

A Data Integration System Managed by Users

Vladimir Ovchinnikov

(ovch@lipetsk.ru)

Abstract. A data integration system is a system providing a general interface to row of distributed and heterogeneous information systems. The article proposes an architecture of a data integration system based on Semantically Complete Model with the following distinctions from existent integration approaches: a) the global schema is represented as a flat concept space being transparent for users; b) creation of GUI view points, including query writing, can be fulfilled without involving IT people; c) new local schemas and view points upon them are shared automatically among all users as soon as created, administration is not necessary; d) the semantic browsing feature is supported automatically, without any programming; e) view points functionality is based on functionality of visual components (grids, charts, trees, graphs, reports, and others), which can be supplied for the system by independent software producers; f) master-detail dependencies within view points are defined in the declarative way.

Keywords: heterogeneous data integration, conceptual model, Semantically Complete Model, Concept Space

1 Introduction

Increase of Internet data sources quantity motivates many researchers to find solutions for data integration task. A data integration system is a system providing a general data access interface to a row of distributed and heterogeneous information systems. A separate local information system has a wrapper which abstract it as a local schema. All local schemas are generalized to a global schema. A global schema is used for interacting with a data integration system. The system defines a set of mappings of local schemas to the global schema, reformulates global schema queries to local schemas queries, executes the latter, and combines results to the result of the initial query.

All integration approaches can be differentiated according to models used for local and global schemas representation. Using ontologies for this purpose is developing most rapidly now [35]. But other models are used for data integration as well [1, 10, 16, 24, 32].

Semantic equivalence of local (source) schemas and a global (target) one is the next differentiation dimension. This dimension implies two schema mapping approaches: a) a target schema has no independent semantics and gathers semantics of

all source schemas; b) a target schema and a source schema have their own semantics and are mapped partially in the general case [11].

Another differentiation dimension is method used for mapping schemas: local schemas can be created as views on a global schema (Local-as-View) [26], a global schema can be created as view on local schemas (Global-as-View) [7], and there are different intermediate approaches [14, 36].

All existent integration ways require administration of schema mapping during local schemas creation or modification. This work should be done by a specially trained IT specialist.

This paper is devoted to architecture, engineering and use of a data integration system being managed by users in many respects. This system can be classified in the following way: a) it is based on Semantically Complete Model [28, 29, 30, 31] as a model for local schemas representation; b) it does not store a global schema explicitly, a global schema is calculated from local schemas automatically; c) it is semantically equivalent, in the sense that a global schema has no independent semantics.

The paper has the following structure. Section 2 issues problems of current heterogeneous information systems engineering and use. Section 3 enumerates features of a conceptual model meant as means for solution of the issued problems. Section 4 shows architecture of the data integration system having no enumerated problems. Section 5 enumerates features of the system. Discussion of the system and related work is given in section 6. Section 7 enumerates open issues to be solved to implement the proposed architecture in practice. Main contributions of the paper are enumerated in section 8.

2 Problem Statement

Present techniques of information systems engineering, including heterogeneous and distributed ones, require significant effort of highly skilled IT specialists. Improvement of the techniques in the direction of slight effort decrease can produce tremendous practical effect in engineering quality and speed. What problems exist on this way?

Problem 1. To create and use a conceptual model a person should be a skilled IT specialist.

Present conceptual modeling approaches can be used by specialists only because of a) great quantity of special constructions with special meaning: inheritance, objectification, power types, attribute-entity difference, etc.; b) to work with a conceptual model one should operate proper relation names explicitly. This problem becomes more evident if we use a conceptual model not only as a knowledge representation mean, but as a data access interface for a distributed and heterogeneous information system.

Conceptualization principle [17] implies that a conceptual model's construction set should be sufficient to describe any application domain, and it should not contain constructions encouraging creation of models having system implementation details.

How to keep true the conceptualization principle, and simultaneously to simplify conceptual modeling to make possible modeling and model use by end users?

Objective 1. Creation and use conceptual models by persons not being IT specialists.

Considerable risk of information systems creation lies in communication of a customer and an analyst. If a customer becomes able to create a system conceptual model, ample quantity of design mistakes can be avoided. In this case, an IT person receives already formalized description of an application domain from a customer's hands directly, there can be no communication mistakes.

Problem 2. To create a new view point (user interface form) for an information system a person should be a skilled IT specialist and should be familiar with a current project implementation.

Present methods of user view points creation require programming since they are based on implementation-specific structures implicitly reflected to application domain concepts. A developer deal with the way of data organization, and not with application domain concepts directly. To create a useful view point one does two things: he/she logically maps data structures to concepts known for users, and organizes visualization of concepts and connections. Since the mapping is implicit, in a developer's mind, view points are based on internal system structures, and so a developer should be familiar with current project implementation, and users' needs as well.

Objective 2. Creation of new view points on distributed and heterogeneous data by a user without involving IT specialists.

If we present a data access interface to a distributed and heterogeneous system as a conceptual model being transparent for users, then: a) a view point developer is not to extract data meaning manually since it is encapsulated to the system, b) creation of a new view point does not require programming since it may be based on the semantic data access interface. The most complex thing a designer has to do is preparation of conceptual queries. After that, he/she uses visualization components for the queries, and arranges layout of components over a form.

So, if a data access interface is simple enough and no programming is required, then elaboration of view points can be done by users themselves without involving IT specialists. To be freed of view points programming, the palette and functionality of components used for query visualization should be sufficient to cover all users' needs.

Problem 3. To integrate separate information systems to one, it is necessary to fulfill additional complex project by skilled IT specialists that are familiar with the current implementation of the base information systems.

Present integration approaches do not allow to impose the integration responsibility on a local information system provider only. There should be one who knows implementation details of all base information systems and creates, manually or semi-automatically, mappings of local schemas to a global one [36]. When a data integration system grows, mappings administration effort grows exponentially.

