Integrating clinical data from hospital databases*

Kostis Karozos, Iosif Spartalis, Artem Tsikiridis, Despoina Trivela, and Vasilis Vassalos

Athens University of Economics and Business, Greece

Abstract. We present the Data Harmonization module from the Data Factory pipeline of the Human Brain Project which makes all the data transformations that are necessary to conform the local hospital datasets into a global schema and import them in the Medical Informatics Platform. To that scope, we encountered several challenging problems mostly due to the syntactic and semantic heterogeneities of the hospitals data. We elaborate on these data integration issues and present our implemented solutions.

Keywords: schema matching · data integration · clinical data

1 Introduction

In the last decades there has been a growing demand for integrating medical data deriving from scattered sources [28]. Research in various fields of medicine, like Neuroscience or Genome medicine, often requires the process and analysis of large amounts of possibly highly heterogeneous data scattered in different sources, like hospitals or scientific laboratories. By correlating such data, researchers are able to extract new knowledge related to their field of study, that are not able to obtain when working with each data source independently. Therefore, data integration technologies are often employed to enhance scientific research. In general, the goal of data integration is to provide a uniform access over a set of data sources that have been created and stored autonomously [8]. The use of such technologies in order to gather the disparate clinical data has been studied in the literature [18, 16, 17] resulting in practical systems [14, 4, 3]

In order to fully exploit the integrated clinical data it is important to be able to reveal the semantic relationships among them. Therefore, it is important that data integration is not limited to structural integration. For example, in order to translate patients data describing dementia symptoms into effective Alzheimer's disease diagnosis, it is important that these data are related to additional patients information, such as genetic data, as well as to biological markers, such as proteins and electroencephalography. The semantic description of the relations between the clinical presentation of patients and the biological markers can lead to more accurate diagnosis and possibly to the identification of new such relations that characterize the Alzheimer's disease. Furthermore, the

* Technical Paper

semantic data integration provides means to increase the expressive power of the queries that are posed over the integrated data [17].

In our setting, we are interested in integrating clinical data related to the human brain. Our work has been developed to meet the needs of the Medical Informatics Platform (MIP) of the Human Brain Project (HBP). The Human Brain Project aims to develop technologies that enhance the scientific research related to human brain. Towards this effort, the MIP provides a data integration mechanism to collect clinical data, such as Electronic Health Records (EHR) and imaging features that are stored in hospitals local databases. The data collected from the hospitals are mapped to the MIP schema and can be exploited by sophisticated Data Mining algorithms and techniques in order to identify biological signatures of diseases and predict features of the brain. The analytic functionalities of the platform along with the integrated data will be made available for re-use, further analysis and research without compromising the sensitive patient information.

In this paper we present our work on the data integration system we developed for the MIP. In particular, we present our work concerning the Data Mapping and Harmonization process which applies all the necessary transformations on the hospital data so that it is imported into the platform.

2 The process of data integration in MIP hospitals

2.1 MIP's Architecture and Basics

MIP is roughly composed of three different layers. Following a top-down approach, the first one is the Web Portal which, as the user interface of the platform, it provides access to the platform's analytic functionalities. The second layer is the Federation Layer which is responsible for federating the queries posed over the Web Portal to queries that are executed over the hospitals' datasets. Additionally, it is responsible for collecting and merging the relevant queries' answers. The third layer is the Local Layer and it exposes to the platform hospital data that are mapped into a common schema. This schema is populated by a set of (currently 168) common variables, namely the Common Data Elements (CDEs), that describe the knowledge about patients' brain structure, diagnosis, clinical tests results, genetic and demographic information (derived from EHR data). MIP users are able to design models and perform experiments based on these variables. The design of the common schema is based on the I2B2[1] data mart star schema which is widely used for implementing clinical data warehouses as well as research data warehouses. The star schema is consisted by the central fact table `OBSERVATION_FACT` surrounded by five dimension tables [Figure 1]. In health-care, a logical fact is an observation on a patient, thus in the case of the MIP, a CDE is considered as such. The `OBSERVATION_FACT` is filled with the basic attributes about each CDE, such as patient and provider numbers, a

