**课程报告设计**

**—— 教学管理系统的设计与开发**

**计算机学院 07111306**

**张滔 1120131939**

## 需求分析：

* 系统能够为不同角色（学生、老师，管理员等）提供不同的服务。
* 不同角色通过不同类型账号登录系统进行教学相关工作
* 学生能查询接收教学通知，查询课表、成绩、考试、作业等信息，能够修改个人信息（如密码）
* 老师能够发布教学通知，能够修改个人信息（如密码），管理学生信息，进行课程安排、考试安排，给学生评分，给学生布置作业等
* 管理员能够管理所有账户信息（查询、添加、删除、修改账户），能够发布系统通知

## 功能设计：

1. 登录部分，系统的入口。不同角色（老师、学生、管理员）能够选择身份进行账号密码登录。能够根据账号密码是否匹配进行相应提示或跳转。
2. 通知的发布与接收。教师能够发布教学通知，学生能够接收教学通知。管理员能发布系统通知，教师学生能够接收系统通知。
3. 课程管理。老师能够安排上课时间、地点，能够管理选修该课程的学生。学生能够查询自己的课程信息。
4. 作业管理。老师能够布置相应科目的作业。学生能够查询自己需要完成的作业。
5. 考试管理。老师能够安排自己课程的考试，包括考试时间、地点、考试要求（是否开卷等）。学生能够查询考试信息。
6. 成绩管理。老师能够给学生进行评价给分，以及相应描述。学生能够查询自己的考试成绩。
7. 个人信息管理。各个角色能够修改自己的登录密码。
8. 账户管理。管理员能够对所有角色（包括自己）的账号进行管理，包括增加新账户，删除账户，查询账户信息，修改账户角色、权限等。

## 功能框图：

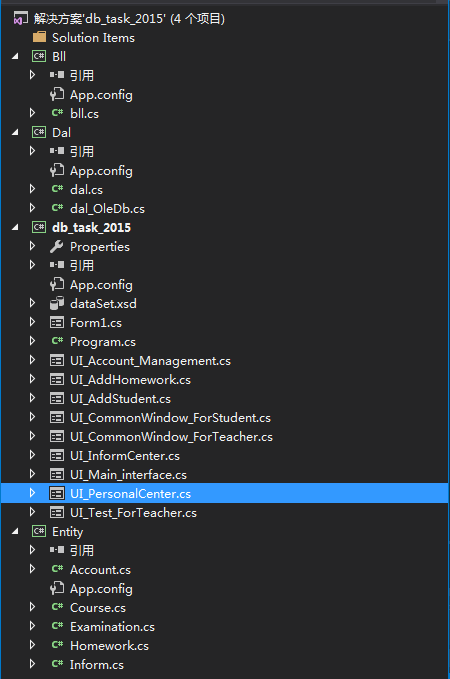
Dal

Bll

Bll

框图如上，在满足功能设计的基础上，采用三层开发模式，将功能剥离便于管理与控制。三类用户通过表示层(USL – User Show Layer)，UI层向业务逻辑层(BLL – Business Logical Layer)索取服务，Bll层向数据访问层(Dal – Data Access Layer)索取服务，Dal层分出OleDb子层，便于之后使用不同数据组件或者数据库时进行扩展。底层组织数据后向上层反馈。当开发时，清晰的分层有助于逻辑上的组织，工作上的划分，统一好各层的数据接口即可。