Newly published local schema can not be integrated automatically today. And present integration approaches permit structure and instantiation conflicts. Solving these conflicts is nontrivial task and requires deep familiarity with information systems implementation. Resultant integration system remains separate with respect

to other data integration systems. So, we have several nonintegrated mediators based on different principles.

Objective 3. Automatic integration of all local information systems as soon as an appropriate local schema is created without necessity to construct explicit mappings of local schemas to a global one and any other additional effort.

If integration of information systems does not require explicit mappings constructing, this integration can be done automatically as soon as a new local schema is published. But how to preserve semantic integrity not using mappings? For this we share elements (concepts) between local schemas (see below).

Problem 4. Programming of rich navigation and filtering mechanisms requires significant effort from IT specialists, especially in distributed and heterogeneous environment.

Present approaches to user interface creation for information systems require separate programming for each way of transition from one view point to others. This problem becomes more complex when we consider a data integration system since quantity of possible transitions increases significantly.

Filtering mechanisms are also programmed for each case separately today. As a result, each system, and perhaps each view point in a system, has its own data filtering mechanism, that complicates users' work significantly and increases programming effort.

Objective 4. Automatic support of semantic navigation and filtering mechanisms in distributed and heterogeneous environment.

If data semantics is encapsulated to a system, then the system can construct a list of possible semantic transitions from one view point to others automatically. If the system is a data integration system, this way of navigation can be applied within all distributed and heterogeneous information systems and between them. Also, data semantics encapsulation allows to create a general mechanism of information filtering, not required programming.

Problem 5. Evolution of structures storing requires reconstruction of all applications that use changing structures.

Information system applications are based on structures tightly connected with the way of data storing today. Changes in structures storing result in changes of all applications used these structures. How to decrease the level of dependence upon structures storing?

Objective 5. Preservation of data access interface when storing structures are changed (without modification of their semantics, for example, to optimize storing and queries efficiency).

Present DBMS use two levels of storing abstraction: the physical and logical levels. But the latter does not ensure the maximal independence from implementation details. If, additionally, we use the conceptual level as a data access interface, degree of independence will increase since reflection of a logical model to a conceptual model can be flexibly adjusted to save data access interface.

3 Semantically Complete Model Features

Semantically Complete Model was introduced as a mean of conceptual modeling that allows for IT specialists and users to work with a model considering entities (object types) only [29, 30]. The model's name follows from the property that the model's relations describe interconnection of underlying entities in semantically complete way [28]. The model was introduced as self-dependent conceptual modeling approach, and it can be considered as restriction of Object-Role Model [6, 19]. Object-Role Modelling (ORM) uses the following terminology [6]:

- a) an object type represents an elementary concept;
- b) a fact type is a relation based on a multi-set of object types;
- c) an inheritance type is a biunique relation on two object types;
- d) an objectified fact type is an object type representing a fact type (instances of the object type are instances of the appropriate fact type);
- e) and others (objectified model, power type, sequence type).

SCM has the following differences with respect to ORM [28, 29]:

Difference 1. A relation (connection in terms of SCM) is based on a set of entities (concepts).

Note that a relation is not based on a multi-set of entities, but on a set of entities. Also there is no distinction between an inheritance type and a fact type in the model, both are special cases of a relation.

Difference 2. A relation is based on an entity set that is unique within a SCM model.

In other words, there are no two relations based on the same entity set within a SCM model. So, each entity set represents an identifier of an appropriate relation.

Difference 3. There are no alternative relations within a SCM model.

Two relations are considered to be alternative if one relation is based on an entity set that is proper subset of another relation's entity set. This is the key difference which realizes the semantic completeness property.

Difference 4. SCM has no such complex for non-IT people aspects as objectification, power types, sequence types, distinction between fact types and inheritance types, distinction between attributes and entities (which exists within ER approach [8, 9]).

A SCM model represents only flat concept space, with connections upon concepts. All other constructions are special cases of these two. It is very simple and may be used by users after little training. Further we use terms "concept" and "connection" in the context of SCM as synonyms for "entity" ("object type") and "relation" ("fact type" or "inheritance type").

Semantically Complete Model has the following features distinguished it from the row of other conceptual modeling techniques.

Feature 1. Connections are complete (i.e. they carry complete information about interconnection of concepts), elementary, implicit, and have no names.

As each connection is identified by a concept set (see difference 2), there is no necessity to assign proper names to connections. And so, connections can be used implicitly by enumerating appropriate concepts.

Since connections are complete (see difference 3), one should define a connection in the way that it can not be projected on several connections, join of which results in

the initial connection. Indeed, if a connection is decomposable, it contains several potential connections; if we do not define these connections explicitly, we restrict expandability of a model. So, it is recommended to define connections in the elementary (indecomposable) way.

Feature 2. SCM allows to think about model considering concepts only.

Since connections have no proper names and can be referred using concepts, handling of concepts is sufficient to work with a SCM model.

Feature 3. Model development and discussion are performed in the same terms.

Concepts are recommended to name so that names sound clear for application domain experts. For this full unabbreviated word combinations should be used. As a result, model creator may use these names in development process as well as in model discussion process.

Feature 4. Inheritance is a special case of a connection, and is not an independent construction.

An inheritance can be represented as a biunique connection based on two concepts. Generalization and specialization [6] are special cases of inheritance merely, differences are in additional constraints imposed on them. A general way of concept connecting simplifies use of SCM by people who are not specialists in IT. Separate navigation mechanisms for each relation type is not necessary, and so model querying is generalized maximally.

Feature 5. SCM has less quantity of equivalent representations of one application domain than other models have.

Less quantity of expressive means in comparison with other conceptual models is not only reason of this feature. The main reason is semantic completeness property: a model does not have alternative relations. As a result, a modeler is not to switch between equivalent, from the semantic view point, application domain representations, and he/she abstracts from implementation details to a greater extent.

Feature 6. If we draw an analogy with Relational Model, SCM is in the 5th normal form.

