集中所有的结果都大，即比子查询结果中最大的大才成立。

```
SELECT * FROM readers WHERE sex=1 AND birthday>(SELECT birthday FROM readers
WHERE reader_name='孙倩')
```

功能说明：查询 readers 表中 sex 为 1 且 birthday 比孙倩的 birthday 大的读者信息。

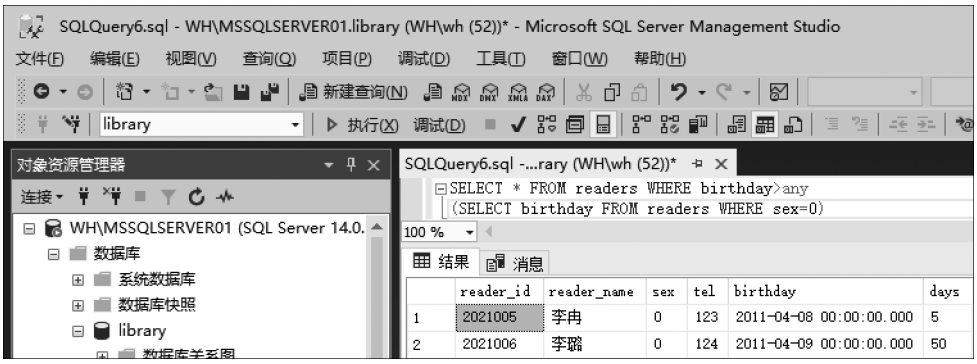


图 3-23　加入 ANY 关键字后的查询结果

几乎所有的内连接查询都可以用子查询实现，例如：

```
SELECT distinct readers. * FROM readers INNER JOIN reader_book ON readers .
reader_id=reader_book .reader_id
```

等价于

```
SELECT * FROM readers WHERE reader_id IN(SELECT reader_id FROM reader_book)
```

提示：前一个查询要去除重复值，否则结果与后一个语句不同。

4. 相关子查询

相关子查询是指在子查询的条件中引用了父查询表中的字段值。相关子查询与前面的嵌套子查询执行顺序不同，嵌套子查询先执行子查询，然后将子查询作为父查询的条件；相关子查询是以父查询中的行为单位，先选取父查询中的第一行，然后子查询利用此行中的相关字段值进行查询，父查询根据子查询返回的结果，判断此行是否满足条件，满足就记录在结果集中，不满足就抛弃，然后继续选取父查询中的下一行，直到父查询中的所有行都判断完。

(1) 查询没有借阅 1000001 图书的读者姓名和性别。

使用嵌套子查询语句，如下：

```
SELECT reader_name, sex FROM readers WHERE reader_id NOT IN(SELECT reader_id
FROM reader_book WHERE book_id='1000001')
```

使用相关子查询语句，如下：

```
SELECT reader_name, sex FROM readers WHERE NOT EXISTS(SELECT reader_id FROM
reader_book WHERE readers.reader_id=reader_book.reader_id AND book_id!=
'1000001')
```

结果如图 3-24 所示。

提示：相关子查询中的查询字段列表可以用 *，而嵌套子查询不可以。

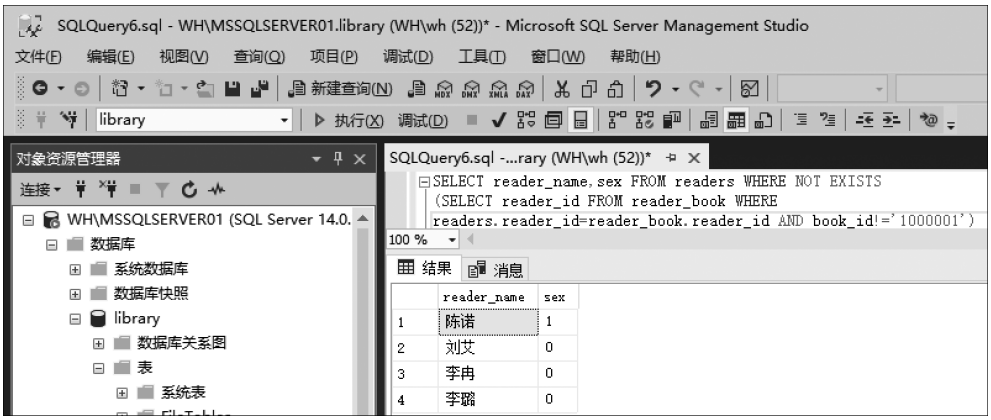


图 3-24　没有借阅 1000001 号图书的读者信息

(2) 查询所有借书读者的编号、姓名和性别。

使用相关子查询的语句如下：

```
SELECT reader_id, reader_name, sex FROM readers WHERE EXISTS (SELECT * FROM
reader_book WHERE readers .reader_id=reader_book .reader_id )
```

