

# 11.3 视图

## 11.3.1 视图概述

数据表设计过程中,通常需要考虑数据的冗余度低、数据一致性等问题,而且对数据表的设计也要满足范式的要求,因此也会造成一个实体的所有信息保存在多个表中的现象。当检索数据时,往往在一个表中不能得到想要的所有信息。为了解决这种矛盾,在 SQL Server 中提供了视图。

### 1. 引入视图的原因

数据存储在表中,对数据的操纵主要是通过表进行的。但是,仅通过表来操纵数据会带来一系列的性能、安全、效率等问题。下面,对这些问题进行分析。

(1) 从业务数据角度来看,由于数据库设计时考虑到数据异常等问题,同一种业务数据有可能被分散在不同的表中,但是对这种业务数据的使用经常是同时使用的,要实现这样的查询,需要通过连接查询或嵌套查询来实现。但是,如果经常要查询相同的内容,每次都要重复写相同的查询语句,就会增加用户的工作量。如

图 11- 1 所示的视图是建立在三张表上的。

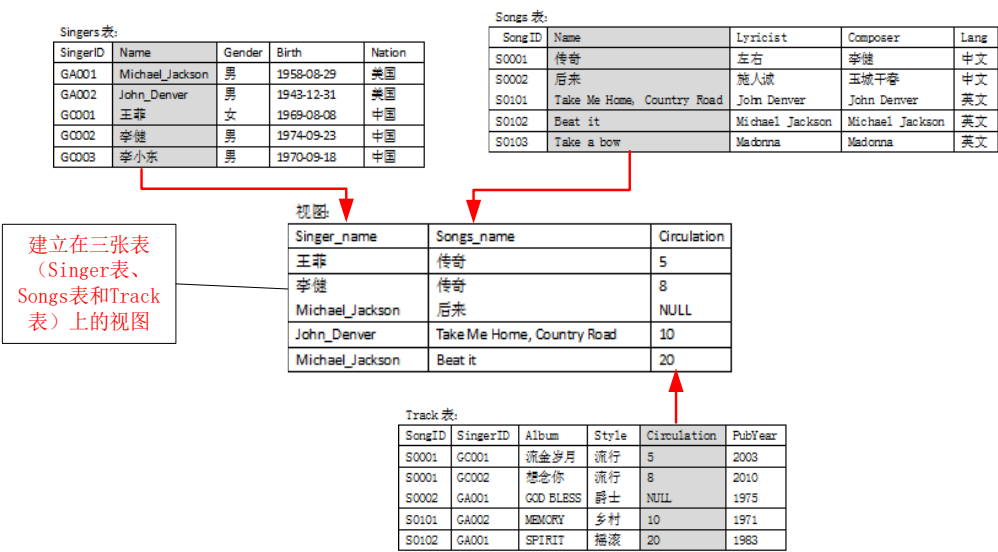
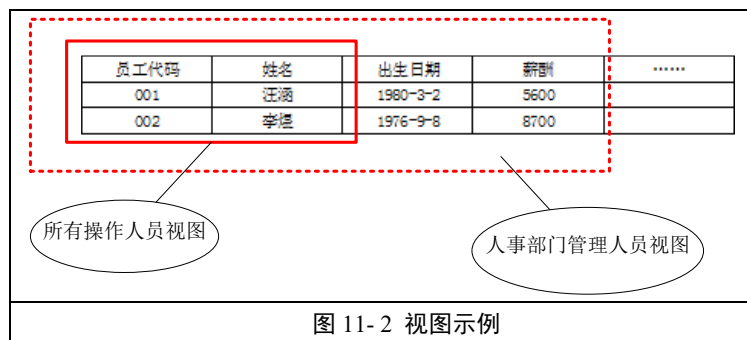


图 11- 1 视图和表之间的关系示例图

(2) 从数据安全角度来看,由于工作性质和需求不同,不同的操作人员只能查看表中的部分数据,不能查看表中的所有数据。例如,人事表中存储了员工的代码、姓名、出生日期、薪酬等信息。一般地,员工的代码和姓名是所有操作人员都可以查看的数据,但是薪酬等信息则只能由人事部门管理人员查看。这说明不同用户对数据的需求是不一样的,如果允许用户直接对表进行操作,那么表的安全将受到威胁。如所示,可以为不同的用户建立不同的视图,从而保证安全性。