If we consider a connection as a relation of Relational Model, then a set of SCM model connections are in 5NF [12, 15] according to the feature 1: connections are not decomposable in the projection-join way.

Though, if the recommendation of the feature 1 is not fulfilled, a model can be in non-5NF (and even non-3NF). But normalization rules are recommended to be applied for SCM models as for RM for greater expandability and update anomaly absence. Further, we consider SCM models as models satisfying 5NF.

Feature 7. Model state can be represented as one set of connection instances.

A connection instance can be represented as a set of concept instances since a connection is not based on a concept multi-set, but a set (see difference 1). Therefore, a connection instance should not refer to a connection explicitly: if we know concept instance set, we know concept set; using this concept set, we can find an appropriate connection since there is only one connection for each concept set (see difference 2). So, if we have a set of instance connections of a SCM model, we can reconstruct a state of each connection of the model.

This feature is valid if we do not use null values, in other words, if a connection instance has one concept instance for each concept of the connection. In principle, the last statement can be violated, but it is not recommended since

- a) connections are elementary (see feature 1);
- b) tables of a relational schema contain null values because of their semantic complexity, null values arise only when combining several semantic connections to one table.

So, null values are not recommended to use in connections. Further we consider SCM models as having no null values.

Feature 8. Query formulation is performed without connections' proper names.

Semantically Complete Query Language (SCQL) is a query language based on SCM and used its properties [30]. An expression of SCQL represents a tree of relation calculation. Each step calculates a target relation on basis of source relations; connections or calculable relations can be used as source relations equally. An expression as a whole has one target relation representing a query result. Syntax of SCQL does not require use of connections' proper names [29, 30]. It follows from the difference 2: each connection is identified by a concept set; and therefore to refer to a connection it is sufficient to enumerate appropriate concepts.

Feature 9. A general criterion of source relations matching holds implicitly when executing composition, union, or minus; it is not necessary to indicate the criterion in an explicit form.

One step of SCQL expression calculation is named as representation. A representation is a way of target relation calculation on basis of source relations. SCQL has several representations being near to generally known ones: projection, union, minus, etc. But there are some specific representations, one of which is composition.

Composition is a superposition of source relations. The superposition is performed not according to concept coincidence, but according to role concept coincidence. A role concept represents a concept extended by an integer role index showing a role in which the concept is used in the expression. A relation participating in a SCQL expression are considered to be based on a set (not multi-set) of role concepts, and not concepts as such. It allows not to indicate join criterion explicitly, but to use role concept coincidence for this purpose.

Another specific representation is positioning. It is applied implicitly or explicitly to each connection participating in an expression: a) explicitly if a role index not equal to 1 should be assigned to a concept; b) otherwise, the role index 1 is assigned to a concept implicitly.

Union and minus have the following distinctions from generally known ones:

- a) source relations matching is performed automatically according to role concept coincidence;
- b) source relations can have different signatures, in other words, can be based on different sets of role concepts because of the first distinction;
- c) source relations can have different arities because of the first distinction.

So, all representations of SCQL use the general criterion of source relations matching applied implicitly. This criterion is role concepts coincidence.

Feature 10. A composition of binary connections is represented as a chain of concepts.

If connections being composed are binary, SCQL allow to represent their composition as a chain of concepts. In this case, each linked concept pair of a chain

represents a connection since a concept pair is sufficient to identify an appropriate connection.

Feature 11. A selection is a special case of a composition.

There is no necessity to introduce selection as a self-dependent representation of SCQL since a composition of source relations, one of which is based on a logical condition, is equivalent to a selection. This feature is the consequence of the feature 9: explicit join criterion is not necessary.

Feature 12. A request of interconnection of indirectly connected concepts, being included in a default closure, is represented as an enumeration of concepts.

If a subset of SCM model connections has no cycles, there is only one path from one concept to another concept within the subset. Each SCM model can have only one default closure that represents an explicitly defined connections subset without cycles. Within a default closure, a connection of indirectly connected concepts can be requested as if they are connected directly: by their simple enumeration [29].

4 Heterogeneous Concept Space Architecture

A data integration system using SCM as data access interface is named as Concept Space since it provides flat concept space interface without complex constructions. Concept Space consists of the following interacting parts: SCM Data Servers, SCM Integration Servers, Shared Object Servers, and User Machines. Each of these parts contains several interacting components in turn. Further, we enumerate these components as “<part name>: <component name>” and describe their roles. The overview of Concept Structure architecture is shown on the figure 1: thick lines outline parts (computers), thin lines outline components within parts, arrows show data flows between components.

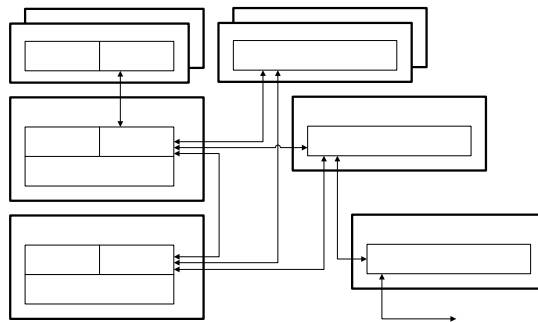


Fig. 1. Concept Space Architecture

SCM Data Server: Storage. A storage fulfills base functions of data storing and query execution.

Data of Concept Space are stored physically using one of known technology within a data server, and are accessed logically with SCM. Relational DBMS is considered as a main technology for Concept Space. Therefore SQL is considered as a main

query language to which SCQL is translated for data querying and modification. Further the row of storage types can be extended with other storing technologies. Concept Space is heterogeneous and can operate data servers of different storage types simultaneously and transparently.

SCM interface can be based not only on a relational schema as such, but on stored procedures not modifying data too. It allows to implement connections not only as stored associations of concept instances, but as calculable associations as well.

SCM Data Server: SCM Adaptor. SCM Adaptor implements SCM-based access interface to data stored in the storage.

