Exchange, Integration, and Consistency of Data. Report on the ARISE/NISR Workshop.

Leopoldo Bertossi (Carleton University), **Jan Chomicki** (University at Buffalo),
Parke Godfrey (York University), **Phokion G. Kolaitis** (IBM Almaden Research Center),
Alex Thomo (University of Victoria), and **Calisto Zuzarte** (IBM CAS, Toronto Lab.)

1 Introduction

The "ARISE/NISR Workshop on Exchange and Integration of Data" was held at the IBM Center for Advanced Studies, Toronto Lab., between October 7-9, 2004.

The Advanced Research Initiative for Software Excellence (ARISE), now referred to as the National Institute for Software Research (NISR), was founded by the National Research Council's Institute for Information Technology, the IBM Toronto Lab., the University of Toronto, the University of Waterloo, and York University, as an effort to create a national institute that will enhance knowledge in the area of software, helping Canada to increase its competitive advantage in the field. NISR's program includes the creation and support of workshops, seminars, courses, and research projects. This first ARISE workshop was organized by Leopoldo Bertossi, Parke Godfrey, Paul Smith (IBM Toronto Lab.), and Calisto Zuzarte; it congregated a large number of researchers. Details and presentations are available at the workshop web site: http://www.scs.carleton.ca/∼diis/arise/workshop.html.

2 Workshop Contents

Kelly Lyons (IBM Toronto Lab. and technical steering committee of NISR) provided the opening remarks. The three keynote presentations, "INFOMIX: Data Integration meets Nonmonotonic Deductive Databases" by Thomas Eiter, "Model Management: Generic Operators for Schema Mappings and Database Integration" by Philip A. Bernstein, and "Hyper: A Framework for P2P Information Integration" by Maurizio Lenzerini, represented well the three areas of research that emerged during the workshop: (a) Virtual Data Integration; (b) Data Exchange and Schema Mappings; and (c) Inconsistency Handling in Databases. It was particularly interesting to see the connections between them. Data exchange and mediator-based data integration share several ideas, concepts, and techniques, but the corresponding communities have stayed relatively distant from each other. Consistency issues naturally arise in both of them.

In data exchange, where data is shipped from a source database in order to populate a target schema, integrity constraints (ICs) imposed at the target level have to be kept satisfied. Instead of restoring the consistency of data at the target *a posteriori*, after the population process, a more appealing alternative takes into account the ICs at the target when the data mappings between the source and the target are being established and/or used. In virtual data integration there is no centralized consistency maintenance mechanism that makes the data in the mediated system satisfy certain global ICs. Again, these ICs have to be captured by the mappings between the sources and the global schema or at query time [5], when global queries are being answered. In the two scenarios, the plans for data transfer or query answering have to deal with potential inconsistencies of data.

Research around the management, mapping, and integration of database schemata is also relevant to both data exchange and integration. It is common that research in these two latter areas starts from given source and target schemas on one side, or source and global schemas on the other. Given those schemas, the problem is to design exchange or query plans. However, there is not much research that addresses the impact of schema design on the latter tasks. This perception was an additional motivation for gathering people from those different areas.

Industrial presentations gave overviews of trends in the area of metadata management for information integration in the data warehousing industry, and of the IBM® DB2® Information Integrator (now called WebSphere® Information Integrator). A hands on demo of aspects of federation, replication and enterprise search features of WebSphere Information Integrator, can be found at http://db2ii2.dfw.ibm.com/wps/myportal/!ut/p/.scr/LoggedIn.[1]

Break-out sessions were run in parallel around each of the three areas mentioned above. Acting as group discussion leaders Alex Thomo (data integration), Phokion Kolaitis (schema-mappings and data exchange), and Jan Chomicki (consistency). The following sections describe relevant research problems and trends that were identified by the working groups.

3 Data Integration

In this session, the main focus was on view-based data integration. The goal of data integration is to provide a uniform interface for querying a collection of disparate and heterogeneous data sources. The two main approaches, namely the *global-as-view* (GAV) and the *local-as-view* (LAV) were overviewed. In both, the user poses queries

[1] DB2, IBM, and WebSphere are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

on a mediated global schema. The main difference lies in the way the data sources are represented. In GAV, each data source exposes a view defined over the local schema, and the view name is an element of the global schema. On the other hand, in LAV, the data sources expose views defined over the global schema. The pros and cons of each approach are as follows. In GAV the query answering is simple: It amounts to view unfolding. However, adding sources to the data integration system is non-trivial. In particular, given a new source, we need to figure out all the ways in which it can be used to obtain tuples for each of the views in the global schema, and this limits the ability of the GAV approach to scale to a large collection of sources. In contrast, in the LAV approach each source is described in isolation. It is the system's task to figure out (at query time) how the sources interact and how their data can be combined. In this case, query answering and query reformulation are harder, and sometimes require recursive queries over the sources.

