A Call to Arms: Revisiting Database Design

Antonio Badia
University of Louisville, USA
abadia@louisville.edu

Daniel Lemire
Université du Québec à Montréal, Canada
lemire@acm.org

1. INTRODUCTION

Good database design is crucial to obtain a sound, consistent database, and — in turn — good database design methodologies are the best way to achieve the right design. These methodologies are taught to most Computer Science undergraduates, as part of any Introduction to Database class [33]. They can be considered part of the "canon", and indeed, the overall approach to database design has been unchanged for years. Moreover, none of the major database research assessments identify database design as a strategic research direction [1, 2, 8].

Should we conclude that database design is a solved problem?

Our thesis is that *database design remains a critical unsolved problem*. Hence, it should be the subject of more research. Our starting point is the observation that traditional database design is not used in practice — and if it were used it would result in designs that are not well adapted to current environments [5]. In short, database design has failed to keep up with the times. In this paper, we put forth arguments to support our viewpoint, analyze the root causes of this situation and suggest some avenues of research. The point of view espoused here has been put forth more or less explicitly in other places (see [18] for a recent and notable example); but here we put together several strands that have received isolated attention, and focus them on an issue that we feel is particularly important — database design.

In the next section (§ 2), we sketch the traditional database design process: we argue that it manages to be, at the same time, *over-engineered* and *under-engineered*. The contradiction is only apparent: as any complex problem, this one is multifaceted. Traditional design does too little with respect to some areas and too much with respect to others. In § 3, we analyze the causes of the problems presented in § 2. We then briefly the current status of research on database design (§ 4). Finally, we present some ideas for a research renewal in § 5.

2. TRADITIONAL MODELING

Relational modeling is usually broken down into three steps:

- **Conceptual modeling**, which includes *requirement gathering and specification*, and results in a *conceptual model* of the database. At this stage, the designer focuses on issues of scope — *what belongs in the database?* — and organization — *how is the information to be structured?* Entity-relationship diagrams [15] and UML class models are the two best known conceptual models, but not the only ones; alternatives like Object Role Modeling have been proposed [29].
- **Logical modeling**, which takes as input the conceptual model produced in the previous step and yields a database schema. This step is well developed [60]. Normalization enforces *functional dependencies* by removing redundancy.
- **Physical modeling**, which takes as input the database schema produced in the previous step and produces storage structures to implement the schema in computer systems. It can be automated to a large extent [13, 24].

Each step focuses on only one aspect of the problem which helps tame the complexity. Also, each step produces an output that feeds into the next step, creating a linear structure that is easy to follow.

2.1 Problems with the traditional approach

The problem of database design is difficult, and it encompasses issues that may not be amenable to formalization [52]. Hence, any method is likely to have some limitations and drawbacks. However, this is not a reason to ignore the serious problems that the traditional approach is running into. Here

we summarize what we see, from our experience and perspective, as the most troublesome issues.

2.1.1 Failures of use and guidance

We claim that the traditional approach is not followed in practice. Indeed, Fitzgerald et al. [23] found that only about 11% of the consulted organizations claimed using an unmodified commercial information system methodology. Furthermore, Brodie and Liu [11] report that while 90% of all information systems inside a Fortune 100 company are relational, they could not find a single instance of an entity-relationship modeling in over ten such large corporations. The lack of modeling is not due to the lack of complexity: they report that a typical Fortune 100 company has about 10 thousand different information systems, that a typical relational database is made of over 100 tables, each containing between 50 to 200 attributes. Formalized conceptual models, as well as the theory developed around normalization, are not used. Physical modeling is frequently delayed until performance problems arise. In a very real way, we have entered a post-methodological era as far as the design of information systems is concerned [5]. The emergence of the Web has coincided with the death of the dominant methods based on the analytic thought and lead to the emergence of sense-making as a primary paradigm.

If one agrees that the traditional method is not used, the obvious question is: Why? Why do practitioners dismiss a method that has a solid theoretical basis and is the distillation of years of thought? It would be easy (and tempting!) to blame the designers or their training. But the tools themselves share a good part of the blame. They fail to give what designers need most, *guidance as to how to apply them*: for conceptual models, not enough guidance is given as how to create one, how to assess its quality, and — importantly — how to handle all information that does not fit into the conceptual model but may be relevant later for data quality of other purposes.

