An Information Integration Framework for E-Commerce

Ilario Benetti, Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio Vincini, *Università di Modena e Reggio Emilia*

The Web has transformed electronic information systems from single, isolated nodes into a worldwide network of information exchange and business transactions. In this context, companies have equipped themselves with high-capacity storage systems that contain data in several formats. The problems faced by these companies often emerge because the storage systems lack structural and application homogeneity in addition to a common ontology. The semantic differences generated by a lack of consistent ontology can lead to conflicts that range from simple name contradictions (when companies use different names to indicate the same data concept) to structural incompatibilities (when companies use different models to represent the same information types).

One of the main challenges for e-commerce infrastructure designers is to retrieve data from different sources and create a unified view that overcomes contradictions and redundancies. Virtual catalogs, such as the Momis project, can help synthesize data and present it in a unified manner to the customer.

One of the main challenges for e-commerce infrastructure designers is information sharing and retrieving data from different sources to obtain an integrated view that can overcome any contradictions or redundancies. Virtual catalogs can help overcome this challenge because they act as instruments to retrieve information dynamically from multiple catalogs and present unified product data to customers. Instead of having to interact with multiple heterogeneous catalogs, customers can instead interact with a virtual catalog in a straightforward, uniform manner.

This article presents a virtual catalog project called Momis (*m*ediator envir*o*nment for *m*ultiple *i*nformation *s*ources). Momis is a mediator-based system for information extraction and integration that works with structured and semistructured data sources.[1,2] Momis includes a component called the SI-Designer for semiautomatically integrating the schemas of heterogeneous data sources, such as relational, object, XML, or semistructured sources. Starting from local source descriptions, the Global Schema Builder generates an integrated view of all data sources and expresses those views using XML. Momis lets you use the infrastructure with other open integration information systems by simply interchanging XML data files.

Momis creates XML global schema using different stages, first by creating a common thesaurus of intra and interschema relationships. Momis extracts the intraschema relationships by using inference techniques, then shares these relationships in the common thesaurus. After this initial phase, Momis enriches the common thesaurus with interschema relationships obtained using the lexical WordNet system (www.cogsci.princeton.edu/wn), which identifies the affinities between interschema concepts on the basis of their lexicon meaning. Momis also enriches the common thesaurus using the Artemis system,[3] which evaluates structural affinities among interschema concepts.

System architecture

Like other integration projects, Momis relies on a semantic approach based on the conceptual schema—or metadata—of the information sources, and on the I^3 architecture[4] (see Figure 1). The system consists of the following functional elements that communicate using the Corba standard: a common data model, data wrappers, and a mediator.

Momis defines the common data model, ODM_{I3}, according to the ODL_{I3} language, which describes source schemas for integration purposes. Momis treats ODM_{I3} and ODL_{I3} as subsets of the corresponding languages in the ODMG specification—according to the proposal for a standard mediator language developed by the I^3-POB working group.[5] In addition, ODL_{I3} introduces new constructors to support the semantic integration process. Momis uses the second functional element, data wrappers, over each piece of source data to translate metadata descriptions of the source into common ODL$_{I3}$ representations and to translate a global query expressed in OQL$_{I3}$—which is a subset of OQL-ODMG—into queries expressed in the source languages. Wrappers then serve to export the resulting data set.

The third functional element, the mediator, consists of two modules: the global schema builder and the query manager. The global schema builder processes and integrates ODL$_{I3}$ descriptions received from wrappers to derive the information source representations. The query manager module performs query processing and optimization, generates OQL$_{I3}$ subqueries for the sources, and synthesizes a unified global answer for the user.

Momis creates an integrated view of all sources—called the global virtual view—and performs revision and validation of the various kinds of knowledge used for the integration. To accomplish this, Momis combines the reasoning capabilities of description logics with affinity-based clustering techniques. Momis then exploits a common ontology for the sources constructed using lexical knowledge derived from WordNet and validated integration knowledge. Momis expresses the global virtual view using XML to guarantee interoperability with other open integration systems.

The user application interacts with Momis by straightforwardly querying the global schema using the OQL$_{I3}$ language. The query manager performs this phase by generating the OQL$_{I3}$ queries for the wrappers. Using mapping-description techniques, the query manager generates the queries automatically by formulating and optimizing the generic OQL$_{I3}$ queries into different subqueries, one for each involved local data source, and synthesizes a unified global answer.