(3) 从数据的应用角度来看，一个报表中的数据往往来自于多个不同的表中。在设计报表时，需要明确地指定数据的来源途径和方式。通过视图机制，可以提高报表的设计效率。

因此，视图是 RDBMS 提供给用户以多种角度来观察数据库中数据的重要机制。

## 2. 视图概述

视图是一种数据库对象，是按某种特定要求从数据库的表或其他视图中导出的虚拟表。视图所引用的表称为视图的基表。从用户角度看，视图也是由行和列组成的二维表；但视图展示的数据并不以视图结构实际存在，而是其引用的基本表的相关数据的映像。行和列的数据来自于基表，并且在引用视图时动态生成。因此，通过视图看到的数据是存放在基表中的数据。当对通过视图看到的数据进行修改时，相应的基表的数据会发生变化；同样，若基表的数据发生变化，这种变化也会自动地反映到视图中。

视图的内容由查询定义，一经定义便存储在数据库中。注意：数据库存储的是视图的定义（确切地说是 SLELECT 语句），而不是视图所看到的数据。

因此，总结来说，视图的特点如下：

- 1) 是从一个或几个基本表（或视图）导出的虚表。
- 2) 只存放视图的定义，不存放视图对应的数据。
- 3) 基表中的数据发生变化，从视图中查询出的数据也随之改变。

---

说明： 以上所述的视图均指标准视图，不包含索引视图。

---

视图的优点可体现在以下几个方面：

### 1) 简化查询语句

通过视图可以将复杂的查询语句变得简单。利用视图，用户不必了解数据库及实际表的结构，就可以方便地使用和管理数据。可以把经常使用的联接、投影和查询语句定义为视图，这样在每一次执行相同查询时，不必重新编写这些复杂的语句，只要一条简单的查询视图语句就可以实现相同的功能。因此，视图向用户隐藏了对基表数据筛选或表与表之间联接等复杂的操作，简化了对用户操作数据的要求。

### 2) 增加可读性

由于在视图中可以只显示有用的字段，并且可以使用字段别名，因此能方便用户浏览查询的结果。在视图中可以使用户只关心自己感兴趣的某些特定数据，而那些不需要的或者无用的数据则不在视图中显示出来。视图还可以让不同的用户以不同的方式看同一个数据集内容，体现数据库的“个性化”要求。

### 3) 保证数据逻辑独立性

视图对应数据库的外模式。如果应用程序使用视图来存取数据，那么当数据表的结构发生改变时，只需要更改视图定义的查询语句即可，不需要更改程序，方便程序的维护，保证了数据的逻辑独立性。

### 4) 增加数据的安全性和保密性

针对不同的用户，可以创建不同的视图，此时的用户只能查看和修改其所能看到的视图中的数据，而真正的数据表中的数据甚至连数据表都是不可见不可访问的，这样可以限制用户浏览和操作的数据内容。另外视图所引用的表的访问权限与视图的权限设置也是相互不影响的，同时视图的定义语句也可加密。

#### 5) 减少数据冗余

数据库中只需要将所有基本数据最合理、开销最小地存储在各个基本表中。对于不同用户的不同数据要求，可以通过视图从各基本表中提取、聚集，形成他们所需要的数据组织。从而不需要在物理上为满足不同用户的需求重复组织数据存储，大大减少数据冗余。

#### 6) 方便导出数据

可以建立一个基于多个表的视图，用 SQL Server Bulk Copy Program(批复制程序，BCP)复制视图引用的行到一个平面文件中，该文件可加载到 Excel 或类似的程序中。

### 3. 视图的类型

#### 1) 标准视图

标准视图就是通常意义上理解的视图，是 SELECT 查询语句。一般情况下我们自己创建的用于对用户数据表进行查询、修改等操作的视图都是标准视图。它是一个虚拟表，不占物理存储空间，也不保存数据。

#### 2) 索引视图

对于标准视图而言，为每个引用视图的查询动态生成结果集的开销很大，特别是那些涉及对大量行进行复杂处理（如聚合大量数据或连接许多行）的视图。可对视图创建一个唯一的聚集索引来提高性能，即创建索引视图。索引视图是被具体化了的视图，即它已经过计算并存储。索引视图尤其适于聚合许多行的查询。但它们不太适于经常更新的基本数据集。

