An Efficient and Adaptive Framework to Access Heterogeneous Health Information Sources

Shokooh Kermanshahani, Hamid Reza Hamidi

Computer Engineering Department, Imam Khomeini International University, Qazvin,IRAN

***Abstract*—In health informatics, the need for a consistent and integrated view of distributed and heterogeneous information sources is inevitable. Healthcare, medical education and research could benefit of integrating medical information of patients or about a disease, a treatment or side effects of a drug. This article proposes a flexible incremental update method for the materialized part of the integration system. It permits us to manage the integration system according to the characteristics of the data sources, which can be changed. We present a hybrid data integration approach.In this approach the materialized part of the system in mediator is the object indexation structure based on an instance classification of the sources objects which correspond to the global schema. The object identifier of each object in the indexation structure is materialized together with the attributes which are needed for the incremental updating of this indexation (classifying attributes).**

***Keywords*—DataBase Integration, Data Warehouse and repository,Schema and Subschema, System Integration.**

I. INTRODUCTION

IN a data integration system, data from different sources are integrated into a global integrated schema which satisfies the needs of users and is managed by a centralized system. All heterogeneities are hidden from the user, who queries the global schema as a single database schema [1]. One of the most important challenges for integrating different autonomous data sources is the heterogeneity which can appear at different levels. The hardware on which two information sources are developed, the network protocols, the software, the data and the query languages may be different. However, the essential and more complicated aspect of heterogeneity is semantic heterogeneity. Semantic heterogeneity characterizes the differences in signification, interpretation or utilization of the same data [14].

Kermanshahani S.is with the Computer Engineering Department, Imam Khomeini International University, Qazvin,IRAN (corresponding author, phone: 0098-28-33901287; fax: 0098-28-33780084; e-mail: kermanshahani@eng.ikiu.ac.ir).

Hamidi H.R.is with the Computer Engineering Department, Imam Khomeini International University, Qazvin, IRAN (e-mail: hamidreza.hamidi@eng.ikiu.ac.ir).

A. Motivation Application

One of the most important challenges in health informatics is to have a consistent integrated view of information about each person, either patient or expert. During their life, people refer to different medical centers in different times. By the way, medical centers collect several distributed but inter-related data. Extract useful information from these data is an essential need of a physician while consulting a patient. With increasing growth in the use of computers, today most medical centers including hospitals, clinics, analyzing laboratories, etc., have their own autonomous sources such as documents, data sources and database systems with a large spectrum of heterogeneity from hardware to conceptual and logical data schemas. Health informatics need to integrate these sources to create a unified view and handle unified data [14], [21].

In addition, integrating medical information itself is an essential step towards providing the level of personalization required in the next generation of healthcare provision and in order to provide the computer-based decision support systems for medical uses [21]. Finding an efficient method to extract useful information is an essential challenge of health informatics, in research as well as in practice.

The main idea of this paper is to develop a hybrid data integration framework, which represents a new aspect of a hybrid method focusing on flexible data refreshing. First,we review the literature of data integration; then we present our method in details. Finally, we conclude this articleand figure out some further research directions.

II. DATA INTEGRATION

To integrate multiple data sources, we needto design a universal (global) schema and an integration mechanism. In this section we present the current knowledge including theoretical and methodological contributions to data integration.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

A. Schema Mapping

To query the data of several local sources via a global schema, this global schema has to relate to the schemas of the local sources. Different methods have been developed to map several local schemas to a global schema: Global-As-View (GAV), Local-As-View (LAV) and Both-As-View are the most important ones. In a GAV mapping approach, the global schema is defined as a view of the underlying source schemas. In a LAV mapping, the schema of each local source is defined as a view of the global schema, and in a BAV mapping, these two methods are combined [2]–[4]. In the BAV method we can obtain a local schema from the relations of the global schema and vice-versa.

B. Fully Materialized Data Integration

The global schema of a data integration system can be fully materialized. It means that a new repository is developed by a data management system and a copy of all the data which correspond to the global schema is saved in this repository. Data from local sources are Extracted, Transformed and Loaded (ETL process) to this repository [5]–[8]. Query evaluation in such an approach is similar to that of a single database and we have access to a powerful query language and to query optimization. However, in such an approach, the data of local sources cannot be directly accessed and the data repository has to be periodically updated. Therefore, there is always a delay to access the last updated data. In addition, building a new repository and ETL processes are expensive.