A critical failure in the traditional approach is that there is little guidance on how to discover important information (e.g., functional dependencies) in the real world. It would not be a concern if the rest of the design assumed that we lacked information. Yet unless we have *all* functional dependencies, there is no guarantee of normal form in the logical design. Thus, the logical design phase is *brittle*.

Ironically, the step where most research has focused on giving guidance is the last one, physical design [13], perhaps because it is easier to simulate realistically the problems and their solutions in a laboratory. However, this third step relies on the previous ones; while it can sometimes result in modifications of the database schema — as when denormalization is recommended, most approaches still assume that a schema has been well designed. By analogy, we could say that we know how to build the walls, as long as the foundation of the house is, somehow, done properly.

2.1.2 Failures of imagination

Even if one were to follow the steps of the traditional design method, and have a perfectly normalized, by-the-book database, what does one obtain?

We consider database design a matter of semantics: we are trying to capture the semantics of a domain, to represent information about that domain faithfully, and to (only) allow operations with the data that are meaningful. But traditional database design focuses on structure. In exchange for all the effort, we have insufficient semantics. This is the sense in which databases are *under-engineered*.

Consider, for instance, the problem of *information integration* [20, 54, 63]. Relational databases fail to provide enough information to determine automatically whether two databases contain information about distinct, overlapping, or similar domains. And yet, integration of information is increasingly critical: 40% of the cost associated with information systems is due to data integration problems [11]. To exemplify this trend toward greater integration and collaboration even in the most conservative settings, consider that the 9/11 Commission report urged the intelligence community to move from its need-to-know standard to a need-to-share approach [35]. Experts believe that the 9/11 tragedy could have been avoided with better data integration. The traditional way to design databases does not capture enough information to enable information integration — in fact, it falls short of capturing precisely the kind of information that would be more valuable for integration. Hence, traditional design not only fails to alleviate the problem, *it is helping to perpetuate it*. Most data integration approaches start by trying to determine the similarity between attributes.[1] Since most design approaches treat attributes as barely more than labels, one has usually only a string to work with: information *about* at-

[1]Several approaches rely on statistical properties of data, and choose not to try to interpret it [36]. It is unclear whether this is done in search of generality or due to need; but we believe that, while this approach provides important information, statistical properties cannot *establish* semantic similarity *by themselves*—but see Halevy et al. [28] for a different viewpoint.

tributes (metadata) is usually absent [53]. As long as design focuses on how to structure attributes in tables and not in what attributes mean, the problem will be with us. In the end, we ask practitioners to follow a model that is demanding and yields, in return, some very limited results.

The lack of appropriate metadata is even more acute in new applications, ranging from financial to legal systems. A prominent example is *e-science*: scientists need not only to store larger and larger amounts of data. They also need to be able to assess provenance [57], access rights, workflows, etc. in order to comply with ever increasing regulations, to be able to share the data, and to achieve the goal of *reproducible research* [58]. On this, traditional design offers no guidance.

To make matters worse, the focus on structure creates rigidity. Kiely and Fitzgerald [37] found that traditional information systems development methods were sometimes perceived to be of limited use within modern projects because they are too cumbersome and inflexible. This is the sense in which databases may be considered *over-engineered*. Consider the NoSQL movement [40]. A large force behind it are programmers for which database design makes no sense. Tired of the rigid structure of relational databases, other systems (Raven DB,[2] Amazon's SimpleDB,[3] Apache's CouchDB,[4] MongoDB[5]) are emerging. What good is it to design if it fails to make the developers more productive? Unfortunately, the mismatch between objects and program structures on one hand, and database structures on the other, is still largely unresolved. Motivated by this problem, Microsoft has proposed the Language-Integrated Query (LINQ) framework [44]. Other initiatives to bridge the gap have been developed over the years — witness to the fact that the problem is still with us.