A relational schema (or another structure) of a data server is mapped to SCM to organize SCM data access interface. This mapping is created by an IT person who is responsible for publishing an information system to Concept Space. The mapping can be generated automatically if a relational schema is generated fully from a SCM model created previously. No restrictions are imposed on a relational schema to access it through SCM.

SCM Adaptor gets SCQL queries or modifications, translates them to an appropriate query and modification language (for example, SQL) using the mapping, executes queries, and then transforms the result to SCQL form.

Also an adaptor manages a local SCM schema (concepts and connections). It allows to create a new concept (connection) or delete an existent one. Concepts from other servers can be shared during connection creation. The global SCM schema represents simple union of all local ones as they are already integrated since creation.

SCM Data Server: SCM Integrator. SCM Integrator implements an entry point to Concept Space by execution of distributed SCQL queries and modifications.

SCM Integrator holds information about distribution of connections among data servers. This information is used for execution of distributed queries and modifications. If a user requests a connection which is unknown for an integrator, then the integrator requests SCM Integration Server for this information (see below).

A user communicates with only one integrator; it is not necessary to switch between different data servers. When an integrator gets a query or a modification, it communicates with other integrators and a local adaptor to execute the operator. In addition, integrators manage distributed and local transactions.

Changing of the global SCM schema are fulfilled through SCM Integrator too. Concepts are created on a data server of the integrator by default, but a user can indicate a data server for concept creation explicitly. Concept identifier is a combination of a data server identifier, which stores the concept, and an internal for the server sequential concept identifier. This way of identification allows to share concepts between data servers simply and provides high level of data servers independence.

Connections are identified with concept identifier set (see difference 2, feature 1) and are created on data servers according to connection creation policy (see below). This policy unambiguously determines what data server should be used for connection creation; a user cannot manage connection distribution manually since Concept Space must guarantee that there are no alternative connections in the global SCM schema as a whole.

Created connections can be implemented (for example, as tables) within an appropriate data server automatically or they can be mapped manually from existent

structures (tables, for example) by a developer. As soon as a connection is mapped, manually or automatically, it can be queried or modified by users and used for view point creation.

Also each integrator allows for users publishing shared objects and controlling access to the objects. There are tree base types of shared objects: components, view points, stored queries (filters). All objects are used by Semantic Browser (see below) for dynamic interface maintaining.

Objects are stored in Shared Object Servers and are managed by SCM Object Managers. There is one register of Shared Object Servers on specialized servers which provides general way of object requesting. Also, identifiers of associated view points and queries (filters) are stored with each concept. The identifiers contain codes of appropriate object servers, so requesting of associated view points and stored queries (filters) by a concept can be performed directly, that is obligatory for effective Semantic Browser implementation. To provide this mechanism there is an appropriate interchange protocol between object managers, registers and integrators.

SCM Integration Server: SCM Domain Integrator. SCM Domain Integrator manages a domain of data servers, providing communication within the domain, and with other domains.

Domain integrators form a tree and can be of two level types: a) a first-level domain integrator covers data servers only, b) a high-level domain integrator covers other integration servers only. A domain integrator holds information about connection distribution among subordinate servers of the domain, reports this information to subordinate servers and higher domain integrators on request. If a request of subordinate server cannot be solved, the request is passed to a higher domain integrator. Each connection creation or deletion is reported to the higher domain integrator, so a line of domain integrators from the first level to the maximal level are aware about connection accessibility.

Domain integrators provide connection creation policy that determines which subordinate server should be used for creation of a connection. The default policy is the following: a) if a connection is based on concepts of one subordinate server, the connection should be created within this server (or its subordinate server); b) else from several subordinate servers, which contain concepts of the connection, one having maximal priority is selected. This policy is described within a domain integrator as a priority defined for each subordinate server.

Each domain integrator has its own connection creation policy. Policy application begins from a domain integrator which covers all concepts participating in the connection and continues for selected subordinate servers in the cascade way. The result is a data server which is used for creation of the connection. Connection creation policies allow to create connections transparently for users, to guarantee SCM constraints correctness, and to integrate new connections to Concept Space as soon as created.

Shared Object Server: SCM Object Manager. SCM Object Manager manages shared objects (components, view points, stored queries), maintains their publication, deletion, access management, and returning on request from SCM integrators.

If a user wish to publish or to manage a shared object, he(she) does it through an integrator, which redirects the user's request to an object manager. Publication of an

object allows to use it through any integrator using one of object searching mechanisms:

- a) inquiry of a common object manager register for an object manager containing a requested object, and then requesting the object manager for the object;
- b) requesting an object manager containing a view point visualizing a certain concept or a stored query (filter) based on the concept directly (this information is stored in the integrator directly, see above).

The last searching mechanism is used for effective dynamic maintenance of Semantic Browser user interface. The mechanism (b) is the most important for maintenance of automatic view point navigation (see below).

Components (for example, grids, charts, reporters, trees, graphs, and others) are published by developers for common use. There can exist mechanism of component access management on basis of payment. But certainly, there should be several free components fulfilling base functionality to browse any view point.

View points and stored queries are published by users themselves for common use or in their own context. As soon as a new object is published it becomes accessible for all users of Concept Space (if the access is not restricted by the publisher of course).

User Machine: SCM Client. SCM Client interacts with an SCM integrator and implements entry point to the Concept Space on a user machine.

SCM Client has the same program interface as SCM integrator and represents an integrator on the side of a user's machine. A user should have at least one client instance for an integrator to use Concept Space, and he/she might have several client instances for one or several integrators if necessary.

User Machine: Semantic Browser. Semantic Browser implements GUI for browsing and modifying Concept Space (its structure and instantiation).

View point is the central element of Semantic Browser. A view point describes layout of view point elements on a user form. Each view point element is visualized with a component. If a component is not accessible (for example, is not paid), the nearest analogue is used. Visualization of each view point element can be adjusted to fit user's needs according to a component's functionality in design time as well as in run time. Design time adjustments is saved as default settings for the view point element, and run time adjustments are saved in a user's context. The adjusting parameters should be standardized for all components to provide compatibility of settings, for example, in the case of a user wishes to switch one component to another without settings change.

