UNIVERSAL DATA MODEL FOR SOLVING RESEARCH PROBLEMS

D.E. Yablokov, V.A. Saleev

Samara National Research University, Samara, Russia

Abstract. The article deals with the methodology of creation a universal data storage for applications oriented to the specialists who participate in scientific researches. It is assumed that the storage structure is not rigidly bound to any scientific domain, and it will be demanded in many projects related to the data accumulation, data analysis and processing. The proposed approach to the organization of the storage system is based on fundamental principles, concepts and technologies that used in the software design process. This process is based on the subject domain analysis and the detection criteria of commonality and variability which guarantee high abstraction level as one of the main tools for working with data.

Keywords: universal data model, storage structure, the abstraction level, data typification, internal classification, data categorization, external classification.

Citation: Yablokov DE, Saleev VA. Universal data model for solving research problems. CEUR Workshop Proceedings, 2016; 1638: 828-837. DOI: 10.18287/1613-0073-2016-1638-828-837

1 Introduction

The creation of software systems for scientific researches is a difficult task. As increasing their complexity, the appropriate software products building require more and more time, and the development costs grow exponentially. It also related to design process and implementation of the storage subsystems for applications, which are used in scientific activity, and also provide researchers of diverse, but at the same time actual and complete information. Many scientific laboratories or research centers could work with applications based on the proposed data model because of its universality and the high level of abstraction when describing the properties of objects and their classification. For the specialists making an experiment it is necessary to obtain and process the correct and complete information connected with research problems, but during of such researches is negatively influenced some factors requiring change the structures of data storage [1] with information about studied data domain or the solvable tasks. Among them: need of careful selection of the experimental data according to a storage format, use of varied initial information for complex research and, as a result, need for data interpretation or normalization for use their within ex-

periment. To overcome these disadvantages it is necessary to move to a new level of the software development of such class. It is necessary to create universal data model [2] for storing and processing various information. This data model could be a basis for building of different information systems. It is also necessary to create the environment for accumulation of the formalized data which can be applied to calculations, processes of diagnostics, prediction or identification [3] in any field of the experimental sciences.

2 Main idea

Objects of any type can be described in terms of some simplified data model for which the basic concepts are entities and attributes. Attributes are the predefined lexical units or descriptors needed to describe the basic semantic meaning of the key concepts in the data domain.

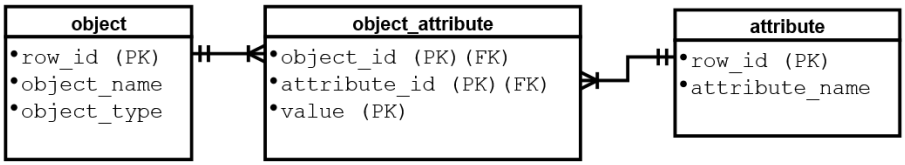

Fig. 1. Entities and attributes

When such concepts are defined, it is not difficult to move to a more specific description. It is possible to extend semantics of the selected conceptual model to the entities relationships and relationships attributes.

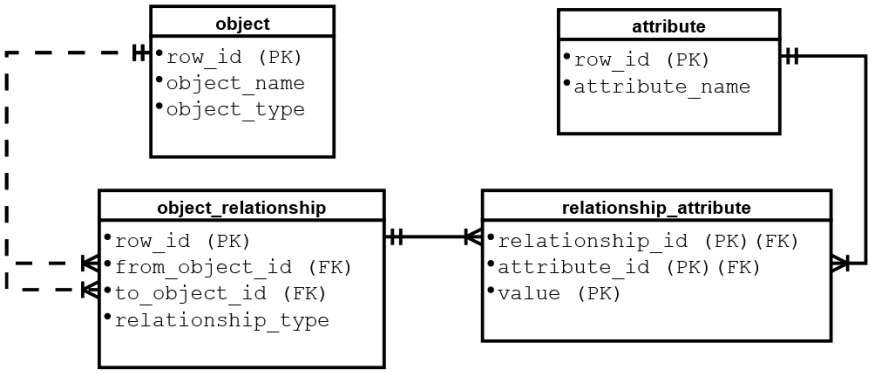

Fig. 2. Entities relationships and relationships attributes

This example is a very simplistic, but at the same time it can support most applications and provide adding data of any type without specific names of tables or fields

that are required when using a relational data model [1] in native form. When using the universal model it is possible to input information which structure is not defined in advance, and change of structural links like “entity–attribute”, “entity–entity” or “relationship–attribute” can be made in runtime.