3. WHY DOES IT FAIL?

The traditional design method was developed in the early seventies, when mainframes dominated information technology. It is in this era that the relational [17] and entity-relationship models [15] were invented. Accordingly, there are several assumptions behind the traditional design which reflect its age:

• *Users are faceless objects for whom (or on whose behalf) the systems are designed [31].*[6] In the seventies, the management of data was left in the hands of few experts who served the needs of technologically unsophisticated employees. Nowadays, the boundaries between users, whether they are employees or clients, and developers are blurred [46]. This is best illustrated with how *hashtags* emerged on the microblogging platform Twitter. Hashtags are a metadata convention among Twitter users [39], in the spirit of folksonomies [55]. Yet Twitter itself had no support for metadata. We can trace back the current convention to a single user who informally proposed it in a 140-character post in August 2007. Later, Twitter engineers recognized the convention and added software support for it. For example, Twitter detects "trending topics" using popular hashtags. A few other conventions, like the "retweet" were first initiated by the users. Sundara Nagarajan has recently expressed the same idea [48]: "Empowered end users cause application systems to evolve at tremendous speeds and continuously create new requirements for interoperation. For instance, a social networking site user can add content and pointers from a website, by simply dragging and dropping. The evolution of mashups that combine data and functionality from multiple sources is another example of this new design paradigm. This is leading to the evolution of the user experience, along with computation and data management." When systems are designed without the users, a lack of user engagement may result: 93% of all accounts in Business Intelligence systems are never used [45].

• *The information system is strongly consistent.* It has been estimated that Google alone has more than 1 million servers. Using cloud computing, anyone can use a distributed network of servers at a modest cost. With multiply located servers and deeply integrated web services, the CAP theorem [25] implies in practice that we have to choose between strong consistency and strong availability: we cannot have both. As a possible illustration of this constraint, the recent failure of an Oracle database at JPMorgan Chase, which froze $132 million in assets and lost thousands of loan applications, was blamed on an database [47] which required strong consistency for all data.

• *Semantics is absolute.* The original design assumed a centralized architecture. This architectural assumption had a reflection on the conceptual level, where one main viewpoint was assumed. While semantic relativism is pointed out [21, 51, 56], a choice must be made for a single model: there is no mech-

[2] `http://ravendb.net`
[3] `http://aws.amazon.com/simpledb/`
[4] `http://couchdb.apache.org/`
[5] `http://www.mongodb.org/`
[6] In the quote sometimes attributed to Frederick the Great, it's "everything for the people, but without the people".

anisms to derive other. Yet when different systems must interact routinely, we cannot expect that they all share the same viewpoint.

• *The models are static.* In the traditional setting, there is little need for evolution. Yet databases, even in large conventional corporations, are fast evolving: 30% of all information systems are modified significantly every year [11] in Fortune 100 companies. Chen, the father of the entity-relationship model, recently recognized the difficulty by pointing out the inability of existing modeling techniques to cope with fast-varying world states [16].

4. WHAT NOW?

Despite these difficulties, *research on database design has failed to make major progress in the last ten years or so.* This is not to say that no research is done. For instance, the series of conferences on entity-relationship modeling [50], while not totally focused on design issues, devote most of the program to them. Theoretical work is still ongoing in logical modeling [38]. Physical design research is still strong, sometimes driven by database vendors [3]. And there is still a community of dedicated researchers including notable researchers like Thalheim [61] and Olivé [49] among others.

But overall the topic is not widely pursued. For instance, if one checks the last 10 years of the "major" database conferences (SIGMOD, VLDB, ICDE), the number of papers in database design is low: leaving aside *physical design* (that is, the fine tuning of storage structures for better performance), we found less than ten talks *mainly* about database design [4, 9, 26, 34, 41, 42, 58, 62] and most of them examine design issues within the confines of a restricted context (sensor databases [41, 42], XML schema evolution [9], user interaction [62], scientific databases [58], data warehousing [34]).[7] There are certainly papers which, without having design as their main goal, bring considerable contributions to the table. For instance, the research on *dataspaces* by Halevy et al. [27] has brought forth the possibility of databases where the schema is implicit or at least not separated from the data and can evolve with it. It opens up some possibilities, but no paper on this project is about design *per se*. Likewise, research on semistructured data (e.g., XML) has exposed the database community to the thought that design must be more flexible [6]. However, little of this seems to have percolated to more traditional (relational, object relational) data models, and the design methodology for them. Finally, the total number of contributions remain low, even including this work, for such a crucial topic.