编程架构图如下：



**附件1**中给出了工程部分源代码。

## 业务流程：

考虑到在Erwin建模过程中会进行数据项的交代，在分析数据流部分只给出数据流涉及的数据结构和部分数据项，部分数据流的交代以处理过程为载体呈现。

数据结构：账户

含义说明：每个登录角色的唯一标识

组成数据项：账号，密码，角色类型

数据结构：个人信息

含义说明：与该登录角色相关的信息

组成数据项：账户，名称

数据结构：课程

含义说明：与该课程相关信息

组成数据项：课程号，课程名称，上课时间，上课地点

数据结构：考试

含义说明：考试相关信息

组成数据项：课程号，学生账号，考试时间，考试成绩

数据结构：通知

含义说明：通知相关信息

组成数据项：通知编号，通知类型，通知内容

处理过程：账户登录

含义说明：对账户信息进行验证

输入数据流：账户

输出数据流：登录结果提示

处理：成功则登入主界面，失败进行相应提示

处理过程：个人信息修改

含义说明： 修改登录密码

输入数据流： 个人信息

输出数据流： 修改后的个人信息

处理：原密码输入错误报错，否则修改成功

处理过程：课表查询

含义说明： 查询该学生该学期所有课程安排

输入数据流： 学生账号

输出数据流： 该学生该学期所有课程安排

处理过程：考试查询

含义说明：查询该学生该学期所有考试安排

输入数据流： 学生账号

输出数据流：该学生该学期所有考试安排

处理过程：成绩查询

含义说明：查询该学生该学期所有考试成绩

输入数据流： 学生账号

输出数据流：该学生该学期所有考试成绩

数据流名：通知下发

说明：由发送者账号和通知数据项构成

来源：教师或者管理员

去向：学生或者教师

数据流名：管理课表

说明：任课教师管理所教授课程

来源：原课程表（若新创建课表则为空）

去向：存储进数据库

数据流名：管理考试

说明：任课教师管理所教授课程的考试安排

来源：原考试安排（若新创建考试安排，则为空）

去向：存储进数据库

数据流名：管理成绩

说明：任课教师管理所教授课程，给某一学生打分评估

来源：老师给出成绩

去向：存储进数据库

数据流名：账户管理

说明：管理员管理账户信息

来源：原有账户信息（若新创建账户，则为空）

去向：存储进数据库

## C:\Users\lenovo\Desktop\捕获.PNGErwin数据建模：

结合上表，简要给出建表逻辑与实体定义与联系。在**附件2**中给出Erwin向Sql Server中转换时的建表代码。

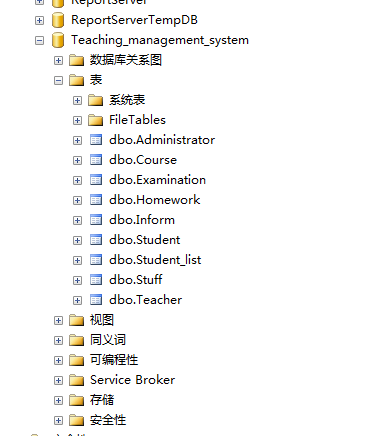
关键实体：

老师、学生、管理员构成主要用户

课程、考试、成绩构成师生交互的主体基础

通知是另一个各用户之间交互的主体

以课程实体为基础，产生课程学生名单实体

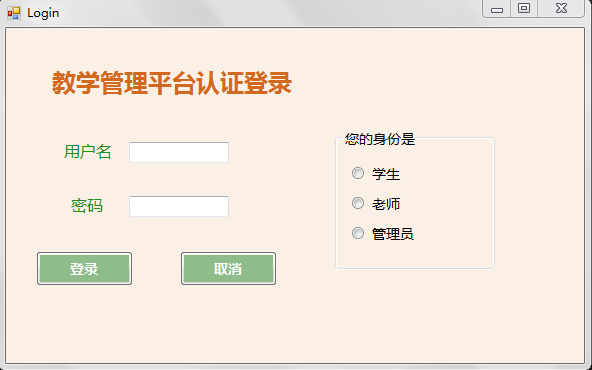
在Sql Server中建立数据库如下：

## VS .NET编程

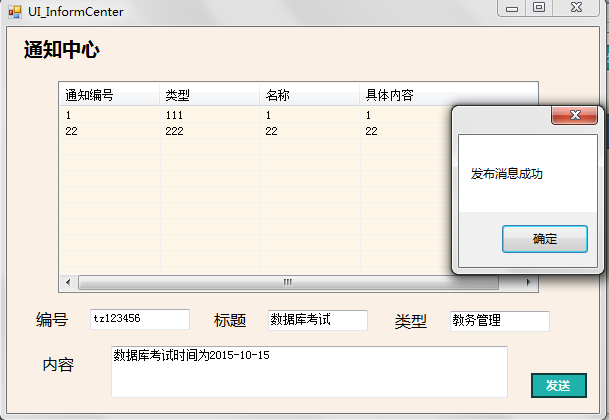
下面给出软件界面效果，进行简要功能说明。详细说明见**系统使用说明。**程序入口处，在系统后台(Sql Server)插入一条管理员(admin1)的信息，由admin1来开始流程操作。

insert into Stuff values('1','0001','1','管理员','admin1')

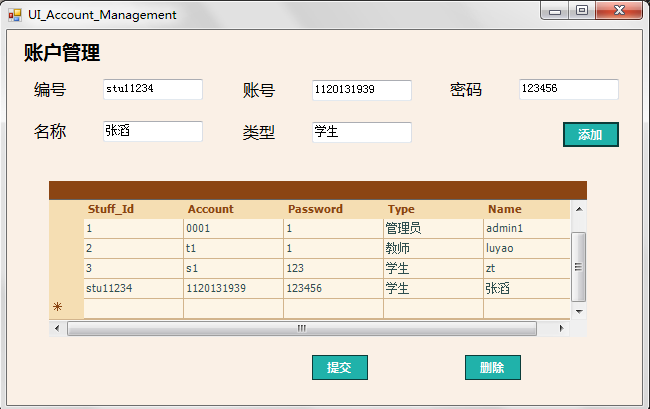
1.登录界面，选择类型登录，若信息填写不完整或者账户信息不匹配则会提示。



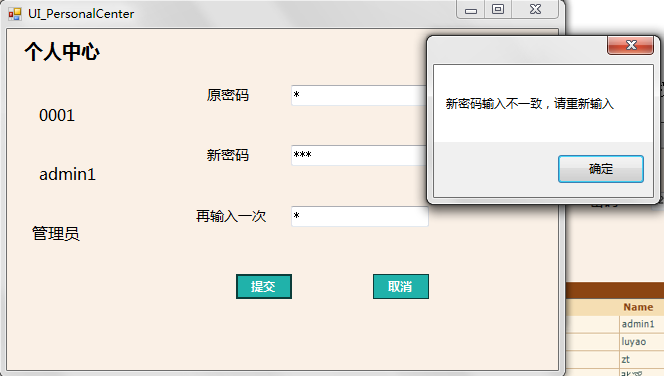
2.以管理员身份登录进入主界面。

3.进入通知界面，可以发布通知。

4.进入账户管理界面，可以添加，修改，管理账户信息(管理员特有权限)。



5.进入个人信息界面，可以修改自己的密码，若填写有误则提示。



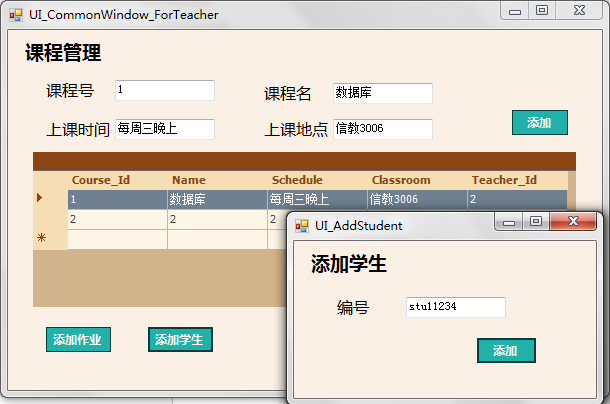
6.用管理员添加老师账号后，登录老师用户主界面。通知与个人信息界面与之前相同，不再赘述。



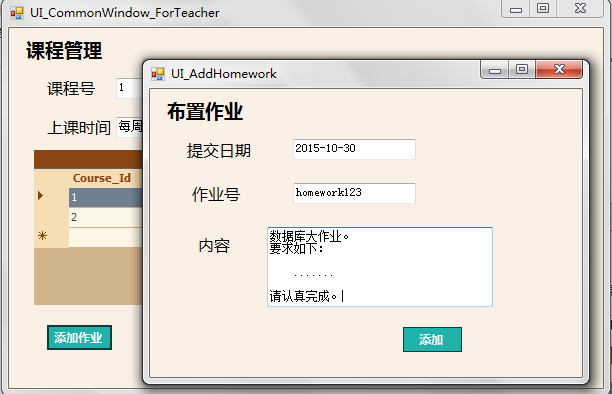
7.点击管理课程按钮。此界面可以添加课程，添加该课程作业，将学生加入该课程



在添加学生界面中，将之前用管理员账户添加的学生加入“数据库”课程。



在添加作业界面中，为参加数据库课程所有学生布置作业。



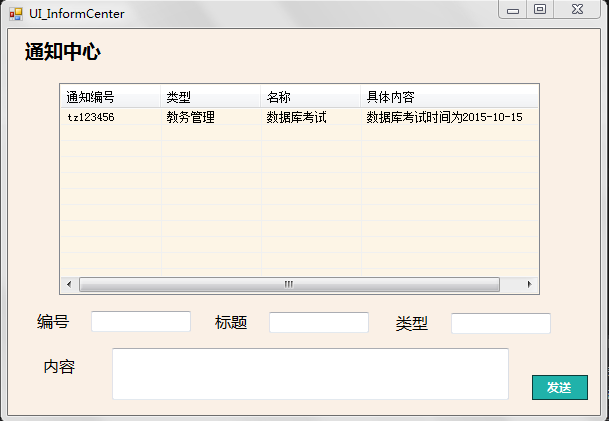
8.退回上一级界面，点击管理考试按钮。在此界面可以对所教授课程的考试进行管理，可以评定成绩。