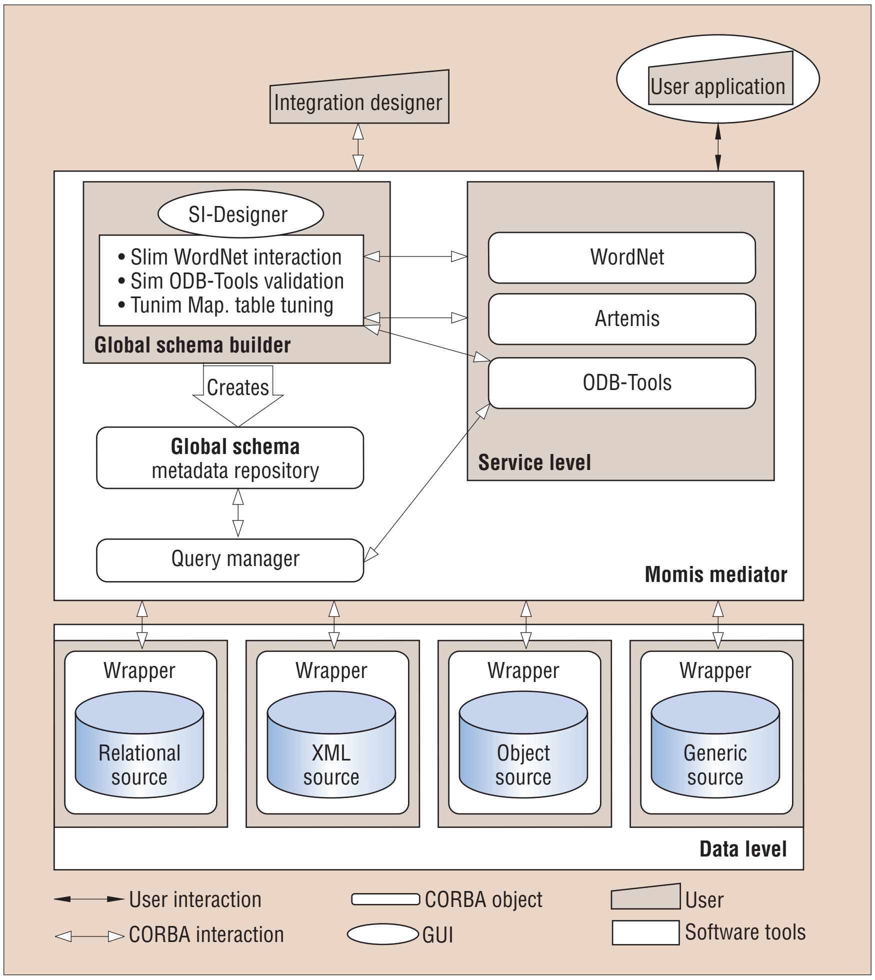


Figure 1. The Momis architecture is divided into three levels: a data level to manage the information sources, a mediator level to perform the integration of the involved sources by interacting with integration designer and to execute the global query processing, and a user level to interact with the designer.

ODL$_{I3}$ language

For a semantically rich representation of source schemas and object patterns, Momis uses an object-oriented language called ODL$_{I3}$. ODL$_{I3}$ is close to the ODL language because it is source-independent and can be used to describe heterogeneous schemas of structured and semistructured data sources. ODL$_{I3}$ extends ODL with intensional and extensional relationships expressing intra- and interschema knowledge for the source schemas.

You can specify the following relationships in ODL$_{I3}$:

- SYN (synonym of) is a relationship defined between two terms t_i and t_j (where $t_i \neq t_j$) that are synonyms in every involved source. For example, you can use t_i and t_j in every source to denote a certain concept.
- BT (broader terms) is a relationship defined between two terms t_i and t_j, where t_i has a broader, more general meaning than t_j. BT relationships are not symmetric. The opposite of BT is NT (narrower terms).
- RT (related terms) is a relationship defined between two terms t_i and t_j that are generally used together in the same context in the considered sources.

An intensional relationship has no implications on the extension or compatibility of the structure of the two involved classes. You can strengthen the intensional relationship syn, bt, and nt between two classes C1 and C2 by establishing that they are also extensional relationships. Consequently, you can define the following extensional relationships in ODLI3:

- C_1 SYN$_{ext}$ C_2 means that the instances of C_1 are the same as C_2.
- C_1 BT$_{ext}$ C_2 means that the instances of C_1 are a superset of the instances of C_2.
- C_1 NT$_{ext}$ C_2 means that the instances of C_1 are a subset of the instances of C_2.
- C_1 DISJ$_{ext}$ C_2 means that the instances of C_1 are disjoint from the instances of C_2.

We assume a default overlap relationships among two classes if no extensional relationship is specified. Moreover, extensional relationships constrain the structure of the two classes C_1 and C_2. If an extensional relationship C_1 NT$_{ext}C_2$ is issued, we enforce a strict inheritance between C_1 and C_2 for the common attributes. Both C_1 and C_2 might have additional attributes as we adopt usual description logics semantics (open world semantics).

To illustrate how Momis works, we use two examples involving different data

```
<!ELEMENT fiat(car*)>
<!ELEMENT car(name, engine, dimensions, tires, performance, price)>
<!ELEMENT engine(name, cylinders?, layout?, capacity?, compression_ratio?, power_w, fuel_system)>
<!ELEMENT dimensions(length, width, height, luggage_capacity)>
<!ELEMENT performance (urban_consumption, combined_consumption, speed)>
<!ELEMENT name (#pcdata)>...
```

Figure 2. By using the Momis system, the user can integrate the XML data source. This DTD represents the schema of a source that stores information about the Fiat.

sources that collect information about vehicles. The first data source, shown in Figure 2, is the Fiat catalog. It contains semistructured XML information. The second data source, shown in Figure 3, is the Volkswagen database, a relational database containing information about Volkswagens.

Integration process

The Momis approach generates a common thesaurus, uses affinity analysis to analyze classes, then clusters the classes and generates the mediated schema. As mentioned above, the common thesaurus is a set of terminological intensional and extensional relationships that describe intra- and interschema knowledge about classes and attributes of the source schemas. Momis uses relationships in the common thesaurus to evaluate the level of affinity between classes' intra- and intersources. We introduced the concept of affinity to formalize the kind of relationships that can occur between classes.

Momis groups classes (that have affinity in different sources) together in clusters using hierarchical clustering techniques. The goal is to identify the classes that Momis will integrate. Momis defines a class for each cluster; this class becomes representative of all the classes belonging to the cluster. The global mediated schema for the analyzed source data consists of all the classes derived from clusters and is the basis for posing queries against the sources.

Extracting lexicon relationships

WordNet, a lexical database developed by the Princeton University cognitive science lab, was inspired by current psycholinguistic lexical memory theories. Many regard WordNet as the most important research resource for computational linguistics, textual analysis, and other related areas. The WordNet database contains 64,089 lemma organized in 99,757 synonym sets. WordNet's starting point for lexical semantics comes from a conventional association between the forms of the words—that is, the way in which the words are pronounced or written—and the concept or meaning they express. These associations, which are of the many-to-many kind, give rise to several properties, including synonymy, polysemy, and so forth.

Synonymy is the property of a concept or meaning that you can express with two or more words. In WordNet, you call a synonym group a *synset*. Only one synset exists for each concept or meaning. In WordNet, you indicate a synset with s, while S indicates the synsets set. In contrast to synonymy, polysemy denotes the property of a single word that has two or more meanings. WordNet synthesizes the correspondence between the form of the words and their meaning in the lexical matrix $\boldsymbol{M}$, in which the words' meaning are reported in rows (where each row represents a synset) and columns that represent the form of the words.

In Momis, we characterize each matrix element as $e = (f, m)$, where f is the base form and m is the meaning counter. For example, name,1 refers to the language unit by which a person is known, whereas name,3 refers to a person's reputation. In the remainder of this article, we indicate the base form and the meaning counter of an element $e = (f, m)$ with $e.f$ and $e.m$. An element of the $\boldsymbol{M}$ matrix can be null or indefinite. Because Momis only associates one $\boldsymbol{M}$ row with an s, we use $s \in S$ as an $\boldsymbol{M}$-row indicator. In other words, the non-null elements of the $\boldsymbol{M}[s]$ row represent each s element. In the same way, because only one $\boldsymbol{M}$ column is associated with a base form, we use the base forms as an $\boldsymbol{M}$ columns index.

