Data Integration: After the Teenage Years

Behzad Golshan Alon Halevy George Mihaila Wang-Chiew Tan

Recruit Institute of Technology
{behzad, alon, george, wangchiew}@recruit.ai

ABSTRACT

The field of data integration has expanded significantly over the years, from providing a uniform query and update interface to structured databases within an enterprise to the ability to search, exchange, and even update, structured or unstructured data that are within or external to the enterprise.

This paper describes the evolution in the landscape of data integration since the work on rewriting queries using views in the mid-1990's. In addition, we describe two important challenges for the field going forward. The first challenge is to develop good open-source tools for different components of data integration pipelines. The second challenge is to provide practitioners with viable solutions for the long-standing problem of systematically combining structured and unstructured data.

Keywords

data integration; views; open-source; structured and unstructured data

1. INTRODUCTION

The field of data integration has expanded in many directions as the technology landscape evolved in recent years. At the outset, the goal of data integration was to build systems that provide a uniform query (and for the ambitious, update) interface to multiple databases within an enterprise. As the Web came into prominence, data integration addressed challenges such as incorporating data from external business partners, data exchange between multiple sources, peer-to-peer data sharing architectures, and querying multiple deep-web data sources. The field also had to consider semi-structured data such as XML and unstructured textual data. When personal data management came to the forefront, researchers developed methods for integrating data from streams available to a single user. Most recently, the focus has shifted to integrating extremely large data sets, and extremely large numbers of data sets, such as the collection of HTML tables on the Web.

Data integration has also been a field that pushed on the combination of techniques from data management and artificial intelligence. Initially, the contribution of AI was to propose knowledge representation formalisms as a mechanism for describing the contents of heterogeneous data sources. With time, AI techniques became even more critical to the field, since data integration operators need to handle uncertainty, need to extract facts from text, need to learn from previous experience, and often, they need to do all of the above at the same time.

This paper accompanies a PODS 2017 Gems talk describing the evolution of data integration since the work on rewriting queries using views in the mid-1990's. The paper can also be considered to be a follow on to the paper "Data Integration: the Teenage Years" published in 2006 [25]. We begin by describing the role that views had in the work on data integration (Section 2) and surveying some of the subsequent developments (Section 3). Like its predecessor [25], the paper is *not* meant to be a comprehensive survey of the field (see [14, 16] for recent textbooks).

We then turn to the main focus of the paper, where we describe two important challenges for the field. The first challenge (Section 4) is that progress of data integration and its application in practice are hindered by the fact that there are very few quality tools with which practitioners and researchers can freely experiment. As a result, the progress on research does not accumulate in the form of open-source tools available for use. We describe an attempt to address this challenge, the BigGorilla open-source platform for data integration in Python. One of the important aspects of BigGorilla is that it offers powerful tools for data integration operators that are needed in practice, and each of them can be used by itself in a data pipeline without requiring the others. We hope that the community will contribute to the platform and improve it with time.

The second challenge we discuss (Section 5) is the current-day version of the long-standing challenge to combine structured and unstructured data. We argue that despite the multiple efforts to address this problem, we still do not offer practitioners viable solutions.

2. VIEWS IN DATA INTEGRATION

Data integration systems provide a uniform query interface to multiple heterogeneous data sources. To provide that interface, they introduce the abstraction of a *global schema* (often called a *mediated schema*). Users pose queries against the global schema, rather than having to know the details of each of the source schemas. Early work on data integration focused on modeling the contents and capabilities of heterogeneous data sources and mapping them to the global schema.

There were two predominant approaches to modeling data sources. *Global-as-view* (GAV) modeled the global schema as a set of view definitions over the schemas of the data sources. In contrast, *local-as-view* (LAV) proposed to model the contents of a data source as

PODS'17, May 14 - 19, 2017, Chicago, IL, USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4198-1/17/05...$15.00

DOI: http://dx.doi.org/10.1145/3034786.3056124

a view over the global schema. The approach known as *Global-local-as-view* (GLAV) combined the two formalisms, where a view over the data sources is defined as a view over the global schema. The GLAV formalism is also known as tuple-generating dependencies [5] and is used widely in the study of data integration and exchange [2, 17, 30, 33]. In the GAV approach, reformulating a query that is posed over the global schema into a query on the data sources is conceptually straightforward–the query is unfolded using the view definitions. In the LAV approach, reformulating a query posed over the global schema is trickier and amounts to the problem of rewriting queries using views.

