Encyclopedia of Business Analytics and Optimization, J. Wang (Editor), IGI-Global, 2014, 502-511.

Concept-Oriented Model

Alexandr Savinov, *SAP AG, Germany*

INTRODUCTION

In recent years, we observe a significant expansion of data analytics driven by such factors as very low cost of data acquisition, storage and processing. To make business decisions, users have to carry out complex analysis which is fed by data. Data is fuel and prerequisite for any kind of analysis starting from simple charts and ending with complex data mining processes. However, data analysis is getting more and more difficult with the explosion of data volume and the variety of data sources which differ in their underlying data models, representation formats and conventions.

In order to represent and analyze data from different data sources a unified model is needed. One such model, called the concept-oriented model (COM), is described in this article. COM (2009a) is a general purpose unified data model aimed at describing many existing views and patterns of thoughts currently used in data modeling. As a unified model, its main goal is to significantly decrease differences and incongruities between various approaches to data modeling such as transactional, analytical (Savinov, 2011b), multidimensional (Savinov, 2005a), object-oriented (Savinov, 2011a), conceptual and semantic (Savinov, 2012c). The motivation behind COM and its practical benefits are similar to those for the Business Intelligence Semantic Model (BISM) (Russo, Ferrari, & Webb, 2012) introduced in Microsoft SQL Server 2012. Like BISM, COM also aims at creating a unified model for all user experiences by significantly simplifying typical analysis tasks and facilitating self-service analytics.

The creation of this model was motivated by the following problems which are difficult to solve within traditional approaches:

Domain-specific identities. Identification in most existing models is based on two approaches: primitive references like oids or surrogates, and identifying properties like primary keys. These approaches do not provide a mechanism for defining strong *domain-specific* identities with arbitrary structure, and the focus in existing models is made on describing entities. The goal of COM is to make identities and entities equal elements of the model both having some structure.

Hierarchical address spaces. Most models focus on describing properties of elements without providing containment relation as a basic construct. A typical solution consists in modeling spaces and containment like any other domain-specific relationship. The goal of COM is to model containment as a core notion of the model by assuming that elements cannot exist outside of any space, domain or context.

Multidimensionality. There exist numerous approaches to multidimensional modeling which are intended for analytical processing. The problem is that analytical and transactional models rely on different assumptions and techniques. The goal of COM in this context is to rethink dimensions as a first-class construct of the data model which plays a primary role for describing both transactional and analytical aspects.

Semantics. Existing models represent data semantics separately from data itself. Normally semantics is supported at the level of conceptual model or using some semantic model. Logical data models have rather limited possibilities for representing semantic relationships and it is a strong limiting factor for the effective use of data. The goal of COM here is to make semantics integral part of the logical data model so that the database can be directly used for reasoning about data.

To solve these problems, COM introduces three main structural principles which distinguish it from other data models and naturally account for various typical data modeling issues:

Duality principle answers the question *how* elements exist by assuming that any data element is composed of two parts: one identity and one entity (also called reference and object, respectively). Thus an element is defined as an identity-entity couple.

Inclusion principle answers the question *where* elements exist by postulating that any element is *included* in some domain (also called scope or context). Thus all elements exist in a hierarchy.

Order principle answers the question what an element *is*, that is, how it is defined and what is its meaning by assuming that all elements are partially ordered. Thus any element has a number of greater and lesser elements which define its semantics.

Syntactically, COM is described by the concept-oriented query language (COQL) (Savinov, 2011b). This language reflects the principles of COM by introducing a novel data modeling construct, called *concept* (hence the name of the approach), and two relations among concepts, *inclusion* and *partial order*. Concepts generalize conventional classes and are used for describing domain-specific identities. Inclusion relation generalized inheritance and is used for describing hierarchical address spaces. Partial order relation among concepts is intended to represent data semantics and is used for complex analytical tasks and reasoning about data.

Formally, COM relies on the notion of nested partially ordered sets (nested posets). Nested poset is a novel formal construct that can be viewed either as a nested set with partial order relation established on its elements or as a conventional poset where elements can themselves be posets. An element of a nested poset is defined as a couple consisting of one identity tuple and one entity tuple.