Semantic relationships between schema terms

The concept *term* lets us associate a definition with each class or attribute name. We form a term in Momis with the $t = (n, e)$ couple, where n indicates a class or attribute name and e indicates a definition. We qualify a class or attribute name n by the name of the source schema to which the class belongs, as in source_name.class_name. We qualify an attribute name with the name of the class to which it belongs, as in source_name.class_name.attribute_name.

These relationships between synsets form the starting point for Momis to define semantic relations among words. Some of these relations exist between single words, while others exist between synsets. In this context, we use the following relations between synsets: synonymy, hypernymy, hyponymy, olonymy, meronymy, and correlation. Correlation is a relation that links two synsets sharing the same hypernym—that is, the same father. Because hyponymy and meronymy are inverse relations to hypernymy and olonymy, we represent the set of relations among synsets with the following equation: $W = \{\boldsymbol{S}_{ynonymy}, \boldsymbol{H}_{ypernymy}, \boldsymbol{O}_{lonymy}, \boldsymbol{C}_{orrelation}\}$.

Given the synset set S and the relations set W, we define the function $\phi\colon S \times W \rightarrow 2^S$ for each synset s as the set of synsets associated through the $r \in W$ relation, as in the following equation:

$$\phi\,(s,r) = \{s' \mid s' \in S,\ r \in W,\ \langle s' r s\rangle holds\}$$

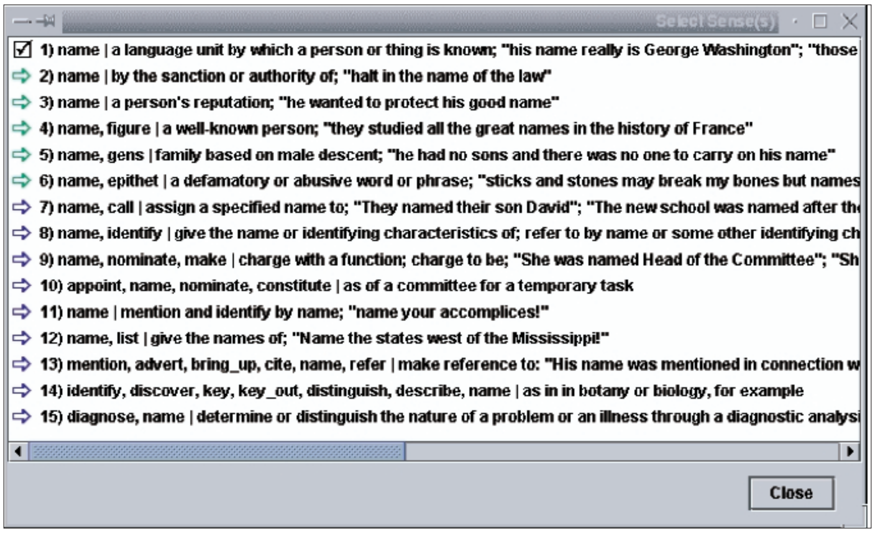
Given a synset S set and a set I of words S, we define the function $H\colon S \rightarrow 2^I$ on the basis of the lexical matrix $\boldsymbol{M}$ as in the following equation:

$$H(s) = \{n \in \mathbf{N} \mid \exists t\,(n,e),\ \boldsymbol{M}\,[s][e.f] = t.n\}$$

We can therefore obtain the relations among the words on the basis of relations existing among the synsets that contain those words. Given a set of words I, we can define the set of relations among words R, $R \subseteq I \times W \times I$ as follows:

```
Vehicle(name, length, width, height)
Motor(cod_m, type, compression_ratio, KW, lubrication, emission)
Fuel_Consumption(name, cod_m, drive_trains, city_km_l, highway_km_l)
Model(name, cod_m, tires, steering, price)
```

Figure 3. Another example of a data source that you can manage with the Momis system. In this case, it is a relational database that stores information about the Volkswagen.


Select Sense(s)

☑ 1) name | a language unit by which a person or thing is known; "his name really is George Washington"; "those

⇨ 2) name | by the sanction or authority of; "halt in the name of the law"

⇨ 3) name | a person's reputation; "he wanted to protect his good name"

⇨ 4) name, figure | a well-known person; "they studied all the great names in the history of France"

⇨ 5) name, gens | family based on male descent; "he had no sons and there was no one to carry on his name"

⇨ 6) name, epithet | a defamatory or abusive word or phrase; "sticks and stones may break my bones but names

⇨ 7) name, call | assign a specified name to; "They named their son David"; "The new school was named after th

⇨ 8) name, identify | give the name or identifying characteristics of; refer to by name or some other identifying ch

⇨ 9) name, nominate, make | charge with a function; charge to be; "She was named Head of the Committee"; "Sh

⇨ 10) appoint, name, nominate, constitute | as of a committee for a temporary task

⇨ 11) name | mention and identify by name; "name your accomplices!"

⇨ 12) name, list | give the names of; "Name the states west of the Mississippi!"

⇨ 13) mention, advert, bring_up, cite, name, refer | make reference to: "His name was mentioned in connection w

⇨ 14) identify, discover, key, key_out, distinguish, describe, name | as in in botany or biology, for example

⇨ 15) diagnose, name | determine or distinguish the nature of a problem or an illness through a diagnostic analysi

Figure 4. By using a graphical interface within SI-Designer, you can select the correct meaning of each schema field. By querying the WordNet lexical system, the graphical interface suggests a list of meanings.

$$R=\{\langle t_i r t_j\rangle | r\in W, t_i t_j \in I, \exists (s): t_i \in H(s), t_j \in \phi(s,r),\ t_i \neq t_j\}$$

The relationships formed from the interaction with WordNet become semantic relations inserted in Momis' common thesaurus according to the following flow: synonymy ⟹ SYN, hypernymy ⟹ BT, olonymy ⟹ RT, and correlation ⟹ RT.

Using these base principles, we developed an algorithm that uses the terms related to the schemata to be integrated as input and the detected semantic relations as output:

Input $I = \{t_i \mid t_i.n \in \mathbf{N}\}$
Output $R = \{\langle t_i r t_j\rangle,\ r \in \{\text{SYN, BT, RT}\}\}$

Starting from the schema to be integrated, Momis creates the *I* set. For example, given a name *n*, Momis chooses the associated words. This choice consists of choosing both a base form and a meaning.