This session focused mostly on the LAV approach. It was discussed that there is sometimes a confusion about how the answer to a query should be characterized. There are two ways of doing that. The first is syntactical: The query answer is the one that can be computed by evaluating the maximal view-based query rewriting expressed in some fixed language. The second characterization is semantical: We try to find the *certain answers*, that belong to all the answers sets that can be obtained on each database that is consistent with the views (on the global schema). The source of confusion is that these two characterizations coincide in the case of conjunctive queries and views. Thus, they have been often used interchangeably in previous works, but this coincidence is not true for more complex queries and views, such as those containing recursion.

As pointed out in [24], in the context of semistructured data, a natural and quite general fragment of recursive Datalog did emerge in the mid 1990s: The class of regular queries, whose basic element is that of regular path queries. For such queries and views to compute the certain answers is computationally hard. For example, to decide whether a tuple is a certain answer is CoNP-complete with respect to the size of data [10]. On the other hand, given a view-based rewriting expressed in some fixed language (e.g., regular language, or Datalog), computing the answer is in PTIME on the size of the data. Clearly, the answer computed by evaluating the maximal (w.r.t. to some fixed language) view-based rewriting is only a subset of the certain answers, but this is often an acceptable approximation. In the session it was concluded that the best sources of information regarding the rewriting-based answers vs. certain answers, are the seminal papers [9, 10]. Moreover, a recent important achievement was pointed out: In [11] the notion of view-based containment and equivalence for regular path queries was investigated and positively solved. By using the solution in [11], one can test, in compile time, whether a computed view-based rewriting will generate the full certain answers when evaluated on the views.

The session was concluded outlining some new promising directions for further research, among others, the optimization of view-based rewritings and data-integration in P2P systems.

4 Data Exchange and Metadata

This session focused on research issues and directions in data exchange and metadata management. A common thread in both these areas is the systematic use of *schema-mappings*, which are high-level specifications in some logical formalism that describe the relationships between schemas.

Issues in data exchange. Data exchange is the problem of taking data structured under a source schema and translating them into data structured under a target schema. Although there are clear similarities with data integration, the main difference between the two frameworks is that, given a source instance, the goal in data exchange is to actually materialize a target instance such that (i) it satisfies the specifications of the schema-mapping between the source schema and the target; (ii) it reflects the given source data as accurately as possible. The challenges in data exchange arise because typically there are more than one target instances (called *solutions*) that satisfy the specifications of the schema-mapping. This state of affairs raises both semantical issues and algorithmic issues in data exchange: Given a source instance, which solutions are better than others? Which solution should one choose to materialize? How difficult is to compute such a good solution? What is the semantics of target queries and how difficult is it to evaluate such queries?

In recent years, the study of the theoretical underpinnings of data exchange has mainly centered on data exchange settings between relational schemas and with schema-mappings specified by source-to-target tuple generating dependencies that can be thought of as global-and-local-as-view (GLAV) constraints [15, 16]. The next step is to attempt to extend this study to richer data exchange settings. One concrete direction is to study the semantics of data exchange between semistructured schemas and XML schemas, and to relate this investigation to the actual practice of existing data exchange tools, such as the CLIO system [21].

During the break-out session, there was a vigorous discussion about rethinking the semantics of query answering in data exchange. Thus far, *the certain answers* semantics has been used as the standard semantics in both data exchange and data integration, and much of the research has focused on the complexity of computing the certain answers to target queries. The definition of the certain answers is based on the entire space of solutions to the data exchange problem. What is so sacred about this semantics? Are there meaningful alternatives to the certain answers semantics that also take into account the "pragmatics" of the situation at hand? Is there a way to gauge the "quality" of the answer to a query, in addition to mainly focusing on the complexity of computing this answer?

Issues in metadata management. Schema-mappings *are* metadata. Bernstein [4] has made a compelling case for the importance of developing both the theory and the practice of metadata management, which in this framework, is achieved by combining certain basic generic operators on schema-mappings, such as *composition*, *merge*, *match*, and *change*. Repeated combinations of these operators can produce complex transformations on schema-mappings and can be used to analyze schema evolution.