BACKGROUND

As a unified model, COM is able to describe many wide-spread mechanisms and data modeling patterns existing in other model.

The support for hierarchies in COM makes it very similar to the classical hierarchical data model (HDM) (Tsichritzis & Lochovsky, 1976) and therefore COM can be viewed as a reincarnation of the hierarchical model using new principles. In both models data exist within a hierarchy where any element has a unique position. The main difference of COM is that it proposes to use domain-specific identities as hierarchical addresses. Another difference is that inclusion relation is simultaneously used to model inheritance as it is done in object data models.

Hierarchies in COM make it similar to object data models (Dittrich, 1986; Bancilhon, 1996). The difference is that inclusion relation in COM generalizes inheritance and is used to model both containment and inheritance (reuse). The assumption that inheritance is a particular case of containment is one of the major contributions of the concept-oriented approach. From this point of view, containment in the hierarchical model and inheritance in object models are particular cases of inclusion relation in COM. The treatment of inclusion in COM is very similar to how inheritance is implemented in prototype-based programming (Lieberman, 1986; Stein, 1987; Chambers, Ungar, Chang, & Hölzle, 1991) because in both approaches parent elements are shared parts of children.

The structure of dimensions in COM can be used for navigational purposes without using joins. This makes it similar to the functional data model (FDM) (Sibley & Kerschberg, 1977; Shipman, 1981; Gray, King, & Kerschberg, 1999; Gray, Kerschberg, King, & Poulovassilis, 2004) if COM dimensions are represented as functions. The difference is that COM uses only single-valued dimensions while multi-valued functions are modeled by means of reversed dimensions and de-projection operator.

COM is similar to the relational model (Codd, 1970) because both of them are tuple-based and set-based models. The main difference of COM is that it introduces two types of tuples: identity tuples and entity tuples. COM makes a clear statement that identities should be at least as important as entities and introduces special means for modeling them using concepts. COM identities can be thought of as surrogates with arbitrary domain-specific structure. Another difference is that instead of using conventional sets, COM considered nested partially ordered sets.

The idea of using partial order for data management was also developed by Raymond (1996) where “a partial order database is simply a partial order”. However, this approach assumes that partial order underlies type hierarchies while COM proposes to use a separate inclusion relation for modeling types. It also focuses on manipulating different partial orders and relies more on formal logic while COM focuses on manipulating elements within one nested poset with strong focus on dimensional modeling, constraint propagation and inference.

The use of partial order in COM makes it similar to multidimensional models (Pedersen & Jensen, 2001; Li & Wang, 1996; Agrawal, Gupta, & Sarawagi, 1997; Gyssens & Lakshmanan, 1997) widely used in OLAP and data warehousing. The main difference is that COM does not use special roles of dimensions, cubes and facts at the level of the model by assuming that these terms describes specific analysis scenarios rather than data itself.

There has been a large body of research on semantic data models (Hull & King, 1987; Peckham & Maryanski, 1988) but most of them propose a conceptual model which needs to be mapped to some logical model. COM proposes a new approach to representing and manipulating data semantics where different abstractions of conventional semantic models such as aggregation, generalization and classification are expressed in terms of a partially ordered set.

PRINCIPLES AND STRUCTURE OF THE MODEL

Duality principle

Like in most other models, the smallest data unit in COM is a *primitive value* also called atomic, system or platform-specific value. More complex elements are produced by means of *tuples* which are treated in their common mathematical sense as a combination of primitive values or other tuples. Tuple members are distinguished by their position which is referred to as a *dimension* (also called attribute, column, property, slot, variable or characteristic in other models). A tuple consisting of *n* members is called *n*-tuple where *n* is called *arity*. If $e = \langle x{:}\,a, y{:}\,b, z{:}\,c, \ldots \rangle$ is a tuple then *x*, *y* and *z* are dimensions while *a*, *b* and *c* are members of this tuple which are other tuples or primitive values.