3 Data classification

Classification is the most simple and at the same time, the most often solvable task of the universal model of data storage. The result of classification is the detection of the signs characterizing groups of the researched objects as classes to which is possible to refer new object. It is important to keep common concepts that defines the commonality and variability, and also other features already classified or again obtainable or analyzable information. In fact, classification is any system distribution of objects, phenomena or concepts by any essential signs selected for convenience of their representation and processing. Thus, the classified information can be provided as the set of the abstract or concrete entities ordered by some principle, which have similar classification criteria (one or more properties). It is necessary for determination of criteria of a commonality in the description, behavior or any other dimensions of source unstructured data. Usually classification tools are separate by a method of their impact on the classified element of information. For example, it is possible to organize internal classification, which can be made on the essential signs characterizing a commonality of objects, concepts or phenomena. The data domain, in this case, describes by the elementary units of data, related to some primitives, which are determined a priori. The main concepts are “meta-type”, “instance type”, “instance”, “hierarchy type” and “relationship attribute”.

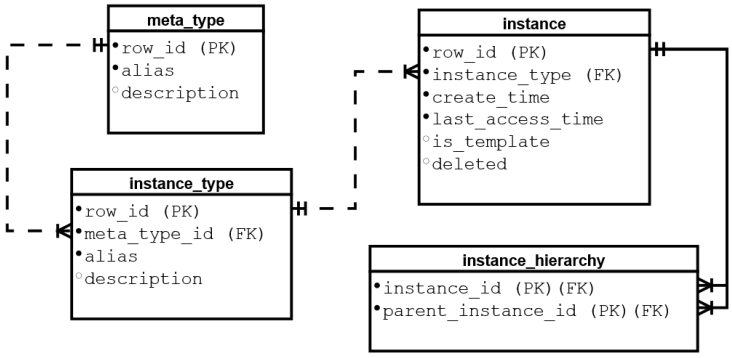

Fig. 3. Data typification and entity relationships

These concepts determine a set of tables that fixes the entity-relation structure in the database. A unique feature of such approach is the possibility of arrangement of unlimited number of objects over a limited set of concepts. For example, the concepts in the materials science are connected to definitions of graph theory that allows one to

describe crystal-chemical data [4] using graph abstractions. With interrelated concepts from chemistry and discrete mathematics, it is possible to classify the basic objects, such as atom or bond as well as more difficult objects, such as molecules, rings, ligands, nets, etc… Applying such classification for the applications of storage and processing crystallographic data, the information about the molecule as a set of atoms, can be represented as a spatial graph, i.e. non-empty set of vertices and the set of its two-element subsets – edges. Applying the concept of "is a" [5] allows to store information on objects with similar behavior and to consider their relations as hierarchy by the principle "from the general to the particular". Applying the concept "part of" [5] allows storing information on objects and their relations, using the principles of aggregation and composition. This allows considering their relationships in accordance with the principle "from the whole to the part". The concept "is a" allows storing information about similar behavior of objects, such as atoms and void centers or bonds and channels. Concept "part of" allows describing the data hierarchies "atom–structure fragment–net" or "atom-bond-ring-tile", etc. Also possible the artificial or additional classification by external sign and used for giving to a set of the researched elements of data necessary criterion of ordering. Such classification can be expressed through categorization when categories provide the necessary level of indirection. In this case, the commonality criteria definition for data elements is independent of their belonging to a specific class of data. The entity can be connected to one or more categories that makes possible in the analysis or decomposition obtaining the information about entity in terms of a set of the related categories. In this case the main concepts are "category type", "category", "subcategory" and "linked category".

