ANALELE UNIVERSITĂȚII
"EFTIMIE MURGU" REȘIȚA
ANUL XVIII, NR. 3, 2011, ISSN 1453 - 7397

Cornelia Victoria Anghel

Code description of a C++ Object Oriented Program

This article presents the description of a C++ Object Oriented Application functional in an industrial Company. Changes of the existing application were made on three levels, the result being very advantageous for employees and management of the company.

***Keywords**: object oriented program, C++ language, code programs, sequences.*

1. Description of the new application

Implementing planned change was made on three levels:
- on database level: statement's SQL and stored procedures
- on application level: C++ code program
- on user interface level: HTML program.

The first phase of implementation was changes in the database. We created a new table called "d_cost_center_assignment" (allocation cost centers) to store each employee's cost centers. This table contains the fields:
- cost_center_assignment_id;
- employee_id;
- cost_center_id;
- function_group_id;
- start_date – for this cost center;
- end_date – final data activities for current center cost.

To run, this new table needs and few restrictions, namely:
- ID of the cost center assignment is a primary key – automatically generated number, unique for each record in this table;
- Cost center ID - a foreign key, that is to validate the entry in the database, this ID should be the primary key in existing table "m_cost_center";
- ID for a foreign key group functions, meaning to validate the entry in the database, this ID should be the primary key in existing table "m_function_group".

The last two restrictions are necessary in order not to introduce cost centers, groups of functions that exist in the database.

2. Implementation issues

To use the new code table in the database, we created a new C + + class that corresponds, I called Assignment DBCostCenter as naming rules followed in the application, using the code:

```
private static string[] TableFieldsDB = {
                              "cost_center_assignment_id",
                              "employee_id",
                              "cost_center_id",
                              "function_group_id",
                              "start_date",
                              "end_date"
                         };
internal DBCostCenterAssignment(ProjectCore pc) :
base(pc, "d_cost_center_assignment", TableFieldsDB)
```

Sequence code now calculates an employee cost center select from the table "cost center assignment" all records containing the column "employee_id" meaning, employee ID that you choose. The result is all the cost centers employee associations (in which he worked at a time). Only then looking in the record that the current cost center, comparing the start and end dates of activity center that costs the current date.

If the current system date is greater than "Start_date" and less than "End_date" means that the current cost center found.

C++ function that does this is:

```
public DataRow GetCurrentCostCenterAssignmentByEmployeeId(int employeeId)
{
DBCostCenterAssignment costCenters =
GetCostCenterAssignmentsByEmployeeId(employeeId);
DataRow currentCostCenter = null;
foreach (DataRow costCenterRow in costCenters.Rows)
{
DateTime startDate =
Convert.ToDateTime(costCenterRow[DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.TE_StartDate)]);
DateTime endDate =
Convert.ToDateTime(costCenterRow[DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.TE_EndDate)]);
            // is the current date between startDate and endDate?
DateTime today = System.DateTime.Today;
if ((startDate <= today) && (today <= endDate)) {
currentCostCenter = costCenterRow; }
 }
if (currentCostCenter==null)
{
string errorMessage = " ";
foreach (DataRow costCenterRow in costCenters.Rows)
 {
DateTime startDate =
Convert.ToDateTime(costCenterRow[DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.TE_StartDate)]);
DateTime endDate =
Convert.ToDateTime(costCenterRow[DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.TE_EndDate)]);
errorMessage += "startDate = " + startDate + " endDate =" + endDate +
System.Environment.NewLine;
 }
DBEmployee dbEmployee = new DBEmployee(PC);
dbEmployee.LoadByPK(employeeId);
string firstName = (string)dbEmployee.Rows[0][DBEmployee.
GetFieldName(DBEmployee.Fields.TE_Firstname)];
string lastName = (string)dbEmployee.Rows[0][DBEmployee.
GetFieldName(DBEmployee.Fields.TE_Lastname)];
throw new ArgumentException("Employee " + firstName + " "+ lastName+ " does
not have a current Cost Center assignment.Assignments are " + errorMessage);
}
return currentCostCenter;
}
```

Also, by comparing the start date and end time of the current cost center, you can find employees who meet the necessary conditions.
This is done by the code sequence:

```
public List<Int32> GetEmployeeIdsByCostCenter(int costCenterId)
{
DBCostCenterAssignment employeesWithCostCenter = new
DBCostCenterAssignment(PC);

employeesWithCostCenter.LoadByFK(DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.FK_CostCenterId), costCenterId);
List<Int32> employeesWithCurrentCostCenter = new List<Int32>();
foreach (DataRow costCenterRow in employeesWithCostCenter.Rows)
{
```

```
DateTime startDate = Convert. ToDateTime (costCenterRow
[ DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment. Fields. TE_StartDate)]);
DateTime endDate =
Convert.ToDateTime(costCenterRow[DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.TE_EndDate)]);
DateTime today = System.DateTime.Today;
if (startDate <= today && today <= endDate)
 {
int employeeID = Convert.ToInt32(costCenterRow[DBCostCenterAssignment.
GetFieldName(DBCostCenterAssignment.Fields.FK_EmployeeID)]);
employeesWithCurrentCostCenter.Add(employeeID);
  }
}
return employeesWithCurrentCostCenter; }
```

3. Aplications results and Conclusions

By comparing the start date and end time of the current cost center, you can find employees who meet the conditions normarii hours. The result obtained, with employee associations are all cost centers (where he worked at a time). We can concluded that the program codes described in detail in this article makes a number of facilities in the initial existing application.

References

[1] Anghel, C.V. *Programare orientata pe obiecte,* Ed. Eftimie Murgu, Resita, 2009.

[2] Jamsa K, Klander L. - *Totul despre C şi C++*, Editura Teora, Bucuresti, 2007.

[3] Alfons Kemper, André Eickler - „*Datenbanksysteme. Eine Einführung.*" Oldenbourg, München 2004.

Address:

- Lect. Dr. Eng. Cornelia Victoria Anghel, "Eftimie Murgu" University of Reşiţa, Computers Science Engineering Department, Traian Vuia Square, nr. 1-4, 320085, Reşiţa, c.anghel@uem.ro.