For example, as shown in Figure 4, by selecting the car.name attribute, we obtain the name base form from the morphologic processor. If Momis does not find a base form, if there is an ambiguity, or if the base form is not satisfactory, you can introduce a base form manually. You can relate a name to one, more than one, or even no meaning at all. You might choose to have a name not related to any meaning for several reasons: because the concept is too complex and it cannot be expressed with one word; because the concept belongs to a generic category and is more closely related to the whole; because the concept is already a substitute key and does not add any knowledge; or because the concept is used as a foreign key and has already been used to extract relations from the schema structure.

The designer selects one or more meanings from those found in WordNet starting from the base form. All words related to the same name share the same base form. Figure 4 illustrates how WordNet obtains all 15 meanings related to the name base form. By selecting them all—that is, by considering 15 words for the car.name attribute—we would obtain results not suitable for the examined context. We show some of them in the following example:

⟨fiat.car.name SYN vw.fuel_consumption.name⟩
⟨fiat.car.name SYN fiat.engine.name⟩

Note that some of these relationships can look quite strange but they are true in some particular context. The problem is how to resolve the meaning ambiguity so that you can supply a context-suitable couple (base form and meaning counter) to WordNet for each source concept. To help you choose the right meaning for each couple, WordNet indicates a syntactic category. This semiautomatic approach helps reduce the task's complexity. In fact, you can divide a difficult problem into several more easy ones and choose each term's meaning from a list.

In practice, this is an 80–20 problem. That is, you can work out 80 percent of the words in 20 percent of the time, while the remaining 20 percent occupies 80 percent of the time. To speed up the 80 percent part, Momis uses a cache of the already selected couple. In Figure 5, the symbol ⟹ denotes the meaning already chosen by the designer for the name concept.

The SI-Designer can show the generalization hierarchy of the meanings to help you make the most difficult choices. For example, in Figure 5, "engine#2" inherits only from "causal_agent..." and "engine#3" from "wheeled_vehicle" and "concern railway contest," whereas "engine#1" inherits also from "machine" and "motor#1." Thus we select "engine#1." At the end of this phase, the SI-Designer shows the relationships derived using WordNet (see Figure 6). At this point, you can delete any of these relationships or add new ones.

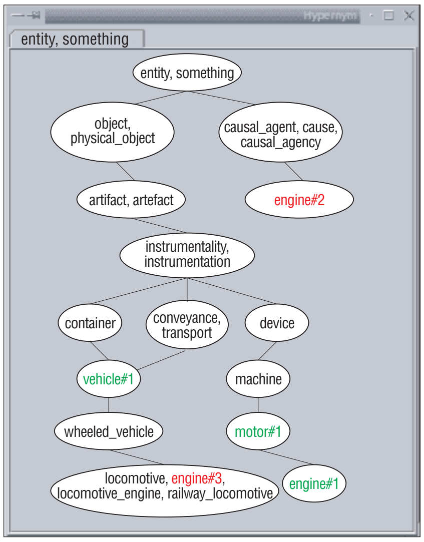


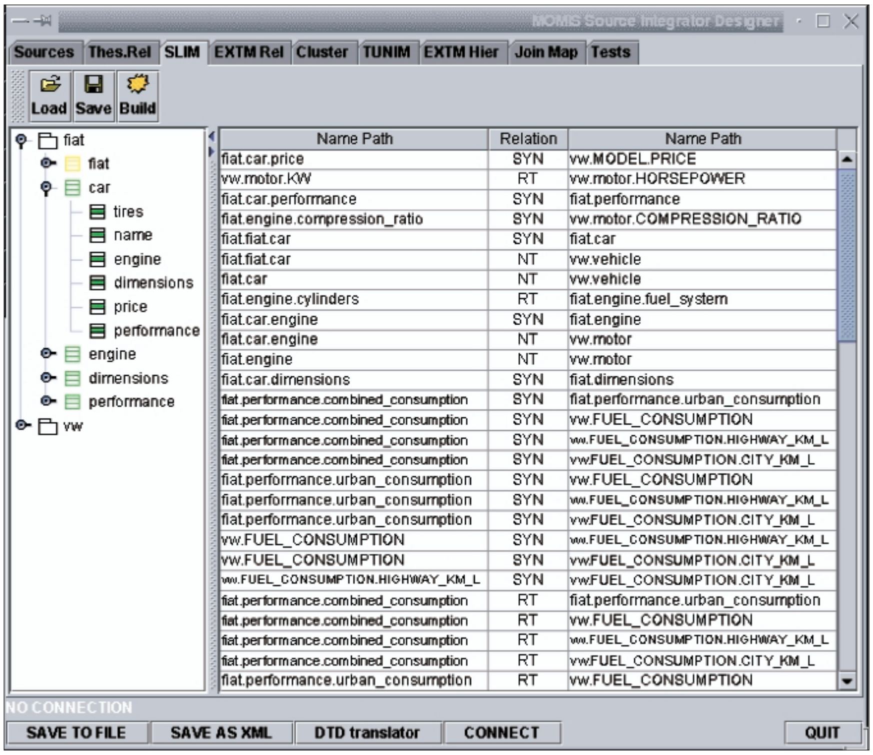
Figure 5. An example of a hypernymy hierarchy of the term engine.

The common thesaurus

In the common thesaurus, we express interschema knowledge in the form of terminological relationships (such as SYN, BT, NT, and RT) and extensional relationships (such as SYN_{ext}, BT_{ext}, and NT_{ext}) between classes or attribute names. Momis constructs the common thesaurus through an incremental process during which relationships are added in the following order: schema-derived relationships not modified by the user, lexicon-derived relationships, user-supplied relationships, and inferred relationships. You can refine lexicon-derived and user-supplied relationships at each step of the integration process.