C. Fully Virtual Data Integration

The global schema of a data integration system may be virtual [9]–[11]. In this case, all data remains in the local sources and a middleware containing the global schema is developed. This middleware, generally called a *mediator*, decomposes or reformulates a user query over a set of local sources. It then recomposes the partial responses into a single response for the user. Data and query languages of local sources may be different from those of the middleware.

Wrappers which contain schema mappings, translate data and queries between the sources and the mediator [12]. With this approach, users have online access to the data of the local sources, but query processing is complicated and time consuming.

D. Hybrid Data Integration

A third possible solution for data integration is to develop a partially materialized integrated schema. A data integration approach which uses such a global schema is called a Hybrid approach [13]. Fully materialized and fully virtual data integration approaches obey to different priorities. In a fully materialized approach, the main priority is the query response time, and in fully virtual data integration, data freshness is more important. However, in many data integration scenarios different priorities may be associated with different data, and a trade-offbetween query response time and data freshness may be preferred to satisfying only one of these two issues. A flexible approach which permits some data to be materialized and other data to be virtual can satisfy both of these goals. In the existing hybrid approaches the global view is partitioned into materialized and virtual parts. Some objects or relations are chosen to be materialized and others reside in the local sources and will be extracted at query time [14].

E. Efficient Query Processing

Efficient query processing is one of the most important challenges in data integration. Because of the dynamic nature of the data integration context, new challenges arise to optimize the processing of queries. At the mediator level, the optimizer may not have enough information to decide on a good plan and in addition, at execution time, a source may not respond exactly as it had been considered at optimization time [15].

In a data integration system, two level of query optimization can be considered: global optimization for query plan and local optimization for subqueries that retrieve data from individual data sources [16]. The restriction of the search space for a query, using semantic knowledge at the mediator level, is a solution to provide global optimization for query processing.

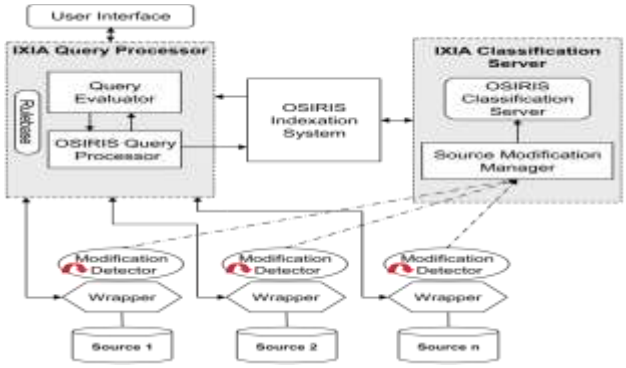

Fig. 1. IXIA data integration architecture [17].

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

III. An Adaptive Data Refreshing Approach

The existing hybrid approaches provide a rapid access to the materialized data. Other data remain in the local sources and are queried directly from the sources when necessary. As a consequence, only the queries to the materialized part of the system are optimized. A typical example of integration scenarios compatible with such approaches is the integration of geographical data, hotel and tourism information and

weather information for a travel agency. In this example, the data of geographical and tourism centers are stable and can be materialized while other information such as weather data change more frequently and are integrated in a virtual manner [22]. Many other data integration scenarios can profit from the trade-off that a hybrid approach offers between query response time and data freshness.

In the next section, we review the architecture of IXIA an Index-based data Integration Approach [17]. It provides a query optimization to the integration system. Then we demonstrate the flexibility of data refreshing, according to the needs of the integration applications.

A. The Architecture

Like a mediator approach, IXIA has a mediator-wrapper architecture, although with some materialization. It has been developed based on the Osiris system in order to take advantage of its object indexation system. Osiris is an object-based database and knowledge base system based on a hierarchy of views where views are similar to concepts defined by logical properties [18], like in a Description Logic approach [19]. Fig. 1 shows a presentation of the IXIA architecture. We briefly describe how it works.

The Osiris system implements the P-type data model with two principal objectives: data sharing through the views and automatic verification of integrity constraints. An object is an instance of one and only one P-type, but it can belong to several of its views. Also an object can change the views it belongs to during its lifetime. Classifying an object into the views of its P-type is a characteristic inherent to this model. This is why the Osiris system, which implements the P-type model, can offer functionalities for decision support, alert management, semantic query optimization, etc. In our work, we profit from its object indexation system.

As mentioned above, the main materialized part of the IXIA is the indexation structure which is based on the instance classification of Osiris. A direct advantage of this materialization is query optimization for the integration system.