[1] https://www.i2b2.org

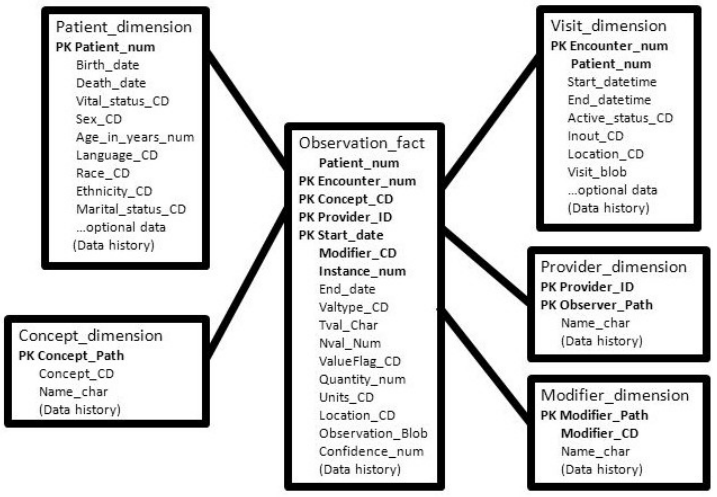


Fig. 1. I2B2 star schema.

concept code for the CDE observed and other parameters. Dimension tables contain further descriptive and analytical information about attributes in the fact table. For example, the `CONCEPT_DIMENSION` table contains information about how CDE variables are organized, the hierarchy they might belong to and other types of categorizations related to the data structure.

2.2 MIP Local Layer

The Local Layer of the MIP is deployed locally in each hospital that participates in the MIP. This is the layer where the data integration process takes place. In general, this layer is responsible for transforming the values of the local variables stored in the hospital's EHR and Picture Archiving and Communication Systems (PACS) into the CDE variables and then populate the common schema. More specifically, the Local Layer consists of three major components: the Data Capture, the Data Factory and the Feature Data Store Sub-system [Figure 2]. In the Data Capture component, the hospital EHR and PACS data is anonymized by the Gnubila FedEHR Anonymizer[2], a third party solution which is developed using Apache Camel framework. Afterwards, the anonymized data is processed in the Data Factory ETL pipeline. Finally, the integrated data is stored in the Feature Data Store module and they is made accessible to the Federation Layer.

The Data Factory has two separate inputs, the anonymized EHR and the PACS data. Therefore, the ETL pipeline has two initial branches which are then merged into a single path. Brain morphometric features are extracted from

[2] https://www.gnubila.fr

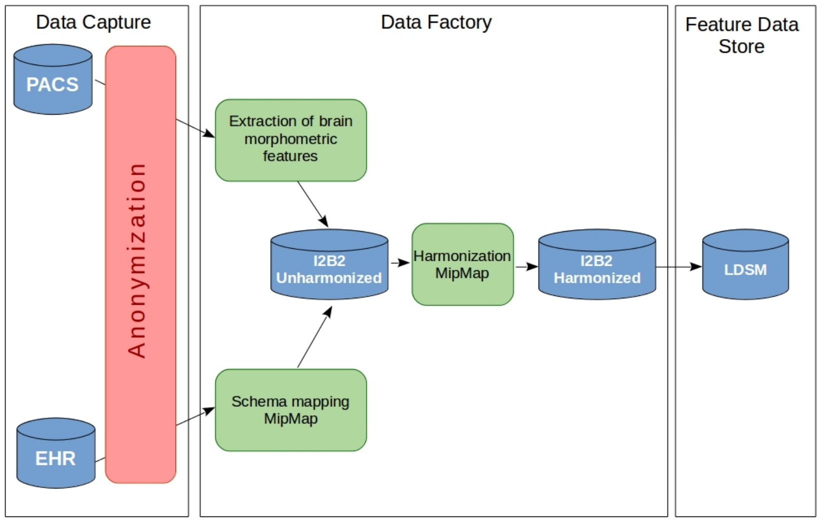


Fig. 2. MIP Local Layer.

the PACS data and are stored into an intermediate database having an I2B2 schema. This database, which is called *Unharmonized*, is the conjunction node where the two branches of the pipeline meet. For the case of the EHR data, a process of schema mapping is performed by using MIPMap [25]. Through this step the initial EHR hospital variables are also stored in the Unharmonized I2B2 database. At this point, the data is ready to be transformed into the corresponding CDE variables following the predefined harmonization rules implemented in MIPMap. These rules are specifically defined for each hospital by a team of hospital's clinicians and the MIP's special committee. By executing the harmonization mapping task the *Harmonized* I2B2 database is populated with CDEs values and the Data Factory pipeline is ended.