The problem of rewriting queries using views received additional attention at the time because there was also a natural application of the techniques in query optimization [12, 21, 32, 40, 41]: if query plans can use materialized views instead of base tables, they can be significantly more efficient.

It is also interesting to note that LAV was a subtle introduction of AI techniques (or more specifically, knowledge representation) into the modeling of data sources (a much less subtle introduction was made earlier in [10]). A view definition is a simple form of knowledge representation, and rewriting queries using views is a reasoning algorithm whose properties are relatively attractive compared to proposals based on description logics that were also discussed at the time. The main benefits of the LAV approach were that it was easier to add data sources as they became available and that it was possible to describe the contents of the data sources more precisely.

The tradeoffs between different schema mapping formalisms (GAV, LAV, or GLAV), the expressive power of queries and of the views provided a rich area of exploration. Some of the main directions included data exchange (defined as the problem of mapping the actual contents of an origin database into a target database with a different schema), adapting the formalisms for tree and graph data, and answering queries from views without necessarily constructing a rewritten query as an intermediate step. A few years later, the area of model management [6] took a comprehensive look at operations that can be performed on database schemas (e.g., merging, composition and inversion of schema mappings, matching schemas). In addition, there was quite a bit of work on building efficient data integration systems and on algorithms for semi-automatically finding matchings and mappings between heterogeneous schemas [7] and work on benchmarking data integration systems [1, 3]. Some of these developments are described in more detail in [25].

3. THE LANDSCAPE CHANGES

To appreciate the developments in data integration in subsequent years, it is important to understand the typical assumptions made by early data integration systems. These assumptions were perfectly reasonable at the time, but they broke down in later years when the applications of data integration expanded. We will refer to the early work as the *classic* data integration paradigm. The assumptions they made were the following.

A1: The global schema is of reasonable size and can be built with modest effort.

A2: The data sources are structured (or at least semi-structured) and have well-defined schemas.

A3: There is a need to integrate all the data sources at hand.

A4: All data integration functionality should be part of an end-to-end data integration system.

A5: The data in the data sources is mostly correct and consistent across data sources (and if not, then getting it into that state is a matter of standard data cleaning).

The fundamental reasons that the assumptions of the classic paradigm break down are the scale and nature of the data sources. Early applications of data integration handled tens of data sources and often less, and they only had to model well-defined domains (e.g., customer support, supply-chain management, certain scientific databases). While it may be easy for a data owner to arrive at a global schema of the data sources she owns, it is often difficult for different data owners to agree on how the global schema of all their data sources should be represented and which sources are more authoritative. To make matters worse, data sources are rarely static in their lifetime. For this reason, collaborative data sharing architectures, such as [28], have been proposed. Today, data integration applications need to consider data sources on the Web and large collections of data sets in enterprises.

The number of structured data sources on the Web is vast. Work from Google and Microsoft report that the number of high-quality HTML tables is in the hundreds of millions [4, 11]. Due to progress on data integration, these HTML tables are now routinely included in responses to queries posed to Web search engines. Earlier work estimated the number of deep web sources to be in the tens of millions [35]. But even within enterprises the number of datasets is enormous. As a somewhat extreme example, [24] reported that the Google dataset search system had 26 billion datasets in its index. In fact, search for datasets is becoming so important that each of the cloud service providers are now starting to support search over datasets.

When you have this number of data sources it is impossible to build a global schema, and hence assumption A1 breaks down. The data sources will also be much more diverse, and therefore assumption A2 does not hold anymore either. In the relatively simple case of modeling all the data in an enterprise, building a global schema would involve understanding every aspect of your enterprise and all the operations it runs. In the case of the Web, structured data covers all domains that are of interest to mankind. Hence, building a global schema involves a Herculean effort of modeling all possible domains, doing it in over 100 languages, and taking into consideration cultural nuances. Even Freebase, the broadest schema, has been shown to cover only a small fraction of the attributes used in HTML tables [23].

