The Vivification Problem in Real-Time Business Intelligence

Patricia C. Arocena, Renée J. Miller, and John Mylopoulos

University of Toronto, Toronto M5S 3G4, Canada
{prg,miller,jm}@cs.toronto.edu

Abstract. In the new era of Business Intelligence (BI) technology, transforming massive amounts of data into high-quality business information is essential. To achieve this, two non-overlapping worlds need to be aligned: the Information Technology (IT) world, represented by an organization's operational data sources and the technologies that manage them (data warehouses, schemas, queries, ...), and the business world, portrayed by business plans, strategies and goals that an organization aspires to fulfill. Alignment in this context means mapping business queries into BI queries, and interpreting the data retrieved from the BI query in business terms. We call the creation of this interpretation the vivification problem. The main thesis of this position paper is that solutions to the vivification problem should be based on a formal framework that explicates assumptions and the other ingredients (schemas, queries, etc.) that affect it. Also, that there should be a correctness condition that explicates when a response to a business schema query is correct. The paper defines the vivification problem in detail and sketches approaches towards a solution.

Key words: data exchange, vivification, incompleteness, uncertainty, business intelligence

1 Introduction

Every time a Business Intelligence (BI) query is evaluated, the data that are returned need to be *interpreted*. Interpretation involves mapping data to business entities, processes and goals that correspond to BI data. For example, a hospital database may contain data about patient names, addresses and health insurance numbers (hi#). Interpretation of these data means that some of them (e.g., name = 'John Smith', address = '40 St. George Street', hi# = 1234567) are ascribed to one patient. Likewise, the database may contain data about events, a hospital admission, an assignment to a ward, and a surgical operation. Interpretation, in this case, means that these events are associated with the entities that participated in them. Moreover, these events are grouped into aggregates, where each represents an instance of a business process. For example, an aggregate such as 'HospitalVisit' models the day-to-day activities in a hospital. This form of interpretation has been called *vivification*, in the sense that it brings data to life (hence, vivifies), or makes data more vivid [13].

The past decade has seen unprecedented interest in BI technologies and services, and a corresponding growth of the BI market. By now, most competitive organizations have a significant investment in BI, much of it technology-related, based on software tools and artefacts. But business people - be they executives, managers, consultants, or analysts - are in general agreement that what helps them the most is business data analyzed in business terms: strategic objectives, business models and strategies, business processes, markets, trends and risks. This gap between the world of business and the world of IT-supplied data remains today the greatest barrier to the adoption of BI technologies, as well as the greatest cost factor in their application to specific projects.

Today's business world requirements for information-on-demand and agility exacerbate this gap. Information-on-demand is hampered when BI tools handle data integration and cleansing of IT-supplied data in isolation from any form of business analysis and interpretation. Pressed with increasing volumes of data and deployment times, this approach usually results in a tendency to integrate and clean as much data as possible with very little business perspective in mind. During the consolidation phase, BI systems (and thus the technologies managing data warehouses over which BI applications are built) make vivification choices when confronted with *incomplete* and *inconsistent* information. But as data and business semantics continuously evolve, *one-size-fits-all* interpretations of the data are rarely sufficient for business analysts. In practice, the evaluation of business queries and interpretation of results amount to a continuous social discovery process between IT and business users, where the latter 'fill in the blanks' and routinely introduce assumptions to obtain the desired level and type of data completeness for decision making.

Agility means that *real-time change* is inevitable and ever-present in a business context, and must be factored into the interpretation and decision making processes as much as in everything else. Real-time requires the ability to access information in an organization whenever it is required [6]. In a real-time enterprise, business policies, objectives and processes change, obstacles and opportunities are encountered and must be dealt in a timely fashion. In our work, we seek solutions to enable BI systems to be able to react to these changes and graciously adapt any interpretation choices made so far without further need of re-architecting the data warehouse or redesigning ETL scripts. This adaptation must be performed at the same time scale as changes in semantics. So when a new business policy is invoked or a law changed, we must be able to adapt the interpretation in a timely fashion.