After defining the integrated schema (an Osiris schema), the classification server makes a first object indexation for all the sources objects which correspond to the global schema and sends the indexation data to the Osiris indexation module.

The indexation data are then incrementally updated by the classification server. The "Modification Detector" modules detect if there is some updating in the sources which results in updating the indexation data from the last indexation maintenance.

The "Modification Detector" of each source functions independently and can be executed with different frequencies.

Updating information obtained from the modification detectors is sent to the "Source Modification Manager" module of the IXIA classification server. This module adds the source information and prepares the "indexation repairing message" for the "Osiris Classification Server", which does the indexation maintenance just as in a single Osiris database.

We note that mappings between the object indexation and data in local sources are made in the wrappers. We save the *(oid, primary-key)* correspondence between the Osiris objects of the global schema and the data in sources. Wrappers also do the mapping between the local sources' schemas and the Osiris global schema.

B. The Modules Design

Fig. 2 shows a detailed architecture of IXIA. This framework is an extension of the Osiris database system in order to develop a platform for data integration. Consequently, the architecture of IXIA consists of some modules of Osiris, some extended modules and some new modules. The following subsections describe these modules.

1) Indexing Structure Descriptor

The Indexation Structure in IXIA is much similar to that of Osiris. The only difference is that we added the sources information to the indexation structure at the mediator level in order to make them unique. Adding sources information to the indexation structure can be done in different ways. In our implementation we add a source field to the object identifier (*Oid)* in the indexation module. Object identifiers at the wrappers level are identified through a simple *Oid*, which is

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

unique in the wrapper of each source. We call it a Local *Oid(Loid)*. The object identifier in the Indexing Structure Descriptor (ISD) space is:

Goid = Loid + Source-ID (Goid : Global Oid)

2) Wrappers

A wrapper in this architecture, like in all mediator architectures, contains the mapping between a source schema and the integrated schema. However, in this approach wrappers contain the assignments between object identifiers *Oid* in the global schema and the primary-keys in the data sources. In other words, the mapping in this approach, is based on the *(Oid, primary-key)* correspondence. A study for the mapping of a relational database schema to an Osiris target schema had been done in [20]. In that study, after defining the identifiers assignments, each attribute of the target schema is mapped to one attribute or a function of several attributes of the source schema. Also the maintenance of the wrappers when adding, deleting or modifying an object was not considered. The "Modification Detector" of IXIA deals with this task (see next section).

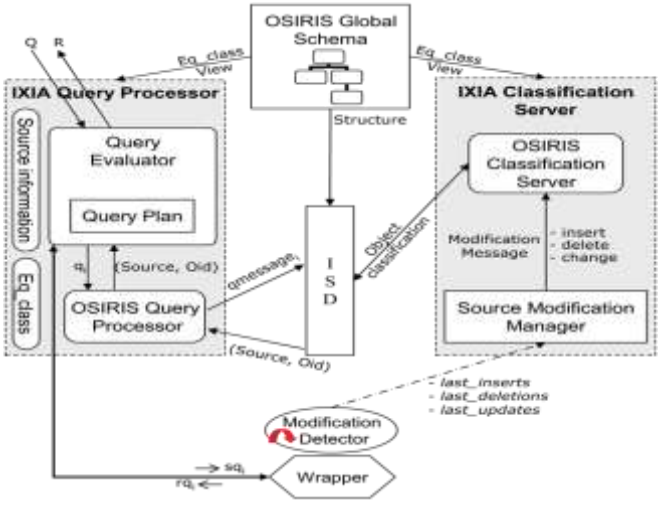

Fig. 2. IXIA modules interactions.

Since the mapping between the global schema and source schemas is made by a mapping between the identifiers, adding a new source is independent from the other sources and a novel source can be integrated as the new wrapper be developed. However, some dependencies may exist between data of different local sources, the wrappers develop independently and the dependencies are managed by IXIA Query Evaluator in query processing.