A unique feature of COM which distinguishes it from all other data models is that it introduces two kinds of tuples: *identity tuples* and *entity tuples*. Identity tuples are values which can be passed and stored only by-copy. Like any value, identity tuples cannot be represented indirectly via any other value such as address or reference. In contrast, the only way to represent and access an entity tuple is via some other value, that is, entity tuples can only be passed and stored by-reference. Identity tuples are supposed to represent transient data while entity tuples represent persistent data which can be shared among many other data elements.

What is even more important is that identity and entity tuples are not modeled and manipulated separately. Instead, according to the *duality principle*, a data element in COM is defined as a *couple* of one identity tuple and one entity tuple. Within such a couple, identity tuple is used as an address, location or reference for the corresponding entity tuple. In other words, COM assumes that any element must have some data that can be shared (entity) and also must have some data that is used to indirectly access this shared data (identity). If one of these parts is empty then we get two particular cases which can be described in existing data models.

Identity-entity couples are modeled via a novel data modeling construct, called *concept*, which is defined as a couple of two classes: one identity class and one entity class. Concept instances are identity-entity couples where identity is supposed to represent the corresponding entity. For example, a book could be described as the following concept:

```
CONCEPT Books
  IDENTITY
    CHAR(10) isbn
  ENTITY
    CHAR(256) title
```

Instances of this concept are represented by custom identities containing an ISBN number rather than using some primitive references like surrogates.

The main purpose of the duality principle is to explain *how* elements exist by assuming that identities manifest the fact of the entity existence. This means that any entity must have some identity defined directly or indirectly (via inheritance) and an entity without identity is supposed to be non-existing. An advantage of the duality principle is that it is possible to freely vary between by-value and by-reference representations. It is not possible in other models where an element is either an object (entity) represented by a primitive reference or it is a value (identity). As a consequence, COM unites two branches of data modeling: domain modeling and relation modeling. Instead of modeling separately domains (sets of values) and relations (sets of tuples), COM considers only one type of domains where elements are identity-entity couples. The duality principle is informally analogous to complex numbers in mathematics which are also composed of two constituents manipulated as one whole. The duality principles can also be thought of as computer memory where both memory cells and their addresses may have arbitrary domain-specific structure.

Inclusion principle

COM assumes that elements cannot exist *in vacuo* outside of any space or scope and the main purpose of inclusion relation is to answer the question *where* elements exist. Any address or reference is always relative and is specified with respect to some other (parent) address which provides a reference point for all internal relative addresses. *Inclusion principle* postulates that any element exists within some other element with respect to which it is identified. Thus all elements exist in a hierarchy where child elements are identified with respect to their parent element. Child elements in this inclusion hierarchy are also referred to as *sub-elements* and parent elements are referred to as *super-elements*.

Inclusion relation in COM is implemented via tuple extension where one tuple member is assigned a special meaning and is referred to as base or *super-dimension*. All other constituents of the tuple taken as one value are referred to as *extension*. The super-dimension has a special name 'super' and/or is separated from other members by the bar symbol: $e = \langle super: s | x: a, y: b, \dots \rangle$. Here s is a super-element while a and b are members of the extension. It can be also written simpler as $e = \langle s | a \rangle$ where s is a base and a is an extension tuple (so a is said to extend s).

If identity tuple is empty then we get the classical object-oriented extension operator applied to objects. If entity tuple is empty then it works as value extension. An important difference from the classical extension is that one element may have many different sub-elements. With the help of extension any element can be represented as a sequence of identities starting from the root element and ending with the identity of the last extension. Such a sequence of identities is referred to as *complex identity*. It is analogous to postal addresses with the main difference that any identity has an associated entity.

At the level of concepts, COM introduces a new relation, called inclusion, so that any concept declares a parent concept it is included in. The parent concept is referred to as a *super-concept* while a child concept is referred to as a *sub-concept*. For example, if bank accounts are identified with respect to their bank (account without a bank is meaningless) then concept `Account` has to be included in concept `Bank`:

```
CONCEPT Account IN Bank
  IDENTITY
    CHAR(10) accNo
  ENTITY
    DOUBLE balance
```

Extension can be also semantically interpreted in terms of containment relation by assuming that extended element is included in its base element: if $e = \langle a | b \rangle$ then $e \subset_1 a$ where $\subset_1$ denotes immediate inclusion relation among elements. All elements are represented as a nested set $(R, \subset)$ where R is a set of elements and $\subset$ is inclusion relation. An important consequence of this approach is that to be a member in a set means to extend this set. Semantically, to be a member of a set means to be more specific than this set and to inherit from the set. In other words, IS-A relation in COM is a particular case of IS-IN relation.