A view point element visualizes one data source which is a SCQL query. Data sources of a view point are arranged in a master-detail hierarchy by a user. Connected data sources of the hierarchy should have common role concepts (see feature 9). These role concepts are used for additional filtering of a subordinate data source according to values of the role concepts in a current row (connection instance) of a higher data source. Each view point has one root data source which plays the key role in semantic navigation (see below).

Each data source can have a filter. A filter is an SCQL query which is used for additional filtering of a data source. During view point element visualization a query of a data source and a query of a filter is composed and then projected on role concepts of the data source. A filter defined for a data source in design time is used as a default filter. View points and filters (that are no more than queries) can be saved in

a user's context or can be published in the shared repository and then used by all users of Concept Space.

Concept Space semantic navigation represents the following process:

- a) a user browsing an initial view point selects one or more concept values (instances) within one or several data sources;
- b) Semantic Browser generates a list of possible transitions to other view points, root data sources of which have the selected concepts;
- c) a user selects a transition;
- d) Semantic Browser loads a selected view point and filters its root data source according to concept values selected before.

Transition context is the core conception of semantic navigation in Concept Space. Transition context is created when a user begins transition from a view point and contains concept values selected before. Concept values can be selected within any data source in arbitrary way (rows, columns, separate cells, or a data source as a whole).

List of possible transitions is generated on basis of concepts represented in a transition context: for each represented concept Semantic Browser finds associated view points and requests them from an object manager, the found view points are listed as possible transitions. Possible transition list can be showed in different modes with grouping, filtering, etc.

When a user selects a view point, the view point begins to load. During loading Semantic Browser gets the view point and all necessary components from the shared repository. Component and view point cache is realized on the client side to hasten view point loading. Further Semantic Browser adjusts a root data source filter according to transition context content. If there are several ways of filter adjustment (for example, a root data source has one concept in different roles), Semantic Browser applies the default one, and a user may change this way after the view point has been loaded.

When all components are loaded and a filter is adjusted, data sources are executed from master to detail ones. If a user clicks on a row of a data source, all its detailed data sources are refreshed according to master-detail dependency.

Semantic Browser records a history tree which reflects transitions made by a user since the last program start. A user can go to any node of the tree to load an appropriate view point state.

Semantic Browser includes several assisting modules:

- a) Query assistant allows for users to write SCQL queries conveniently and transparently as a result of SCM features.
- b) Modeler assistant allows for users to create SCM model conveniently and transparently.
- c) Concept Space Manager allows to manage Concept Space: stored queries (filters), view points, components, access rights.

5 Concept Space Features

Due to SCM properties and Concept Space architecture, Concept Space has the following row of interesting features that distinguish it from other data integration systems.

Feature 1. One deal with the flat concept space only, there are no current relations, objectified relations, separate types of relations (for example, inheritance and fact types), and other complex constructions.

SCM is so simple that practically any user can be trained to create and use models. A user deal with semantic concepts, not with attributes. As it is well-known within Relation Model, relational attributes pertain to only one relation and, as a result, do not carry complete semantics of an appropriate concept. Oppositely, SCM reflects application domain concepts with their semantics directly as it is simplified restriction of ORM. In addition, connections of SCM carry complete semantics. So, Concept Space has a very simple and clear structure in comparison with other conceptual models.

Feature 2. View points upon distributed and heterogeneous data are created by users themselves without involving IT specialists.

Concept Space heterogeneity is transparent for users. A user deal with the flat concept space only. So, creation of a distributed and heterogeneous view point is not more complex for users than creation of a centered one. Query assistant allows for users to make this work conveniently and intuitively clear.

Feature 3. Queries are formulated on basis of concepts only by users, using formal Semantically Complete Query Language (SCQL), by means of query by navigation mechanism.

A user writes a query by browsing the flat concept space. Since connections have no proper names, he/she uses one navigation dimension only – concepts. Combining simple base queries, a user can build a query of arbitrary complexity without involving IT people.

Feature 4. SCM-based data integration does not require to maintain explicit mappings of local schemas to a unified global schema. SCM-based local schemas are integrated as soon as created.

Local schemas in the SCM form can share concepts but can not share connections. It allows to consider a global schema as logical composition of all local schemas. As a result, all concepts and connections are integrated as soon as created. Each information system being integrated to Concept Space have a wrapper which maintains a local schema in the form of SCM. The proposed way of data integration does not need any mediator as opposed to current approaches. To integrate an information system to Concept Space it is only necessary to wrap the information system's structures by using an SCM Adaptor supplied for each DBMS or another storage by IT people. It is done locally on the level of the information system (not the global schema). So, separate efforts for local schemas integration is not necessary.

Feature 5. The integration is done in the semantic way that is used by built-in semantic navigation mechanism of Semantic Browser.

Semantic navigation mechanism allows: a) to find view points concerned selected concept values automatically; b) to switch to a view point within selected context concept values, so that the newly loaded view point visualizes information concerned

these concept values only. This mechanism is transparent for users and takes into account heterogeneous nature of Concept Space.

Feature 6. Data access interface are preserved if a storage structure is changed without modification of its semantics.

Local SCM schema is based on information system structures, and there is flexibility in mapping of the latter to the former. Therefore, when system structures are changed with stable semantics, the mapping can be adjusted so that data access interface (local SCM schema) is not changed at all as opposed to other integration approaches. Moreover, the feature 5 of SCM additionally stimulates to use the same SCM model in the case of such change.

Feature 7. Shared repository of visualization components, view points, stored queries (filters) is used.

There is a common distributed repository of objects that are shared by all users of Concept Space. Components are developed and published by IT specialists. Other objects (view points and stored queries) can be developed and published by users themselves. All objects become accessible for other users as soon as published.