#### 3) 分区视图

分区视图是通过的成员表使用 UNION ALL 所定义的视图，这些成员表的结构相同，但作为多个表分别存储在同一个 SQL Server 实例中，或存储在称为联合数据库服务器的自主 SQL Server 服务器实例组中。

分区视图在一台或多台服务器间水平连接一组成员表中的分区数据，这样数据看上去如同来自于一个表。联接同一个 SQL Server 实例中的成员表的视图是一个本地分区视图；如果视图是在服务器间连接表中的数据，则称其为分布式分区视图。

一般情况下，如果视图为下列格式，则称其为分区视图：

```
SELECT <select_list1> FROM T1
UNION ALL
SELECT <select_list2> FROM T2
UNION ALL
...
SELECT <select_listn> FROM Tn
```

例如，Customers 视图的数据分布在三个服务器位置的三个成员表中：本地的 Customers\_33、Server2 上的 Customers\_66 和 Server3 上的 Customers\_99。

Server1 的分区视图 Customers 通过以下方式进行定义：

```
CREATE VIEW Customers
AS
--从本地成员表中检索数据
SELECT * FROM CompanyData.dbo.Customers_33
UNION ALL
--从 Server2 的成员表中检索数据
```

```
SELECT * FROM Server2.CompanyData.dbo.Customers_66
UNION ALL
--从 Sever3 的成员表中检索数据
SELECT * FROM Server3.CompanyData.dbo.Customers_99
```

说明： SQL Server 2008 中包含本地分区视图只是为了实现向后兼容，当前不推荐使用。本地分区数据的首选方法是通过已分区表。

## 11.3.2 创建视图

定义视图的基本原则：

1) 只能在当前数据库中创建视图，创建视图时，Microsoft SQL Server 首先验证视图定义中所引用的对象是否存在。但是，分区视图除外，因为分区视图所引用的表和视图可以存在于其他数据库甚至其他服务器中。

2) 视图名称必须遵循标识符的规则，且对每个架构都必须唯一。此外，该名称不得与该架构包含的任何表的名称相同。

3) 可以创建嵌套视图（即建立在视图上的视图），但嵌套不得超过 32 层。

4) 不能将默认值、AFTER 触发器与视图相关联，但可以定义 INSTEAD OF 触发器。

5) 定义视图的查询不能包含 SELECT INTO 语句；SELECT 语句中不能包含 ORDER BY 子句，除非在 SELECT 语句的选择列表中还有一个 TOP 子句。

6) 不能创建临时视图，也不能对临时表创建视图。

7) 如果视图引用的基本表被删除，则当使用该视图时将返回一条错误信息。

在 SQL Server 中，视图可以通过 SQL Server Management Studio 图形化地创建或使用 T-SQL 语句来创建。

### 1. 使用 SSMS 图形化界面创建视图

在 SSMS 中使用图形化界面创建视图是一中最快捷的方式。下面以

图 11- 1 中建立在三张表 Singers、Songs 和 Track” 上的视图 v\_s\_track 为例加以说明。步骤如下：

1) 在“对象资源管理器”面板下，选择“数据库”→“视图”选项，单击右键，在快捷菜单中选择“新建视图”。

2) 在如图 11-4 所示的“添加表”对话框中，选择新视图需要的表、视图等作为数据来源。

3) “视图设计器”类似于查询设计器，其区域分为四部分，前三部分同查询设计器，最下面是视图执行结果显示区（图 11- 5）。

4) 视图创建完毕后，关闭“视图设计器”，给视图命名并存盘(本例命名为 v\_s\_track)。刷新后，在“对象资源管理器”窗口下，“视图”节点中就会出现该视图。

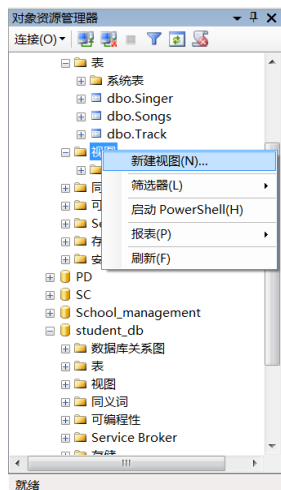


图 11-3 选择“新建视图”