3) Modification Detector

The object indexation information must be refreshed periodically. In order to maintain the indexation data, the modifications of data sources have to be detected. A module called 'Modification Detector" (MD) has been developed for this purpose. A MD module is developed for each source independently from other MDs and the detection process can be executed at different frequencies for different sources. If the classifying attributes of a P-type of the global schema are in two different sources, then to reclassify the objects of this P-type, the corresponding attributes of both sources must be considered. In such case, the MDs remain independent and this question will be dealt with in the "Source Modification Manager". To simplify and according to our motivation application, in the current implementation we consider the situation where all classifying attributes of an object are in a same source. In such case when a real world object is in two different sources, we will create two objects at the mediator level, each for one source, and by joint the source codes to *Oid*s, they will be distinguished. Such implementation permits us to verify the consistency of data in different sources.

```
01. class SourceModificationManager
02. {
03.   OSIRIScServer OSIRIS_cServer     // OSIRIS classification server identifier
04.   SourceModificationManager (OSIRIScServer t){     //constructor
05.       OSIRIS_cServer:=t
06.   }
07.   inserts (ModificationDetector MD, Wrapper wrapper)
08.   { //---- classification of last inserted
09.     Buffer tmp_inserts=new MD.last_inserts     //dump last inserts
10.     MD.I_dumped:=true
11.     while (tmp_inserts has more object) {
12.         extract a local object from tmp_inserts
13.         create a global object based on local object and wrapper
14.         OSIRIS_cServer.insert(global object)
15.     }
16.   }
17.   // ... same implementation for "deletions" and "updates" methods considering:
18.   // ... in line 13. find a global object instead of create a global object
19. } //SourceModificationManager
```

Fig. 3. Java-Like pseudocode of the source modification manager.

Studying the data consistency verification can be considered as a future work. In the current framework, the MD module has been implemented on top of each wrapper and together they make a single component with two parts. Thus it can be used to add or delete the *Oid/Primary-key* mapping in the wrapper. We explain the details of the "Modification

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Detector" and its implementation through the description of the indexation maintenance functionality.

4) Classification Server

The "Classification Server" of IXIA consists of "Osiris Classification Server" (OCS) and "Source Modification Manager" (SMM). It is responsible for object classification in the IXIA mediator. The SMM receives the modification of each source from the MD and prepares a message containing object insertions, object deletions or updated objects to the "Osiris Classification Server". It also adds the source information to the message (it creates a *Goid* from the *Loid* received from the MD). The Osiris classification server receives the modification message and makes the classification updates. Fig. 3 presents a pseudo-code for the Source Modification Manager.

5) Query Processor

Like all query processors in data integration systems, the IXIA query processor receives the user query, performs query decomposition / reformulation and then recomposes partial responses in order to send a response to the user. The IXIA query processor contains two main modules which we describe below: "Osiris Query Processor" and "Query Evaluator".

"**Osiris Query Processor**": This module works as in a single Osiris database. For each partial query, received from the query evaluator, it searches in the ISD Space the objects' *Oids* which satisfy or potentially satisfy that query. However, in an Osiris database, after defining these *Oids*, the query processor does the verification of the complementary conditions and extracts the requested attributes. In IXIA, since data remain in the sources, this second part is not done by the Osiris Query Processor.

"**Query Evaluator**": All the tasks which correspond to the data integration query processing are done by this module. It generates a query plan for the user query. Following this logical query plan, the query evaluator sends partial queries to the Osiris query processor and retrieves the partial response which are the Global *Oids* (*Goids*). It is also responsible for preparing the partial queries to the sources and combining the partial responses in order to provide a final response for the user. We describe the functionality of this module where we review the query processing.

IV. The Functionalities

We continue to present our approach by describing its three main functionalities.

A. Initialization Phase

When developing an integration application using IXIA, the integrated schema is described by an Osiris conceptual schema. In the first step all the Eq-class classes (i.e., collections of SSDs) will be extracted from the existing assertions in the P-types. The structure of indexation is defined by these Eq-classes together with the views of the P-type. In the initialization phase, the classifying attributes for all the objects in the sources which correspond to the integrated schema are extracted and the objects are classified in the indexation table. The extraction of these data is made by the "Modification Detector". In this case all extracted data will be considered as new insertions by the MD and will be sent to the "Source Modification Manager" (SMM), which in turn sends the insert messages to the "Osiris Classification Server".