3 MIPMap overview

MIPMap is a schema mapping and data exchange tool [26]. MIPMap offers a GUI that allows the user to define correspondences between the source and the target schema, as well as join conditions, possible constraints and functional transformations. Users can simply draw arrow lines between the elements of two tree-form representations of the schemata.

Despite the progress in automated and semi-automated schema mapping, such techniques usually come with an error percentage which is inversely proportional to the level of the automation [8]. The requirements of the Human Brain Project leave absolutely no room for errors when processing sensitive medical data from hospitals. However, in order to facilitate the MIPMap users, WebMIPMap's supports collaboration features that enable the reuse of other experts' mappings (Section 5).

MIPMap is based on the open source mapping tool ++Spicy[20] which it extended so as to meet HBP's needs. One key difference between MIPMap and its predecessor is the approach of translating instances. MIPMap is equally focused on data exchange as well as schema mapping. ++Spicy offered high performance during the data exchange process when the source schema had a few instances, however in cases when the data was increased to some thousand rows per table the time and memory requirements grew exponentially. Thus, this all-in-memory approach was not suitable for our use cases. Instead of this, our tool uses temporary tables in a local Postgres database to eliminate the need for immense memory size.

Mappings are expressed in Tuple Generating Dependencies (TGDs)[2]. The target to-populate data is the outcome of the TGD rules' implementation on the source data. Producing the "optimal" solution during the data exchange procedure translates into discarding the redundant produced tuples while at the same time merging tuples that correspond to the same instance. Possible redundant target tuples, called homomorphisms[5], are recognized already in the TGD level and since the number of TGDs is considerably smaller than the number of instances, the computing procedure is much more efficient.

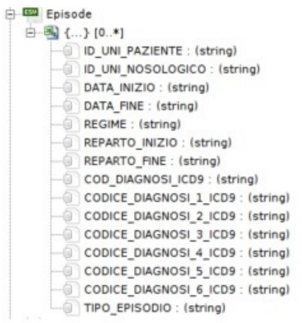
Taking into account the provided correspondences, constant assignments and the constraints declared, the original algorithm of the Clio[1] system is used to generate the candidate TGDs. Next, the rewriting algorithm recognizes possible cases of homomorphism among these rules. After potential generation of redundant tuples has been detected, the candidate TGDs are rewritten to take those into account and hence produce the optimal results.

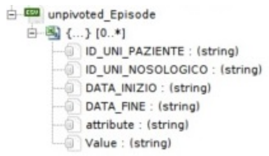
4 Problems faced with while integrating clinical data from heterogeneous sources

Before we started accessing hospital servers and conforming local data to the global MIP schema we were aware of the existing several types of heterogeneity [27] that may be encountered and which are related to structural, naming, semantic, content or other differences. Our initial impression was that following the data harmonization rules and implementing them in the mapping tasks configurations would be a straight forward task. We soon realized that was not the case.

4.1 Unpivoting

Since we use the generic I2B2 star schema and do not have a custom made global schema for the MIP variables, there is no 1:1 correspondence between the local and the global variables. I2B2's fact table, named `OBSERVATION_FACT`, is designed to store information about patients' visits in a self-explanatory way i.e. each tuple is an observation whose nature is defined in the concept_cd column and value in the tval_char or nval_num if it is string or numerical respectively. In order to map the hospitals' local variables to it we firstly had to generate an unpivoted version of some of the local datasets' tables and then designed the correspondences. Otherwise, an alternative would be to duplicate each target table for each "folding" element which clearly is too confusing and would mess up visually the configuration on the GUI. When unpivoting a table we select some stable key columns and "fold" the rest ones in an attribute-value duo of variables. An example of this preprocessing unpivoting procedure is shown in Figures 3 and 4 for Niguarda-Milano's table Episode. When configuring map-


```
Episode
  {...} [0..*]
    ID_UNI_PAZIENTE : (string)
    ID_UNI_NOSOLOGICO : (string)
    DATA_INIZIO : (string)
    DATA_FINE : (string)
    REGIME : (string)
    REPARTO_INIZIO : (string)
    REPARTO_FINE : (string)
    COD_DIAGNOSI_ICD9 : (string)
    CODICE_DIAGNOSI_1_ICD9 : (string)
    CODICE_DIAGNOSI_2_ICD9 : (string)
    CODICE_DIAGNOSI_3_ICD9 : (string)
    CODICE_DIAGNOSI_4_ICD9 : (string)
    CODICE_DIAGNOSI_5_ICD9 : (string)
    CODICE_DIAGNOSI_6_ICD9 : (string)
    TIPO_EPISODIO : (string)
```
Fig. 3. Columns of Table Episode


```
unpivoted_Episode
  {...} [0..*]
    ID_UNI_PAZIENTE : (string)
    ID_UNI_NOSOLOGICO : (string)
    DATA_INIZIO : (string)
    DATA_FINE : (string)
    attribute : (string)
    Value : (string)
```
Fig. 4. Columns of Table Episode's unpivoted version