Order principle and semantic interpretations

The representation of elements by means of tuples is used to introduce the third structure into the model in the form of the strict partial order relation ‘$<$’ between elements. If $a < b$ then *a* is said to be a *lesser* element and *b* is referred to as a *greater* element. All elements of the model are represented as a partially ordered set (poset) $(R, <)$. According to the *order principle*, a tuple is less than any of its members: if $e = \langle \dots, a, \dots \rangle$ is a tuple then $e <_1 a$ where $<_1$ means ‘immediately less than’ relation (‘less than’ of rank 1). This rule holds for both identity and entity tuples. Essentially, this principle means that tuple members are immediately greater than the tuple they are in. The greatest elements of the poset are primitive values which are then used to compose all other elements down to the least elements which are not used in any other element of the model. Note that according to the duality principle, only identities are included in other tuples by-value which means that entity parts of tuples can be included and shared among many other elements. Since identities are used as references, this principle also means that a referencing element is always less than the referenced elements (references represent greater elements). As a consequence, the reference structure is used to represent partial order relation among elements.

Concepts are also partially ordered by assuming that dimension (field) types specify greater concepts. For example, if a book has a publisher then this means that the `Publisher` concept is greater than the `Book` concept:

```
CONCEPT Publisher ... // Greater

CONCEPT Book // Lesser concept
  ENTITY
    // Type is a greater concept
    Publisher publisher
```

Nested partially ordered sets

Elements in COM are structured by means of two relations simultaneously: inclusion ‘$\subset$’ and partial order ‘$<$’. A connection between these two relations is provided via *type constraint*: $e \subset b \Rightarrow e.x \subseteq b.x$. This constraint means that if element *e* is included in element *b* then any greater element of *e* referenced by dimension *x* is included in the greater element of *b* referenced by this same dimension *x*. In terms of tuple extension, this means that extended dimensions cannot store arbitrary values but rather can only extend the value returned by the super-dimension. If $e = \langle a | b \rangle$ then $b.x$ must be an extension of $a.x$.

Structure consisting of these two binary relations is referred to as a *nested partially ordered set*. More specifically, a nested partially ordered set $(R, \subset, <)$ is a set *R* with strict inclusion ‘$\subset$’ and strict partial order ‘$<$’ relations on its elements which are connected by type constraint. This structure can be produced from a partially ordered set if we assume that an element can itself be a partially ordered set. In this case it can be visualized as a nested Euler diagram (Fig. 1, left). Alternatively, it can be produced from a nested set if we assume that its elements are partially ordered and then it can be visualized as a tree the elements of which are partially ordered (Fig. 1, right). For example, element *a* consists of elements *c* and *d* which are its extensions. Simultaneously, element *a* has one lesser element *b*.

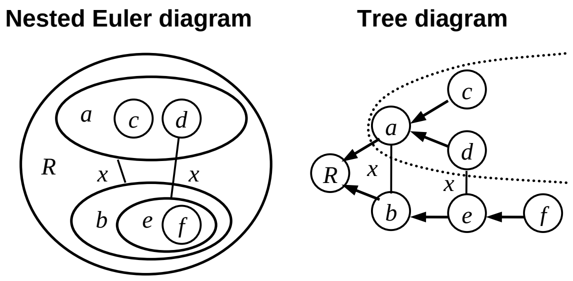

Figure 1. Nested partially ordered set.

A *concept-oriented database* is formally defined as a nested partially ordered set $(R, \subset, <)$ where elements are identity-entity couples and strict inclusion and strict partial order relations are induced by extension and references, respectively.