B. Indexation Maintenance

The Indexation maintenance processing is done by an incremental update and when some data in one or several sources is modified, the object indexation update becomes necessary. Generally, in materialized approaches, incremental update (compared with static update) is best applied in conditions where changes on the data are visibly smaller than the size of the data set in a certain period of time. Here, the data which we are interested in updating (classifying attributes) are significantly smaller than the size of data which participate in integration scenario. The modification at the object level can be:

1 Delete an existing object.
2 Add a new object.
3 Modify the value of an attribute.

Deleting an object from a source consists of deleting its corresponding *Oid* from the ISD space in the mediator. If this object is the only object in an ISD, this ISD (one line of ISD table) is also deleted. When adding a new object or modifying an attribute value in the sources, a classification process may be necessary. It is important to note that all the new entries in a local source do not correspond to the creation of a novel object in the Osiris schema. For example, in a relational data source, only the insertion or deletion of the tables whose primary-keys are mapped to an *Oid* of the global schema, imply creating a novel object or deleting an existing object.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Thus only the classifying attributes modifications which cause a SSD change are important for the indexation system. On the other hand, all of the attribute modifications do not require an indexation modification.

The most complex problem is to find an approach to distinguish the important modifications in local sources in an optimal way in time and feasibility. The challenge is that the sources are autonomous and that often there are no direct accesses to the structure of the sources to distinguish if there have been modifications (for example by a trigger). The Modification Detector has been developed for this purpose.

The Modification Detector (MD) over each source detects new objects, deleted objects and modifications of classifying attributes. This detection is made independently and in predefined periods for each source. A MD is linked to the classification server and is joined to the wrapper of its corresponding source.

The data extracting process makes use of the mapping information in the wrappers. When detecting modifications, it informs the Source Modification Manager (SMM), which adds the source information and sends the corresponding message (insert, delete or change) to the Osiris Classification Server. If the modification is an “insert", a Local Oid (*Loid*) is assigned to the novel object at the wrapper level and is joint to the ID of the corresponding source, in order to create the Global Oid (*Goid*) and classify it in the ISD Space of the mediator. This last task is done by Osiris Classification Server.

We propose three detecting technologies for the Modification Detector; log system, database trigger and polling strategy. All or some of these technologies can be used in an integration application depending on the sources' characteristics and the application needs.

C. Query Processing

Like in Osiris, the query processing in IXIA is made by using the indexation system. However, there are some important differences. The first difference is that in IXIA there is not any access to object attributes at the mediator level. Thus for all complementary verifications and to extract the requested attributes, data sources must be queried.

The second difference arises when the condition part of a query corresponds to more than one P-type. To process such query, it must be decomposed into single Osiris queries, each corresponding to a P-type. In an Osiris database, to answer a query, when attributes of several P-types are needed, Osiris searches the valid and potential response *Oids* for a first P-type. For potential *Oids*, it makes the necessary verifications and then extracts the requested attributes. This is because often the value of some attributes of the first P-type may be needed for querying the second part of the query (which corresponds to the second P-type). In IXIA one does not have access to the value of attributes at the mediator level.

A user query in IXIA is an Osiris query that the general form of it is:

(Context | conditions) [attributes]

Two scenarios are possible for this query:

1 Both context and conditions correspond to a single P-type of the global schema. The query is sent to the Osiris Query Processor in order to extract the response Oids.

2 The conditions of the user query correspond to more than one P-type of the global schema (two P-types, for example). In this case, the user query must be decomposed into several Osiris queries (each corresponding to a P-type). This decomposition is necessary because when a query corresponds to more than one P-type, we may need the value of one or several attributes in one P-type in order to query other attributes in another P-type. In the context of data integration, attributes are in different sources. Thus, often a part of the query may need the result of another part.

For each Osiris partial query, the Osiris Query Processor (OQP) searches the valid and potential *Oids*, which are Global Oids (*Goid*). The Query Evaluator prepares the source partial queries and sends them to the wrappers to verify all the complementary conditions and extract attributes. In this process, before sending partial queries to the sources, *Goids* are decoded in order to obtain the *Loids*and the corresponding sources. We note that some attributes found in the materialized part of the Modification Detector may be used in the query processing.

The partial responses will be recomposed by the IXIA query evaluator into an Osiris Response (a partial Osiris query). The final response for the user is prepared after

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

receiving all the partial Osiris responses. In IXIA we then use the Osiris Query Processor only in single P-type query option.

The algorithm which must be followed by the IXIA query processor in order to make the decomposition, evaluation and re-composition of a user query is generated by the query evaluator using the sources information. It is memorized in the Query Plans module until the end of the processing of a user query.

D. Experimental Results

To test the cost of query processing, we measured it in a local area network. Two local data sources are considered for data integration. The cost of Osiris and IXIA query processing are measured. As you can see in Fig 4, the query distribution and the aggregation of responsesare less than 3%.