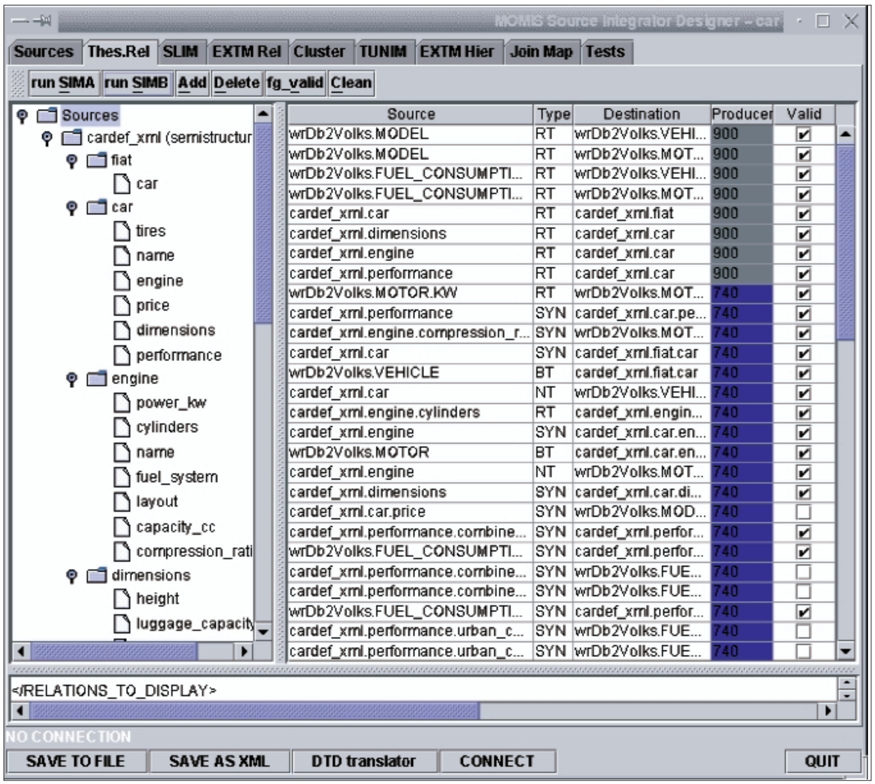
Momis extracts these relationships using the Source Lessical Integration Module (SLIM) by analyzing each ODL schema separately. In particular, Momis extracts each intraschema RT relationship from the specification of foreign keys in relational source schemas. When a foreign key is also a primary key both in the original and in the referenced relation, Momis extracts a BT–NT relationship.

The SLIM module extracts the intensional



Name Path	Relation	Name Path
fiat.car.price	SYN	vw.MODEL.PRICE
vw.motor.KW	RT	vw.motor.HORSEPOWER
fiat.car.performance	SYN	fiat.performance
fiat.engine.compression_ratio	SYN	vw.motor.COMPRESSION_RATIO
fiat.fiat.car	SYN	fiat.car
fiat.fiat.car	NT	vw.vehicle
fiat.car	NT	vw.vehicle
fiat.engine.cylinders	RT	fiat.engine.fuel_system
fiat.car.engine	SYN	fiat.engine
fiat.car.engine	NT	vw.motor
fiat.engine	NT	vw.motor
fiat.car.dimensions	SYN	fiat.dimensions
fiat.performance.combined_consumption	SYN	fiat.performance.urban_consumption
fiat.performance.combined_consumption	SYN	vw.FUEL_CONSUMPTION
fiat.performance.combined_consumption	SYN	vw.FUEL_CONSUMPTION.HIGHWAY_KM_L
fiat.performance.combined_consumption	SYN	vwFUEL_CONSUMPTION.CITY_KM_L
fiat.performance.urban_consumption	SYN	vw.FUEL_CONSUMPTION
fiat.performance.urban_consumption	SYN	vw.FUEL_CONSUMPTION.HIGHWAY_KM_L
fiat.performance.urban_consumption	SYN	vwFUEL_CONSUMPTION.CITY_KM_L
vw.FUEL_CONSUMPTION	SYN	vw.FUEL_CONSUMPTION.HIGHWAY_KM_L
vw.FUEL_CONSUMPTION	SYN	vwFUEL_CONSUMPTION.CITY_KM_L
vw.FUEL_CONSUMPTION.HIGHWAY_KM_L	SYN	vwFUEL_CONSUMPTION.CITY_KM_L
fiat.performance.combined_consumption	RT	fiat.performance.urban_consumption
fiat.performance.combined_consumption	RT	vw.FUEL_CONSUMPTION
fiat.performance.combined_consumption	RT	vw.FUEL_CONSUMPTION.HIGHWAY_KM_L
fiat.performance.combined_consumption	RT	vwFUEL_CONSUMPTION.CITY_KM_L
fiat.performance.urban_consumption	RT	vw.FUEL_CONSUMPTION

Figure 6. An example of interschema relationships extracted by SLIM.



Source	Type	Destination	Producer	Valid
wrDb2Volks.MODEL	RT	wrDb2Volks.VEHI...	900	✓
wrDb2Volks.MODEL	RT	wrDb2Volks.MOT...	900	✓
wrDb2Volks.FUEL_CONSUMPTI...	RT	wrDb2Volks.VEHI...	900	✓
wrDb2Volks.FUEL_CONSUMPTI...	RT	wrDb2Volks.MOT...	900	✓
cardef_xml.car	RT	cardef_xml.fiat	900	✓
cardef_xml.dimensions	RT	cardef_xml.car	900	✓
cardef_xml.engine	RT	cardef_xml.car	900	✓
cardef_xml.performance	RT	cardef_xml.car	900	✓
wrDb2Volks.MOTOR.KW	RT	wrDb2Volks.MOT...	740	✓
cardef_xml.performance	SYN	cardef_xml.car.pe...	740	✓
cardef_xml.engine.compression_r...	SYN	wrDb2Volks.MOT...	740	✓
cardef_xml.car	SYN	cardef_xml.fiat.car	740	✓
wrDb2Volks.VEHICLE	BT	cardef_xml.fiat.car	740	✓
cardef_xml.car	NT	wrDb2Volks.VEHI...	740	✓
cardef_xml.engine.cylinders	RT	cardef_xml.engin...	740	✓
cardef_xml.engine	SYN	cardef_xml.car.en...	740	✓
wrDb2Volks.MOTOR	BT	cardef_xml.car.en...	740	✓
cardef_xml.engine	NT	wrDb2Volks.MOT...	740	✓
cardef_xml.dimensions	SYN	cardef_xml.car.di...	740	✓
cardef_xml.car.price	SYN	wrDb2Volks.MOD...	740	☐
cardef_xml.performance.combine...	SYN	cardef_xml.perfor...	740	✓
wrDb2Volks.FUEL_CONSUMPTI...	SYN	cardef_xml.perfor...	740	✓
cardef_xml.performance.combine...	SYN	wrDb2Volks.FUE...	740	☐
cardef_xml.performance.combine...	SYN	wrDb2Volks.FUE...	740	☐
wrDb2Volks.FUEL_CONSUMPTI...	SYN	cardef_xml.perfor...	740	✓
cardef_xml.performance.urban_c...	SYN	wrDb2Volks.FUE...	740	☐
cardef_xml.performance.urban_c...	SYN	wrDb2Volks.FUE...	740	☐

