OBJECT-RELATIONAL MAPPING OF MULTIDIMENSIONAL DATA

Rocsana TONIS, Radu BUCEA-MANEA

Center for Studies and European Mobility

ABSTRACT. This paper presents the basic modeling and transformation rules for data warehouses in ranked object collections. Starting with these collections we've developed a service that allows dynamic allocation of objects during the querying. Furthermore, a basic API was created to interface the crud methods. accessing data sources through services is safe, effective and does not involve expansive hardware/software resources, accordingly to Cloud Computing paradigm.

Keywords: data warehouse, SMEs, Object-Relational Mapping, network business environment, performance economical-financial indicators, Cloud computing, CRUD.

ABSTRACT: În acest articol se prezintă principalele reguli de modelare şi transformare a depozitelor de date în colectii ierarhizate de obiecte. pe baza acestora am dezvoltat un serviciu care permite preluarea datelor, din orice tip de sursă şi alocarea dinamică în structuri de date pe durata interogării lor. Accesarea surselor de date prin intermediul serviciilor este sigură, eficientă şi nu implică resurse hardware/software costisitoare, conform paradigmei Cloud Computing.

Cuvinte cheie: depozit de date, imm, object - relational mapping, mediu virtual de afaceri, indicatori de performanță economico-financiari, cloud computing.

1. INTRODUCTION

It is considered a star shaped data warehouse, which includes data regarding Romanian SMEs and theirs activities, so that to be able to evaluate each other and to form a network businesses environment to ensure fair competition and business opportunities. Starting from this data warehouse it is developed a web service that allows access to data, using dynamic memory management.

The dimensions and fact table choice was done in accordance with the cooperation and competition requirements between Romanian SMEs, presenting relevant information on which we can measure the firms economic and financial performance, as to be able to fundament decisions in order to establish long-term, medium-term and short-term business strategies. Decisions will reflect the strategies coherency, materialized in firms' profitability, gearing and solvency. Historical data allows What IF analyses, scenarios and forecasts, which also influences the firm business strategy. The data warehouse highlights trends, directions and exceptions for long periods of time.

2. THE PLANNING AND ANALYSIS DATA WAREHOUSE PHASES

The planning and analysis data warehouse phases involve a thorough study of the field, searching data warehouse alternatives, constraints and documentation. We analyze the possible alternatives and decide what strategy to follow. Data warehouse is built to allow updating and handling data, without affecting pre-existing data, in a reasonable time. The process of creating a data warehouse is continuous; meaning that always may be added data silos (Data Marts, Data Repository etc.). [Rizescu, 2005]

We take into account many SMEs who decide to join in a network business environment, after Cloud computing philosophy. The network is based on a data warehouse containing historical and persistent data regarding the firms' business activity. The data warehouse is operated by a distributed decision support system with a BI interface, forming a multi-tier architecture.

Data consistency is achieved when following restrictions on the data fields' length such:

- firms' name is set to maximum 40 characters,
- activity categories have maximum 250 characters,
- e-mail addresses length is 50 characters,
- web addresses must have no more than 100 characters
- contact person have is 40 characters long.

The **validating rule** for the tables' fields says that measures' values must be positive numbers, exception being made for net and gross profit, which may be negative. For the foreign keys fields we modify the *Required* property so that they cannot be NULL.

3. DATAWAREHOUSE DESIGN

The data warehouse model includes the following items (Figure 1):

▪ *Activity measures* - we apply statistical functions on data in order to obtain the aggregated quantitative data.

▪ *Model dimensions* - are following tables: *Firme, Timp, CAEN,* with sub-dimensions *localitați, județe, regiuni*. Each key of the dimension tables is unique. Dimensions describe data in facts table (*Financiar*). The tables' keys represent the most relevant detailed level for this problem.

▪ *Facts table* - stores activity measures grouped by dimensions. Each key is a unique combination of external keys to dimension tables, including the *Timp* dimension. Facts table must achieve quantification of data described by dimensions.