pings for an unpivoted source table, MIPMap user has to keep in mind that the transformations she sets address to all folded variables, therefore these have to be valid for all of them. If any folded variable is in need of special treatment then the condition function has to take action.

4.2 Efficient Mapping Plan

Given a source and a target schema along with their semantic equivalences and transformation rules there are many ways the MIPMap user can implement them and configure the mapping task, specially when the source information is dispersed in many tables which have to be joined. Even though many mapping configurations may have the same semantically correct result, not all of them are optimal. In case a non-optimal solution is designed major delays will occur in the execution of the mapping task when dealing with large volumes of data. To avoid that, the MIPMap user has to have in mind the TGDs and the query plan[3] her GUI configurations are translated to. MIPMap as a tool is not responsible for these delays since it translates visual correspondences and joins to SQL on

[3] Actually we do not have to think constantly of the exact query plan; it is supposed to be intuitive for an database engineer which operations are more expensive than others

the fly; Postgres executing them and populating the target data is obviously what needs time.

Towards configuring efficient mapping tasks, MIPMap supports natural and outer joins (Figure 5) so the user can choose the most appropriate type. It also gives the option the joined values to be substrings of each other instead of matching the whole string. It is up to the MIPMap user, though, to configure mappings efficiently. One mapping efficiency rule is to prefer Selection Condition instead of If Condition function when we want to isolate some specific tuples, meaning select directly the tuples with the certain criteria instead of selecting all table's tuples and browsing one-by-one to check whether the criteria are true or not. Another tip is to avoid joins with unpivoted tables when possible. The reason is that from an original tuple many unpivoted tuples are generated; in fact they are as many as the selected variables to "fold". So when joining with an unpivoted version of a table we join with a multiple amount of the real tuples.

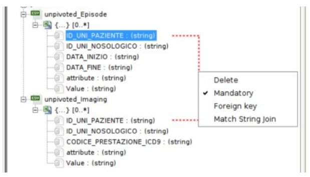


Fig. 5. Columns of Table Episode's unpivoted version

4.3 Data Clean(s)ing

As we have been examining local hospital data along with their semantics we witnessed an non-negligible amount of typos and value errors. Hospital IT departments seem to still have error-prone procedures when it comes to export data from their local EHR information system. In the meantime, batches of incoming (to the MIP) data tend to differentiate their initial local schema which leads to the conclusion that the information exported by the hospitals has not been finally standardized. In an environment like this, the need for efficient and accurate data cleansing[19] seems prominent.

Currently, there is no separate cleansing module in the pipeline since we do it manually with the use of MIPMap. When configuring the mapping tasks we exclude or correct faulty values via the mapping transformations. A separate, integratable to MIPMap, robust data cleansing module is in our plans. Until it is implemented and tested, we continue to correct errors in the input data using MIPMap's typical transformation functionalities.

5 Crowdsourcing semantic mapping

MIPMap's anti-proposal to automation is collaboration. WebMIPMap[25], an online version of MIPMap, allows users to create, share mappings as well as view, edit and evaluate other users' mappings. When a user loads another one's mapping she decides whether to accept or reject the correspondences one-by-one. A public mapping-task when firstly loaded by another user has all its correspondences green (Figure 6) until the user decides what to accept and what not (Figure 7). From that evaluation the mappings' owner credibility is updated accordingly. We aim at making WebMIPMap's basics known to HBP hospitals IT departments so as to actually use it. We believe this collaborativeness will benefit people's understanding for the clinical variables' semantics of our interest and the process of describing and importing data from new coming hospitals will flow in an easier way.