The first main challenge in metadata management is to develop rigorous semantics for each of the basic operators. Once this is achieved, the next challenge is to investigate the properties of these operators for different schema-mapping languages. One concrete issue has to do with the *closure* properties of the schema-mapping languages. For instance, is a given schema-mapping language closed under composition? In other words, can the composition of two schema-mappings be expressed in the same language used to express each of the two schema-mappings? Another issue has to do with the algorithmic properties of the basic operators and the schema-mapping languages used to express them. In particular, for which schema-mapping languages can the outputs of these operators be efficiently computed? Progress has been made in the study of several operators, including composition [20, 17] and merge [22]. Nonetheless, much more remains to be done for the remaining operators.

Another major challenge in this area is the development of better user interfaces for visualizing and manipulating schema-mappings. Techniques from other areas, such as graph drawing, may have a role to play in designing more effective user interfaces that will make it possible to achieve large-scale metadata management.

Finally, there is a need to create a suite of benchmarks to be used to carry out experiments to compare tools for data exchange and metadata management. Research in this area will undoubtedly benefit from the existence of a public-domain repository with data, schemas that have evolved over time, complex schema-mappings, and challenging queries. Building such a repository will be a real service to the community and will advance the field.

5 Handling Inconsistency of Data

The notion of *inconsistency* has been extensively studied in many contexts. In classical logic, an inconsistent set of formulas implies every formula (*triviality*). In databases, a database instance is inconsistent if it does not satisfy integrity constraints (ICs) (*constraint violation*). Those two kinds of inconsistency are closely related. Triviality is not a problem in the database context because the semantics of query answers does not take into account ICs.

Inconsistent databases arise in data integration, during long-running database activities, and in other situations in which ICs cannot or would not be enforced. In order to deal with inconsistency in a flexible manner, database research has developed different approaches that we will illustrate using a simple example.

Consider a database schema consisting of two unary relations P and Q and the IC: $\forall x.\neg(P(x) \wedge Q(x))$. Assume a database instance consists of the following facts: $\{P(a), Q(a), P(b)\}$. Under *prevention* (usual constraint enforcement), such an instance could not arise: only one of $P(a)$ and $Q(a)$ could be inserted into the database. Under *ignorance* (constraint non-enforcement), no distinction is made between $P(a)$ and $P(b)$, despite that the latter, not being involved in a constraint violation, appears to represent more reliable information. Under *isolation* [7], both $P(a)$ and $Q(a)$ would be dropped (or ignored in query answering). Under *weakening* [3, 19], $P(a)$ and $Q(a)$ would be replaced by $P(a) \vee Q(a)$ or some other form of disjunctive information. Allowing *exceptions* [6], means that the constraint is weakened to $\forall x.\neg(P(x) \wedge Q(x) \wedge x \neq a)$. *Materialized repairing* [14] produces a *repair*: a consistent instance minimally different from the original one, in this case $\{P(a), P(b)\}$ or $\{Q(a), P(b)\}$. *Virtual repairing* [1] does not change the database but rather returns query answers true in all repairs (*consistent* query answers). So the query asking for all such x that $P(x)$ is true, returns only $x = b$. Finally, under the attack/support approach [23], $P(a)$ attacks $Q(a)$ and vice versa, and thus the support for both is lower than for $P(b)$.

Research has focused on materialized or virtual repairing, and different notions of repair have been proposed, to capture the notion of minimal change in different ways [1, 8, 25]. Also, different evaluation mechanisms for computing consistent query answers have been developed and the complexity of this problem studied [2, 13, 18, 8, 12].

The following research issues in the area of inconsistent databases were viewed as important by the participants:

1. How to *generalize/integrate/parameterize* existing approaches to the computation of consistent query answers.
2. What kind of *preferences* are useful and do they help? Although preferences may reduce the number of repairs, they introduce additional minimization criteria, which may negatively influence computational complexity.
3. Should *null* and *default* values be considered in constructing repairs?
4. How to identify contexts suitable for a specific notion or repair. For example, if the database is assumed to be complete, all repairs are obtained by deleting facts. However, in the absence of such an assumption, insertions of facts should also be considered.
5. How to integrate inconsistency resolution with various

data cleaning tasks: Data normalization/standardization, record linkage and merging.

6. How to design more powerful query languages by capturing the notion of *possible* query answer (answer true in some repair) and embedding the computation of both consistent and possible answers in a first-order query language.

7. How to deal with *intractability*. Trade-offs between expressive power and complexity should be further identified and approximate answers, possibly with confidence factors, studied.

8. How to test different algorithms. Measures of inconsistency should be defined and the issue of benchmarking and systematically generating inconsistent data explored.