Operations

As a set-theoretic model, COM supports the well known Cartesian product operation which is applied to a number of source sets $E_1, E_2, \ldots, E_n$ and returns a set consisting of all combinations of the source elements:

$$E_1 \times \ldots \times E_n = \{e = \langle e_1, \ldots, e_n \rangle | e_i \in E_i, i = 1, \ldots, n\}$$

In COM, this classical operation has the following properties. First, elements of the product are identity-entity couples and their identities are defined as combinations of the source identities. A consequence of such a definition is that arity of the product is equal to the number of the source sets *n*. In the relational model, arity of the product is equal to the sum of the arities of the source sets because all the source attributes are included in the result tuple. The second difference a consequence of the order principle: elements of the product are less than the source elements they are composed of. Thus new sets in COM are always connected with the existing sets they were built from.

COM also provides operations for set-based navigation which rely on the dimension structure for retrieving existing elements given constraints on other elements. Two operations of *projection* and *de-projection* allow for moving up along dimensions and down against dimensions, respectively. Projection, denoted by right arrow, returns all elements of a greater set *D* referenced from the lesser set *E* along dimension *d*:

$$E \rightarrow d \rightarrow D = \{a \in D | \exists e \in E : e <_d a\}$$

Note also that an element can be included in projection only one time even if it has many lesser elements.

De-projection is defined as an operation that returns elements of a lesser set *F* which reference the elements of the given greater set *E* along dimension *f*:

$$E \leftarrow f \leftarrow F = \{b \in F | \exists e \in E : b <_f e\}$$

De-projection can be also defined as a set of elements which are projected to this set along the specified dimension.

If a dimension is not specified then it can be found automatically. This mechanism is used for constraint propagation and inference (Savinov, 2006a, 2012b). Projection and de-projection operations can also be used to access data along inclusion hierarchy. In this case, 'super' is used as a dimension name. Projection and de-projection can be combined so that data is accessed using a zig-zag path leading from the source set to the destination set. This approach is a basis for set-based navigation mechanism (Savinov, 2005b).

FUTURE RESEARCH DIRECTIONS

Concepts and inclusion are used also in a novel approach to programming, called concept-oriented programming (COP) (Savinov, 2012a, 2009b, 2008). Therefore in future COM will be more tightly integrated with this novel programming paradigm with the goal of unification and decreasing differences between data modeling and programming. In particular, it is important to unify language constructs used in COP and COQL.

Another future goal consists in further developing the formal basis of COM. In particular, it is interesting to describe this model only in terms of sets and functions which will make it closer to the functional paradigm. From the point of view of database systems, it is important to demonstrate benefits of COM in modeling column-stores and in-memory databases with the main applications in data analysis.

CONCLUSION

In this article we have described a novel unified data model with the following main distinguishing features:

Identity-entity couples and concepts instead of classes. COM assumes that data elements are identity-entity couples where identities are used to represent entities. This creates a nice ying-yang style of symmetry between two sides of one model. To model these couples, COM proposes to use a novel construct, concept, which generalizes classes. Data modeling is then broken into two orthogonal branches: identity modeling and entity modeling.

Inclusion instead of inheritance. Inclusion relation introduced in COM permits objects to exist in a hierarchy where they are identified by hierarchical addresses. Data modeling is then reduced to describing such a hierarchical address space. Importantly, inheritance is considered a particular case of inclusion so that inclusion is used to model two classical relations simultaneously: IS-A and IS-IN.

Partial order instead of graph. COM proposes to partially order all data elements by assuming that references represent greater elements and concept dimension types represent greater concepts. Data modeling is then reduced to ordering elements so that other properties and mechanisms are derived from this relation.

The main advantage of any unified data modeling approach is the ability to describe many existing data models using a small number of data modeling constructs. In other words, the more data modeling patterns can be represented (generality) and the fewer major notions are used (simplicity), the better the unified model is. In this sense, COM has clear benefits because it can be applied to a wide range of tasks but at the same time it is still a quite simple approach which uses only a few basic notions to derive many important data modeling and analysis techniques. In particular, this model can be used to solve the following general problems:

- Uniting analytical and transactional modeling. Currently these are two branches in data modeling. As a consequence, one and the same data is modeled and managed two times: in a transactional database and in a data warehouse. COM allows us to eliminate this transactional-analytical impedance mismatch by simplifying system design.
- Uniting conceptual and logical modeling. Currently conceptual modeling and logical modeling are two separate layers in data modeling. This leads to the necessity to have some kind of translation or mapping procedure which result in additional complexity.
- Uniting graph-based, deductive and object-based models. These models emphasize certain aspects which are important for data modeling and data management. Unfortunately they are not unified and therefore have to be described using different approaches and constructs. COM generalizes and unites these approaches by simplifying many data management tasks.