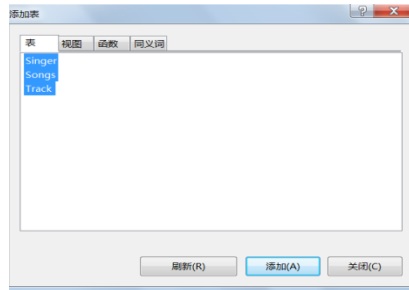


图 11-4 “添加表”对话框

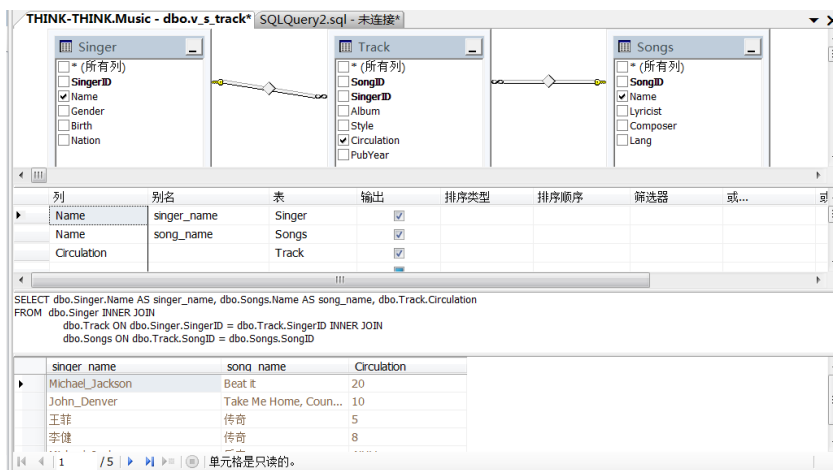


图 11-5 “视图设计器”

## 2. 使用 T-SQL 语句来创建视图

可以使用 T-SQL 中的 CREATE VIEW 语句来创建视图。语法如下：

```
CREATE VIEW view_name          /*指定视图名称 */
[(column [, column]...)]      /*指定视图中的列 */
AS select_statement           /*定义视图的 SELECT 语句*/
[WITH CHECK OPTION];          /*强制针对视图执行的所有数据修改语句都必须符合在视图定义中设置的条件（即 select_statement 中的条件表达式） */
```

说明如下：

1) 组成视图的属性列名，要么全部省略，要么全部指定，没有第三种选择。如果省略了各个属性列名，则隐含该视图由子查询中 select 子句目标列中的各字段组成；在下列情况下必须明确指定组成视图的所有列名：

- (1) 某个目标列是聚集函数、常量或列表表达式；
- (2) 多表连接时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定义新的名字。

2) WITH CHECK OPTION: 为防止用户通过视图对数据进行增删改时，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。

3) WITH CHECK OPTION 强制针对视图执行的所有数据修改语句 (insert, delete, update) 都必须符合在视图定义中设置的条件 (即子查询中的条件表达式)。例如，创建一个定义视图的查询，该视图从表中检索员工的薪水低于 \$30,000 的所有行。如果员工的薪水涨到 \$32,000，因其薪水不符合视图所设条件，查询时视图不再显示该特定员工。但是，WITH CHECK OPTION 子句强制所有数据修改语句均根据视图执行，以符合定义视图的 SELECT 语句中所设条件。如果使用该子句，则对行的修改不能导致行从视图中消失。任何可能导致行消失的修改都会被取消，并显示错误。

4) RDBMS 执行 CREATE VIEW 语句时只是把视图定义存入数据字典，并不执行其中的 SELECT 语句。在对视图查询时，按视图的定义从基本表中将数据查询出来。

**例 11-1：** 创建视图，显示发行量不小于 10 万的歌手名、歌曲名、歌曲类型及发行量。代码如下：

```
CREATE VIEW v_ Circulation
AS
SELECT Singer.Name AS singer_name, Songs.Name AS song_name,
Track.Style, Track.Circulation FROM Singer INNER JOIN Track
ON Singer.SingerID = Track.SingerID INNER JOIN Songs
ON Track.SongID = Songs.SongID
WHERE (Track.Circulation>=10)
```