The fact is, *traditional database design is not a mainstream research topic.* We believe that this is due to two main facts: first, for most researchers, work on database conceptual models is seen as too difficult, because the subject is "soft", not clearly formalized, and does not yield itself well to the typical paper that one expects to see published in most conferences and technical journals. Second, work on relational design is considered useless as the topic is commonly taken as basically a mature and closed one. Certainly, there is always some more work that can be done (for instance, extending the idea of key and functional dependence to other models, like XML, has received some recent attention [14], as well as extending the concept itself to 'soft' functional dependencies [32]), but the subject is often considered "a solved problem".

On teaching, we note that while some textbooks are quite good about pointing out to students the limits and difficulties of the process, others simply gloss over the issues and give the impression that this a "case closed" situation — which may contribute to the lack of research in the area.

5. WHAT NEXT?

Traditional database design fails to provide the tools needed to design databases in today's environment, but researchers have not updated or expanded the methodologies enough to keep up with the times. Should we continue teaching methodologies which disappoint practitioners?

A first step towards renewed emphasis on database design research is to come up with a fresh and timely

[7] We do not claim any statistical validity for this observation. For one, the sample used is limited — more conferences, and certainly some journals, should be included. For another, there is a subjective aspect to this analysis. For the sake of transparency, we explain our method: first, the web page of each conference, reached through the web site of the organization behind the conference was used: this gives access to session titles, paper titles and sometimes abstracts. A search was made for keyword "design", another for keyword "normal" (to obtain 'normalization' and so on), and another for "semantic". We checked the title and abstract for each match (the last keyword generated quite a few matches). As stated, papers on physical design (database tuning and index design) were excluded. The list of papers obtained is given for interested readers to judge by themselves in the references above. Others may reasonably disagree exactly as to what to include, i.e., how to define 'mainly about database design' (but note that we include an invited talk and two tutorials!). However, unless one uses a generous notion of 'database design', we believe other people's results will be in the same order of magnitude as ours.

approach. Different researchers will likely have different viewpoints as to what are the most crucial or interesting problems. We submit the following research plan to open up a discussion.

Design for a distributed world.

- We must update database design methodologies for new environments that did not exist in the 1970s. Though there were many failed attempts to replace the ACID-compliant relational database systems with *better* alternatives, the landscape has finally begun changing with the adoption of cloud computing. For example, the data consistency requirements (and other issues affecting distribution) should be made explicit during the design phase, so that they can be exploited when deciding an architecture. In fact, many NoSQL designs assume that most operations can be kept local in order to ensure scalability, which means that one needs to know which data is likely to be involved in a transaction (logically related) in order to distribute the data ([59] makes the same point, implicitly). Along the same lines, deciding what can be made eventually consistent (versus what needs to be kept consistent at all times), and what to do in the face of inconsistency, should be based on the semantics of data. Hence, such issues should be part of the design phase.

- It is fashionable to talk about Big Data: one the main driver being this trend is our ability to quickly integrate diverse data sets to create new services. Correspondingly, *easier data integration should become one of the primary goals of good database design.* Another issue that Big Data brings is the distributed nature of the model. Do we need a 'distributed design' approach? For instance, should design produce more or less independent modules or 'chunks' of connected data, which can in turn be connected to each other in one or more ways? How would such a distributed design relate to Berners-Lee's linked data [7]? Or perhaps we should propose methodologies which, instead of starting from a clean slate, begin with the existing schemas (both within the organization, and public ones) and build on top of them. Should we shift the focus towards *extensions of what there is*?

Rethink functional dependencies.

- If Helland is right and normalization is for sissies ([30]), then one should question the focus on functional dependencies in database design. If this idea seems far-fetched, recall that data warehousing practitioners proposed a different design methodology (the star schema) that does not use the idea of functional dependency at all (rationalization of star schema in normal form came after the fact). The question then is whether there are other concept or concepts that can replace 'functional dependence' and be a good basis for design.

- We know that enforcing functional dependencies in the schema is insufficient to ensure that the data is semantically consistent. There are many rules, some expressible as constraints, assertions, or triggers, that could be enforced to ensure meaningful data. But current design mostly ignores this information. Shouldn't we attempt to capture this information, which is most likely to have an impact on the quality of our data, during the design? (In which case we need to define a way to measure the impact of different types of rules in data quality and consistency.) How can these various rules be used *together* in design? Note that to answer this question one has to answer other, more basic, questions: how do these different rules interact?

