# 实验二：OLAP与数据立方体之操作数据表

## 一、目的与要求

数据表是数据库中最基本的操作对象，通常说的把数据存放在数据库中其实就是存放在数据库中的一张张表中。本实验的主要内容是介绍SQL Server中的数据表和视图，并对数据表的基本操作进行详细介绍。

## 二、操作环境

实验采用头歌实践教学平台（www.educoder.net）进行自动测评，课程团队通过平台提供的公用云和公共镜像的方式提前在平台配置好软硬件环境及实验任务，操作系统为Ubuntu 16.04.3 LTS，SQL Server数据库为sqlserver2012，该环境适用于SQL Server基础课程使用，如数据库管理。

## 三、实验内容

### 第1关：表的创建和使用

#### 任务描述

本关任务：通过实验加深对数据库表的理解，掌握数据表的基本操作。

#### 相关知识

##### 什么是表

如果说数据库是一个仓库，那么表就是存放着物品的架子。

**数据表** 是数据库中最基本的操作对象，通常把数据存放在数据库中其实就是存放在数据库中的一张张表中，且一个数据库可以包含一个或多个表。

数据表中的数据按照行和列的规则来进行数据存储，每一行为一条数据记录，一条数据记录是由多个字段的描述信息组成的，每一列就是一个字段。

例如：有一个记录了员工信息的employee表，每一列就是用来描述员工特定信息的字段，比如工号、姓名等等；每一行则用于描述某一员工的所有字段信息：工号、姓名、性别和学历等，这些信息的集合称之为一条记录，如下图所示。

![表格

描述已自动生成]()

##### 操作数据表

SQL Server提供了非常丰富的数据表操作方法，用户可以通过企业管理器和T-SQL语言进行操作。使用T-SQL操作数据表有着灵活、快捷等特点，也是数据库管理人员使用最多的一种方式。

对于数据表的操作主要分为：

* CREATE TABLE创建数据表；
* ALTER TABLE修改数据表结构；
* DROP TABLE删除数据表。

###### 创建数据表

数据表的创建主要是对基本结构的构建，例如列属性的设定，数据完整性的约束。创建数据表使用CREATE TABLE语句，基本语法格式如下：

CREATE TABLE table\_name  
(  
 column\_name1 data\_type(size),  
 column\_name2 data\_type(size),  
 column\_name3 data\_type(size),  
....  
);

column\_name：列名；

data\_type：列的数据类型（例如 varchar、integer、decimal、date等等）；

size：列的最大长度。

例如，使用下面的CREATE TABLE语句创建一个名为 Persons的表，包含五列PersonID、LastName、FirstName、Address 和 City：

CREATE TABLE Persons  
(  
 PersonID int,  
 LastName varchar(255),  
 FirstName varchar(255),  
 Address varchar(255),  
 City varchar(255)  
);

利用select name, length, COLLATION from syscolumns where id=object\_Id('Persons')查看创建的Persons表结构：

![文本

描述已自动生成]()

###### 插入数据

创建完表后，可用INSERT INTO语句向表中插入数据。其语法如下：

INSERT INTO 表名称 VALUES (值1, 值2,....)

我们也可以指定所要插入数据的列，如下：

INSERT INTO 表名称 (列1, 列2,...) VALUES (值1, 值2,....)

例如向Persons表中插入Tom的个人信息，可以使用下列的语句完成。

INSERT INTO Persons VALUES('1001','Tom','Green','West Road','New York')

使用select \* from Persons语句查看插入结果。

手机屏幕截图

描述已自动生成

###### 修改表结构

在表已经创建好的前提下可以使用ALTER TABLE语句对表中的 **列** 进行 **增加、删除或修改数据类型** 等操作。

* 新增列：

ALTER TABLE table\_name  
ADD column\_name datatype

使用如下语句可以为数据表Persons添加一列AddedCol：

ALTER TABLE Persons  
ADD AddedCol varchar(50)

新增列后的表结构如下图：

![文本

描述已自动生成]()

* 删除列：

ALTER TABLE table\_name  
DROP COLUMN column\_name

例如，使用下列语句可以删除数据表Persons中的列City：

ALTER TABLE Persons  
DROP COLUMN City

删除City列后，表Persons的结构如下图：![文本

描述已自动生成]()