Figure 7. An example of the common thesaurus, which is a set of terminological intensional and extensional relationships that describe intra- and interschema knowledge about classes and attributes of the source schemas.

relationships by analyzing different source schemas according to the WordNet-supplied ontology. You can supply intensional and extensional relationships to capture specific domain knowledge about the source schemata, such as in the Volkswagen source data in which the model entity is a specialization of the vehicle entity. You cannot extract this relationship automatically using both the lexical and the structural approaches. Because of this we supplied the following relationship: <VW.Model NT fiat.car>.

This is a crucial operation because the new relationships are required to belong to the common thesaurus and will be used to generate the global virtual view. This means that if you insert a nonsense or incorrect relationship, the subsequent integration process can produce an incorrect global schema. Our system helps you detect incorrect relationships by performing a validation step with ODB-Tools. Validation is based on the compatibility of domains associated with attributes. In this way, you can determine valid and invalid terminological relationships. In particular, let $a_t = <n_t, d_t>$ and $a_q = <n_q, d_q>$ be two attributes with a name and a domain, respectively. Momis executes the following checks on terminological relationships defined for attribute names in the thesaurus:

- $<n_t$ SYN $n_q>$: The relationship is valid if d_t and d_q are equivalent or if one is a specialization of the other.
- $< n_t$ BT $n_q>$: The relationship is valid if d_t contains or is equivalent to d_q.
- $< n_t$ NT $n_q>$: The relationship is valid if d_t is contained in or is equivalent to d_q.

When you define an attribute domain $d_t(d_q)$ using the union constructor, Momis recognizes a valid relationship if at least one domain $d_t(d_q)$ is compatible with $d_q(d_t)$. Momis infers intensional and extensional new relationships by exploiting ODL's inference capabilities. In the sample domain schema, ODL infers the following relationships:

<VW.Model RT fiat.dimensions>
<VW.Model NT fiat.engine>
<VW.motor NT fiat.car>

Momis adds these relationships to the common thesaurus (see Figure 7) and considers them in the subsequent phase of constructing global schema. Terminological relationships defined in each step carry over into the intensional level by definition. Furthermore, in each of the above steps, you can strengthen a terminological relationship SYN, BT, and NT between two classes C_1 and C_2 by establishing that they hold up at the extensional level.

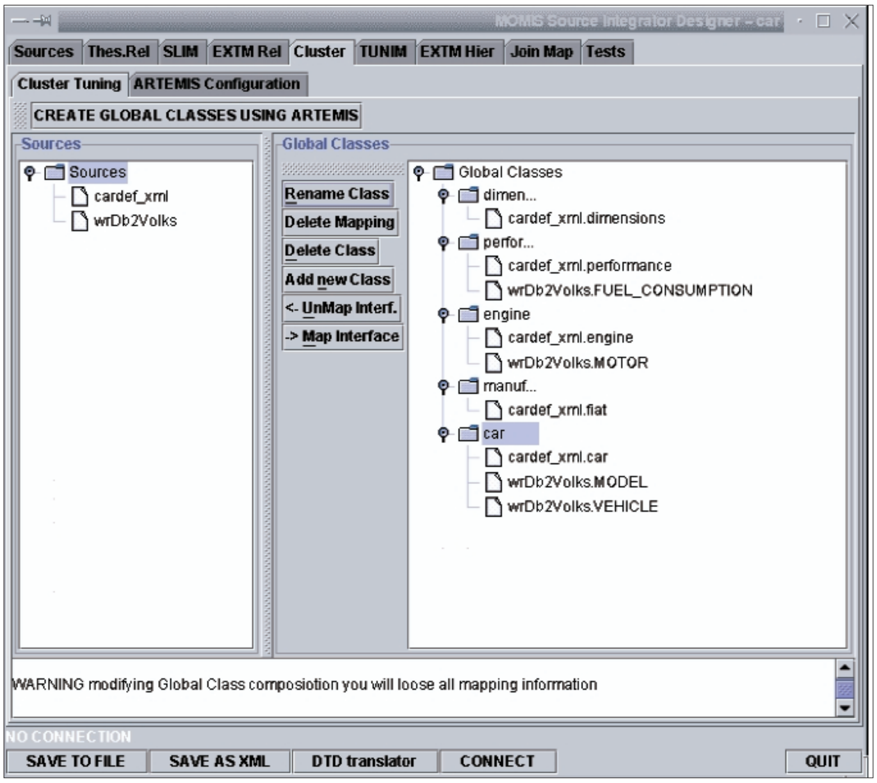
Specifying extensional relationships enables subsumption computation and consistency checking between the two classes C_1 and C_2. The inferred relationships shown in this article are simple. You can compute them using an inheritance mechanism. On the other hand, you need description logic inferences to compute subsumption and to check consistency in more complex schemata and in the presence of extensional interschema relationships.

Clustering ODL$_{I3}$ classes

Providing an integrated representation of heterogeneous information requires that you determine whether the source schemas contain semantically related ODL$_{I3}$ classes. We exploit the knowledge in the common thesaurus to assess the level of semantic affinity between ODL$_{I3}$ classes. For this purpose, Momis evaluates a set of affinity coefficients (numerical values in the range [0, 1]) for all possible pairs of ODL$_{I3}$ classes based on the intensional relationships derived from the common thesaurus. Momis assigns a strength to each type of intensional relationship.

Affinity coefficients determine the degree of semantic relationship between two classes based on their names (*name affinity coefficient*) and their attributes (*structural affinity coefficient*). Two classes have *name affinity* if they are connected through a path in the common thesaurus: Momis obtains their level of affinity as the product of the intensional relationship strengths that connect the two classes involved. We define *structural affinity* between two classes by using Dice's function, which returns an affinity value in the range [0, 1] proportional to the number of attributes that have affinity in the considered classes. A comprehensive affinity value, called a *global affinity coefficient*, is the linear combination of the name and structural affinity coefficients.