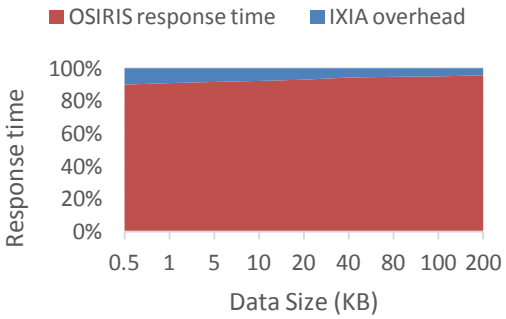

Fig. 4. Query processing costs of integration.

V. CONCLUSION

In all the hybrid data integration approaches that we have studied, a horizontal hybrid method is used. This means that in the integrated schema some objects or relations are implemented virtually and others by a materialized method. All data manipulation in the virtual parts is made in a virtual manner and data of the materialized views are manipulated by materialized paradigms. A user query in such approaches may correspond to materialized or virtual parts or it can be decomposed between virtual and materialized parts. Consequently, the performance of the query processing as well as the data accessing delay are similar to those of fully materialized or fully virtual approaches respectively for materialized and virtual parts of the data integration system.

The hybrid approach that we propose in this paper, implements a vertical hybrid approach. It means that at the mediator level, some data of each object are materialized and others are virtual. The attributes of the objects remain in the local sources and generally data are extracted from the sources at query time. The object identifier of each object in an indexation structure is materialized together with the attributes which are needed for the refreshing of this indexation. We call this kind of partition of the integration system to the materialized and virtual parts a vertical partition, and we name our approach a vertical hybrid approach.

We proposed a data refreshing solution. The modification detector is the core of this solution. It offers:

1. Data refreshing for different sources, hence for different data can be done in different time periods. This flexibility permits us to consider the characteristics of different sources in each data integration application.
    a. The rate of data updating.
    b. The importance of the data.
    c. The querying frequency.
    d. The query power.
    e. The availability of the source.
2. The arrangement of refreshing for different data sources can be changed by changing the frequencies of different MDs, which is a decision of the system administrator.

3. Adding a novel source to the system is independent from other sources. Because of its special mapping system (object-to-object), when defining a wrapper for a novel source, it can be integrated to the system. The novel source information, however, must be added to the rule base of the query evaluator and the Source Modification Manager

There are some restrictions and difficulties in IXIA architecture. In the wrappers we need two levels of mapping: schema mapping and *(Oid, primarykey)* mapping. This second kind needs to be updated each time an object is deleted or inserted in a source. When an object is inserted an *Oid* must be created and assigned to the primary-key. However, this process can also be done by the Modification Detector in order to be optimized.

Finally, it must be considered that like all the approaches which use some materialization in their architecture, we often do not have access to on-line data and generally there is a slight delay in receiving updated data. Therefore, this approach cannot be applied to applications in which on-line querying is crucial. IXIA cannot either be used if there are sources which do not give access to the primary-key of data.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

REFERENCES

[1] Doan, A., Halevy, A., & Ives, Z. (2012). "The Future of Data Integration. Principles of Data Integration", 453–457. doi:10.1016/b978-0-12-416044-6.00019-3.

[2] Vassalos, V., & Papakonstantinou, Y. (2000). "Expressive capabilities description languages and query rewriting algorithms". Logic Programming, 43(1), 75–122. doi:10.1016/s0743-1066(99)00026-6.

[3] SLi, C., Yerneni, R., Vassalos, V., Garcia-Molina, H., Papakonstantinou, Y., Ullman, J., & Valiveti, M. (1998). "Capability based mediation in TSIMMIS. ACM SIGMOD Record", 27(2), 564–566. doi:10.1145/276305.276382.

[4] Boyd, M., Kittivoravitkul, S., Lazanitis, C., McBrien, P., & Rizopoulos, N. (2004). "AutoMed: A BAV Data Integration System for Heterogeneous Data Sources". Active Flow and Combustion Control 2018, 82–97. doi:10.1007/978-3-540-25975-6_8.

[5] Thakur, G., & Gosain, A. (2011). "A comprehensive analysis of materialized views in a data warehouse environment". IJACSA) International Journal of Advanced Computer Science and Applications, 2(5).

[6] Richard Hull and Gang Zhou, "A framework for supporting data integration using the materialized and virtual approaches", pp. 481-492, 1996. doi: 10.1145/235968.233365.