* 修改列的数据类型：

ALTER TABLE table\_name  
ALTER COLUMN column\_name datatype

下列语句把数据表Persons中AddedCol列的数据类型改为int类型：

ALTER TABLE persons ALTER COLUMN addedCol int

修改数据类型后的表结构如下：

![文本

描述已自动生成]()

###### 删除数据表

删除数据表是对数据库中已建立的表进行删除，在删除表的同时会对表中定义的数据、索引和视图同时进行清除。在做任何删除操作前应做好备份工作，我们使用DROP TABLE语句对数据库中的数据表进行删除，语法格式如下：

DROP TABLE table\_name

#### 编程要求

本次编程任务是：

* 在create table至create end中补全代码，创建Student、Course和Score三个表；
* 在insert begin和insert end中补全代码，向Score表中插入下面给出的数据；
* 在alt student table至alt end中补全代码，修改Student表结构，新增一列addr用于描述地址信息；
* 在del score table至del end中补全代码，删除Score表。

Student表结构如下：

| 字段名 | 数据类型 | 说明 |
| --- | --- | --- |
| sno | varchar(50) | 学号 |
| name | varchar(50) | 姓名 |
| sex | varchar(10) | 性别 |
| birthday | Date | 出生日期 |
| discipline | varchar(50) | 专业 |
| school | varchar(100) | 学院 |

Course表结构如下：

| 字段名 | 数据类型 | 说明 |
| --- | --- | --- |
| cno | varchar(50) | 课程号 |
| cname | varchar(50) | 课程名称 |
| description | varchar(256) | 课程说明 |
| credit | float | 学分 |
| cinstitution | varchar(128) | 开课单位 |

Score表结构如下：

| 字段名 | 数据类型 | 说明 |
| --- | --- | --- |
| sno | varchar(50) | 学号 |
| cno | varchar(50) | 课程号 |
| grade | float | 成绩 |

Score表待插入的数据：

| 学号 | 课程号 | 成绩 |
| --- | --- | --- |
| 09011101 | 101 | 89 |
| 09011101 | 102 | 78 |

#### 测试说明

平台将对你补充的代码进行评测，正确的输出结果请在测试集中查看。

### 第2关 数据库视图的创建和使用

#### 任务描述

本关任务：掌握视图的创建和使用。

#### 相关知识

##### 视图是什么

**视图**是一个虚拟表，其结构和内容是通过SQL查询获得的。用户可以通过SQL查询语句，可以像其他普通关系表一样，对视图中的数据进行查询。视图同样支持表的相关操作，并可以直接修改、添加、删除数据库中的真实数据。

视图通常用来集中、简化和自定义不同用户对数据库的不同认识，如视图可用作安全机制，方法是只允许用户由视图访问数据，而不授予用户直接访问数据表的权限。视图还可以用于提供向后兼容接口来模拟曾经存在但其架构已更改的表。

##### 视图的优缺点

###### 视图的优点

* 安全保密性。通过视图用户只能查询和修改他们所能见到的数据，数据库中的其他数据则既看不见也取不到。通过视图，用户可以被限制在数据的不同子集上；
* 查询简单性。视图能够从几个不同的关系表中提取数据，并且用一个单表表示出来，利用视图，将多表查询转换成视图的单表查询；
* 结构简单性。视图能够给用户一个“个性化”的数据库结构外观，用一组用户感兴趣的可见表来代表这个数据库的内容；
* 隔离变化。视图能够代表一个一致的、非变化的数据。即使是在作为视图基础的数据表（基表）被分割、重新构造或者重命名的情况下，也是如此；
* 逻辑数据独立性。视图可以使应用程序和数据库表在一定程度上独立。如果没有视图，应用一定是建立在表上的。有了视图之后，程序可以建立在视图上，从而使程序独立于数据表。

###### 视图的缺点

虽然视图存在上述的优点，但在定义数据库对象时，不能不加选择地来定义视图，因为视图也存在一些缺点：

* 性能。SQL Server必须把视图查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，即使是视图的一个简单查询，SQL Server也要把它变成一个复杂的结合体，需要花费一定的时间；
* 修改限制。当用户试图修改视图的某些信息时，数据库必须把它转化为对基本表的某些信息的修改，对于简单的视图来说，这是很方便的，但对于比较复杂的视图，可能是不可修改的。