Feature 8. Inserting and deleting of connection instances are sufficient palette of modifications within Concept Space.

SCM schema represents a conceptual data representation level, so optimization issues are not concerned in it. Optimization is done when SCQL queries and modifications are translated to, for example, SQL queries and modifications. In addition, SCM model has the feature 1: connections are elementary (see above). As a result, the update operation becomes surplus. An updating is reduced to the sequence of a deleting and an inserting. But for shortening the update operation can still be used.

6 Discussion of Concept Space and Related Work

Discussion 1. Concept Space in comparison with ontology-based integration systems.

Today there are many data integration systems based on different approaches. Ontology-based integration systems are developing dynamically now [35]. Why have we chosen such architecture of Concept Space in comparison with ontology-based systems?

Concept Space is flat and all concepts are connected using one general way without complex constructions. While ontologies form a class hierarchy and bisects concepts and connections: concepts are represented as classes and slots, connections are represented as inheritances and slot references. So, Concept Space uses more simple knowledge representation model having less equivalent representations for one application domain. Simple nature of Concept Space results in that SCM model creation can be fulfilled by users themselves. While ontology creation requires a skilled IT person's efforts.

Global ontologies are mapped to local ones explicitly and the mapping should be created by a skilled IT person [13, 25], while Concept Space global SCM schema is a composition of local SCM schemas, it does not require explicit mapping definition. As a result, administration of Concept Space is reduced to wrappers creation, but

administration of ontology-driven integration systems additionally requires mapping management.

During integration system use, local ontologies are transformed to global ones, while SCM model of Concept Space does not undergo transformation to use. As a result, Concept Space has more transparent structure without implied transformations.

Concept Space can be browsed by users directly using one tool (Semantic Browser), and view points are created by users themselves. While ontology-driven integration systems require programming of each new view point and creation of client programs each time anew.

Concept Space allows to change storage structure without change of data access interface (see feature 6 of Concept Space). While ontologies are often stored and used in the form as they are defined.

Discussion 2. Semantically Complete Modeling in comparison with Object-Role Modeling.

There are several conceptual modeling techniques offering different types of conceptual models: Object-Role Model [6, 18], its particular cases [4, 5, 20, 23, 27, 34], ER model [8, 9], and others. All these models can be used as data access interfaces of integration systems in principle. Why is Semantically Complete Model used for Concept Space? Let us clarify this in comparison with the most advanced conceptual model – Object-Role Model.

First of all, SCM represents flat concept space, while ORM provides many-dimensional navigation space: object types, fact types, inheritance types, objectified fact types, objectified models, sequence types, power types. So, ORM is more complex for users.

Fact types of ORM do not carry complete information about interconnection of underlying object types, while connections of SCM carry complete meaning. To understand real interconnection of object types, ORM forces to analyze several fact types and inheritance types based on these object types jointly, while analysis of one SCM connection is sufficient to understand interconnection of underlying concepts completely. As a result, SCM is simpler for model studying.

SCM has more restricted way of connection declaration as opposed to ORM, and as a result one application domain can be represented in less equivalent forms. It simplifies creation and studying of an SCM model and its comparing with another one. In addition, SCM features allow to build a query language (SCQL) not using proper relation names during query formulation, while ORM features do not. So, SCM has a simpler query language that is better comprehended by users.

Users may not know what is a relation in its mathematical sense and how a fact type differs from an inheritance type, but everyone intuitively knows what concepts and connections of concepts are. SCM just allows to think in this level without complicated constructions which ORM have. Moreover, SCM allows to stay in flat concept space without relations having self-dependent meaning. It is natural for people not being IT specialists and is simpler for IT specialists as well.

During natural language text analysis using ORM, a fact type can be distinguished only by verb way of writing. As a result, a natural language text should contain verbs defined for fact types in a model. Whereas using SCM, a natural language text can contain any known verbs ignoring the synonymy problem since a connection can be identified on basis of concept names or parts of concept names.

Discussion 3. SCQL in comparison with SQL.

There are many query languages having different purposes: LISA-D [21, 22], ConQuer [2, 3], RDQL [33], and others. The most important one is SQL which is technological standard for RDBMS and is prevalent now. There are several SQL-like languages for conceptual models (for example, RDQL). Why do we introduce a new query language (SCQL) having constructions differing from SQL constructions significantly, and why do we use it in Concept Space? Let's clarify this.

SQL deals with columns of tables merely, while SCQL deals with concepts as such. One concept of SCM can be represented in several tables columns of a relational schema, mapping of columns to concepts is out of SQL (and relational schema as well).

It is not necessary to define and use relations' proper names in SCQL, while SQL requires to mention table names, for example, selection of SQL has the clause FROM where all participated tables are enumerated explicitly.

Join criterion is defined automatically according to role concepts coincidence in SCQL, while SQL requires to define the criterion explicitly in the clause WHERE of selection. This distinction plays the key role in decision to use SCQL for Concept Space since it allows semantic-oriented view points creation and semantic data navigation without programming.

Union and minus of SCQL allow for source relations to have different signatures and arities, do not require manual relation alignment, and do not allow semantics mixing. The way of union and minus execution is determined by role concepts underlying source relations automatically. Whereas SQL requires the same signature and arity of source relations, requires manual relation alignment, and allows to mix different semantics (concepts) to one column. This distinction additionally emphasizes semantic orientation of SCQL as opposed to SQL.

SCQL has very transparent way of definition of binary relations composition: as a chain of concepts (see feature 10). Selection in SCQL represents a particular case of composition (see feature 11). Also, there is a mechanism (closures) which allows to request interconnection of indirectly connected concepts as if they are connected directly (feature 12). All these distinctions lead to greater simplicity of SCQL for non-IT and IT people in comparison with SQL. In addition, SCQL allows intuitively transparent way of query creation with the help of a query assistant.

7 Open Issues

Practical implementation of the proposed architecture requires to fulfill researches in several directions which are uncovered yet. All these open issues are connected with SCM closely since it is the core of the architecture.