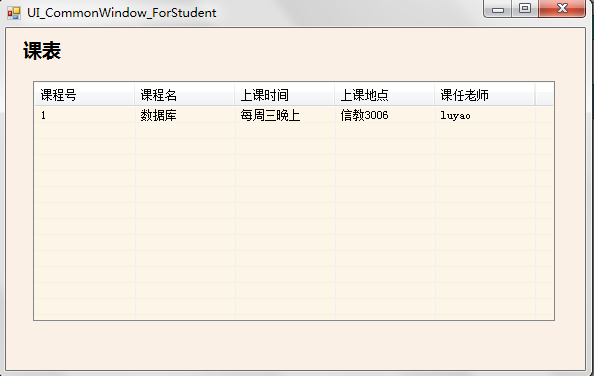
9.下面以学生角色登录，进入主界面。



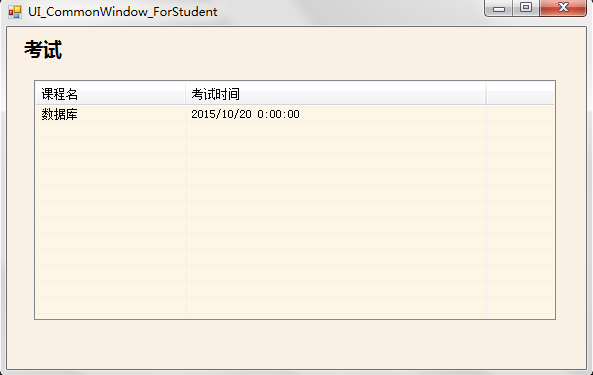
10.进入通知界面，可以看到接收到了之前的通知。



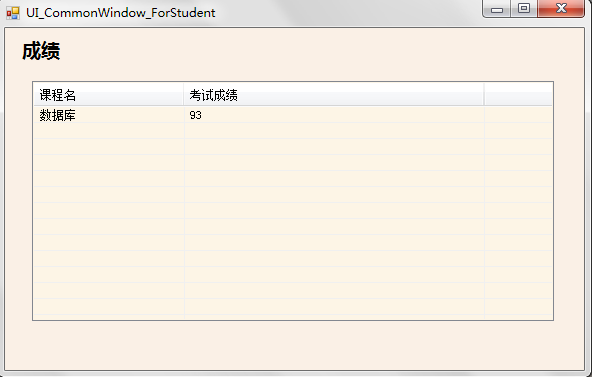
11.进入课表界面，可以查看当前课表。



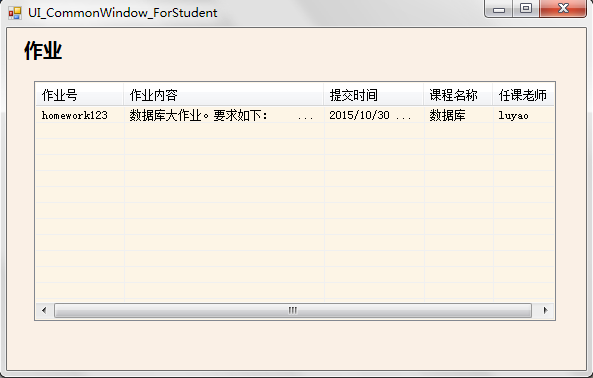
12.进入考试界面，可以查看考试。



13.进入成绩界面，可以查看成绩。



14.进入作业界面，可以查看作业。



15.点击退出按钮，退回到登录界面。

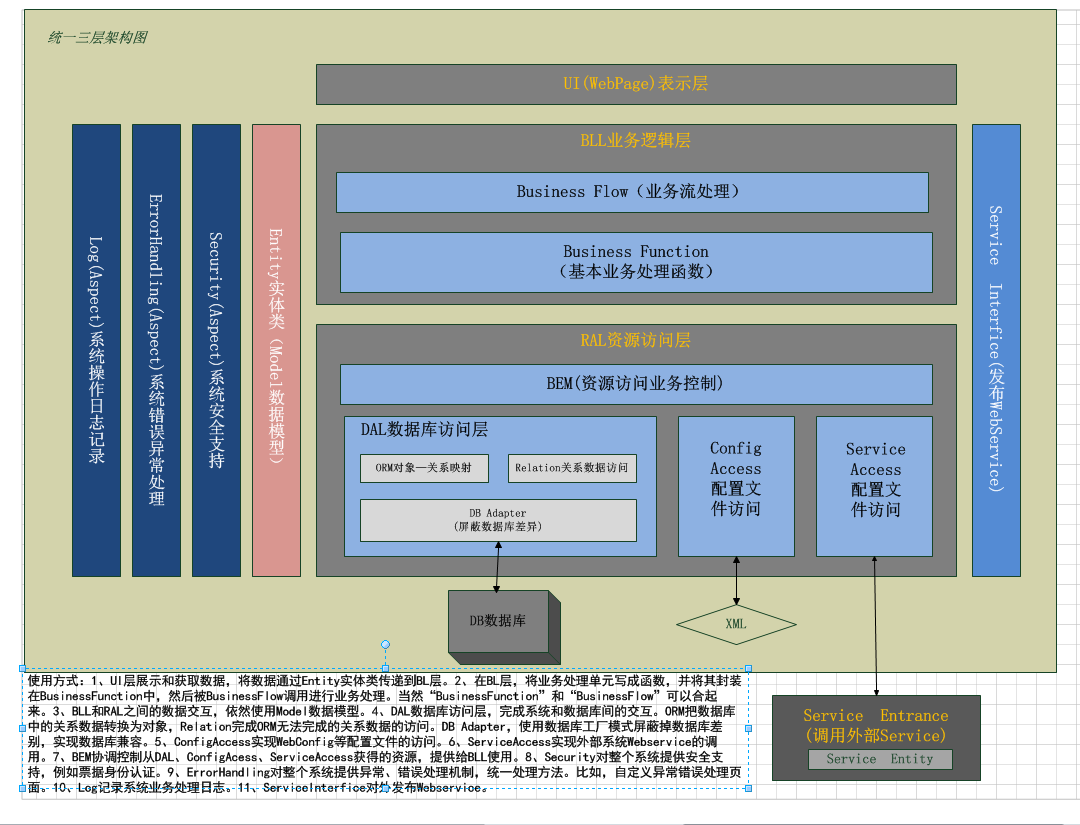
## 功能检查与程序调试：

在本次实验中。由于功能较多，代码编写量较大，出现了一些问题。如：

* .NET编程与sql server数据库对接时出现增、删、改、查异常
* 编写UI层代码时对于控件的不熟悉
* 建模过程中几次删改模型，在尽量减少连接查询的基础上消除数据冗余
* Bll层的一些简单的数据处理细节的错误
* …….

有赖于三层模型架构，使得工程结构较为清晰，每次都能较快地发现问题并解决。

## 总结：

本次数据库大作业，选择了教学管理系统进行开发。参考了我校教学管理系统的UI以及功能模块：<http://10.5.2.80/(rh1dpcnzmrop1v45rl2nlm55)/default2.aspx>，在此基础上进行功能剥离与模型简化，得到了本次作业结果。开发过程中运用了三层架构模型，极大地方便了工程的模块化，增加了编程效率与碎片化编程的可能。

在实验中，综合运用了小学期所学到的理论知识和实践工具，对于数据库系统的理解更为全面与透彻，编程水平也有了很大提高。

## 附件1：

**登录界面，Form1.cs：**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Bll;

using Entity;

namespace db\_task\_2015

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void login\_Click(object sender, EventArgs e)

{

String account = this.account\_textbox.Text;

String password = this.password\_textBox.Text;

if (account == "" || password == "")

{

MessageBox.Show("请填写完整的账户信息");

}

else

{

int flag = -1;

if (this.student\_radioButton.Checked)

{

flag = 0;

}

else if (this.teacher\_radioButton.Checked)

{

flag = 1;

}

else if (this.admin\_radioButton.Checked)

{

flag = 2;

}

if (-1 == flag)

{

MessageBox.Show("请选择您的登录类型");