To bridge the gap between the business and IT worlds, we propose to use a *business model* (also known as business schema) to represent business concepts (e.g., business goals, entities and processes [12]) and, moreover, *schema mappings* from a database schema to a business model to specify interpretations. Data are then interpreted in terms of these concepts. Unfortunately (but unsurprisingly!), the vivification of data in terms of business model concepts is not a simple one-to-one mapping. Much like the task of finding who visited a crime scene on the basis of footprints, there are in general many possible mappings from data to business entities and processes. For instance, the hospital database may mention

N times patients named 'John Smith'. These may correspond to 1, 2, ... or N different patients. Likewise, a surgical operation may be part of one patient's hospital visit or another's, depending on the data associated with the operation, and how these data have been interpreted.

In this position paper, we elaborate on the need for a systematization of the vivification process, to make vivification rigorous and traceable. Specifically, we argue that solutions to the vivification problem should be based on a formal framework that explicates assumptions, records context, provenance and the other ingredients that entail an interpretation. Moreover, vivification should come with a correctness condition for a response to a business schema query to be correct. In other words, we call for vivification to be practiced with the same rigour that has been applied to query processing for more than three decades.

The rest of the paper is structured as follows. In Section 2, we discuss our research baseline. In Section 3, we consider how vivification can be used in BI, that is, we outline elements of a formal framework to bring data to life. In Sections 4 and 5, we present some steps involved in our envisioned vivification process and we illustrate using preliminary results from our case study. In Section 6, we summarize related work. Last, we conclude in Section 7.

2 Research Baseline

A business schema is most useful if it is instantiated with complete and accurate information. This information is naturally derived from the IT-supplied databases (or warehouses), but since these databases are often designed for a different purpose (supporting operational systems or supporting BI reporting), it is rare that they provide complete and consistent business data. Hence, business users must deal with missing and uncertain information on a daily basis. Next, we review two foundational techniques for interpreting data.

2.1 Knowledge Base Vivification

Consider a knowledge base (KB) as a finite collection of facts about the world of interest. We say a KB is *incomplete* when it asserts that one of many facts is true, but without being able to individually determine the truth of each individual fact. A KB indicating that 'John Smith' as a recently admitted inpatient is either in the Cardiology Ward or in the Maternity Unit, without saying which of the two locations holds, is considered incomplete. Incompleteness in this example arises due to a disjunction in the representation of the facts. In general, incompleteness may be attributable to a number of logical constructs, such as disjunction, existential quantification and negation [13].

To act in the face of incompleteness, Levesque proposed a technique called *vivification* [13] for transforming an incomplete KB into a complete representation, where reasoning can be reduced to standard database querying. Vivifying a KB amounts to eliminating incompleteness, and most importantly, placing symbols in the KB into a *one-to-one correspondence* to objects of interest in the

world and their relationships. For example, confronted with the disjunctive fact that 'John Smith' is either in the Cardiology Ward or in the Maternity Unit, we might be able to resolve this issue by looking up the location of the admitting physician's office. In doing so, we may use the relationship between admitted inpatients and admitting doctors in the business to establish proper connections among the data.

The notion of vivification is relevant to the task of interpreting the world of IT-supplied data in terms of the world of business. First, it corresponds well to the common *ad hoc* practice of 'filling in the blanks' to coerce data into a form that is complete and adequate for business querying. Second, various types of business data, such as business plans, strategies and goals, are already conveyed using vivid or visual descriptions. This is, ultimately, the type of data business users have in their mind when issuing queries. We propose to use vivification to provide a principled foundation for doing the data-to-business interpretation.

2.2 Data Exchange

Vivification is also important in translating between two databases [17]. Schema mappings are used to express the relationship between two database schemas. Given a schema mapping, the problem of data exchange was introduced by Fagin et al. [11] and defined as the problem of translating data structured under a source schema into an instance of a target schema that reflects the source data as accurately as possible. An important aspect of this work was the recognition that even in exchanging data between two IT database schemas, it is rare that a complete target instance (database) can be created. The notion of a *universal solution* was introduced as the best solution for representing data as accurately as possible, but these instances are often *incomplete* as we exemplify next.