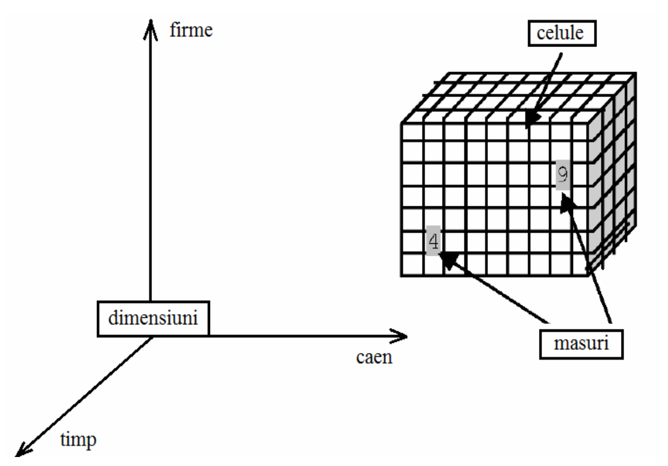


Fig. 1. Model dimensions.

The schema central table, called the facts table, is composed of external keys to dimension tables whose combination form the primary key of this table that uniquely identify data to be analyzed, ensuring keys consistency. The central table also includes attributes that characterizes an event, called action measure. For measures, the value of numeric attribute varies continuously. When is constant is called dimensional attribute.

The facts from the central table can be measured only when an event occurs and do not coincide with attributes dimensions. They are formed from activity measures and dimensions which characterize the context event.

Financiar facts table, consist of data warehouse measures: we choose *turnover, net profit, number of employees, assets, liabilities, capital*. Thus it is obtained a star model. The measures are additives on any of the three dimensions.

This table contains the external keys of the dimensions mentioned above such as: *Id_Timp, Id_CAEN*. The facts table contains additives measures for dimensions like: company, city, county, region, time and CAEN (*Județ, Regiune, Timp, CAEN)*. The SUM operator can be applied for attributes aggregation of values along all defined dimensions hierarchies; the ROLLUP and CUBE queries are significant for each dimension. On these measures may be called upon other aggregation operators (MAX, MIN, AVG). These measures concern accounting financial items such as: *CA-turnover, PN-net profit, nrsal-number of employees, Acirc-current assets, Aimob-fixed assets, stocks, claims, DT- total debt, ksoc-share capital, Kt-Total capital, VNT-total revenues, Cht-total expenditure*.

The facts table and the whole structure of the data warehouse allow comparative analysis between different years, certain towns and companies, or several criteria in the same time. It may also achieve a financial analysis on the situation of each enterprise: analysis of assets structure rates, cash-flow and other indicators. According to the relationship between the interest rate, the economic profitability rate and financial profitability rate it may consider whether a company can obtain credit, being a trustfull business partner. They can calculate solvency firms or make forecasts and What IF analysis.

Dimensions chosen to be applied for every record in the facts table are: time, firms, CAEN (*timp, firme, CAEN*) (Figure 2).

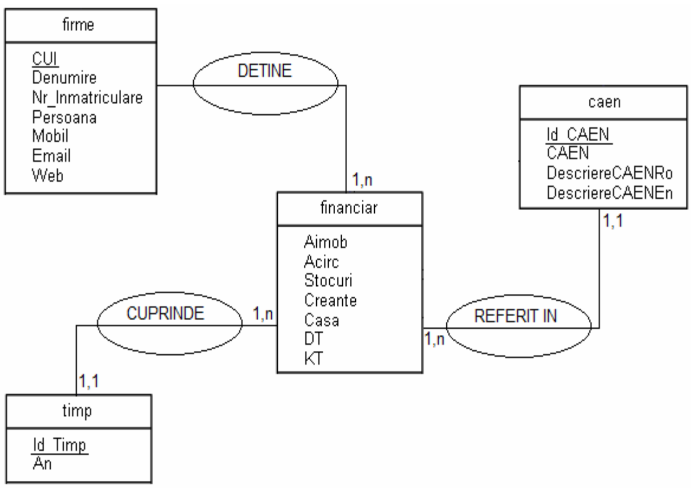


Fig. 2 . Logic data model.

Firms dimension has a primary key that uniquely refers it, *Idfirma*, and other attributes that characterizes it that may be subject to complex queries and analyses: Name, cui, web address, email, contact person. Companies may originate from different towns and counties, but the same company may not have headquarters in two different towns or counties. Any company is also related to sub-dimension *localitati*, through external key *idloc* from the table *localitati*.