}

else

{

bll a = new bll();

Account info\_account = a.check\_login\_info(account, password, flag);

if(null == info\_account)

{

MessageBox.Show("您填写的信息有误，请重新登录！");

}

else

{

UI\_Main\_interface main\_interface = new UI\_Main\_interface(info\_account.stuff\_id,

info\_account.account, info\_account.password,

info\_account.name, info\_account.type, flag);

//this.IsMdiContainer = true;

// main\_interface.MdiParent = this;

//this.Hide();

main\_interface.Show();

//this.Close();

}

}

}

}

private void Form1\_Load(object sender, EventArgs e)

{

this.password\_textBox.PasswordChar = '\*';

}

}

}

**主界面，UI\_Main\_Interface.cs:**

public partial class UI\_Main\_interface : Form

{

String stuff\_id;

String account;

String password;

String name;

String type;

int flag;

public UI\_Main\_interface(String stuff\_id, String account, String password,

String name, String type, int flag)

{

InitializeComponent();

this.stuff\_id = stuff\_id;

this.account = account;

this.password = password;

this.name = name;

this.type = type;

this.flag = flag;

}

private void UI\_Main\_interface\_Load(object sender, EventArgs e)

{

this.account\_label.Text = account;

this.name\_label.Text = name;

this.type\_label.Text = type;

init\_interface(flag);

}

private void init\_interface(int flag)

{

this.tab\_bt\_1.Text = "通知";

this.tab\_bt\_5.Text = "个人信息";

this.quit\_bt.Text = "退出";

if (flag == 2) //若是管理员

{

this.tab\_bt\_6.Visible = false;

this.tab\_bt\_3.Visible = false;

this.tab\_bt\_4.Visible = false;

this.tab\_bt\_2.Text = "管理账户";

}

else if(flag == 1)

{

this.tab\_bt\_6.Visible = false;

this.tab\_bt\_4.Visible = false;

this.tab\_bt\_2.Text = "管理课程";

this.tab\_bt\_3.Text = "管理考试";

//this.tab\_bt\_4.Text = "成绩测评";

}

else

{

this.tab\_bt\_2.Text = "课表";

this.tab\_bt\_3.Text = "考试";

this.tab\_bt\_4.Text = "成绩";

this.tab\_bt\_6.Text = "作业";

}

}

private void tab\_bt\_5\_Click(object sender, EventArgs e)

{

UI\_PersonalCenter personal\_center = new UI\_PersonalCenter(stuff\_id, account,

password, name, type);

personal\_center.Show();

}

private void quit\_bt\_Click(object sender, EventArgs e)

{

this.Close();

}

private void tab\_bt\_2\_Click(object sender, EventArgs e)

{

if(2 == flag) //若是管理员则tab2为管理账户

{

UI\_Account\_Management personal\_center = new UI\_Account\_Management();

personal\_center.Show();

}else if(1 == flag) //若是教师则tab2为管理课程

{

UI\_CommonWindow\_ForTeacher common\_window = new UI\_CommonWindow\_ForTeacher(stuff\_id);

common\_window.Show();

}

else //若为学生则为查看课表

{

show\_CommonWin\_ForStu(2);

}

}

private void tab\_bt\_1\_Click(object sender, EventArgs e)

{

//bool is\_student = flag == 0 ? true : false;

UI\_InformCenter inform\_center = new UI\_InformCenter(stuff\_id);

inform\_center.Show();

}

private void tab\_bt\_3\_Click(object sender, EventArgs e)

{

if(1 == flag)

{

UI\_Test\_ForTeacher test\_forTeacher = new UI\_Test\_ForTeacher();

test\_forTeacher.Show();

}

else

{

show\_CommonWin\_ForStu(3);

}

}

private void tab\_bt\_4\_Click(object sender, EventArgs e)

{

show\_CommonWin\_ForStu(4);

}

private void tab\_bt\_6\_Click(object sender, EventArgs e)

{

show\_CommonWin\_ForStu(5);

}

private void show\_CommonWin\_ForStu(int flag)

{

UI\_CommonWindow\_ForStudent common\_window = new UI\_CommonWindow\_ForStudent(flag, stuff\_id);

common\_window.Show();

}

}

**Bll层，bll.cs:**

public class bll

{

public Account check\_login\_info(String account, String password, int flag)

{

String table\_name = "";

switch (flag)

{

case 0:

table\_name = "Student";

break;

case 1:

table\_name = "Teacher";

break;

case 2:

table\_name = "Administrator";

break;

default:

break;

}

String selectAccountInfo\_str = "select Stuff.\* " + "from Stuff," + table\_name + " where " +

"Stuff.Account='" + account +

"' and Stuff.Password='" + password +

"' and Stuff.Stuff\_Id = " + table\_name + ".Stuff\_Id ";

dal a = new dal();

Account info\_account = a.sql\_Query(selectAccountInfo\_str);

return info\_account;

}

public bool change\_password(String stuff\_id, String new\_password)

{

bool if\_success = false;

String updateAccountInfo\_str = "update Stuff set password='" + new\_password + "' where Stuff\_Id='" + stuff\_id + "'";

dal a = new dal();

if\_success = a.sql\_update\_AccountInfo(updateAccountInfo\_str, new\_password);

return if\_success;

}

public bool add\_student(String student\_id, String course\_id)

{

bool if\_success = false;

String sql = "insert into Student\_list values('" + student\_id + "','" + course\_id + "')";

dal a = new dal();

if\_success = a.sql\_add\_Student(sql);

return if\_success;

}

public bool add\_homework(String homework\_id , String content, String date,

String course\_id, String teacher\_id)

{

bool if\_success = false;

String sql = "insert into Homework values('" + homework\_id + "','" + content

+ "','" + date + "','" + course\_id + "','" + teacher\_id + "')";

dal a = new dal();

if\_success = a.sql\_add\_Homework(sql);

return if\_success;

}

public List<Inform> get\_Inform()

{

String sql = "select \* from Inform";

dal a = new dal();

return a.sql\_get\_Inform(sql);

}

public List<Course> get\_Course(String student\_id)

{

String sql = "select \* from Course where Course\_Id=" +

"(select Course\_Id from Student\_list where Student\_Id='" + student\_id + "')";

dal a = new dal();

return a.sql\_get\_Course(sql);

}

public List<Examination> get\_Exam(String student\_id)

{

String sql = "select \* from Examination where Student\_Id='" + student\_id + "'";

dal a = new dal();

return a.sql\_get\_Exam(sql);

}

public List<Homework> get\_Homework(String student\_id)

{

String sql = "select \* from Homework,Student\_list where Student\_Id='" +

student\_id + "'" + "and Student\_list.Course\_Id=Homework.Course\_Id";

dal a = new dal();

return a.sql\_get\_Homework(sql);

}

public bool insert\_Inform(String info\_id, String type,

String title, String content, String stuff\_id)

{

bool if\_success = false;

String date = DateTime.Today.Date.ToString();

String sql = "insert into Inform values('" + info\_id +"','"+ title +

"','" + content + "','" + type + "','" +

date + "','" + stuff\_id + "')";

dal a = new dal();

if\_success = a.sql\_insert\_Inform(sql);

return if\_success;

}

}

**Dal层，dal.cs:**

public class dal

{

public void sql\_NonQuery(String sql)