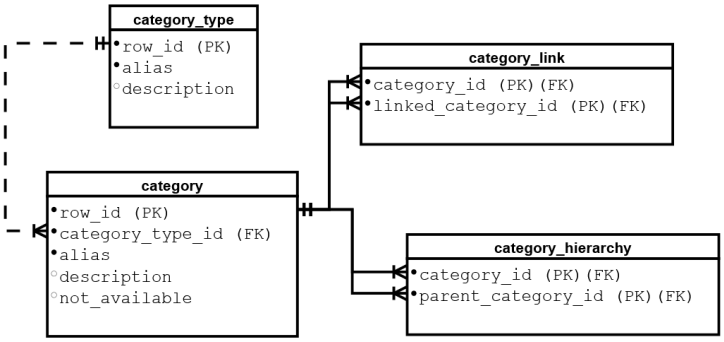

Fig. 4. Data categorization

Such classification can be used for periodic table of chemical elements where the mechanism of categorization can be applied to specifying to what group belongs chemical element, for example, of alkali metals, halogens, etc.

4 Scope of application

The storage structure is constructed by the principles of universal data model logically can be separated into some interdependent components. These include: classification

tools (typification and categorization), entity instances (with a possibility of assignment for instance type a specific property set), entity relationships (allowing to specify the relationship types and values of the appropriate attributes), properties palette (combining mechanisms to define primitive and composite properties).

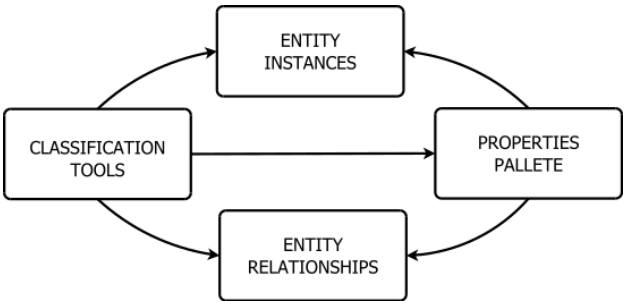

Fig. 5. Logical storage structure

This makes it possible to use the proposed universal data model in different fields of scientific research. These problems can be in the scope of theoretical and experimental chemistry connected with design and structural analysis of compounds and new materials [6], classification and systemized data representation about compound framework types [7], prediction chemical properties of the crystal structures [8]. In addition, in number of physics problems the proposed universal data model can be used to store various information about results of experiments with different level of details or to be a source of data for creation of mathematical models during researches. These include the problem of analysis and recognition of the nanoscale images [9, 10], spectral-spatial classification of hyperspectral images [11], increasing of sensor sensitivity [12], etc.

5 Implementation

The following are code samples from the DDL-script for the database creation using the universal model of data storage.

1. Code sample of creation of the meta-types table, as a set of the elementary primitives related to some internal base concepts of data domain, which are defined a priori.

```
CREATE TABLE IF NOT EXISTS meta_type
(
  row_id uuid NOT NULL DEFAULT uuid_generate_v4(),
  alias varchar(128) NOT NULL,
  description text,
  CONSTRAINT meta_type_pkey PRIMARY KEY(row_id),
  CONSTRAINT meta_type_uindex UNIQUE(alias)
);
```

2. Code sample of creation of the instance types table, as a list of the concepts defining the criterion of identity that distinguish the corresponding entity from other entities, which aren't corresponding to this criterion.

```
CREATE TABLE IF NOT EXISTS instance_type
(
  row_id uuid NOT NULL DEFAULT uuid_generate_v4(),
  meta_type_id uuid NOT NULL,
  alias varchar(128) NOT NULL,
  description text,
  CONSTRAINT instance_type_pkey PRIMARY KEY(row_id),
  CONSTRAINT instance_type_fkey
    FOREIGN KEY(meta_type_id)
      REFERENCES meta_type(row_id),
  CONSTRAINT instance_type_uindex  UNIQUE(alias)
);
```