[7] A. Gupta, H. V. Jagadish, I. S. Mumick, "Data integration using self-maintainable views", in : In Proceedings of International Conference on Extending Database Technology (EDBT, 1996, pp. 140-144.

[8] Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P., "Fundamentals of Data Warehouses", 2nd Edition, Springer, ISBN 978-3-662-05153-5, 2003.

[9] F. Afrati, R. Chirkova, H.V. Jagadish (2017), "Answering Queries Using Views", Morgan & Claypool Publishers, ISBN: 978-1681730318.

[10] Abiteboul, S., Arenas, M., Barceló, P., Bienvenu, M., Calvanese, D., David, C., Hull, R., Hullermeier, E., Kimelfeld, B., Libkin, L. and Martens, W., (2017),"Research directions for principles of data management". Dagstuhl Manifestos, 7(1), pp.1-29.

[11] Francois Goasdoue Veronique Lattes, Marie-Christine Rousset, "The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System", International Journal of Cooperative Information Systems, V9, N4, pp. 383-401, doi: 10.1142/S0218843000000181.

[12] M. Lenzerini, "Data integration: A theoretical perspective", in: PODS, 2002, pp. 243-246.

[13] Y. Kalfoglou, M. Schorlemmer, "Ontology mapping: The state of the art", The Knowledge Engineering Review 18 (2003) 2003.

[14] Zhang, H., Guo, Y., Li, Q. et al., "An ontology-guided semantic data integration framework to support integrative data analysis of cancer survival", BMC Med Inform Decis Mak (2018) 18(Suppl 2): 41. https://doi.org/10.1186/s12911-018-0636-4

[15] A. Halevy, A. Rajaraman, J. Ordille, "Data integration : the teenage years", in : VLDB '06 : Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment, 2006, pp. 9-16.

[16] C.-N. Hsu, C. A. Knoblock, "Semantic query optimization for query plans of heterogeneous multidatabase systems", IEEE Trans. on Knowl. And Data Eng. 12 (6) (2000) 959{978. doi:http://dx.doi.org/10.1109/69.895804.

[17] S. Kermanshahani, H. R. Hamidi, "A Data Integration Approach Based on Indexation". International Journal of Computer Science and Network Security, VOL.17 No.7, July 2017.

[18] Bassolet, C. G., Simonet, A., & Simonet, M. (n.d.). "Probabilistic classification in Osiris, a view-based OO DBMS and KBMS", Proceedings of 7th International Conference and Workshop on Database and Expert Systems Applications: DEXA 96. doi:10.1109/dexa.1996.558273.

[19] Roger, M., Simonet, A., & Simonet, M. (2002). "Bringing Together Description Logics and Database in an Object Oriented Model", Database and Expert Systems Applications, 504–513. doi:10.1007/3-540-46146-9_50.

[20] E. Gay, "Vues osiris sur une base relationnelle", CNAM, Centre de Grenoble, France (Juin 1999).

[21] Hauer, T., Rogulin, D., Zillner, S., Branson, A., Shamdasani, J., Tsymbal, A. McClatchey, R. (2008). "An Architecture for Semantic Navigation and Reasoning with Patient Data - Experiences of the Health-e-Child Project". The Semantic Web - ISWC 2008, 737–750. doi:10.1007/978-3-540-88564-1_47.

[22] BERGAMASCHI, S., BENEVENTANO, D., GUERRA, F., & VINCINI, M. (2005). BUILDING A TOURISM INFORMATION PROVIDER WITH THE MOMIS SYSTEM. Information Technology & Tourism, 7(3), 221–238. doi:10.3727/109830505774297265

Shokooh KERMANSHAHANI (She was born in 1972, Qazvin, Iran) studied Computer Engineering, B.S. in Esfehan University (1995, Iran), M.S. and Ph.D. in Joseph Fourier University (2003, 2009 France). She is currently assistant professor of computer engineering, Imam-Khomeini International University, Qazvin, Iran.

Hamid Reza HAMIDI (He was born in 1971, Birjand, Iran) received B.S. from Sharif University of Technology (1994, Iran), M.S. from Tehran University (1997, Iran) and Ph.D. from Institute National Polytechnic de Grenoble (2005, France) in Computer Engineering. He is currently assistant professor of computer engineering, Imam-Khomeini International University, Qazvin, Iran.

[1] http://coron.loria.fr/site/downloads_datasets.php.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731