In case more than one users have shared the same mapping task but have differences between them, WebMIPMap can make recommendations. A weight is given to each correspondence and the one with the highest is selected. The correspondences' weights are computed taking into account the user's PageRank[23], her credibility, the connection's credibility and the user's total connections.

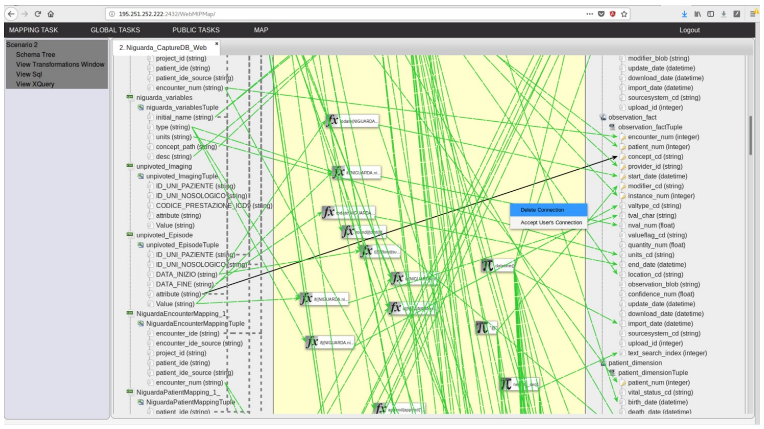


Fig. 6. Milano-Niguarda Public Mapping-task ready to get accepts/deletes on its proposing mappings

6 Mappings implementation and evaluation

The pipeline for MIP Local has been fully deployed in CHRU Lille[4] and Milano-Niguarda ICT[5] hospitals. The deployment is done in docker[22] containers mak-

[4] http://www.chru-lille.fr/

[5] https://comitatoeticoareac.ospedaleniguarda.it/

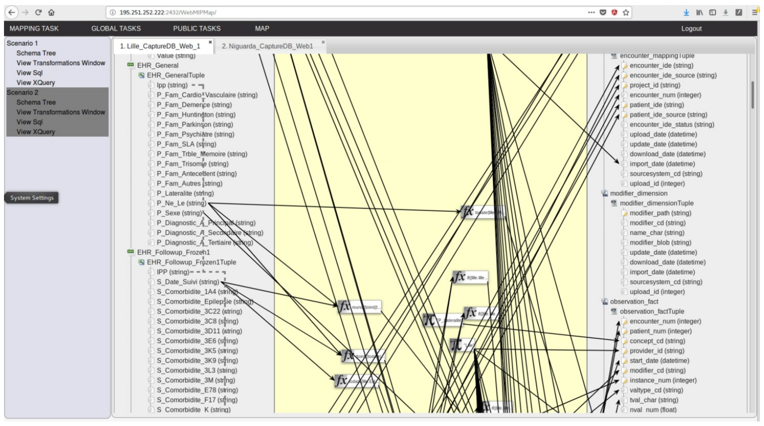


Fig. 7. Lille Accepted Mapping-task

ing it easier to transfer and setup our software components in diverse hospital operating system environments. The project's goal for the next phase is to recruit and setup MIP in a total of 30 hospitals setting the bar to collect data for more than 30.000 patients with brain diseases. From the 2 hospitals up until now we have processed and imported data for 1599 patients.

6.1 Incremental Data Integration

Clinical data from hospitals that are intended to be imported to MIP arrive in small batches and on a regular basis. In the stable version of MIPMap, the schema mapping process does not take into account the information about the frequency and load of this data integration workflow. To be more precise, the SQL-script that is produced by MIPMap could be of subpar performance if executed on a regular basis because it may execute all TGDs of the schema mapping plan from scratch every time. We examine whether an equivalent schema mapping that allows us to compute only the incremental changes to the target relations of MIP, as new batches of data arrive.

To tackle this problem, one could view the relations of the target-schema of MIP as a set of materialized views of the source-schema relations and take advantage of an established line of work, the theory of Incremental View Maintenance. The challenge for us is to apply this theory to a Data Exchange setting, the setting of MIPMap. To the best of our knowledge, it is the first time this type of approach is attempted. We present a brief outline of our work and our proposed approach.