Open Issue 1. Basics of a conceptual model representing a flat concept space.

Partially this issue is discussed in [28, 30]: SCM core and base constraints are explained and formalized. But complex constraints are not defined yet, this work is underway.

Open Issue 2. A query language formulating queries considering concepts only.

This issue is discussed in [29, 30]: SCQL is introduced, main representations (operations) are defined. Further research is directed at extension of SCQL: adding outer composition (join), grouping, analytical, and other query operations, plus modification operations.

Open Issue 3. Principles of mapping of Relational Model to SCM.

Solving of this issue is necessary for creation of SCM Adaptor component upon an RDBMS. This work is underway.

Open Issue 4. Mapping SCQL queries and modifications to SQL.

This issue is necessary for the same purpose as the third one. Solving it, one should take into account not only query optimization aspects, but also modification optimization aspects. For example, several SCQL deletions and insertions can be transformed to one SQL update.

Open Issue 5. Built-in dictionary support, multi-language support, and concept complex structure support in Concept Space.

High generality of Semantic Browser requires that SCM Adaptor maintains the value domain conception with the following features:

- a) A value domain can have a restricting list of possible values. This list is used by Semantic Browser to provide for users with possible values and to check the appropriate constraint.
- b) A value domain can have a complex structure like a class of OO programming languages. It can be used for mapping object-relational schemas to SCM.
- c) All string values (including possible values of value domains) should be multilingual so that Concept Space can work with many languages without additional efforts.

Open Issue 6. Support of manipulation with metainformation (concepts, connections, constraints) through SCM data access interface.

How to arrange access to metainformation of Concept Space? Apparently, the access should be organized using the same SCM data access interface. Modification of metainformation can be fulfilled through the same interface too. This issue is not investigated yet.

Open Issue 7. The principle of access control within Concept Space.

Access to all objects of Concept Space should be restricted according to access rights granted by object owners to other users. How this rights should be managed is not investigated yet.

To complete architecture of Concept Space, several architectures of its components should be designed in detail:

Architecture 1. Architecture of a program layer maintaining SCM data access interface (by means of SCQL) to process data stored in a relational schema.

Solving of discussed open issues is preparing the way to SCM Adaptor architecture elaboration for RDBMS (see SCM Adaptor role within Concept Space above). Author is now developing a version of such architecture with somewhat restricted functionality.

Architecture 2. Architecture of a program layer maintaining SCM data access interface that transparently deals with distributed and heterogeneous data wrapped by SCM Adaptors of different types.

SCM Adaptor does not carry full functionality for Concept Space, it covers a single data server only. SCM Integrator should be created to provide integration of different

data servers to entire Concept Space. Its architecture should comply with all properties described above to guarantee all declared Concept Space features, for example, it should be able to fulfill distributed CSQL queries and modifications effectively.

Architecture 3. Architecture of a program layer integrating several domains of data servers to one information space.

SCM Integrator requires to interact with a program layer covering a domain of data servers: SCM Domain Integrator. A domain integrator architecture should take into account that, for example, domain integrators communicates each other forming one hierarchy. This architecture does not investigated yet in detail.

Architecture 4. Architecture of a program layer providing access to Concept Space from clients' computers transparently.

SCM client provide API on behalf of an integrator. It should allow to connect and interact with Concept Space as a whole using one entry point.

Architecture 5. Architecture of distributed shared repository of view points, stored queries (filters), visualization components.

This architecture includes specification of SCM Object Manager and protocol of interaction with integrators. This architecture does not investigated yet in detail.

Architecture 6. Architecture of general GUI for browsing of Concept Space.

This architecture should describe principles of Semantic Browser: view points constructing and visualization, semantic navigation, and others. Auxiliary modules should be designed too: administration module, modeler assistant, query assistant, and others.

8 Conclusion

This paper introduces a general architecture of a data integration system managed by users. The proposed approach makes the following contributions to methodology of data integration systems creation:

- a) In this approach, the work of conceptual model and view points creation and integration is done by users themselves without involving IT people because of high simplicity of SCM and its high ability to reflect semantics.
- b) The approach provides semantic data integration within a global space automatically as soon as new concepts and connections are published without any administration, with automatically supported semantic navigation.
- c) Embodiment of this approach to Internet will allow to hasten development of Internet applications and to increase data integration level up to semantic level.

The author makes further investigations in the directions outlined in the article. In particular, architecture of SCM Adaptor is prepared and is being implemented now.