**例 11-2：** 建立中文歌曲的视图，并要求进行修改和插入操作时仍需保证该视图只有中文歌曲。代码如下：

```
CREATE VIEW v_lang
as
select SongID,Name,Lang from dbo.Songs where Lang='中文'
WITH CHECK OPTION
```

当对视图 v\_lang 进入插入、修改和删除操作时，都必须符合 “Lang='中文'” 的条件，否则，将出错。如果对该视图做插入操作，代码如下：

```
insert into v_lang(SongID,Name,Lang)
values('S0104','Rolling in the Deep','英文')
```

因为不符合 “Lang='中文'” 的条件，执行时报错，如图 11-6 所示。

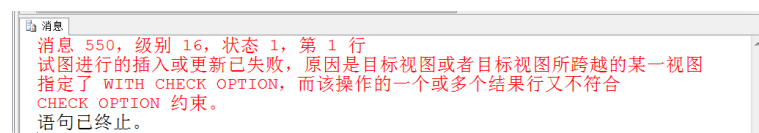


图 11-6 WITH CHECK OPTION 示例图

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或某些列，但保留了主键，称为行列子集视图。例 11-2 中的视图 v\_lang 就是一个行列子集视图。

**例 11-3：** 建立一个视图，统计各个国家的歌手人数。

```
CREATE VIEW v_singer_num
AS
select nation,count(singerid) as singer_num
```

```
from singer group by nation
```

上例中使用带有聚集函数和 GROUP BY 子句的子查询来定义视图。这种视图称为**分组视图**。

**例 11-4：**将中国歌手的名字、出生日期、年龄定义成一个视图。

```
CREATE VIEW v_age
AS
select name,birth,YEAR(getdate())-YEAR(birth) as age
from Singer where Nation='中国'
```

上例中的年龄是由计算派生出的，称为**带表达式的视图**。

### 11.3.3 管理视图

#### 1. 查看视图

在 SSMS 中，右击相应的视图名，在弹出菜单中选择“属性”，可以查看视图的属性，包括创建日期、视图名称等。

使用系统存储过程 `sp_help`，可以查看视图中列的名称、数据类型等详细信息。图 11-7 显示了视图 `v_s_track` 的详细信息。

THINK-THINK-Music - dbo.Songs

SQLQuery2.sql - TH1..NK.Music (sa (55))\*

SQL-sql - THINK-THINK-Music (sa (54))

sp\_help v\_s\_track

结果

消息

|   | Name      | Owner | Type | Created_datetime        |
|---|-----------|-------|------|-------------------------|
| 1 | v_s_track | dbo   | view | 2012-10-01 16:19:15.223 |

|   | Column_name | Type    | Comput... | Length | Pr... | Sca... | Nulla... | TrimTrailingBlar |
|---|-------------|---------|-----------|--------|-------|--------|----------|------------------|
| 1 | singer_name | varchar | no        | 50     |       |        | no       | no               |
| 2 | song_name   | varchar | no        | 50     |       |        | yes      | no               |
| 3 | Circulation | int     | no        | 4      | 10    | 0      | yes      | (n/a)            |

|   | Identity                    | Seed | Increment | Not For Replicati... |
|---|-----------------------------|------|-----------|----------------------|
| 1 | No identity column defined. | NULL | NULL      | NULL                 |

|   | RowGuidCol                    |
|---|-------------------------------|
| 1 | No rowguidcol column defined. |

查询已成功执行。

THINK-THINK (10.0 RTM)

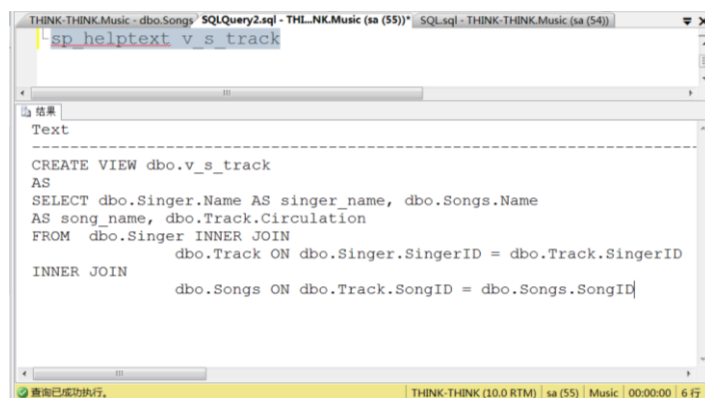
sa (55)