The maintenance of materialized views of relations has been a topic of interest of the Databases community for more than thirty years. By precomputing and storing the contents of (traditionally) virtual views, e.g. in relational DBMS, the user can benefit from better query performance. However, as the data of the underlying source schema relations is modified, the goal is to compute and apply only the incremental changes to the materialized view and avoid recomputing its contents from scratch. Different techniques have been developed as the problem of maintaining materialized views turns out to be quite multidimensional[13].

MIPMap's schema mapping engine is based on ++Spicy[20] which can be viewed as the next-generation extension of +Spicy[21]. This family of schema mapping tools was among the first to implement algorithms that attain the important formalism in the theory of data exchange called the core[10]. The core is, in some sense, the *optimal* solution to a data exchange problem. The core schema mapping, is the smallest set of TGDs that preserves the semantics of the data exchange. These tools implement a TGD rewriting algorithm that, given a schema mapping, outputs a core schema mapping, as described in [21], in polynomial time.

When we are given a schema mapping scenario (a set of tuple-generating dependencies) we first transform each TGD describing a source-schema to target-schema correspondence to its "incremental" version according to the rules described in [24, 12]. We then feed the TGD rewriting algorithm of MIPMap with the "incremental" version of the TGDs.

The transformations of relational expressions we employ from [24, 12] to obtain the incremental version of the TGDs guarantee that no redundant computations are performed when each TGD is run independently. Moreover, we have shown that the core schema mapping with the incremental TGDs coincides with the core of the non-incremental TGDs that MIPMap produced previously. Therefore, this new schema mapping is in fact equivalent to the one that consists of non-incremental TGDs in terms of the semantics of the data exchange. However, we expect this data integration task to be superior in terms of performance.

7 Related work

Integrating clinical and biomedical data from multiple sources has been a goal for many other projects and initiatives in the bioinformatics community. A platform with major similarities to MIP is GAAIN[28], a federation platform for searching and integrating data about Alzheimer. It is a shared network of data and analysis tools that prompts scientists to join and share their repositories of relevant data while still keeping ownership rights to them. Part of GAAIN's architecture is GEM (GAAIN Entity Mapper)[3] which is a tool that maps pairs of schemas in an automated way. GEM identifies the name, description, type and range of each element and calculates the pairs' similarity scores.

As far as the architecture is concerned, data integration systems have two approaches: i)Data Warehouses and ii)Virtual Integration Systems. In Data Warehouses all the volume of data is consolidated into a single database using a unique global schema. The Human Genome Project has been an international research mega-project that built genes warehouses trying to identify and map all the genes of the human genome. Between others, the UCSC Genome Browser[11] and Ensembl[15] were two of the products developed in the project. Another warehousing system is DataFoundry[7] which is mainly designed for bioinformatics data but its infrastructure has the ability to run on other scientific domains' data as well. It combines features of a typical warehouse and a federated database. On the other hand, in Virtual Integration data is scattered across sources and does not conform to a global schema. Wrappers have the mission to translate queries and return results. In this category, we have TAMBIS[4] which uses ontologies and Description Logic. It is based on a biological knowledge database that covers aspects associated with proteins and nucleic acids, biological processes, tissues and taxonomy. BioMediator[9] has a similar approach as it has a knowledge database and uses a query language of its own, called PQL. Finally, IBM's DiscoveryLink[14] is a classic federation database for biomedical data which -between many others- was influenced by the Stanford-IBM Manager of Multiple Information Sources (TSIMMIS)[6]. SQL is the language in which queries are submitted and wrappers are defined. For a wrapper to access a data source, it has to register and be configured accordingly.

References

1. Chapter 52 - translating web data. In Philip A. Bernstein, Yannis E. Ioannidis, and Dimitris Ramakrishnan, Raghu Papadias, editors, {*VLDB*} *'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 598 – 609. Morgan Kaufmann, San Francisco, 2002.
2. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases: the logical level*. Addison-Wesley Longman Publishing Co., Inc., 1995.
3. Naveen Ashish, Peehoo Dewan, Jose-Luis Ambite, and Arthur W Toga. Gem: the gaain entity mapper. In *International Conference on Data Integration in the Life Sciences*, pages 13–27. Springer, 2015.
4. Patricia G Baker, Andy Brass, Sean Bechhofer, Carole A Goble, Norman W Paton, Robert Stevens, et al. Tambis: Transparent access to multiple bioinformatics information sources. In *Ismb*, volume 6, pages 25–34, 1998.
5. Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, pages 77–90, 1977.
6. Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The tsimmis project: Integration of heterogenous information sources. 1994.
7. Terence Critchlow, Krzysztof Fidelis, Madhavan Ganesh, Ron Musick, and Tom Slezak. Datafoundry: information management for scientific data. *IEEE Transactions on Information Technology in Biomedicine*, 4(1):52–57, 2000.
8. AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of data integration*. Elsevier, 2012.
9. Loren Donelson, Peter Tarczy-Hornoch, Peter Mork, Cindy Dolan, Joyce A Mitchell, M Barrier, and Hao Mei. The biomediator system as a data integration tool to answer diverse biologic queries. In *Medinfo*, pages 768–772. Citeseer, 2004.
10. Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: Getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, March 2005.
11. Pauline A Fujita, Brooke Rhead, Ann S Zweig, Angie S Hinrichs, Donna Karolchik, Melissa S Cline, Mary Goldman, Galt P Barber, Hiram Clawson, Antonio Coelho,