References

1. Bergamaschi S., Castano S., Vincini M., Beneventano D. Semantic integration of heterogeneous information sources // *Data & Knowledge Engineering*, 36(3), 2001, pp.215-249.
2. Bloesch A.C., Halpin T.A. ConQuer: A Conceptual Query Language // *Proceedings of ER'96: 15-th International Conference on Conceptual Modeling*, LNCS, no. 1157, 1996, pp. 121-133.
3. Bloesch A.C., Halpin T.A. Conceptual Queries using ConQuer-II // *Proceedings of ER'97: 16-th International Conference on Conceptual Modeling*, 1997.
4. van Bommel, P.; ter Hofstede, A.H.M.; van der Weide: Semantics and verification of object-role models: *Information Systems*, 16(5), 1991; pp.471-495.
5. Brouwer, S.J.; Martens, C.L.J.; Bronts, G.H.W.M.; Proper, H.A.: Towards a Unifying Object Role Modelling Approach: *Proc. of the First International Conference on Object-Role Modelling (ORM-1)*, Magnetic Island, Australia, 1994; pp. 259-273.
6. Bronts G.H.W.M., Brouwer S.J., Martens C.L.J., Proper H.A. A Unifying Object Role Modelling Approach. *Information Systems*, 20(3), 1995, pp. 213-235.
7. Chawathe S., Garcia-Molina H., Hammer J., Ireland K., Papakonstantinou Y., Ullman J., Widom J. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, 1994, pp.7-18.
8. Chen P.P.S. The entity-relationship model – towards a unified view of data // *ACM Transactions on Database Systems*, 1(1), 1976, pp. 9-36.
9. Chen P.P.S. A Preliminary Framework for Entity-Relationship Models // *Entity-Relationship Approach to Information Modeling and Analysis*, Saugus, Calif., 1981.
10. Collins S.R., Navathe S., Mark L. XML schema mappings for heterogeneous database access // *Information and Software Technology*, 44(4), 2002, pp. 251-257.
11. Claypool K.T., Rundensteiner E.A. Gangam: A Transformation Modeling Framework // *8th International Conference on Database Systems for Advanced Applications (DASFAA'03)*, 2003, p. 47.
12. Date C.J., Fagin R. Simple conditions for quaranteeing higher normal forms in relational databases // *ACM Transactions on Database Systems*, 17(3), 1992, pp. 465-476.
13. Ermolayev V., Keberle N., Shapar V., Vladimirov V. Ontology-Driven Sub-Query Extraction for Distributed Autonomous Information Resources in UnIT-Net IEDI // In (Doroshenko A., Halpin T., Liddle S., Mayr H. eds.) *Proc. of the 3rd International Conference ISTA'2004: Information Systems Technology and its Applications*, Salt Lake City, GI Lecture Notes in Informatics P-48, 2004, pp. 137-150.
14. Friedman M., Levy A., Millstein T. Navigational plans for data integration. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI'99)*, 1999, pp. 67-73.
15. Fagin R., Vardi M.Y. The Theory of Data Dependencies – A Survey // *IBM Research Report RJ4321*, 1984.
16. Gartner H., Bergmann A., Schmidt J. Object-oriented modeling of data sources as a tool for the integration of heterogeneous geoscientific information // *Computers & Geosciences*, 27(8), 2001, pp. 975-985.
17. van Griethuysen J.J., editor. *Concepts and Terminology for the Conceptual Scheme and the Information Base*. Publ. nr. ISO/TC97/SC5-N695, 1982.
18. Halpin T.A. *Information Modeling and Relational Databases*. Morgan Kaufmann, San Francisco, 2001.
19. Halpin T.A. *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, Australia, 2nd edition, 1995.
20. Halpin, T.A.; Orłowska, M.E.: Fact-Oriented Modelling for Data Analysis: *Journal of Information Systems*, 2(2), 1992; pp. 1-23.

21. ter Hofstede A.H.M., Proper H.A., van der Weide. Formal Definition of a Conceptual Language for the Description and Manipulation of Information Models // *Information Systems*, 18(7), 1993, pp. 489-523.
22. ter Hofstede A.H.M., Proper H.A., van der Weide. A Conceptual Language for the Description and Manipulation of Complex Information Models. In *Seventeenth Annual Computer Science Conference*, vol. 16 of *Australian Computer Science Communications*, 1994, pp. 157-167.
23. ter Hofstede, A.H.M.; van der Weide: Expressiveness in conceptual data modeling: *Data & Knowledge Engineering*, 10(1), 1993; pp. 65-100.
24. Jensen M.R., Moller T.H., Pedersen T.B. Converting XML DTDs to UML diagrams for conceptual data integration // *Data & Knowledge Engineering*, 44(3), 2003, pp. 323-346.
25. Li Y., Liu D., Zhang W. A Data Transformation Method Based On Schema Mapping // In (Doroshenko A., Halpin T., Liddle S., Mayr H. eds.) *Proc. of the 3rd International Conference ISTA'2004: Information Systems Technology and its Applications*, Salt Lake City, GI Lecture Notes in Informatics P-48, 2004, pp. 67-79.
26. Levy A.Y., Rajaraman A., Ordille J.J. Querying heterogeneous information sources using source descriptions // In *Proc. of the 22nd International Conference on Very Large Data Bases (VLDB'96)*, 1996, pp/ 251-262.
27. Nijssen, G.M., Halpin, T.A.: *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989.
28. Ovchinnikov V.V. A Conceptual Modeling Technique without Redundant Structural Elements // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 29, 2003.
29. Ovchinnikov V.V. A Conceptual Modeling Technique Based on Semantically Complete Model, its Applications // In (Doroshenko A., Halpin T., Liddle S., Mayr H. eds.) *Proc. of the 3rd International Conference ISTA'2004: Information Systems Technology and its Applications*, Salt Lake City, GI Lecture Notes in Informatics P-48, 2004, pp. 25-38.
30. Ovchinnikov V.V. A Semantically Complete Conceptual Modeling Technique // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 32, 2004.
31. Ovchinnikov V.V. Improving Controllability of Vast Conceptual Models // *Journal of Conceptual Modeling* (www.inconcept.com/jcm), 31, 2004.
32. Pontieri L., Ursino D., Zumpano E. An approach for the extensional integration of data sources with heterogeneous representation formats // *Data & Knowledge Engineering*, 45(3), 2003, pp.291-331.
33. RDQL – A Query Language for RDF. W3C Member Submission, 2004, URL: <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>
34. de Troyer, O.M.F.: The OO-Binary Relationship Model: A Truly Object Oriented Conceptual Model.: *Proc. of the Third International Conference CaiSE'91 on Advanced Information Systems Engineering*, vol. 498 of *Lecture Notes in Computer Science*, Trondheim, Norway, 1991; pp. 561-578.
35. Wache H. Ontology-Based Integration of Information – A Survey of Existing Approaches // In (A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, M. Uschold) *Proc. of the IJZAI-01 Workshop on Ontologies and Information Sharing*, 2001, pp.108-118.
36. Xu L., Embley D.W. Combining the Best Global-as-View and Local-as-View for Data Integration // In (Doroshenko A., Halpin T., Liddle S., Mayr H. eds.) *Proc. of the 3rd International Conference ISTA'2004: Information Systems Technology and its Applications*, Salt Lake City, GI Lecture Notes in Informatics P-48, 2004, pp. 123-135.