As a general purpose unified model, COM can be applied to quite different problems like database architectures, query languages, schema matching, information retrieval and many others.

Taking into account its simplicity and generality, COM seems rather perspective direction for further research and development in the area of data modeling and analytics.

REFERENCES

Agrawal, R., Gupta, A., & Sarawagi, S. (1997). Modeling multidimensional databases. In *13th International Conference on Data Engineering (ICDE'97)* (pp. 232-243).

Bancilhon, F. (1996). Object databases. *ACM Computing Surveys (CSUR), 28*(1), 137-140.

Chambers, C., Ungar, D., Chang, B., & Hölzle, U. (1991). Parents are shared parts of objects: Inheritance and encapsulation in Self. *Lisp and Symbolic Computation, 4*(3), 207-222.

Codd, E.F. (1970). A relational model of data for large shared data banks. *Communications of the ACM, 13*(6), 377-387.

Dittrich, K.R. (1986). Object-oriented database systems: The notions and the issues. In *Proc. Intl. Workshop on Object-Oriented Database Systems* (pp. 2-4).

Gray, P.M.D., King, P.J.H., & Kerschberg, L. (eds.) (1999). Functional approach to intelligent information systems. *J. of Intelligent Information Systems, 12*, 107-111.

Gray, P.M.D., Kerschberg, L., King, P., & Poulovassilis, A. (eds.) (2004). *The functional approach to data management: Modeling, analyzing, and integrating heterogeneous data*. Heidelberg, Germany: Springer.

Gyssens, M., & Lakshmanan, L.V.S. (1997). A foundation for multi-dimensional databases. In *Proc. 23rd Intl. Conf. on Very Large Data Bases (VLDB'97)* (pp. 106-115).

Hull, R., & King, R. (1987). Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys (CSUR), 19*(3), 201-260.

Li, C., & Wang, X.S. (1996). A data model for supporting on-line analytical processing. In *Proc. Conference on Information and Knowledge Management* (pp. 81-88). Baltimore, MD.

Lieberman, H. (1986). Using prototypical objects to implement shared behavior in object-oriented systems. In *Proc. OOPSLA'86* (pp. 214-223).

Peckham, J., & Maryanski, F. (1988). Semantic data models. *ACM Computing Surveys (CSUR), 20*(3), 153-189.

Pedersen T.B., & Jensen C.S. (2001). Multidimensional database technology. *IEEE Computers, 34*(12), 40-46.

Raymond, D. (1996). *Partial order databases*. Ph.D. Thesis, University of Waterloo, Canada.

Russo, M., Ferrari, A., & Webb, C. (2012). *Microsoft SQL Server 2012 Analysis Services: The BISM tabular model*. Microsoft Press.

Savinov, A. (2012a). Concept-oriented programming: Classes and inheritance revisited. In *Proc. 7th International Conference on Software Paradigm Trends (ICSOFT 2012)* (pp. 381-387).

Savinov, A. (2012b). Inference in hierarchical multidimensional space. In *Proc. International Conference on Data Technologies and Applications (DATA 2012)* (pp. 70-76).

Savinov, A. (2012c). Concept-oriented model: Classes, hierarchies and references revisited. *Journal of Emerging Trends in Computing and Information Sciences, 3*(4), 456-470.

Savinov, A. (2011a). Concept-oriented model: Extending objects with identity, hierarchies and semantics. *Computer Science Journal of Moldova (CSJM), 19*(3), 254-287.

Savinov, A. (2011b). Concept-oriented query language for data modeling and analysis. In L. Yan & Z. Ma (Eds), *Advanced database query systems: Techniques, applications and technologies* (pp. 85-101). IGI Global.