If you cannot build a global schema, the question of whether it is really necessary to integrate all the available data arises. In fact, sometimes data integration involves a great deal of human effort as (external) facts have to be carefully verified before they can be merged into a database [9]. In the ideal world it would be nice to be able to answer any question over these vast data collections. In practice, however, you can get by with less lofty goals, putting into question assumption A3. Instead of integrating the data, the focus shifts to building a system that provides useful search services over data sources. That does not mean that integration will never happen – when the benefit of fully integrating a smaller number of data sources arises, then it is the time to expend the effort. In practice, this trend has led to the rise of two complimentary types of systems: data lakes (often referred to as *dataspaces* [20]), and search systems over the collection of datasets in a data lake.

The key idea of data lakes is *pay-as-you-go* data management. Given the collection of data sources available in the lake, the system should offer several useful services. The first such service is to locate sources that are relevant to a particular query. Given a specific data source, the data lake should provide as much useful information about it, such as usage statistics and provenance infor-

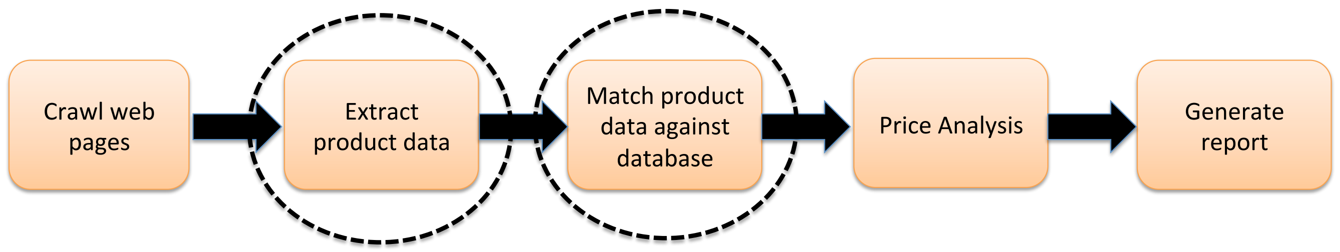


Figure 1: A data pipeline for comparing the prices of products with one's competitor. The pipeline starts by crawling the competitor's site, then extracts the product's attributes from each product page. Next, the pipeline matches products across the two databases, and finally the price comparison is performed. The first, second, and third steps of the pipeline are data integration operators.

mation (which data sources it depends on, and which data sources depend on it). The system should also monitor the data source as it is updated and send an alert when something unusual happens that may indicate a problem.

The data lake should also help a user find similar data sources, including sources that could be joined or unioned with a given data source. When a collection of related data sources are found, the tools of the data lake should help to build an integrated global view over them.

The above discussion should make it clear that even though the focus has shifted from integration to looser federation architectures, the technical challenges in both cases have many similarities. A good search system still needs to understand the semantics of the underlying data sources, needs to be able to find matches between schemas of disparate datasets and needs to find how entities in one dataset correspond to those in another. Hence, we see that the value of the basic data integration operators, such as schema matching and merging, data matching and merging, schema mappings, that were developed in the context of classic data integration are useful in isolation as well, leading to questioning assumption A4. We return to this point in detail in Section 4.

Finally, assumption A5 breaks down when there is such a large number of data sources from independent organizations. In such a case, there are likely to be more inconsistencies between data sources [18, 27] and cases where the actual truth is a matter of perspective [15].

4. MAKING IT ON OUR OWN

As we have alluded to above, data integration operators are useful as standalone services, not just as part of a complete data integration system. In particular, operations such as data matching, schema matching, and data transformation arise in common scenarios such as data-analysis pipelines and data-lake services. Consider the example in Figure 1.

The example describes a pipeline used by an online merchant (e.g., Walmart or Amazon) who wants to periodically compare its prices with the corresponding prices of a competitor. The pipeline begins by crawling the competitor's Web site and extracting from every product page a structured description of the product including the price. Once the pipeline created a database of the competitor's products, the next task is to match the competitor's products with the vendor's own product database so a direct comparison of prices can be performed. After the correspondence between the product databases is found, the actual analysis can begin and the appropriate reports can be generated.