{

dal\_OleDb.myExecuteNon(sql);

}

public Account sql\_Query(String sql)

{

Account account = dal\_OleDb.myQuery(sql);

return account;

}

public bool sql\_update\_AccountInfo(String sql, String new\_password)

{

bool if\_success = dal\_OleDb.update\_AccountInfo(sql, new\_password);

return if\_success;

}

public bool sql\_add\_Student(String sql)

{

bool if\_success = dal\_OleDb.add\_Student(sql);

return if\_success;

}

public bool sql\_add\_Homework(String sql)

{

bool if\_success = dal\_OleDb.add\_Homework(sql);

return if\_success;

}

public List<Inform> sql\_get\_Inform(String sql)

{

return dal\_OleDb.Inform\_query(sql);

}

public List<Course> sql\_get\_Course(String sql)

{

return dal\_OleDb.get\_Course(sql);

}

public List<Examination> sql\_get\_Exam(String sql)

{

return dal\_OleDb.get\_Exam(sql);

}

public List<Homework> sql\_get\_Homework(String sql)

{

return dal\_OleDb.get\_Homework(sql);

}

public bool sql\_insert\_Inform(String sql)

{

bool if\_success = dal\_OleDb.insert\_Inform(sql);

return if\_success;

}

}

**OleDb层，dal\_OleDb.cs:**

public class dal\_OleDb

{

public static void myExecuteNon(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

public static Account myQuery(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

Account info\_account = new Account();

if (reader.HasRows)

{

while (reader.Read())

{

info\_account.stuff\_id = (String)reader[0];

info\_account.account = (String)reader[1];

info\_account.name = (String)reader[4];

info\_account.type = (String)reader[3];

}

}

else

{

info\_account = null;

}

reader.Close();

conn.Close();

return info\_account;

}

public static List<Inform> Inform\_query(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

List<Inform> list = new List<Inform>();

if (reader.HasRows)

{

while (reader.Read())

{

Inform inform = new Inform();

inform.info\_id = (String)reader[0];

inform.content = (String)reader[2];

inform.type = (String)reader[3];

inform.title = (String)reader[1];

list.Add(inform);

}

}

else

{

list = null;

}

return list;

}

public static List<Course> get\_Course(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

List<Course> list = new List<Course>();

if (reader.HasRows)

{

while (reader.Read())

{

Course course = new Course();

course.Course\_Id = (String)reader[0];

course.Name = (String)reader[1];

course.Schedule = (String)reader[2];

course.Classroom = (String)reader[3];

course.Teacher\_Id = (String)reader[4];

String sql\_forSelect = "select Name from Stuff where Stuff\_Id='" + course.Teacher\_Id + "'";

course.Teacher\_Name = sql\_selectOneString(sql\_forSelect);

list.Add(course);

}

}

else

{

list = null;

}

return list;

}

public static List<Examination> get\_Exam(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

List<Examination> list = new List<Examination>();

if (reader.HasRows)

{

while (reader.Read())

{

Examination exam = new Examination();

exam.Course\_Id = (String)reader[0];

exam.Date = reader[2].ToString();

exam.Score = reader[3].ToString();

String sql\_forSelect = "select Name from Course where Course\_Id='" + exam.Course\_Id + "'";

exam.Course\_Name = sql\_selectOneString(sql\_forSelect);

list.Add(exam);

}

}

else

{

list = null;

}

return list;

}

public static List<Homework> get\_Homework(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

List<Homework> list = new List<Homework>();

if (reader.HasRows)

{

while (reader.Read())

{

Homework homework = new Homework();

homework.homework\_id = (String)reader[0];

homework.content = (String)reader[1];

homework.date = (String)reader[2].ToString();

String course\_id = (String)reader[3];

String teacher\_id = (String)reader[4];

String sql\_forSelect = "select Name from Course where Course\_Id='" + course\_id + "'";

homework.course\_name = sql\_selectOneString(sql\_forSelect);

sql\_forSelect = "select Name from Stuff where Stuff\_Id='" + teacher\_id + "'";

homework.teacher\_name = sql\_selectOneString(sql\_forSelect);

list.Add(homework);

}

}

else

{

list = null;

}

return list;

}

public static String sql\_selectOneString(String sql)

{

String target = "";

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

OleDbDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

target = (String)reader[0];

}

}

return target;

}

public static bool update\_AccountInfo(String sql, String new\_password)

{

return sql\_insert(sql);

}

public static bool insert\_Inform(String sql)

{

return sql\_insert(sql);

}

public static bool add\_Student(String sql)

{

return sql\_insert(sql);

}

public static bool add\_Homework(String sql)

{

return sql\_insert(sql);

}

public static bool sql\_insert(String sql)

{

OleDbConnection conn = initial();

OleDbCommand cmd = new OleDbCommand(sql, conn);

conn.Open();

try

{

cmd.ExecuteNonQuery();

return true;

}

catch

{

return false;

}

finally

{

conn.Close();

}

}

private static OleDbConnection initial()

{

String conn\_str = "Provider=SQLNCLI11;Data Source=(local);Integrated Security=SSPI;Initial Catalog=Teaching\_management\_system;";

OleDbConnection conn = new OleDbConnection(conn\_str);

return conn;

}

}

## 附件2：

CREATE TABLE [Administrator]

(

[Stuff\_Id] varchar(20) NOT NULL

)

Execution Successful

ALTER TABLE [Administrator]

ADD CONSTRAINT [XPKAdministrator] PRIMARY KEY CLUSTERED ([Stuff\_Id] ASC)

Execution Successful

CREATE TABLE [Course]

(

[Course\_Id] varchar(20) NOT NULL ,

[Name] varchar(20) NOT NULL ,

[Schedule] varchar(20) NOT NULL ,

[Classroom] varchar(20) NOT NULL ,

[Teacher\_Id] varchar(20) NULL

)

Execution Successful

ALTER TABLE [Course]

ADD CONSTRAINT [XPKCourse] PRIMARY KEY CLUSTERED ([Course\_Id] ASC)

Execution Successful

CREATE TABLE [Examination]

(

[Course\_Id] varchar(20) NOT NULL ,

[Student\_Id] varchar(20) NOT NULL ,

[Date] datetime NOT NULL ,

[Score] integer NOT NULL

)

Execution Successful

ALTER TABLE [Examination]

ADD CONSTRAINT [XPKExamination] PRIMARY KEY CLUSTERED ([Course\_Id] ASC,[Student\_Id] ASC)

Execution Successful

CREATE TABLE [Homework]

(

[Homework\_Id] varchar(20) NOT NULL ,

[Content] varchar(100) NOT NULL ,

[Commit\_Date] datetime NOT NULL ,

[Course\_Id] varchar(20) NOT NULL ,

[Teacher\_Id] varchar(20) NOT NULL

)

Execution Successful

ALTER TABLE [Homework]

ADD CONSTRAINT [XPKHomework] PRIMARY KEY CLUSTERED ([Homework\_Id] ASC,[Course\_Id] ASC,[Teacher\_Id] ASC)

Execution Successful

CREATE TABLE [Inform]

(

[Info\_Id] varchar(20) NOT NULL ,

[Title] varchar(20) NULL ,

[Content] varchar(100) NULL ,

[Type] varchar(20) NULL ,

[Time] datetime NULL ,

[Stuff\_Id] varchar(20) NULL

)