3. Code sample of the table creation of category types, as a set of the elementary primitives related to some external concepts of data domain, which are determined a priori.

```
CREATE TABLE IF NOT EXISTS category_type
(
  row_id uuid NOT NULL DEFAULT uuid_generate_v4(),
  alias varchar(128) NOT NULL,
  description text,
  CONSTRAINT category_type_pkey PRIMARY KEY(row_id),
  CONSTRAINT category_type_uindex   UNIQUE(alias)
);
```

4. Code sample of creation of the categories table as a tool providing necessary flexibility through the additional level of indirection in case of determination of criteria of commonality for entities independent of their typification.

```
CREATE TABLE IF NOT EXISTS category
(
  row_id uuid NOT NULL DEFAULT uuid_generate_v4(),
  category_type_id uuid NOT NULL,
  alias varchar(128) NOT NULL,
  description text,
  not_available boolean,
  CONSTRAINT category_pkey PRIMARY KEY(row_id),
  CONSTRAINT category_fkey
    FOREIGN KEY(category_type_id)
      REFERENCES category_type(row_id),
  CONSTRAINT category_uindex UNIQUE(alias)
);
```

5. Code sample of creation of the categories hierarchy table for determining external criteria of commonality by the principle of similar behavior.

```
CREATE TABLE IF NOT EXISTS category_hierarchy
(
  category_id uuid NOT NULL,
    parent_category_id uuid NOT NULL,
  CONSTRAINT category_hierarchy_pkey
    PRIMARY KEY(category_id, parent_category_id),
  CONSTRAINT category_hierarchy_fkey_1
    FOREIGN KEY(category_id)
      REFERENCES category(row_id),
  CONSTRAINT category_hierarchy_fkey_1
    FOREIGN KEY(parent_category_id)
      REFERENCES category(row_id)
);
```

6. Code sample of creation the table of linked categories that contains data on the external of commonality by the principle from part to whole.

```
CREATE TABLE IF NOT EXISTS category_link
(
  category_id uuid NOT NULL,
  linked_category_id uuid NOT NULL,
  CONSTRAINT category_link_pkey
    PRIMARY  KEY(category_id, linked_category_id),
  CONSTRAINT category_link_fkey_1
    FOREIGN KEY(category_id)
      REFERENCES category(row_id),
  CONSTRAINT category_link_fkey_2
    FOREIGN KEY(linked_category_id)
      REFERENCES category(row_id)
);
```

7. Code sample of procedures for creation the data row in the table meta_type.

```
CREATE OR REPLACE FUNCTION create_meta_type_row
(
  meta_type_id uuid,
  alias varchar(128),
  description text = NULL
)
RETURNS VOID AS
$BODY$
BEGIN
  INSERT INTO meta_type(row_id, alias, description)
  VALUES(meta_type_id, $2, $3);
```

```
END;
$BODY$
LANGUAGE plpgsql;
```

8. Code sample of procedure for update the data row in the table instance_type.

```
CREATE OR REPLACE FUNCTION update_instance_type_row
(
  instance_type_id uuid,
  bit_mask integer,
  meta_type_id uuid = NULL,
  alias varchar(123) = NULL,
  description text = NULL
)
RETURNS void AS
$BODY$
BEGIN
  UPDATE instance_type t SET
  meta_type_id =
    CASE bit_mask & 1
      WHEN 1 THEN $3
      ELSE t.meta_type_id
    END,
  alias =
    CASE bit_mask & 2
      WHEN 2 THEN $4
      ELSE t.alias
    END,
  description =
    CASE bit_mask & 4
      WHEN 4 THEN $5
      ELSE t.description
    END
  WHERE t.row_id == instance_type_id;
END;
$BODY$
LANGUAGE plpgsql;
```

9. Code sample of data reading from the table category_type.

```
CREATE OR REPLACE FUNCTION read_category_type_rows
(
  category_type_ids VARIADIC UUID[]
)
RETURNS SETOF record AS
$BODY$
```

```
BEGIN
  RETURN QUERY SELECT t.* FROM category_type t
  WHERE t.row_id == ANY(category_type_ids);
END;
$BODY$
LANGUAGE plpgsql;
```