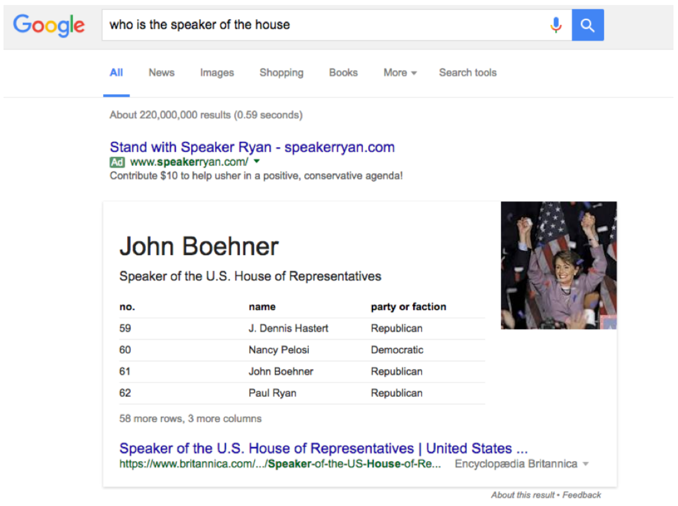
The second, and third steps of the pipeline (structured product extraction and matching across databases) are prototypical data integration operators. It would be highly beneficial to the data scientists or engineers if they can simply plug existing tools into the appropriate stages of their pipeline. In general, a data pipeline is typically more complex than what is depicted in the figure. Also, some operators may be invoked repeatedly for different purposes. In any case, significant time is almost always involved to collect and integrate data before a data scientist can apply her favorite algorithm for analysis [34].

The above example illustrates an important overarching challenge our community faces, namely, that data integration problems are not recognized as such when practitioners encounter them. Specifically, if an engineer is building a pipeline as the one described above, and she needs to match data records from two sources, she probably would not even realize that data matching is an example of a bigger set of tasks known collectively as data integration. Even if she does recognize the problem, she will not consider buying a data integration product to solve her problem, because it is probably too expensive for the problem at hand. More importantly, the data integration system will not provide a module that fits into the existing pipeline. One of the causes of this recognition problem is that data integration operators have always been part of a comprehensive data management solution, and we have never tried to go at it on our own. To emphasize, we describe the problem not because we lack professional pride or yearn for more recognition for our work as a community. Our goal is to considerably expand the impact we can have in practice.

In response to this challenge, we and others have started developing an open-source platform for data integration and preparation (learning some of the lessons from a previous attempt [37]). BigGorilla[1] exposes data integration and preparation operators as modules in Python [8]. BigGorilla currently uses Python because it is the easiest environment for data scientists to quickly perform experiments. However, as the project develops, implementations in other programming languages will be encouraged. Currently, BigGorilla has modules for string matching, entity matching [31], and a simple schema matcher that considers both schema attributes and data if available. In addition, it has consolidated open-source tools for other modules such as acquisition, extraction, and workflow management. New modules are being added as they are required by applications. BigGorilla also aims to enrich its repository of data pipelines, which can serve as instructive examples for the development of other pipelines, over time.

Though BigGorilla focuses on data integration operators, we do

[1]Data integration is a big, dirty, and hairy problem. Hence, BigGorilla. http://www.biggorilla.org



who is the speaker of the house

About 220,000,000 results (0.59 seconds)

Stand with Speaker Ryan - speakerryan.com
Ad www.speakerryan.com/
Contribute $10 to help usher in a positive, conservative agenda!

John Boehner
Speaker of the U.S. House of Representatives

no.	name	party or faction
59	J. Dennis Hastert	Republican
60	Nancy Pelosi	Democratic
61	John Boehner	Republican
62	Paul Ryan	Republican

58 more rows, 3 more columns

Speaker of the U.S. House of Representatives | United States ...
https://www.britannica.com/.../Speaker-of-the-US-House-of-Re... Encyclopædia Britannica

About this result • Feedback

Figure 2: Google's answer to the query "speaker of the house" circa Fall, 2016.

acknowledge the important role and benefits arising from building end-to-end systems, recent examples of which include Data Tamer [39], Deepdive [36], High-level Integration Language [26], Data Civilizer [13] and also integration and visualization systems such as [22, 29].