Execution Successful

ALTER TABLE [Inform]

ADD CONSTRAINT [XPKInformation] PRIMARY KEY CLUSTERED ([Info\_Id] ASC)

Execution Successful

CREATE TABLE [Student]

(

[Stuff\_Id] varchar(20) NOT NULL ,

[Class] varchar(20) NOT NULL

)

Execution Successful

ALTER TABLE [Student]

ADD CONSTRAINT [XPKStudent] PRIMARY KEY CLUSTERED ([Stuff\_Id] ASC)

Execution Successful

CREATE TABLE [Student\_list]

(

[Student\_Id] varchar(20) NOT NULL ,

[Course\_Id] varchar(20) NOT NULL

)

Execution Successful

ALTER TABLE [Student\_list]

ADD CONSTRAINT [XPKCourse\_list] PRIMARY KEY CLUSTERED ([Student\_Id] ASC,[Course\_Id] ASC)

Execution Successful

CREATE TABLE [Stuff]

(

[Stuff\_Id] varchar(20) NOT NULL ,

[Account] varchar(20) NOT NULL ,

[Password] varchar(20) NOT NULL ,

[Type] varchar(20) NOT NULL ,

[Name] varchar(20) NOT NULL

)

Execution Successful

ALTER TABLE [Stuff]

ADD CONSTRAINT [XPKStuff] PRIMARY KEY CLUSTERED ([Stuff\_Id] ASC)

Execution Successful

CREATE TABLE [Teacher]

(

[Stuff\_Id] varchar(20) NOT NULL ,

[Position] varchar(20) NULL

)

Execution Successful

ALTER TABLE [Teacher]

ADD CONSTRAINT [XPKTeacher] PRIMARY KEY CLUSTERED ([Stuff\_Id] ASC)

Execution Successful

ALTER TABLE [Administrator]

ADD CONSTRAINT [R\_3] FOREIGN KEY ([Stuff\_Id]) REFERENCES [Stuff]([Stuff\_Id])

ON DELETE CASCADE

ON UPDATE CASCADE

Execution Successful

ALTER TABLE [Course]

ADD CONSTRAINT [R\_8] FOREIGN KEY ([Teacher\_Id]) REFERENCES [Teacher]([Stuff\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Examination]

ADD CONSTRAINT [R\_4] FOREIGN KEY ([Course\_Id]) REFERENCES [Course]([Course\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Examination]

ADD CONSTRAINT [R\_5] FOREIGN KEY ([Student\_Id]) REFERENCES [Student]([Stuff\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Homework]

ADD CONSTRAINT [R\_18] FOREIGN KEY ([Course\_Id]) REFERENCES [Course]([Course\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Homework]

ADD CONSTRAINT [R\_19] FOREIGN KEY ([Teacher\_Id]) REFERENCES [Teacher]([Stuff\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Inform]

ADD CONSTRAINT [R\_20] FOREIGN KEY ([Stuff\_Id]) REFERENCES [Administrator]([Stuff\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Inform]

ADD CONSTRAINT [R\_21] FOREIGN KEY ([Stuff\_Id]) REFERENCES [Teacher]([Stuff\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Student]

ADD CONSTRAINT [R\_1] FOREIGN KEY ([Stuff\_Id]) REFERENCES [Stuff]([Stuff\_Id])

ON DELETE CASCADE

ON UPDATE CASCADE

Execution Successful

ALTER TABLE [Student\_list]

ADD CONSTRAINT [R\_22] FOREIGN KEY ([Student\_Id]) REFERENCES [Student]([Stuff\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Student\_list]

ADD CONSTRAINT [R\_23] FOREIGN KEY ([Course\_Id]) REFERENCES [Course]([Course\_Id])

ON DELETE NO ACTION

ON UPDATE NO ACTION

Execution Successful

ALTER TABLE [Teacher]

ADD CONSTRAINT [R\_2] FOREIGN KEY ([Stuff\_Id]) REFERENCES [Stuff]([Stuff\_Id])

ON DELETE CASCADE

ON UPDATE CASCADE

Execution Successful

CREATE TRIGGER tD\_Administrator ON Administrator FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Administrator \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Administrator Inform on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002201d", PARENT\_OWNER="", PARENT\_TABLE="Administrator"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_20", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Inform

WHERE

/\* %JoinFKPK(Inform,deleted," = "," AND") \*/

Inform.Stuff\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Administrator because Inform exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Administrator on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Administrator"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_3", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Stuff

WHERE

/\* %JoinFKPK(deleted,Stuff," = "," AND") \*/

deleted.Stuff\_Id = Stuff.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Administrator

WHERE

/\* %JoinFKPK(Administrator,Stuff," = "," AND") \*/

Administrator.Stuff\_Id = Stuff.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Administrator because Stuff exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Administrator ON Administrator FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Administrator \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insStuff\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Administrator Inform on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="000261fd", PARENT\_OWNER="", PARENT\_TABLE="Administrator"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_20", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Inform

WHERE

/\* %JoinFKPK(Inform,deleted," = "," AND") \*/

Inform.Stuff\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Administrator because Inform exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Administrator on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Administrator"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_3", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Stuff

WHERE

/\* %JoinFKPK(inserted,Stuff) \*/

inserted.Stuff\_Id = Stuff.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Administrator because Stuff does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Course ON Course FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Course \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Course Student\_list on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0003eecd", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_23", FK\_COLUMNS="Course\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Student\_list

WHERE

/\* %JoinFKPK(Student\_list,deleted," = "," AND") \*/

Student\_list.Course\_Id = deleted.Course\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Course because Student\_list exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Course Homework on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_18", FK\_COLUMNS="Course\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Homework

WHERE

/\* %JoinFKPK(Homework,deleted," = "," AND") \*/

Homework.Course\_Id = deleted.Course\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Course because Homework exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Course Examination on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="Course\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Examination

WHERE

/\* %JoinFKPK(Examination,deleted," = "," AND") \*/

Examination.Course\_Id = deleted.Course\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Course because Examination exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Teacher Course on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Course"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_8", FK\_COLUMNS="Teacher\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Teacher

WHERE

/\* %JoinFKPK(deleted,Teacher," = "," AND") \*/

deleted.Teacher\_Id = Teacher.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Course

WHERE

/\* %JoinFKPK(Course,Teacher," = "," AND") \*/

Course.Teacher\_Id = Teacher.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Course because Teacher exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Course ON Course FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Course \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insCourse\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Course Student\_list on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00048f9c", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_23", FK\_COLUMNS="Course\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Course\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Student\_list

WHERE

/\* %JoinFKPK(Student\_list,deleted," = "," AND") \*/

Student\_list.Course\_Id = deleted.Course\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Course because Student\_list exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Course Homework on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_18", FK\_COLUMNS="Course\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Course\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Homework

WHERE

/\* %JoinFKPK(Homework,deleted," = "," AND") \*/

Homework.Course\_Id = deleted.Course\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Course because Homework exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Course Examination on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="Course\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Course\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Examination

WHERE

/\* %JoinFKPK(Examination,deleted," = "," AND") \*/

Examination.Course\_Id = deleted.Course\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Course because Examination exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Teacher Course on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Course"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_8", FK\_COLUMNS="Teacher\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Teacher\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Teacher