et al. The ucsc genome browser database: update 2011. *Nucleic acids research*, 39(suppl_1):D876–D882, 2010.

12. Timothy Griffin, Leonid Libkin, and Howard Trickey. An improved algorithm for the incremental recomputation of active relational expressions. *IEEE Trans. on Knowl. and Data Eng.*, 9(3):508–511, May 1997.
13. Ashish Gupta and Iderpal Singh Mumick, editors. *Materialized Views: Techniques, Implementations, and Applications.* MIT Press, Cambridge, MA, USA, 1999.
14. Laura M. Haas, Peter M. Schwarz, Prasad Kodali, Elon Kotlar, Julia E. Rice, and William C. Swope. Discoverylink: A system for integrated access to life sciences data sources. *IBM systems Journal*, 40(2):489–511, 2001.
15. Tim Hubbard, Daniel Barker, Ewan Birney, Graham Cameron, Yuan Chen, L Clark, Tony Cox, J Cuff, Val Curwen, Thomas Down, et al. The ensembl genome database project. *Nucleic acids research*, 30(1):38–41, 2002.
16. KA Karasavvas, R Baldock, and Albert Burger. Bioinformatics integration and agent technology. *Journal of Biomedical Informatics*, 37(3):205–219, 2004.
17. Brenton Louie, Peter Mork, Fernando Martin-Sanchez, Alon Halevy, and Peter Tarczy-Hornoch. Data integration and genomic medicine. *Journal of biomedical informatics*, 40(1):5–16, 2007.
18. Bertram Ludascher, Amarnath Gupta, and Maryann E Martone. Model-based mediation with domain maps. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 81–90. IEEE, 2001.
19. Jonathan I Maletic and Andrian Marcus. Data cleansing. In *Data Mining and Knowledge Discovery Handbook*, pages 21–36. Springer, 2005.
20. Bruno Marnette, Giansalvatore Mecca, Paolo Papotti, Salvatore Raunich, Donatello Santoro, et al. ++ spicy: an open-source tool for second-generation schema mapping and data exchange. *Clio*, 19:21, 2011.
21. Giansalvatore Mecca, Paolo Papotti, and Salvatore Raunich. Core schema mappings. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 655–668, 2009.
22. Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.
23. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
24. Xiaolei Qian and Gio Wiederhold. Incremental recomputation of active relational expressions. *IEEE transactions on knowledge and data engineering*, 3(3):337–341, 1991.
25. Giorgos Stoilos, Despoina Trivela, Vasilis Vassalos, Tassos Venetis, and Yannis Xarchakos. Enabling data integration using mipmap. In *International Conference on Data Integration in the Life Sciences*, pages 86–93. Springer, 2017.
26. Giorgos Stoilos, Despoina Trivela, Vasilis Vassalos, Tassos Venetis, and Yannis Xarchakos. Enabling data integration using mipmap. In Marcos Da Silveira, Cédric Pruski, and Reinhard Schneider, editors, *Data Integration in the Life Sciences*, pages 86–93, Cham, 2017. Springer International Publishing.
27. Walter Sujansky. Heterogeneous database integration in biomedicine. *Journal of biomedical informatics*, 34(4):285–298, 2001.
28. Arthur W Toga, Scott Neu, Karen Crawford, Priya Bhatt, and Naveen Ashish. The global alzheimer's association interactive network (gaain). *Alzheimer's & Dementia: The Journal of the Alzheimer's Association*, 11(7):P121, 2015.