5. COMBINING STRUCTURED AND UNSTRUCTURED DATA

Combining structured data and unstructured data is a long-standing challenge to the data management community and to the data integration community in particular. However, like other long-standing challenges, its importance only increases.

The challenge arises because the content that data integration systems manage is both structured and unstructured and because they need to answer queries that can be formulated using both keywords and structured query languages. Web search engines have faced this challenge for a while, but it has become more acute in recent years because they try to answer some of their queries using structured data stored in their knowledge graph [38]. The challenge is faced also by vertical search engines, such as search for jobs, beauty salons or houses, and enterprise data lakes. Even narrower applications such as tracking the sentiment towards a company's products on social media face this challenge head on.

On the query side, users have gotten use to simple keyword search engines and expect every system to offer such an interface. For certain query fields, such as location and date ranges, users still provide structured data in their queries.

On the content side the data may simply be unstructured, such as a stream of social media comments, or a collection of Web documents. However, even when the primary organization of the content is structured, salient aspects of the content may still end up in unstructured text fields. Consider the example of a search engine for beauty salons (e.g., treatwell.com). Each beauty salon is a record in a structured database. To register with the search service, the owner of a beauty salon will fill out a questionnaire with a few fields, such as address and opening hours. However, some of the nuanced content describing the salon is more detailed than the schema of the search service. For example, the owner may want to describe some of the skills and experience of particular hair stylists, or some culturally specific names for the services they offer.

In summary, a data integration system needs to understand unstructured queries, translate them into a structured query when possible, and then find answers from content that may be partially structured and partially in text. There are many facets to this challenge. In the following example we highlight a particularly important facet, that of combining data to produce query answers.

Consider the query *who is speaker of the house*, posed to Google. Figure 2 shows the answer given sometime in the Fall of 2016 (Google answers change constantly for many reasons, including locale, time of day, and device from which the query is issued, and numerous changes to the search engine algorithms). To those not familiar with American politics, or rightfully skeptical about them at this point in time, at the time of writing, the answer to this query is Paul Ryan. Ryan was preceded by John Boehner who was preceded by Nancy Pelosi. The speaker of the house is the head of the U.S. Congress and would become president if both the sitting president and vice-president are unable to fulfill their duties.

As can be seen from the screenshot, Google provides answers from three sources, and in this example, they are all different. The top result, John Boehner, comes from a system that performs information extraction from Web documents. Since Paul Ryan was relatively new on the job at the time of the query, the number of Web documents supporting him may have been considerably smaller than those supporting his predecessor and therefore Boehner was chosen. The second answer (the HTML table) has the correct data, but that is apparent only to a human looking at the table who can interpret the meaning of the order of rows in the table. Finally, the third part of the page (which is not strictly speaking a direct answer to the query, but only a visual adornment to the answer), the image of Nancy Pelosi comes from image search. The three algorithms are independently contributing to the displayed result and are inconsistent with each other. Amusingly, the ad on the top of the search result page refers to the correct answer.

Abstracting from the example, the challenge is the following (see Figure 3). We have developed many techniques to extract structured data from unstructured data, such as the algorithm yielding John Boehner above. We also have algorithms for finding good structured data in the sea of unstructured Web documents, such as the table above. However, we have no principled way to combine the results of different algorithms. Google's case is a relatively extreme one since there are many algorithms at play, and as seen above, there is no attempt to try to reconcile them. But even in less complex query engines, engineers struggle with this challenge. In the current state of the art, the algorithm for combining results is hardwired somewhere in the code of the search engine. As a result, it's hard to know how the engine works, hard to modify it, and impossible to explain to users.

Our community has spent quite a bit of effort addressing this challenge. One of the main lines of research has been on probabilistic databases and more generally, reasoning with uncertainty. However, these techniques have had limited impact in practice because of the difficulty of coming up with reliable probabilities and interpreting them in practical settings. Work on tracing provenance/lineage has a potential of contributing to the solution, because combining evidence often has to do with where they come from. However, provenance is only one component of the solution.