##### 操作视图

###### 创建视图

使用T-SQL命令创建视图的基本语法如下：

--语法  
CREATE VIEW view\_name  
 AS  
 SELECT column\_name(s) FROM table\_name  
 WHERE condition

view\_name：指定创建视图的名称；  
column\_name：指定视图中列名称；  
table\_name：视图的基础表名称；  
condition：select的限定条件。

下面是一个创建视图的实例：

--准备表和数据  
create table studentTable(  
 id int identity(1,1)primary key,  
 name varchar(20),  
 gender char(2),  
 age int,  
)  
  
insert into studentTable (name,gender,age)  
values  
 ('刘备','男',28),  
 ('张飞','男',24),  
 ('关羽','男',26);

--删除已存在的视图student\_view  
if (exists (select \* from sys.objects where name = 'student\_view'))  
 drop view student\_view  
go  
  
--创建视图student\_view  
create view student\_view   
as  
select name,age from studentTable where age>24;  
  
--查询视图  
select \* from student\_view;

通过创建的视图查询的结果：

图片包含 图形用户界面

描述已自动生成

###### 通过视图向基本表中插入数据

--通过视图向基本表中插入数据  
create view stu\_insert\_view(编号,姓名,性别,年龄)  
as  
select id,name,gender,age from studentTable;  
go  
  
select \* from studentTable;  
  
---插入一条数据  
insert into stu\_insert\_view values('曹操2','男',40);  
  
----查看插入记录之后表中的内容。  
select \* from studentTable;

插入数据之前表中的数据如下：

![文本

低可信度描述已自动生成]()

通过视图插入数据后表中的数据如下：

文本

中度可信度描述已自动生成

###### 通过视图修改基本表的数据

--查看修改之前的数据  
select \* from studentTable;  
  
--修改数据  
update student\_view set age=30 where name='刘备';  
  
--查看修改后的数据  
select \* from studentTable;

修改视图之前表studentTable中的数据如下：

文本

中度可信度描述已自动生成

通过视图修改表studentTable中的数据后如下：

文本

中度可信度描述已自动生成

###### 通过视图删除基本表的数据

通过视图删除基本表数据的语法如下：

delete view\_name where condition;

--查看表中源数据  
select \* from studentTable;  
  
--删除记录  
delete student\_view where name ='关羽';  
  
--通过视图删除之后的数据  
select \* from studentTable;

删除之前的数据：

文本

中度可信度描述已自动生成

删除之后的数据：

图形用户界面, 文本

描述已自动生成

###### 修改视图

可以使用 ALTER VIEW 语句来对已有的视图进行修改。

ALTER VIEW view\_name AS select\_statement

语法说明如下：

* view\_name：指定视图的名称。该名称在数据库中必须是唯一的，不能与其他表或视图同名；
* select\_statement：指定创建视图的 SELECT 语句，可用于查询多个基表或源视图。

例如：

--修改视图  
alter view student\_view  
as  
select name,age from studentTable where age>30;

此时再执行select \* from student\_view会得到如下年龄大于30的记录：

图形用户界面, 文本

中度可信度描述已自动生成

###### 删除视图

可以使用drop view语句删除视图：

drop view view\_name1,view\_name2,......,view\_nameN;  
--该语句可以同时删除多个视图，只要在删除各视图名称之间用逗号分隔即可。

例如：

drop view student\_view;

#### 编程要求

本次编程任务是：

* 补全右侧代码片段中create score\_view\_cdept至create end中间的代码，创建学生的成绩单视图，包括学号sno、姓名name、课程名cname和成绩grade字段。

涉及到的数据库和表都已经建立好了，并插入了相应的数据，具体如下：  
Student表结构如下：

| sno | name | sex | birthday | discipline | school |
| --- | --- | --- | --- | --- | --- |
| s001 | Tom | male | 2019-06-01 | computer | information |
| s002 | Bob | male | 2019-06-01 | software | information |
| s003 | Alice | female | 2019-05-02 | computer | information |

Course表结构如下：

| cno | cname | description | credit | cinstitution |
| --- | --- | --- | --- | --- |
| c001 | network | computer network description | 2 | College Of Information Engineering |
| c002 | Java | java description | 2 | College Of Information Engineering |

Score表结构如下：

| sno | cno | grade |
| --- | --- | --- |
| s001 | c001 | 85 |
| s002 | c001 | 90 |

#### 测试说明

平台将对你编写的 SQL 进行评测：

预期输出：

图形用户界面, 文本, 应用程序

描述已自动生成