WHERE

/\* %JoinFKPK(inserted,Teacher) \*/

inserted.Teacher\_Id = Teacher.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

select @nullcnt = count(\*) from inserted where

inserted.Teacher\_Id IS NULL

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Course because Teacher does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Examination ON Examination FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Examination \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Student Examination on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00026467", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="Student\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Student

WHERE

/\* %JoinFKPK(deleted,Student," = "," AND") \*/

deleted.Student\_Id = Student.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Examination

WHERE

/\* %JoinFKPK(Examination,Student," = "," AND") \*/

Examination.Student\_Id = Student.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Examination because Student exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Course Examination on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="Course\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Course

WHERE

/\* %JoinFKPK(deleted,Course," = "," AND") \*/

deleted.Course\_Id = Course.Course\_Id AND

NOT EXISTS (

SELECT \* FROM Examination

WHERE

/\* %JoinFKPK(Examination,Course," = "," AND") \*/

Examination.Course\_Id = Course.Course\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Examination because Course exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Examination ON Examination FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Examination \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insCourse\_Id varchar(20),

@insStudent\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Student Examination on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002c50e", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="Student\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Student\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Student

WHERE

/\* %JoinFKPK(inserted,Student) \*/

inserted.Student\_Id = Student.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Examination because Student does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Course Examination on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_4", FK\_COLUMNS="Course\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Course\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Course

WHERE

/\* %JoinFKPK(inserted,Course) \*/

inserted.Course\_Id = Course.Course\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Examination because Course does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Homework ON Homework FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Homework \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Teacher Homework on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="000255ae", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_19", FK\_COLUMNS="Teacher\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Teacher

WHERE

/\* %JoinFKPK(deleted,Teacher," = "," AND") \*/

deleted.Teacher\_Id = Teacher.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Homework

WHERE

/\* %JoinFKPK(Homework,Teacher," = "," AND") \*/

Homework.Teacher\_Id = Teacher.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Homework because Teacher exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Course Homework on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_18", FK\_COLUMNS="Course\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Course

WHERE

/\* %JoinFKPK(deleted,Course," = "," AND") \*/

deleted.Course\_Id = Course.Course\_Id AND

NOT EXISTS (

SELECT \* FROM Homework

WHERE

/\* %JoinFKPK(Homework,Course," = "," AND") \*/

Homework.Course\_Id = Course.Course\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Homework because Course exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Homework ON Homework FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Homework \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insHomework\_Id varchar(20),

@insCourse\_Id varchar(20),

@insTeacher\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Teacher Homework on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00029b70", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_19", FK\_COLUMNS="Teacher\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Teacher\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Teacher

WHERE

/\* %JoinFKPK(inserted,Teacher) \*/

inserted.Teacher\_Id = Teacher.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Homework because Teacher does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Course Homework on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_18", FK\_COLUMNS="Course\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Course\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Course

WHERE

/\* %JoinFKPK(inserted,Course) \*/

inserted.Course\_Id = Course.Course\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Homework because Course does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Inform ON Inform FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Inform \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Teacher Inform on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00027027", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_21", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Teacher

WHERE

/\* %JoinFKPK(deleted,Teacher," = "," AND") \*/

deleted.Stuff\_Id = Teacher.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Inform

WHERE

/\* %JoinFKPK(Inform,Teacher," = "," AND") \*/

Inform.Stuff\_Id = Teacher.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Inform because Teacher exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Administrator Inform on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Administrator"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_20", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Administrator

WHERE

/\* %JoinFKPK(deleted,Administrator," = "," AND") \*/

deleted.Stuff\_Id = Administrator.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Inform

WHERE

/\* %JoinFKPK(Inform,Administrator," = "," AND") \*/

Inform.Stuff\_Id = Administrator.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Inform because Administrator exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Inform ON Inform FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Inform \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insInfo\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Teacher Inform on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002fc8a", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_21", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Teacher

WHERE

/\* %JoinFKPK(inserted,Teacher) \*/

inserted.Stuff\_Id = Teacher.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

select @nullcnt = count(\*) from inserted where

inserted.Stuff\_Id IS NULL

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Inform because Teacher does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Administrator Inform on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Administrator"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_20", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Administrator

WHERE

/\* %JoinFKPK(inserted,Administrator) \*/

inserted.Stuff\_Id = Administrator.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

select @nullcnt = count(\*) from inserted where

inserted.Stuff\_Id IS NULL

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Inform because Administrator does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Student ON Student FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Student \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Student Student\_list on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0003122f", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="Student\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Student\_list

WHERE

/\* %JoinFKPK(Student\_list,deleted," = "," AND") \*/

Student\_list.Student\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Student because Student\_list exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Student Examination on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="Student\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Examination

WHERE

/\* %JoinFKPK(Examination,deleted," = "," AND") \*/

Examination.Student\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Student because Examination exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Student on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Student"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Stuff

WHERE

/\* %JoinFKPK(deleted,Stuff," = "," AND") \*/

deleted.Stuff\_Id = Stuff.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Student

WHERE

/\* %JoinFKPK(Student,Stuff," = "," AND") \*/

Student.Stuff\_Id = Stuff.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Student because Stuff exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Student ON Student FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Student \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insStuff\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Student Student\_list on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00038994", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="Student\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Student\_list

WHERE

/\* %JoinFKPK(Student\_list,deleted," = "," AND") \*/

Student\_list.Student\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Student because Student\_list exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Student Examination on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Examination"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_5", FK\_COLUMNS="Student\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Examination

WHERE

/\* %JoinFKPK(Examination,deleted," = "," AND") \*/

Examination.Student\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Student because Examination exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Student on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Student"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Stuff

WHERE

/\* %JoinFKPK(inserted,Stuff) \*/

inserted.Stuff\_Id = Stuff.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Student because Stuff does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Student\_list ON Student\_list FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Student\_list \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Course Student\_list on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00028ae7", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_23", FK\_COLUMNS="Course\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Course

WHERE

/\* %JoinFKPK(deleted,Course," = "," AND") \*/

deleted.Course\_Id = Course.Course\_Id AND

NOT EXISTS (

SELECT \* FROM Student\_list

WHERE

/\* %JoinFKPK(Student\_list,Course," = "," AND") \*/

Student\_list.Course\_Id = Course.Course\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Student\_list because Course exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Student Student\_list on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="Student\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Student

WHERE

/\* %JoinFKPK(deleted,Student," = "," AND") \*/

deleted.Student\_Id = Student.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Student\_list

WHERE

/\* %JoinFKPK(Student\_list,Student," = "," AND") \*/

Student\_list.Student\_Id = Student.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Student\_list because Student exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Student\_list ON Student\_list FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Student\_list \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insStudent\_Id varchar(20),

@insCourse\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Course Student\_list on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002ba98", PARENT\_OWNER="", PARENT\_TABLE="Course"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_23", FK\_COLUMNS="Course\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Course\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Course

WHERE

/\* %JoinFKPK(inserted,Course) \*/

inserted.Course\_Id = Course.Course\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Student\_list because Course does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Student Student\_list on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Student"

CHILD\_OWNER="", CHILD\_TABLE="Student\_list"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_22", FK\_COLUMNS="Student\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Student\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Student