Caen dimension has a primary key, *Id_CAEN,* that uniquely refers it and connects it with the *Financial* facts table. Other attributes of this dimension are: code of Romanian classification activities -Caen and Romanian and English description of the code.

Time dimension is present in any data warehouse. It's common to all data warehouses and may be populated in advance for a certain period of time. It has a primary key, *IdTimp*, that uniquely refers it and relates to the facts table. It consist of a single attribute, *year*, because the measures in our data warehouse are recorded only once a year.

The measures that will populate every record in the facts table are: turnover and net profit, assets and debts. The data warehouse allows different degrees of aggregation. Its structure is aimed to output SMEs business characteristics so they can collaborate and find out more details about the competing companies.

The data design is intuitive and can be easily updated both by adding new records to the presented dimensions, and by adding new dimensions without being affected the data warehouse functionality or integrity (Figure 3).

4. OBTAIN DATA WAREHOUSE OBJECT MODEL BY MAPPING AND CONVERTING TABLES IN THEIR CLASSES

Object-Relational Mapping (ORM) is a methodology that offers access to relational databases using object-orientated concepts. An ORM consists of a mapping language between objects and relational tables and an API for objects storage, querying and updating. There are several standards for commercial & Open source implementations, and some nonstandard products. [www1]

The ease of mapping depends on the implementation restrictions, on physical data level, on legacy to which relational and object model are bound and on complexity of these models. ORM can provide significant improvement in programming efficiency, in application quality and maintenance.

Beginning with the entity-association model, each entity may be mapped in a class type. A pivot table having fields made up from external keys to other tables can be decomposed in as many classes, each having one reference type member to the external key referred object. (figure 2)

5. DESIGNING THE OLAP PROGRAMMABLE INTERFACE FOR DATA SOURCE QUERYING

The usual method for accessing relational databases involves the using of APIs for sending SQL formatted queries to database server and to return the results back to the application. It is the programmer decision to use the results directly or to create data structures for loading the results into memory. Unfortunately, these structures cannot model the relationships established between tables or ensure the above mentioned requirements, so that most often the business logic is intermingled with database programming logic.

API allows programmers to meet all CRUD operations (creates, read, update and delete). Because the programmer does not directly accesses lines and columns from the database, an abridged notation describes its behavior.

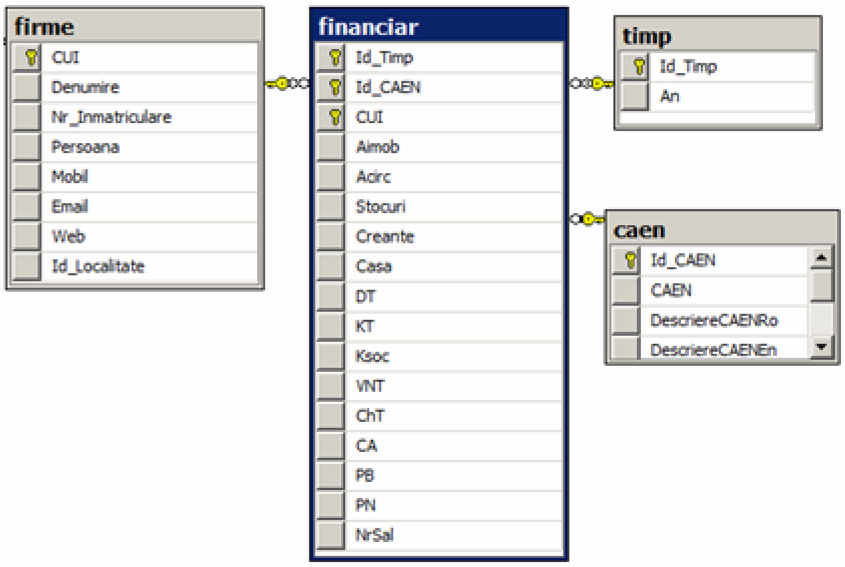


Fig. 3. Data warehouse schema.