Savinov, A. (2009a). Concept-oriented model. In V.E. Ferraggine, J.H. Doorn, & L.C. Rivero (Eds.), *Handbook of research on innovations in database technologies and applications: Current and future trends* (2nd ed., pp. 171-180). IGI Global.

Savinov, A. (2009b). Concept-oriented programming. In Mehdi Khosrow-Pour (Ed.), *Encyclopedia of information science and technology* (2nd ed., pp. 672-680), IGI Global.

Savinov, A. (2008). Concepts and concept-oriented programming. *Journal of Object Technology (JOT), 7*(3), 91-106.

Savinov, A. (2006a). Query by constraint propagation in the concept-oriented data model. *Computer Science Journal of Moldova (CSJM), 14*(2), 219-238.

Savinov, A. (2005a). Hierarchical multidimensional modelling in the concept-oriented data model. *Proc. 3rd international conference on Concept Lattices and Their Applications (CLA'05)* (pp. 123-134).

Savinov, A. (2005b). Logical navigation in the concept-oriented data model. *Journal of Conceptual Modeling, 36*. Retrieved December 5, 2012, from http://conceptoriented.org/savinov/publicat/jcm_05.pdf

Shipman, D.W. (1981). The functional data model and the data language DAPLEX. *ACM Transactions on Database Systems (TODS), 6*(1), 140-173.

Sibley, E.H., & Kerschberg, L. (1977). Data architecture and data model considerations. In *Proceedings of the AFIPS Joint Computer Conferences* (pp. 85-96).

Stein, L.A. (1987). Delegation is inheritance. In *Proc. OOPSLA'87* (pp. 138-146).

Tsichritzis, D.C., & Lochovsky, F.H. (1976). Hierarchical data-base management: A survey. *ACM Computing Surveys (CSUR), 8*(1), 105-123.

KEY TERMS & DEFINITIONS

Concept is a data modeling and programming construct which is defined as a couple of one identity class and one entity class. Concept instances are identity-entity couples. Concepts generalize conventional classes and are used instead of them for declaring types of elements. A concept with the empty identity class is equivalent to a conventional class. A concept with the empty entity class describes values.

Concept-Oriented Model (COM) is a unified approach to data modeling which generalizes several major views on data: relational, multidimensional, object-oriented, conceptual and semantic. It is based on three structural principles: duality, inclusion and partial order.

Duality Principle postulates that any element is an identity-entity couple where identity is a value which uniquely represents the associated entity. This principle is a unique distinguishing feature of the concept-oriented model (COM) because most other models manipulate tuples while COM manipulates couples of tuples. In particular, this principle unifies relation modeling and domain modeling.

Entity Tuple is one of two constituents of an element in the concept-oriented model. It is a tuple which is represented and passed by-reference where the reference is an associated identity tuple which is not part of this entity. Any entity must have a non-empty identity which however can be inherited from the parent element in the inclusion hierarchy. The structure of entities is described by the entity class of concepts.

Identity Tuple is one of two constituents of an element in the concept-oriented model. It is a tuple which is passed by-copy and uniquely represents an entity. An identity without an associated entity (the associated entity is empty) is a value. The structure of identities is described by the identity class of concepts.

Inclusion Principle postulates that all elements exist in an inclusion hierarchy where child elements extend their parent. A consequence of this principle is that any element in the hierarchy is uniquely represented by a sequence of identities starting from the root and ending with this element identity. This sequence is called complex identity and is analogous to conventional postal addresses. Inclusion relation among concepts generalizes classical inheritance. Inclusion also unites inheritance, general-specific (IS-A) and containment (IS-IN) relations.

Nested Partially Ordered Set is a formal basis of the concept-oriented model. It is defined as a set with strict inclusion and strict partial order relations on its elements which are connected by type constraint. It can be viewed as a nested set where elements are partially ordered or as a partially ordered set where elements are also partially ordered sets.

Order Principle assumes that all elements are partially ordered where lesser elements reference their greater elements. Concepts are also partially ordered assuming that lesser concepts specify their greater concepts as field types. This principle plays an important role in COM because it provides several conceptual interpretations: multidimensional, entity-relationship, containment, object role.