WHERE

/\* %JoinFKPK(inserted,Student) \*/

inserted.Student\_Id = Student.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Student\_list because Student does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Stuff ON Stuff FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Stuff \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Stuff Administrator on parent delete cascade \*/

/\* ERWIN\_RELATION:CHECKSUM="00025923", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Administrator"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_3", FK\_COLUMNS="Stuff\_Id" \*/

DELETE Administrator

FROM Administrator,deleted

WHERE

/\* %JoinFKPK(Administrator,deleted," = "," AND") \*/

Administrator.Stuff\_Id = deleted.Stuff\_Id

/\* ERwin Builtin Trigger \*/

/\* Stuff Teacher on parent delete cascade \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Teacher"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_2", FK\_COLUMNS="Stuff\_Id" \*/

DELETE Teacher

FROM Teacher,deleted

WHERE

/\* %JoinFKPK(Teacher,deleted," = "," AND") \*/

Teacher.Stuff\_Id = deleted.Stuff\_Id

/\* ERwin Builtin Trigger \*/

/\* Stuff Student on parent delete cascade \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Student"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="Stuff\_Id" \*/

DELETE Student

FROM Student,deleted

WHERE

/\* %JoinFKPK(Student,deleted," = "," AND") \*/

Student.Stuff\_Id = deleted.Stuff\_Id

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Stuff ON Stuff FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Stuff \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insStuff\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Stuff Administrator on parent update cascade \*/

/\* ERWIN\_RELATION:CHECKSUM="0004251c", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Administrator"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_3", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF @numrows = 1

BEGIN

SELECT @insStuff\_Id = inserted.Stuff\_Id

FROM inserted

UPDATE Administrator

SET

/\* %JoinFKPK(Administrator,@ins," = ",",") \*/

Administrator.Stuff\_Id = @insStuff\_Id

FROM Administrator,inserted,deleted

WHERE

/\* %JoinFKPK(Administrator,deleted," = "," AND") \*/

Administrator.Stuff\_Id = deleted.Stuff\_Id

END

ELSE

BEGIN

SELECT @errno = 30006,

@errmsg = 'Cannot cascade Stuff update because more than one row has been affected.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Teacher on parent update cascade \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Teacher"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_2", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF @numrows = 1

BEGIN

SELECT @insStuff\_Id = inserted.Stuff\_Id

FROM inserted

UPDATE Teacher

SET

/\* %JoinFKPK(Teacher,@ins," = ",",") \*/

Teacher.Stuff\_Id = @insStuff\_Id

FROM Teacher,inserted,deleted

WHERE

/\* %JoinFKPK(Teacher,deleted," = "," AND") \*/

Teacher.Stuff\_Id = deleted.Stuff\_Id

END

ELSE

BEGIN

SELECT @errno = 30006,

@errmsg = 'Cannot cascade Stuff update because more than one row has been affected.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Student on parent update cascade \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Student"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_1", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF @numrows = 1

BEGIN

SELECT @insStuff\_Id = inserted.Stuff\_Id

FROM inserted

UPDATE Student

SET

/\* %JoinFKPK(Student,@ins," = ",",") \*/

Student.Stuff\_Id = @insStuff\_Id

FROM Student,inserted,deleted

WHERE

/\* %JoinFKPK(Student,deleted," = "," AND") \*/

Student.Stuff\_Id = deleted.Stuff\_Id

END

ELSE

BEGIN

SELECT @errno = 30006,

@errmsg = 'Cannot cascade Stuff update because more than one row has been affected.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tD\_Teacher ON Teacher FOR DELETE AS

/\* ERwin Builtin Trigger \*/

/\* DELETE trigger on Teacher \*/

BEGIN

DECLARE @errno int,

@severity int,

@state int,

@errmsg varchar(255)

/\* ERwin Builtin Trigger \*/

/\* Teacher Inform on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0003d925", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_21", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Inform

WHERE

/\* %JoinFKPK(Inform,deleted," = "," AND") \*/

Inform.Stuff\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Teacher because Inform exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Teacher Homework on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_19", FK\_COLUMNS="Teacher\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Homework

WHERE

/\* %JoinFKPK(Homework,deleted," = "," AND") \*/

Homework.Teacher\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Teacher because Homework exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Teacher Course on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Course"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_8", FK\_COLUMNS="Teacher\_Id" \*/

IF EXISTS (

SELECT \* FROM deleted,Course

WHERE

/\* %JoinFKPK(Course,deleted," = "," AND") \*/

Course.Teacher\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Teacher because Course exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Teacher on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Teacher"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_2", FK\_COLUMNS="Stuff\_Id" \*/

IF EXISTS (SELECT \* FROM deleted,Stuff

WHERE

/\* %JoinFKPK(deleted,Stuff," = "," AND") \*/

deleted.Stuff\_Id = Stuff.Stuff\_Id AND

NOT EXISTS (

SELECT \* FROM Teacher

WHERE

/\* %JoinFKPK(Teacher,Stuff," = "," AND") \*/

Teacher.Stuff\_Id = Stuff.Stuff\_Id

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Teacher because Stuff exists.'

GOTO error

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

CREATE TRIGGER tU\_Teacher ON Teacher FOR UPDATE AS

/\* ERwin Builtin Trigger \*/

/\* UPDATE trigger on Teacher \*/

BEGIN

DECLARE @numrows int,

@nullcnt int,

@validcnt int,

@insStuff\_Id varchar(20),

@errno int,

@severity int,

@state int,

@errmsg varchar(255)

SELECT @numrows = @@rowcount

/\* ERwin Builtin Trigger \*/

/\* Teacher Inform on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="000442ca", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Inform"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_21", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Inform

WHERE

/\* %JoinFKPK(Inform,deleted," = "," AND") \*/

Inform.Stuff\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Teacher because Inform exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Teacher Homework on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Homework"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_19", FK\_COLUMNS="Teacher\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Homework

WHERE

/\* %JoinFKPK(Homework,deleted," = "," AND") \*/

Homework.Teacher\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Teacher because Homework exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Teacher Course on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Teacher"

CHILD\_OWNER="", CHILD\_TABLE="Course"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_8", FK\_COLUMNS="Teacher\_Id" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Course

WHERE

/\* %JoinFKPK(Course,deleted," = "," AND") \*/

Course.Teacher\_Id = deleted.Stuff\_Id

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Teacher because Course exists.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

/\* Stuff Teacher on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Stuff"

CHILD\_OWNER="", CHILD\_TABLE="Teacher"

P2C\_VERB\_PHRASE="", C2P\_VERB\_PHRASE="",

FK\_CONSTRAINT="R\_2", FK\_COLUMNS="Stuff\_Id" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(Stuff\_Id)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Stuff

WHERE

/\* %JoinFKPK(inserted,Stuff) \*/

inserted.Stuff\_Id = Stuff.Stuff\_Id

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @numrows

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Teacher because Stuff does not exist.'

GOTO error

END

END

/\* ERwin Builtin Trigger \*/

RETURN

error:

RAISERROR (@errmsg, -- Message text.

@severity, -- Severity (0~25).

@state) -- State (0~255).

rollback transaction

END

Execution Successful

Schema Generation Complete

48 query succeeded.