Though we are actively thinking about this challenge, we are not yet in a position to describe a solution. We believe, however, that there are two principles that should guide the solution. First, it should be possible to declaratively specify the combination of structured and unstructured data using an appropriate language. A

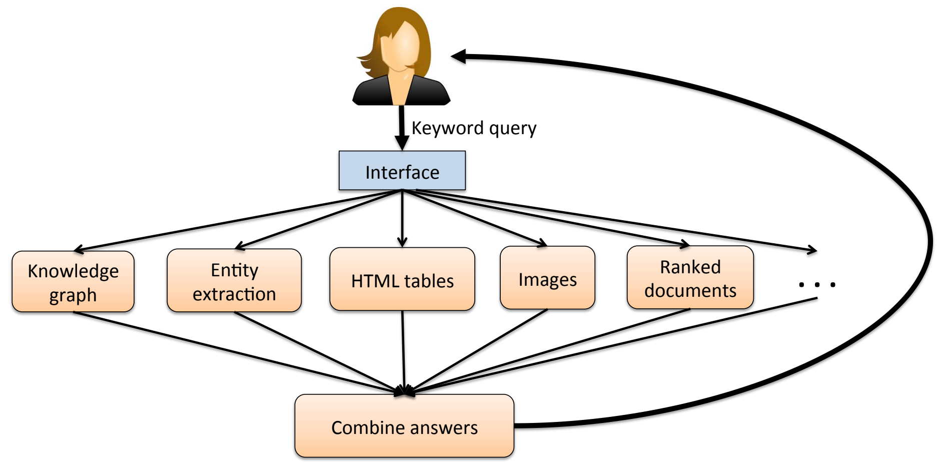


Figure 3: Different algorithms are consulted when answering a query to the search engine. Some of these algorithms consult structured data and others find responses from unstructured data. The challenge we face is to develop a principled method to *combine* the results of all these algorithms into a coherent answer to the user.

declarative specification would enable engineers to easily inspect their systems, debug the outcome, modify them when necessary, and possibly expand the scope of the search as needed [19]. At the same time, that does *not* mean that the rules for the combination need to always be written manually. In particular, some of the rules can be inferred with machine learning techniques.

The second principle is the original data should be accessible at all levels of the system. For example, in the query above, once we realize that extraction of John Boehner may not be consistent with the HTML table, we may want to inspect the documents from which Boehner was extracted to see if they're earlier than those of Ryan. However, if all we have access to is the final result of the extraction algorithm, even if it is a ranked list of names with probabilities, we cannot reason any deeper. In practice, we may have to approximate this principle, and keep around, as much as possible, the context of the original data.

6. CONCLUSION

As computing continues to deepen its impact on every aspect of our lives, new application scenarios create novel data integration challenges. The basic operations that data integration research has addressed since its inception have withstood the test of time and continue to appear in different variants in the new contexts. We believe that it is time for data integration operators to break free of end-to-end data integration systems and be available in the open source to speed up adoption and progress. With the rise of speech and natural language interfaces to our mobile devices, it also becomes critical that we find effective solutions to combining structured and unstructured data. The exciting golden age of data integration is still in front of us.

7. REFERENCES

[1] B. Alexe, W. C. Tan, and Y. Velegrakis. Stbenchmark: towards a benchmark for mapping systems. *PVLDB*, 1(1):230–244, 2008.

[2] M. Arenas, P. Barceló, L. Libkin, and F. Murlak. *Foundations of Data Exchange*. Cambridge University Press, 2014.

[3] P. C. Arocena, B. Glavic, R. Ciucanu, and R. J. Miller. The ibench integration metadata generator. *PVLDB*, 9(3):108–119, 2015.

[4] S. Balakrishnan, A. Y. Halevy, B. Harb, H. Lee, J. Madhavan, A. Rostamizadeh, W. Shen, K. Wilder, F. Wu, and C. Yu. Applying webtables in practice. In *CIDR*, 2015.

[5] C. Beeri and M. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.

[6] P. A. Bernstein. Applying model management to classical meta-data problems. In *CIDR*, 2003.

[7] P. A. Bernstein and L. M. Haas. Information integration in the enterprise. *Commun. ACM*, 51(9):72–79, 2008.

[8] Biggorilla: Data integration and data preparation in python. http://www.biggorilla.org, 2017.