In Momis, a hierarchical cluster algorithm uses global affinity coefficients to classify ODL$_{I3}$ classes according to their degree of affinity. The output of the clustering procedure is an affinity tree, where the classes themselves are the leaves and intermediate nodes have an associated affinity value that helps define the classes in the corresponding cluster. The Artemis tool environment helps perform the affinity-based evaluation and clustering procedures.[3]



WARNING modifying Global Class composiotion you will loose all mapping information

Figure 8. The Artemis module calculates the clusters that compose the Global Virtual View. Through a graphical interface, you can improve the set of proposed clusters.

Global class and mapping tables

Starting from clusters generated at the previous stage, we define a global class for each cluster that represents the mediated view of all the cluster's classes. For each global class, Momis provides a set of global attributes and, for each of these attributes, the intensional mappings of the local attributes—that is, the attributes of the local classes belonging to the cluster. Momis obtains the global attributes in two steps: Union of the attributes of all the classes belonging to the cluster, and fusion of the similar attributes. In the second step, Momis eliminates redundancies semi–automatically by taking into account the relationships stored in the common thesaurus.

For each global class, Momis generates a persistent mapping table that stores all the intensional mappings. The table's columns represent the set of the local classes that belong to the cluster; its rows represent the global attributes. An element $MT[L][ag]$ represents how Momis maps the global attribute *ag* into the local class *L*. Each element $MT[L][ag]$ of the table can assume one of the following values:

- $MT[L][ag] = al$: The global attribute *ag* maps into the *al* local attribute.
- $MT[L][ag] = al_1$ `and` al_2 `and` … `and` al_n: We use this formula when the value of the *ag* attribute is the concatenation of the values assumed by a set of attributes *al* that belong to the same local class *L*.
- $MT[L][ag]$ = `case of` *al* `const`$_1$: al_1, `const`$_n$: al_n: Momis uses this formula when the *ag* global attribute can assume one value in a set of al_i belonging to the same local class *L* and when the value choice depends on a third attribute *al* from the same class.
- $MT[L][ag] = const$: In this case, a global attribute value does not refer to any local attribute and a constant value is set by the designer.
- $MT[L][ag] = null$: In this case, no attribute of the class *L* corresponds to the global attribute *ag*.

SI-Designer framework

As described above, the integration process consists of various steps actually implemented in separate modules. SI-Designer is a framework that helps unify the overall integration process. SI-Designer gives you a graphical interface to obtain the global virtual view, relating to each integration step a specific interaction with a software module. All the modules involved are available as Corba objects and interact using established IDL interfaces.

In particular, SI-Designer performs the following steps:

- In the source acquisition phase, the user selects the sources to be integrated. A wrapper performs the translation from the source description model into the ODL_{I3} description model.
- In the intensional relationship definition phase, Momis embeds new relationships into the common thesaurus. These new relationships are either schema derived (by interacting with ODB-Tools system), lexicon derived (by interacting with the WordNet lexical database), or designer supplied (by being added manually by the designer).
- In the extensional relationship definition phase, Momis defines the relationships by interacting with the integration designer. Momis exploits these relationships to detect extensionally overlapping classes.
- In the clustering phase, based on the knowledge stored in the common thesaurus, Momis creates the global classes. The designer may dynamically refine the Artemis coefficients to improve the clusters's set. In our example (see Figure 8), we obtain a cluster including car data contained in the sources and a cluster for the motor and engine information.
- In the mapping table tuning phase, the user can modify or add attributes for each global class generated in the previous phase.

In the final step of the integration process, you export the global virtual view into an XML DTD by adding the appropriate XML tags to represent the mapping table relationships. As mentioned above, by using XML in defining the global virtual view, you can then use Momis infrastructure with other open integration information systems. In addition, Momis translates the common thesaurus into an XML file so you have a shared ontology that different semantic ontology languages can use.

Related research

Researchers have developed several projects based on a mediator architecture for heterogeneous information integration. The most popular, the Tsimmis project,[6] follows a structural approach and uses a self-describing model to represent heterogeneous data sources. The project uses the Mediator Specification Language to enforce source integration and uses pattern-matching techniques to perform a predefined set of queries based on a query template. You can only execute predefined queries in Tsimmis, which means you must perform a source modification manually to change the mediator rules.

The Garlic project[7] builds on a wrapper architecture to describe the local source data with an OO language, while the Sims project[8] proposes to create a global schema definition by exploiting the use of description logics (specifically, the Loom language) for describing the information sources. The use of a global schema allows the Garlic and Sims projects to support all possible user queries instead of a predefined subset.

In the final step of the integration process, you export the global virtual view into an XML DTD by adding the appropriate XML tags to represent the mapping table relationships.

The Information Manifold system[9] provides a source-independent and query-independent mediator. Its input schema is a set of descriptions of the sources. Given a query, the system creates a plan for answering the query using the underlying source descriptions. The Information Manifold project integrates algorithms to determine useful information sources and to generate the query plan. The Information Manifold system largely requires that the designer define the integrated schema manually.

Another approach based on description logics and ontologies is the Observer system, which supports semantic interoperation and formulation of rich queries over distributed information repositories that use different vocabularies.[10] The idea here is that each repository has its own ontology. The Observer system specifies inter-ontology relationships declaratively (using description logics) in an inter-ontology manager module to handle vocabulary heterogeneities for query processing. The focus here is on representation of interontology relationships to solve vocabulary problems at the intensional and extensional levels rather than on using these relationships for deriving an integrated virtual view of the underlying information sources. In our approach, we try to extract as much information as possible from source descriptions and from WordNet. In this article, we've shown how you can use this information for affinity evaluation and integration purposes.

The analysis, discovery, and representation of interschema properties is another critical aspect of the integration process. The Dike system[11] describes semiautomatic techniques for discovering synonyms, homonyms and object inclusion relationships from database schemas. The Dike system also presents a semiautomatic algorithm for integrating and abstracting database schemes.

We are developing Momis' query manager component with query-optimization and answer-composition functionality based on extensional axioms and integrity constraints.[12] Researchers have studied this approach actively in recent years and know these techniques as query rewriting and query answering. One of the original aspects of the Momis query manager consists of employing components based on description logics to perform semantic optimization steps on both global and local queries to minimize the number of accessed sources and the volume of data to be integrated. We plan to enhance these capabilities.