- Much of the database-design courses focus on functional dependence and normal form. It is often implied that the physical design ought to be a straight-forward application of the logical design. This is because, once, the equation *one relation = one table = one file* held for virtually all relational systems. Yet it not longer applies. For example, many distributed or column-oriented database systems replicate data for speed or reliability. Is it time to *completely* separate logical design from physical design, i.e. consider the relation as a purely conceptual entity?

Design for imperfect knowledge.

- We must cope with incomplete information (about the domain, the users, etc.) since in real systems, the scope or boundary of a database, or its future usage, is often uncertain [43]. Thus, design should proceed with as few assumptions as possible. Until now, a certain *closed world assumption* mentality trickles all the way from the conceptual model to the database. Clearly,

we live in an *open* world. Should be consider schemas as *descriptive* instead of *prescriptive*, which is what they are now? If so, what to do with data that does not follow the schema? Should any such data be allowed? Given the difficulty of determining in advance the type of data that the system may have to deal with, should the design include, for instance, a description of data that should *not* be allowed, and leave the database open to all other data? To some extend, XML schema languages (e.g., XML Schema and Relax NG) seem to adopt such a permissive attitude, with the added requirement that the data be structured in a hierarchical manner. Unfortunately, our experience is that the process is burdensome and is not widely used [10]. Are there lightweight alternatives?

- In turn, adopting an open world point of view will make it easier to support collaborative, evolutionary design as an integrated part of the workflow [18]. The issue here is, how do we design databases with *open*-world model while insuring the necessary consistency? If we are going to give permission to users to modify a schema, how much freedom should users have? For instance, one could study *whether design can be crowdsourced* (and if so, how and under what constraints). In general, one need to decide what kind of changes can be support, whether they come from the users or from a designer. A deeper study of *database evolution* could be of help here: could a system be designed that adapts its storage to changing schemas and requirements? Physical design is currently focused on *query workload*, that is, it adapts itself to the (changing) requirements posed by the database queries. Could some of these ideas be used to make the system reactive to changes in the schema? We find interesting that functional dependencies can be (roughly) classified as *natural* (one that reflects an invariant in the world: a person has only one height) or *artificial* (one that reflects a convention: each employee has to attend X meetings a month). The former are quite stable, but the latter are subject to change (note that all so-called *business rules* are artificial!). Should a system be able to cope with changes in artificial dependencies (old ones cease to hold, new ones are added)?

- New data stores in the NoSQL movement use non-relational data models: key/value, documents, extensible records [12]. Probably the first research task for such data models is a clarification of their exact structure and properties, since the terms are used somewhat loosely. But an immediate second is to decide whether they require a different approach to design (after all, even NoSQL data stores require design) or, to the contrary, whether design decisions can be kept independent of the data model. The question is not as trivial as it may seem: some of these new models allow an *open schema*, that is, one where the user can add attributes at will, while others still require, like relational databases, a *closed schema*, that is, one where all possible attributes are declared beforehand — yet others, like extensible records, combine both parts.

- Though there has been much work done on probabilistic databases [19] and soft functional dependencies [32], such subjects remain almost entirely distinct from database design. Yet semantics are not always absolute: some relationships are merely almost always true. Thus, it is likely that there are many more soft dependencies or conditional dependencies than 'standard' functional dependencies. (A conditional dependency is one that holds only under certain circumstances. For example, at some places, a married couple is always made of a man and a woman, but not at others.) Current design practices tend to ignore all functional dependencies but the standard ones, which are but an extreme case [22]. Should we make room in database-design methodologies for probabilistic metadata and several types of dependencies? If so, how would different types of dependencies be used? How would they behave when put together?

No doubt, different researchers will have different viewpoints on these issues. Some may object to some of the challenges included here; others may wish to direct attention to other problems not included here. We stress again that this plan is meant to start the discussion; let the debate begin.

References

[1] S. Abiteboul et al. The Lowell database research self-assessment. *Communications of the ACM*, 48:111–118, May 2005.

[2] R. Agrawal et al. The Claremont report on database research. *SIGMOD Record*, 37:9–19, September 2008.

[3] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala. Database tuning advisor for Microsoft SQL Server 2005: demo. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 930–932, New York, NY, USA, 2005. ACM.

[4] P. Andritsos, R. J. Miller, and P. Tsaparas. Information-theoretic tools for mining database structure from large data sets. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 731–742, New York, NY, USA, 2004. ACM.