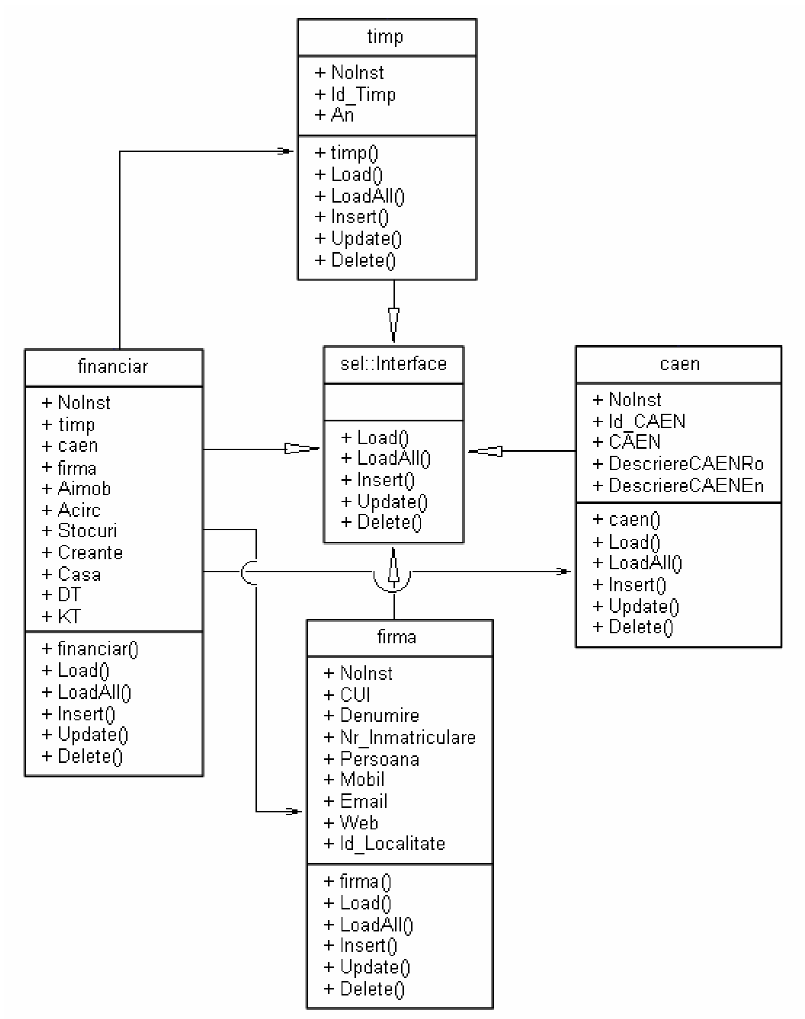


Fig. 4 . Classes diagram.

For example "create an object-instance to a specified class" is synonymous with "to create a line or more in the database that correspond to the instance of the class obtained after mapping. In SQL, the expression **frm.LoadAll()** is synonymous with SQL phrase "SELECT * FROM firme", and the expression **frm.Load("CU"," 53744")**which returns company instances that meet the restriction <CUI=53744>, maps the expression „SELECT * FROM firme WHERE CUI=53744". It's obvious the plus in efficiency obtained using the object syntax.

API includes methods for creating instances to classes obtained after mapping the tables, to retrieve an object-instance belonging to a class on the basis of its identifier, to find all its classes and sub-classes related instances, or to delete a domain(type) instance. Updates are used in the context of one transaction, extracting an object instance and using the class methods which is part of in order to modify the instance values.

Bellow it is shown the LoadAll() method applicable to all classes that inherits the CRUD interface and called passing through <T> generic type:

```
   public void LoadAll()
   {
    Type type = typeof(T);
    String header = "";
    String data = "";
    FieldInfo[] fields = 
type.GetFields(BindingFlags.Instance | 
BindingFlags.NonPublic | 
BindingFlags.Public);//1.
    for (int i = 0; i < noInst; i++)
    {
     foreach (var field in fields)
     {
      if (i == 0) header += ("|" + 
field.Name.Substring(1, 2) + (new String(' ', 
(field.GetValue(myclss[i]).ToString().Length))));
      data += ("| " + 
field.GetValue(myclss[i]));
     }
     data += "\n\r" + new String('-', 
header.Length) + "\n\r";
    }
    Console.WriteLine(header);
    Console.WriteLine(new String('=', 
header.Length));
    Console.WriteLine(data);//2.
   }
```

Steps are the following:

1) The class fields which type was passed through generic parameter <T> are obtained by reflection;

2) Starting with the NoInst static member that stores the instances number, the public properties names and values are iteratively read;

3) The obtained result is displayed in the console.