Another extension we have considered introducing is intelligent and mobile agents in the Momis architecture. Exploiting agents might help improve some of the existing Momis features and add functionality to the system. One method is to introduce intelligent mobile agents to carry out the integration process more efficiently, dynamically and flexibly. Currently, when we add or delete a new source, we have to restart the Momis system. Our aim is to allow source integration at runtime, possibly by exploiting mobile agents that search for interesting new sources or check the request of an administrator for integrating his or her new source. This will help Momis users face the high degree of dynamism of the Internet. Another improvement carried out by the agents could be to broaden the volume of available data and to improve query-processing performance in a scalable environment. The Miks system includes some of these preliminary techniques.[13] ■

Acknowledgments

This article is an extended version of the article "SI-Designer: An Integration Framework for E-Commerce."[14]

References

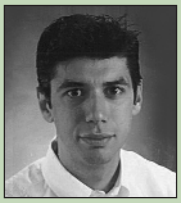
1. D. Beneventano et al., "Information Integration: The Momis Project Demonstration," *Proc. 26th Int'l Conf. on Very Large Databases*, Morgan Kaufmann, New York, Sept. 2000, pp. 611–614.
2. S. Bergamaschi et al., "Semantic Integration of Heterogeneous Information Sources," *J. Data and Knowledge Eng.*, vol. 36, no. 3, March 2001, pp. 215–249.
3. S. Castano et al., "Global viewing of heterogeneous data sources," *IEEE Trans. on Knowledge and Data Eng.*, IEEE Press, Piscataway N.J., 2000.
4. R. Hull et al., "Arpa i^3 Reference Architecture," http://www.isse.gmu.edu/I3_Arch/index.html.
5. P. Buneman, L. Raschid, and J. Ullman, "Mediator Languages: A Proposal for a Standard," *SIGMOD Record*, vol. 26, no. 7, July 1997, pp. 39–44.
6. S. Chawathe et al., "The Tsimmis Project: Integration of Heterogeneous Information Sources," *Proc. 10th Anniv. of IPSJ*, Oct. 1994, pp. 7–18.
7. M.T. Roth and P. Scharz, "Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources," *Proc. 23rd Int'l Conf. on Very Large Databases*, Morgan Kaufmann, New York, Aug. 25, 1997, pp. 266–275.
8. Y. Arens, C.A. Knoblock, and C. Hsu, "Query Processing in the Sims Information Mediator," *Advanced Planning Technology*, May 1996, pp. 61–69.
9. A.Y. Levy, A. Rajaraman, and J.J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions," *Proc. 22nd Int'l Conf. on Very Large Databases*, Morgan Kaufmann, New York, Sept. 3, 1996, pp. 251–262.
10. E. Mena, et al. "Observer: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies," *Distributed and Parallel Databases*, Jan. 2000, vol. 8, no. 2, pp. 223–271.
11. L. Palopoli et al., "Intensional and Extensional Integration and Abstraction of Heterogeneous Databases," *J. of Data and Knowledge Engineering*, vol. 35, no. 3, Dec. 2000, pp. 201–237.
12. D. Beneventano and S. Bergamaschi, "Extensional Knowledge for Semantic Query Optimization in a Mediator Based System," *Proc. Int'l Workshop on Foundations of Models for Information Integration*, Sept. 2001.
13. G. Gelati, F. Guerra, and M. Vincini, "Agents Supporting Information Integration: The Miks Framework," *Proc. AIIA and TABOO Workshop: From Object to Agents*, Pitagora Editrice, Bologna, Sept. 2001.
14. I. Benetti et al., "SI-Designer: An Integration Framework for E-Commerce," *Proc. IJCAI01 Workshop on E-Business and the Intelligent Web*, 2001, pp. 97–104.

The Authors

Ilario Benetti is a PhD candidate in information engineering in the Dipartimento di Ingegneria dell'Informazione at the Università di Modena e Reggio Emilia. His research interests include information integration from heterogeneous data sources, especially e-commerce applications. He received a Laurea in computer engineering from the University of Modena e Reggio Emilia.Contact him at Università di Modena e Reggio Emilia, Dipartimento di Ingegneria dell'Informazione—Via Vignolese 905, 41100 Modena, Italy; benetti.ilario@unimo.it.

Domenico Beneventano is associate professor in information engineering in the Dipartimento di Ingegneria dell'Informazione at the Università di Modena e Reggio Emilia. His research interests include complex object data models and reasoning techniques applied to databases for knowledge representation and query optimization. He received a PhD in computer engineering from the University of Bologna. Contact him at Università di Modena e Reggio Emilia, Dipartimento di Ingegneria dell'Informazione—Via Vignolese 905, 41100 Modena, Italy; beneventano.domenico@unimo.it.

Sonia Bergamaschi is a professor of databases in the Dipartimento di Ingegneria dell'Informazione at the Università di Modena e Reggio Emilia. Her research interests include intelligent integration of information, knowledge representation and management in the context of very large databases that face theoretical and implementation problems. She is a member of the IEEE Computer Society and the ACM. Contact her at Università di Modena e Reggio Emilia, Dipartimento di Ingegneria dell'Informazione—Via Vignolese 905, 41100 Modena, Italy; bergamaschi.sonia@unimo.it.



Francesco Guerra is a PhD candidate in information engineering in the Dipartimento di Ingegneria dell'Informazione at the Università di Modena e Reggio Emilia. His research interests include intelligent integration of heterogeneous information system, especially Web applications. He received a Laurea in computer engineering from the University of Modena e Reggio Emilia. Contact him at Università di Modena e Reggio Emilia, Dipartimento di Ingegneria dell'Informazione—Via Vignolese 905, 41100 Modena, Italy; guerra.francesco@unimo.it.

Maurizio Vincini is a research associate in information engineering in the Dipartimento di Ingegneria dell'Informazione at the Università di Modena e Reggio Emilia. His research interests include the intelligent integration of information system based on reasoning tecniques, object oriented database design, and query optimization. He received a PhD in computer engineering from the University of Modena and Reggio Emilia. Contact him at Università di Modena e Reggio Emilia, Dipartimento di Ingegneria dell'Informazione—Via Vignolese 905, 41100 Modena, Italy; vincini.maurizio@unimo.it.