结果如图 3-25 所示。

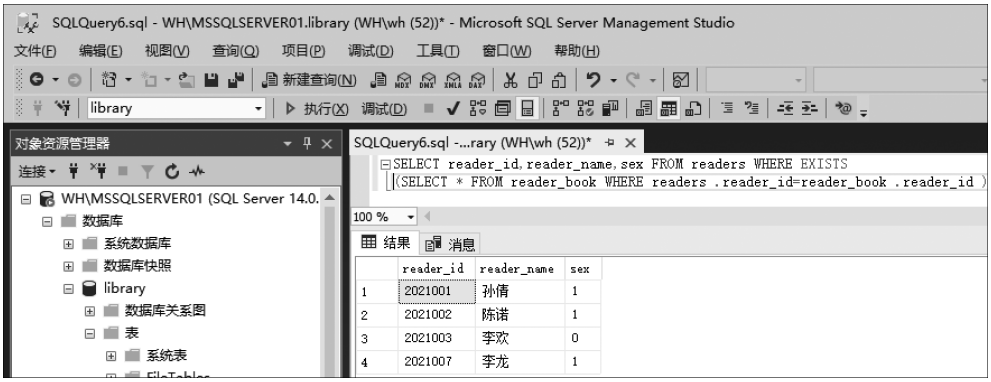


图 3-25　已借书读者的信息

```
SELECT reader_id, reader_name, sex FROM readers WHERE NOT EXISTS (SELECT * FROM
reader_book WHERE readers .reader_id=reader_book .reader_id )
```

加上 NOT 关键字是将未借书读者的信息检索出来，结果如图 3-26 所示。

3.1.7　分组查询

1. 引入问题

(1) 查询 2021001 读者借阅图书的数量。

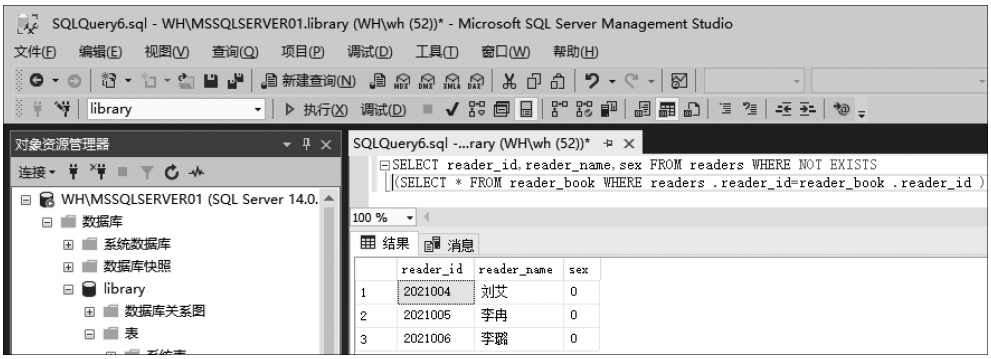


图 3-26　未借书读者的信息

```
SELECT count(*) FROM reader_book WHERE reader_id='2021001'
```

查询结果如图 3-27 所示。

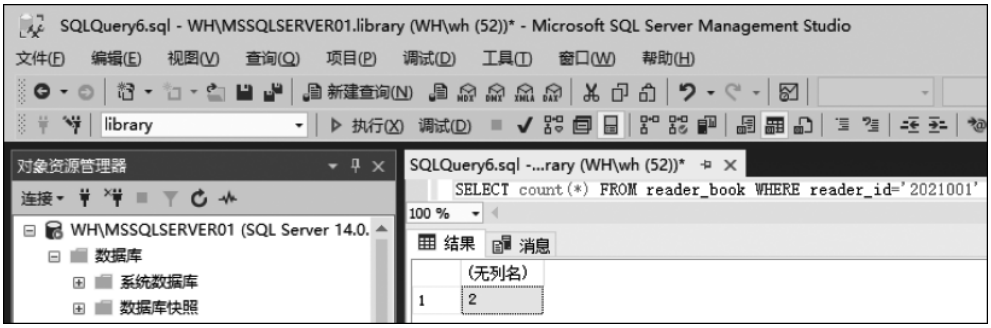


图 3-27　2021001 读者的借书数量

(2) 查询每个读者的借书数量。

查询指定读者的借书数量可以直接查询到。如果要把每个读者的借书数量都计算出来,需要按照读者编号分别计算,此时要用到分组关键字。

2. 分组关键字

GROUP BY 可以按照某字段将将记录分成若干小组,语法格式如下:

```
GROUP BY 字段或计算字段 [HAVING 条件]
```

GROUP BY 关键字要放置在 WHERE 关键字之后,可以按照字段分隔表中数据,将值相同的分成一组。

3. 解决问题

(1) 查询每个读者的借书数量。

```
SELECT reader_id 读者编号,count(*) 借书数量 FROM reader_book GROUP BY reader_id
```

查询结果如图 3-28 所示。