9. How to generalize current approaches to repairing to XML databases and various data integration scenarios (e.g., data exchange, peer-to-peer). In such scenarios an inconsistent database is not necessarily stored but virtual, which creates additional challenges for repairing and computation of consistent query answers. Some current approaches to data integration presently ignore data inconsistencies and have to be extended to deal with them.

References

1. Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pp. 68–79, 1999.
2. Arenas, M., Bertossi, L. and Chomicki, J. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming*, 3(4–5):393–424, 2003.
3. Baral, C., Kraus, S., Minker, J. and Subrahmanian, V.S. Combining Knowledge Bases Consisting of First-Order Theories. *Computational Intelligence*, 8:45–71, 1992.
4. Bernstein, P. A. Applying Model Management to Classical Meta-Data Problems. In *Conference on Innovative Data Systems Research (CIDR)*. 209–220, 2003.
5. Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In *Inconsistency Tolerance*, Springer LNCS 3300, 2004, pp. 42-83.
6. Borgida, A. Language Features for Flexible Handling of Exceptions in Information Systems. *Proc. ACM Transactions on Database Systems*, 10(4):565–603, 1985.
7. Bry, F. Query Answering in Information Systems with Integrity Constraints. In *IFIP WG 11.5 Working Conference on Integrity and Control in Information Systems*, pp. 113–130. Chapman &Hall, 1997.
8. Calì, A., Lembo, D. and Rosati, R. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pp. 260–271, 2003.
9. Calvanese D., De Giacomo, G., Lenzerini, M. and Vardi, M.Y. Rewriting of Regular Expressions and Regular Path Queries. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pp. 194–204, 1999.
10. Calvanese D., De Giacomo, G., Lenzerini, M. and Vardi, M.Y. Answering Regular Path Queries Using Views. *Proc. International Conference on Data Engineering (ICDE)*, pp. 389–398, 2000.
11. Calvanese D., De Giacomo, G., Lenzerini, M. and Vardi, M.Y. View-based Query Containment. *Proc. ACM Symposium on Principles of Database Systems (PODS)*, pp. 56–67, 2003.
12. Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 197(1-2):90-121, 2005.
13. Eiter, T., Fink, M., Greco, G. and Lembo, D. Efficient Evaluation of Logic Programs for Querying Data Integration Systems. *Proc. International Conference on Logic Programming (ICLP)*, pp. 163–177. Springer-Verlag, LNCS 2916, 2003.
14. Embury, S.M., Brandt, S.M., Robinson, J.S., Sutherland, I., Bisby, F.A., Gray, W.A., Jones, A.C. and White, R.J. Adapting integrity enforcement techniques for data reconciliation. *Information Systems*, 26(8):657–689, 2001.
15. Fagin, R., Kolaitis, Ph. G., Miller, R. J., and Popa, L. Data Exchange: Semantics and Query Answering. In *International Conference on Database Theory (ICDT)*. 207–224, 2003.
16. Fagin, R., Kolaitis, Ph. G., and Popa, L. Data Exchange: Getting to the Core. In *ACM Symposium on Principles of Database Systems (PODS)*. 90–101, 2003.
17. Fagin, R., Kolaitis, Ph.G., Popa, L., and Tan, W.-C. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *ACM Symposium on Principles of Database Systems (PODS)*. 83-94, 2004.
18. Greco, G., Greco, S. and Zumpano, E. A Logical Framework for Querying and Repairing Inconsistent Databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1389–1408, 2003.
19. Lin, J. and Mendelzon, A.O. Merging Databases under Constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1996.
20. Madhavan, J. and Halevy, A. Y. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*. 572–583, 2003.
21. Popa, L., Velegrakis, Y., Miller, R. J., Hernandez, M. A., and Fagin, R. Translating Web Data. In *International Conference on Very Large Data Bases (VLDB)*. 598–609, 2002.
22. Pottinger, R., and Bernstein, P. A., Merging Models Based on Given Correspondences In *International Conference on Very Large Data Bases (VLDB)*. 826-873, 2003.
23. Pradhan, S. Argumentation Databases. *Proc. International Conference on Logic Programming (ICLP)*, pp. 178–193. Springer-Verlag, LNCS 2916, 2003.
24. Vardi. M. Y. A Call to Regularity. *Proc. PCK50 - Principles of Computing & Knowledge, Paris C. Kanellakis Memorial Workshop '03*, pp. 11.
25. Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. *Proc. International Conference on Database Theory (ICDT)*, pages 378–393. Springer-Verlag, LNCS 2572, 2003.