[5] D. E. Avison and G. Fitzgerald. Where now for development methodologies? *Communications of the ACM*, 46:78–82, January 2003.

[6] D. Barbosa, J. Freire, and A. O. Mendelzon. Designing information-preserving mapping schemes for XML. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 109–120. VLDB Endowment, 2005.

[7] T. Berners-Lee. Linked data. *International Journal on Semantic Web and Information Systems*, 4(2), 2006.

[8] P. Bernstein et al. The Asilomar report on database research. *SIGMOD Record*, 27:74–80, December 1998.

[9] K. Beyer, F. Özcan, S. Saiprasad, and B. Van der Linden. DB2/XML: designing for evolution. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 948–952, New York, NY, USA, 2005. ACM.

[10] T. Bray. Don't Invent XML Languages. `http://bit.ly/364VEy` [last checked on 10/07/2011], 2006.

[11] M. L. Brodie and J. T. Liu. The power and limits of relational technology in the age of information ecosystems. Keynote at On The Move Federated Conferences, 2010.

[12] R. Cattell. Scalable SQL and NoSQL data stores. *Sigmod Record*, 39(2):12–27, 2010.

[13] S. Chauduri and V. Nrasayya. Self-tuning database systems: A decade of progress. In *Proceedings of VLDB*. VLDB Endowment, September 2007.

[14] H. Chen, H. Liao, and Z. Gao. Functional dependencies for XML. In *Proceedings of the 2010 international conference on Web-age information management*, WAIM'10, pages 110–115, Berlin, Heidelberg, 2010. Springer-Verlag.

[15] P. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[16] P. Chen. Suggested research directions for a new frontier: Active conceptual modeling. In D. Embley, A. Olivé, and S. Ram, editors, *Conceptual Modeling - ER 2006*, volume 4215 of *Lecture Notes in Computer Science*, pages 1–4. Springer Berlin / Heidelberg, 2006.

[17] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13:377–387, June 1970.

[18] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. MAD skills: new analysis practices for big data. *Proc. VLDB Endow.*, 2:1481–1492, August 2009.

[19] N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Commun. ACM*, 52:86–94, July 2009.

[20] A. Doan, P. Domingos, and A. Y. Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3), 2003.

[21] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison Wesley, 6th edition, 2010.

[22] W. Fan, F. Geerts, J. Li, and M. Xiong. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering*, 23:683–698, 2011.

[23] G. Fitzgerald, A. Philippides, and S. Probert. Information systems development, maintenance and enhancement: findings from a UK study. *International Journal of Information Management*, 19(4):319–328, 1999.

[24] K. E. Gebaly and A. Aboulnaga. Robustness in automatic physical database design. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, EDBT '08, pages 145–156, New York, NY, USA, 2008. ACM.

[25] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, 33(2):59, 2002.

[26] L. Golab, H. J. Karloff, F. Korn, and D. Srivastava. Data auditor: Exploring data quality and semantics using pattern tableaux. *PVLDB*, 3(2):1641–1644, 2010.

[27] A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '06, pages 1–9, New York, NY, USA, 2006. ACM.

[28] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009.

[29] T. Halpin. *Conceptual Schema and Relational Database Design*. Prentice Hall, 1989.

[30] P. Helland. Normalization is for sissies. In *Conference on Innovative Database Systems Research (CIDR)*, 2009.

[31] J. Iivari, H. Isomäki, and S. Pekkola. The user, the great unknown of systems development: reasons, forms, challenges, experiences and intellectual contributions of user involvement. *Information Systems Journal*, 20(2):109–117, 2010.

[32] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga. Cords: automatic discovery of correlations and soft functional dependencies. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 647–658, New York, NY, USA, 2004. ACM.

[33] J. Impagliazzo. Computing curricula 2005. *ACM SIGCSE Bulletin*, 38(3):311–311, 2006.

[34] M. Jarke, C. Quix, D. Calvanese, M. Lenzerini, E. Franconi, S. Ligoudistianos, P. Vassiliadis, and Y. Vassiliou. Concept based design of data warehouses: the DWQ demonstrators. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 591–, New York, NY, USA, 2000. ACM.

[35] C. Jones. Intelligence reform: The logic of information sharing. *Intelligence & National Security*, 22(3):384–401, 2007.