API is accessed within a service not exposing the interface implementation. After server starts and before it listens for possible client connection requests, the data sources are mapped and resulting class related object-instances are generated. In this respect, the next code sequence is executed:

```
............................................
Generic<timp> tmp = new Generic<timp>();
Generic<judete> jud = new Generic<judete>();
tmp.select(); jud.select();.................
```

The select() method opens the connection with a data source and invoke the insert() method that scans the source table using Cartesian coordinates and generate an indexed collection of the records related object-instances. According to the string marshaled by the client it is called the appropriate CRUD method. In the example presented in figure 5 there have been executed the following instructions:

```
tmp.Load("An","2006");
```

and

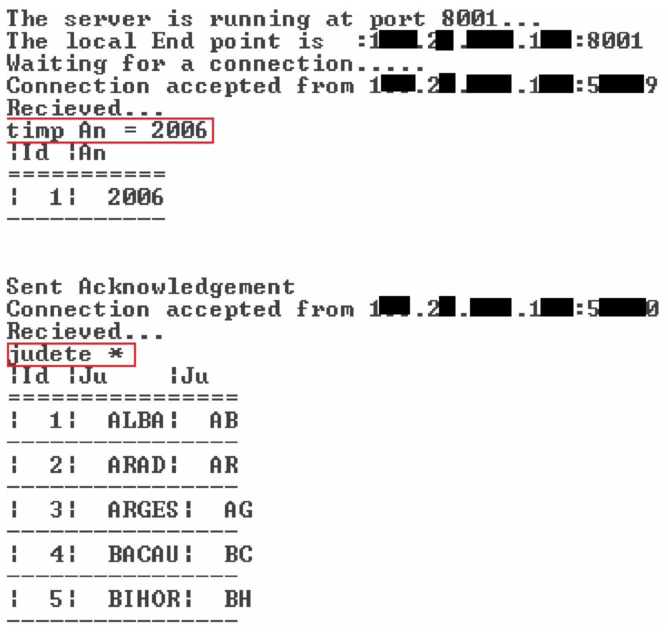
```
jud.LoadAll();
```

To achieve this example we used the Express Edition of the Microsoft Visual C# 2010 and library references to .NetFramework 4.0 *Collections, Reflection* and *Data*.

6. CONCLUSIONS

The relational to object transition involves splitting the business logic from the database programmable component. Using an API standard allows shorter development cycle, giving more time to evaluate mapping alternatives and techniques.

The most effective way for Romania SMEs to become profitable is to cooperate inside a dedicated business network environment with proper competition. In this respect we modeled at object-conceptual level a distributed data warehouse using mapping technology of tables into classes (ORM) and we developed a CRUD compatible interface for accessing data.



```
The server is running at port 8001...
The local End point is  :1[redacted].2[redacted].[redacted].1[redacted]:8001
Waiting for a connection.....
Connection accepted from 1[redacted].2[redacted].[redacted].1[redacted]:5[redacted]9
Recieved...
timp An  = 2006
!Id !An
===========
!  1!  2006
-----------


Sent Acknowledgement
Connection accepted from 1[redacted].2[redacted].[redacted].1[redacted]:5[redacted]0
Recieved...
judete *
!Id !Ju      !Ju
================
!  1!  ALBA!  AB
----------------
!  2!  ARAD!  AR
----------------
!  3!  ARGES!  AG
----------------
!  4!  BACAU!  BC
----------------
!  5!  BIHOR!  BH
----------------
```

Fig. 5. The querry output with timp An = 2006 and judete * expressions.

BIBLIOGRAPHY

[Rizescu, 2005] Rizescu, G. - *Multidimensional data analysis using OLAP Technology*, Revista Informatica Economică, vol. 33, nr. 1: 119, Bucuresti, 2005, ISSN 1453-1305 .

[www1] Bridging the Object-Relational Divide, available on-line at http://queue.acm.org/detail.cfm?id=1394139, accessed at 11.02.2012