Music

0

图 11-7 用 `sp_help` 查看视图 `v_s_track`

使用 `sp_helptext` 可以查看视图定义语句。图 11-8 显示了使用 `sp_helptext` 查看视图 `v_s_track` 定义语句的结果。



```
CREATE VIEW dbo.v_s_track
AS
SELECT dbo.Singer.Name AS singer_name, dbo.Songs.Name
AS song_name, dbo.Track.Circulation
FROM dbo.Singer INNER JOIN
      dbo.Track ON dbo.Singer.SingerID = dbo.Track.SingerID
INNER JOIN
      dbo.Songs ON dbo.Track.SongID = dbo.Songs.SongID
```

图 11-8 使用 `sp_helptext` 查看视图 `v_s_track` 定义

#### 2. 修改视图

修改视图可更改一个视图的定义，但不影响相关的存储过程或触发器，也不更改权限。只需要使用新的 **select** 语句和选项代替原来的定义。

在 SSMS 中，在视图名上右击，在快捷菜单中选择“设计”，即可打开“视图设计器”，在此对视图进行修改。

还可用 **ALTER VIEW** 语句进行修改视图，语法如下：

```
ALTER VIEW view_name [(column_name)]
    [WITH ENCRYPTION]          /*对视图加密*/
AS select_statement [WITH CHECK OPTION]
```

各项参数与 **CREATE VIEW** 语句中的相同，不再赘述。

### 3. 删除视图

在 SSMS 中，在视图名上右击，在快捷菜单中选择“删除”，及“删除对象”对话框中点击“确定”即可删除。

删除视图对应的 T-SQL 语句为 **DROP VIEW** 语句。语法为：

```
DROP VIEW view_name
```

删除视图是从数据字典中删除相应的视图定义。由该视图导出的其他视图定义仍在数据字典中，但已不能使用，必须使用 **DROP VIEW** 语句显式删除。同样，删除基表时，由该基表导出的所有视图定义也必须显式地使用 **DROP VIEW** 语句删除。

## 11.3.4 视图的使用

当视图被定义以后，用户就可以像对基本表进行查询那样对视图进行查询了。RDBMS 实现视图消解法（**View Resolution**）实现对视图的查询：首先进行有效性检查，检查涉及的表、视图等是否在数据库中存在。如果存在，则从数据字典中取出查询所涉及的视图的定义，把定义中的子查询和用户对视图的查询结合起来，转换成对基本表的查询，然后执行这个经过修正后的查询。

**例 11- 5：**利用 11.2.2 节中已创建的视图 **v\_s\_track**，在该视图中查询 **Circulation** 大于 5 万张的歌手名、歌曲名以及发行量。

**v\_s\_track** 的定义如下：

```
CREATE VIEW v_s_track
AS
SELECT  dbo.Singer.Name AS singer_name,  dbo.Songs.Name AS song_name,
        dbo.Track.Circulation
FROM    dbo.Singer INNER JOIN
        dbo.Track ON dbo.Singer.SingerID = dbo.Track.SingerID INNER JOIN
        dbo.Songs ON dbo.Track.SongID = dbo.Songs.SongID
```

对 **v\_s\_track** 进行查询的语句：

```
Select * from v_s_track where Circulation>5
```

DBMS 执行此查询时，将其转换成对三张基本表查询，修正后的查询语句为：

```
SELECT  dbo.Singer.Name AS singer_name,  dbo.Songs.Name AS song_name,
        dbo.Track.Circulation
FROM    dbo.Singer INNER JOIN
        dbo.Track ON dbo.Singer.SingerID = dbo.Track.SingerID INNER JOIN
        dbo.Songs ON dbo.Track.SongID = dbo.Songs.SongID
```



```
where Circulation>5
```

由上例可以看出，当需要对基本表做很复杂的查询时，可以先对基本表建立一个视图，然后只需对此视图进行查询，这样就不必再键入复杂的查询语句，从而简化了查询操作。

有时，将通过视图及查询语句转换为对基本表的查询语句是很直接的。但是在有些情况下，这种转换不能直接进行。

**例 11-6：**利用 11.2.2 节中例 11-3 所创建的视图 v\_singer\_num，查询人数在 2 名以上的国家及人数。

查询语句如下所示：

```
Select * from v_singer_num where singer_num>2
```

如果直接转换成对基本表的查询，将产生以下的查询语句：

```
select nation,count(singerid) as singer_num from singer  
group by nation where count(singerid)>2
```