[36] J. Kang and J. Naughton. Schema matching using interattribute dependencies. *IEEE Transactions on Knowledge and Data Engineering*, 20(10), October 2008.

[37] G. Kiely and B. Fitzgerald. An investigation of the use of methods within information systems development projects. *The Electronic Journal of Information Systems in Developing Countries*, 22, 2005.

[38] S. Kolahi and L. Libkin. An information-theoretic analysis of worst-case redundancy in database design. *ACM Transactions on Database Systems*, 35:5:1–5:32, February 2008.

[39] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 591–600, New York, NY, USA, 2010. ACM.

[40] N. Leavitt. Will NoSQL databases live up to their promise? *Computer*, 43(2):12–14, 2010.

[41] Q. Luo and H. Wu. System design issues in sensor databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 1182–1185, New York, NY, USA, 2007. ACM.

[42] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 491–502, New York, NY, USA, 2003. ACM.

[43] M. Magnani and D. Montesi. A survey on uncertainty management in data integration. *J. Data and Information Quality*, 2:5:1–5:33, July 2010.

[44] E. Meijer, B. Beckman, and G. Bierman. LINQ: reconciling object, relations and XML in the .NET framework. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 706–706, New York, NY, USA, 2006. ACM.

[45] R. Meredith and P. O'Donnell. A functional model of social media and its application to business intelligence. In *Proceeding of the 2010 conference on Bridging the Socio-technical Gap in Decision Support Systems: Challenges for the Next Decade*, pages 129–140, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[46] F. Millerand and K. Baker. Who are the users? Who are the developers? Webs of users and developers in the development process of a technical standard. *Information Systems Journal*, 20(2):137–161, 2010.

[47] C. Monash. Details of the JPMorgan Chase Oracle database outage. `http://bit.ly/ckWLfq` [last checked on 10/07/2011], 2010.

[48] S. Nagarajan. Guest editor introduction. *Data Storage Evolution, Special Issue of Computing Now*, March 2011. IEEE Press.

[49] A. Olivé. *Conceptual Modeling of Information Systems*. Springer, 2007.

[50] J. Parsons, M. Saeki, P. Shoval, C. C. Woo, and Y. Wand, editors. *Conceptual Modeling - ER 2010, 29th International Conference on Conceptual Modeling*, Lecture Notes in Computer Science 6412, Vancouver, BC, Canada, November 2010. Springer.

[51] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, 3rd edition, 2002.

[52] J. F. Roddick, A. Ceglar, D. De Vries, and S. La-Ongsri. Active conceptual modeling of learning. In P. Chen and L. Y. Wong, editors, *Postponing schema definition: low instance-to-entity ratio (LItER) modelling*, pages 206–216, Berlin, Heidelberg, 2007. Springer-Verlag.

[53] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to falilitate interoperability among heterogeneous information systems. *ACM Trans. Database Syst.*, 19(2):254–290, 1994.

[54] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. Tech report dit-04-087, University of Trento, 2005. Also: Journal of Data Semantics, LNCS 3730, pp. 146-171, 2005.

[55] S. Siersdorfer and S. Sizov. Social recommender systems for web 2.0 folksonomies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, HT '09, pages 261–270, New York, NY, USA, 2009. ACM.

[56] A. Silberschatz, H. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, 6th edition, 2010.

[57] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34:31–36, September 2005.

[58] E. Stolte, C. von Praun, G. Alonso, and T. Gross. Scientific data repositories: designing for a moving target. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 349–360, New York, NY, USA, 2003. ACM.

[59] M. Stonebraker and R. Cattell. 10 rules for scalable performance in 'simple operations' datastores. *Communications of the ACM*, 54(6):72–80, 2011.

[60] T. J. Teorey, S. S. Lightstone, T. Nadeau, and H. V. Jagadish. *Database Modeling and Design, Fifth Edition: Logical Design*. Morgan Kaufmann, 5th edition, 2011.

[61] B. Thalheim. *Fundamentals of Entity-Relationship Modeling*. Springer-Verlag, 2000.

[62] D. Tunkelang. Design for interaction. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 969–970, New York, NY, USA, 2009. ACM.

[63] O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. In *Proceedings of SIGMOD*, pages 449–460, 2007.