10. Code sample of procedure for deleting data row from the table category_link.

```
CREATE OR REPLACE FUNCTION delete_catgory_link_rows
(
  category_id uuid,
  use_linked boolean = NULL
)
RETURNS void AS
$BODY$
BEGIN
  DELETE FROM category_link t
  WHERE $1 =
    CASE COALESCE($2, false)
      WHEN true THEN t.linked_category_id
      ELSE t.category_id
    END;
END;
$BODY$
LANGUAGE plpgsql;
```

6 Conclusions

The problem of creation of universal data storage becomes significant in the case of implementation of unstructured data collection using an object-oriented approach [13] for information representation and a relational database. During the creation of system of such type, the most appropriate approach is a deductive method. It provides decomposition of complex concepts into elementary units of data with mathematically and semantic expected behavior. Units of information, in accordance with some predefined concepts, describe the data domain. The universal data model allows storing information of any type and complexity by using elementary primitives to describe the hierarchical links and data relationships. Using such primitives is a prerequisite for the development of an effective and reliable method of the description and extraction of information necessary for researches. The article provides explanations on proposed solutions and methodologies based on fundamental concepts of programming [14] and data analysis. The main advantage of the proposed approach is the possibility of applying a universal model for any type of information, and also a possibility of determination of system of the concepts that provide the foundation for the creation a domain-specific language [15]. Closest to the context of the specific scien-

tific knowledge, such language can represent correlations between structure of the data domain and how it expressed through the criteria of commonality and variability.

Acknowledgements

This work was supported by the Russian government (Grant 14.B25.31.0005).

References

1. Ambler S, Sadalage P. Refactoring Databases: Evolutionary Database Design. Addison-Wesley, 2006; 384 p.
2. Silverstone L. The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises. Wiley, 2001; 542 p.
3. Rajan, K. Informatics for Materials Science and Engineering. Butterworth-Heinemann, 2013; 610 p.
4. Blatov VA, Shevchenko AP, Proserpio DM. Applied Topological Analysis of Crystal Structures with the Program Package ToposPro. Cryst. Growth Des, 2014; 14 (7): 3576–3586.
5. Booch G. Object-Oriented Analysis and Design with Applications. Addison-Wesley, 2007; 720 p.
6. O'Keeffe M, Peskov MA, Ramsden SJ, Yaghi OM. The Reticular Chemistry Structure Resource (RCSR) Database of, and Symbols for, Crystal Nets. Acc. Chem. Res., 2008; 41(12): 1782–1789.
7. Baerlocher C, McCusker LB, Olson DH. Atlas of Zeolite Framework Types Sixth revised edition. London: Elsevier, 2007; 398 p.
8. Blatov VA, Proserpio DM. Periodic-Graph Approaches in Crystal Structure Prediction. Modern Methods of Crystal Structure Prediction, ed. Oganov AR. Weinheim: Wiley-VCH, 2011: 1–28.
9. Soifer VA, Kupriyanov AV. Analysis and recognition of the nanoscale images: Conventional approach and novel problem statement. Computer Optics, 2011; 35(2): 136–144.
10. Kupriyanov AV. Texture analysis and identification of the crystal lattice type upon the nanoscale images. Computer Optics, 2011; 35(2): 151–157.
11. Zimichev EA, Kazanskiy NL, Serafimovich PG. Spectral-spatial classification with k-means++ particional clustering. Computer Optics, 2014; 38(2): 281–286.
12. Egorov AV, Kazanskiy NL, Serafimovich PG. Using coupled photonic crystal cavities for increasing of sensor sensitivity. Computer Optics, 2015; 39(2): 158–162. DOI: 10.18287/0134-2452-2015-39-2-158-162.
13. Fowler M. Patterns of enterprise application architecture. Addison-Wesley, 2003; 560 p.
14. Yablokov DE. Programming paradigms. XIV MNPK “Nauchnoe obozrenie fiziko-tekhnicheskikh nauk v XXI veke”, Moscow, Prospero, 2015; 14(2): 94-98. [In Russian]
15. Fowler M. Domain-Specific Languages. Addison-Wesley, 2010; 640 p.