这种转换显然是错误的，因为 WHERE 子句不能包含聚合函数。

正确地语句应为：

```
select nation,count(singerid) as singer_num from singer  
group by nation having count(singerid)>2
```

目前，对于这种分组视图的查询，大对数关系型数据库管理系统均能进行正确地转换。可用的方法是：DBMS 先执行定义视图 v\_singer\_num 的 SELECT 语句，得到了一个结果，把它作为一个临时表，假设命名为 tmp\_v\_singer\_num，然后将上面的查询语句改写为：

```
Select * from tmp_v_singer_num where singer_num>2
```

同样可以得到正确的结果。因此，对于用户来讲，可以把视图当做表一样进行查询，而不必关心 DBMS 如何处理。

## 2. 更新视图

就像我们可以更新一个表中的数据一样，视图也可以被更新。视图的更新包括插入（INSERT）、删除（DELETE）和修改（UPDATE）操作。对视图进行更新操作时，要注意不能违反基本表对数据的各种约束和规则要求。

由于视图是“虚表”，所以对视图的更新最终转化为对基表的更新。正因为这样，在关系数据库中，并不是所有的视图都是可更新的，因为有些视图的更新并不能有意义地转换成相应表的查询。要通过视图更新表数据，必须保证视图是可更新视图。一般来说，行列子集视图可以更新。

不同的数据库系统对于在视图上执行更新操作指定了不同条件。SQL Server 对于更新视图的限制如下：

- 1) 任何修改（包括 UPDATE、INSERT 和 DELETE 语句）都只能引用一个基表的列。
- 2) 在视图中修改的列必须直接引用表列中的基础数据，它们不能通过其他方式派生，例如通过：

- （1）聚合函数（AVG、COUNT、SUM、MIN、MAX 等）。
- （2）通过表达式并使用列计算出其他列。
- （3）使用集合运算符（UNION、UNION ALL、CROSSJOIN、EXCEPT 和 INTERSECT）形成的列。

- 3) 被修改的列不受 GROUP BY、HAVING 或 DISTINCT 子句的影响。

因此，在 11.3.2 节中创建的视图，只有例 11-2 中的视图因为是行列子集视图可以更新。例 11-3 的视图是分组视图，例 11-4 的视图是带表达式的视图，均不能更新，因为这些更新操作无法转换成对基本表的更新。

**例 11-7：**向例 11-2 建的视图 v\_lang 中插入一条记录。

```
insert into v_lang(SongID,Name,Lang)
values('S0104','因为爱情','中文')
```

转换成对基本表的插入语句为：

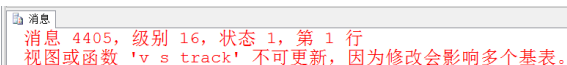
```
insert into Songs(SongID,Name,Lang)
values('S0104','因为爱情','中文')
```

**例 11-8：**在 11.3.2 节所建的视图 v\_s\_track 中执行如下插入操作：

```
insert into v_s_track values('王菲','红豆',10)
```

系统会报错，如图 11-9 所示。因为该操作涉及到三张基表的插入操作，而对于 Singers 表、Songs 表来说，只给出了 name 属性的值，而主键值没有给出，自然无法成功插入数据。

如果对视图更新的限制妨碍直接通过视图修改数据，可以使用具有支持 INSERT、UPDATE 和 DELETE 语句逻辑的 INSTEAD OF 触发器。详见第 13 章。

A screenshot of a database error message box. The title bar says "消息" (Message). The text inside reads: "消息 4405, 级别 16, 状态 1, 第 1 行 视图或函数 'v\_s\_track' 不可更新, 因为修改会影响多个基表。" (Message 4405, level 16, state 1, line 1 View or function 'v\_s\_track' cannot be updated, because the modification affects multiple base tables.)

消息 4405, 级别 16, 状态 1, 第 1 行  
视图或函数 'v\_s\_track' 不可更新, 因为修改会影响多个基表。

图 11-9 视图更新操作示例