[9] P. Buneman, J. Cheney, W. C. Tan, and S. Vansummeren. Curated databases. In *Proc. of PODS*, pages 1–12, 2008.

[10] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, pages 55–62, 1993.

[11] K. Chakrabarti, S. Chaudhuri, Z. Chen, K. Ganjam, and Y. He. Data services leveraging bing's data assets. *IEEE Data Eng. Bull.*, 39(3):15–28, 2016.

[12] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proc. of ICDE*, pages 190–200, Taipei, Taiwan, 1995.

[13] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *CIDR*, 2017.

[14] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[15] X. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava. SOLOMON: seeking the truth via copying detection. *PVLDB*, 3(2):1617–1620, 2010.

[16] X. L. Dong and D. Srivastava. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015.

[17] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[18] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[19] G. H. L. Fletcher, J. V. den Bussche, D. V. Gucht, and S. Vansummeren. Towards a theory of search queries. *ACM Trans. Database Syst.*, 35(4):28:1–28:33, 2010.

[20] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: A new abstraction for information management. *Sigmod Record*, 34(4):27–33, 2005.

[21] J. Goldstein and P.-A. Larson. Optimizing queries using materialized

views: a practical, scalable solution. In *Proc. of ACM SIGMOD*, pages 331–342, 2001.

[22] P. J. Guo, S. Kandel, J. M. Hellerstein, and J. Heer. Proactive wrangling: mixed-initiative end-user programming of data transformation scripts. In *UIST*, pages 65–74, 2011.

[23] R. Gupta, A. Y. Halevy, X. Wang, S. E. Whang, and F. Wu. Biperpedia: An ontology for search applications. *PVLDB*, 7(7):505–516, 2014.

[24] A. Y. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Managing google's data lake: an overview of the goods system. *IEEE Data Eng. Bull.*, 39(3):5–14, 2016.

[25] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In *VLDB*, 2006.

[26] M. Hernández, G. Koutrika, R. Krishnamurthy, L. Popa, and R. Wisnesky. Hil: A high-level scripting language for entity integration. In *Proc. of EDBT*, pages 549–560, 2013.

[27] I. F. Ilyas and X. Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

[28] Z. G. Ives, T. J. Green, G. Karvounarakis, N. E. Taylor, V. Tannen, P. P. Talukdar, M. Jacob, and F. C. N. Pereira. The ORCHESTRA collaborative data sharing system. *SIGMOD Record*, 37(3):26–32, 2008.

[29] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler: interactive visual specification of data transformation scripts. In *CHI*, pages 3363–3372, 2011.

[30] P. Kolaitis. Schema mappings, data exchange, and metadata management. In *Proc. of ACM PODS*, pages 61–75, 2005.

[31] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. *PVLDB*, 9(12):1197–1208, 2016.

[32] P. A. Larson and H. Yang. Computing queries from derived relations. In *Proc. of VLDB*, pages 259–269, 1985.

[33] M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proc. of ACM PODS*, 2002.

[34] S. Lohr. For Big-Data Scientists, 'Janitor Work' is Key Hurdle to Insights. New York Times, 2014.

[35] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep-web crawl. In *Proc. of VLDB*, pages 1241–1252, 2008.

[36] C. D. Sa, A. Ratner, C. Ré, J. Shin, F. Wang, S. Wu, and C. Zhang. Deepdive: Declarative knowledge base construction. *SIGMOD Record*, 45(1):60–67, 2016.

[37] L. Seligman, P. Mork, A. Y. Halevy, K. P. Smith, M. J. Carey, K. Chen, C. Wolf, J. Madhavan, A. Kannan, and D. Burdick. Openii: an open source information integration toolkit. In *Proc. of ACM SIGMOD*, pages 1057–1060, 2010.

[38] A. Singhal. Introducing the knowledge graph: things, not strings. *Official google blog*, 2012.

[39] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.

[40] O. G. Tsatalos, M. H. Solomon, and Y. E. Ioannidis. The GMAP: A versatile tool for physical data independence. *VLDB Journal*, 5(2):101–118, 1996.

[41] H. Z. Yang and P. A. Larson. Query transformation for PSJ-queries. In *Proc. of VLDB*, pages 245–254, 1987.