Example 1. Consider a simple schema mapping stating how to populate a target insurance database from a source inpatient registry table:

$$\texttt{Registry}\ (name, dob, addr) \rightarrow \exists\ \textbf{policyId}\ \texttt{Insurance}\ (\textbf{policyId}, name, dob)$$

We use source-to-target tuple-generating dependencies (s-t tgds) to express the mapping [11]. This mapping indicates that for each inpatient registry tuple, there must be an associated insurance record in the target, whose name and date of birth values are the same as those of the inpatient registry. The variables $name, dob, addr$ are universally quantified. Notice that the target expression (right-hand-side of the implication) uses an existential variable `policyId` to indicate that the policy number is unspecified. Figure 1 shows a source instance I and a universal solution (target instance) J with labeled `NULL` values (i.e., N_1, N_2 and N_3) instead of concrete policy numbers.

Schema mappings record the decisions made in translating data. Data exchange permits the precise modeling of the uncertainty in a translation. In the above target instance, the use of three labeled `NULL`s indicates that the three tuples could refer to three different people, or to just two people (perhaps Marge moved so the first two `NULL`s should share a single value) or to a single person

(perhaps there were inconsistencies in entering information and all three tuples refer to one person). If a business needs to track the number of patients, someone must make a choice on how to resolve this uncertainty. We propose to build on the foundation of data exchange to view the vivification task as one of removing incompleteness or uncertainty from a schema mapping. We advocate a declarative approach where vivification decisions are documented in a formalism that supports automated reasoning (in the same way schema mappings permit formal reasoning about a translation) and in which we can define precisely when an interpretation (or set of vivification decisions) is vivid.

Source Instance I

Name	DOB	Address
Marge Wood	05-05-1927	Markham
Marge Wood	05-05-1927	London
Margret Wood	05-05-1925	London

Target Instance J

PolicyId	Name	DOB
N_1	Marge Wood	05-05-1927
N_2	Marge Wood	05-05-1927
N_3	Margret Wood	05-05-1925

Fig. 1. A Data Exchange Scenario: Source Instance and Universal Solution

3 Formal Framework

We now present elements of a formal framework for interpreting data in BI, using the concepts of a database schema, business schema, business mapping, vivification assumptions and business queries.

Database Schema. A source database schema comprises a number of relations $R_1, R_2, \ldots, R_n$, some of which may be recording information about the instrumentation of business processes. Our proposal is general enough to accommodate not only mainstream relational databases but also specific technologies such as data warehouses.

Business Schema. A target business schema, on the other side, offers a conceptual model of a particular organization in terms of business concepts $T_1, T_2, \ldots, T_m$, such as business entities, business processes, and business events. This description may be given in some business modeling formalism [12].

Business Mapping. To describe the relationship between the source and the target schemas, we use a set $\mathcal{M}$ of *business schema mapping rules*. Some mapping rules indicate a correspondence between database tuples and business entities, and some others between observed events in the data and business events in the process model. The mapping language we use for illustration is based on the widely used database concept of s-t tgds [11]. Each mapping rule may indicate a correspondence between a query over the source database ϕ_{DB} and a query over the target ψ_{BS} involving either business entities or business events. Formally, each mapping rule is of the form:

$$\forall \mathbf{z}, \mathbf{x}(\phi_{DB}(\mathbf{z}, \mathbf{x}) \rightarrow \exists \mathbf{y} \psi_{BS}(\mathbf{x}, \mathbf{y})) \tag{1}$$

Intuitively, $\mathbf{x}$ represents the information that is exchanged from the source into the target, and $\mathbf{y}$ represents information that is unknown in the target.

Assumptions. We also use a set $\mathcal{A}$ of *vivification assumptions* to characterize desired properties or constraints over any instantiation of the target schema. These may be given using any constraint language, such as tuple-generating dependencies (tgds) and equality-generating dependencies (egds) [2]. In addition to having functional dependencies and cardinality constraints, we may also have assumptions that state a particular choice among a set of possible facts to resolve incompleteness or inconsistency.

Business Queries. Last, we have a set $\mathcal{Q}$ of *business queries* in some query language formalism. We assume a traditional query answering semantics over complete databases which will determine for a given query what parts of a given business schema instance must be complete to compute a certain query answer. For example, if we consider Example 1, using this mapping alone, we would not be able to answer a query that counts the number of insurance policies. This mapping leaves the insurance policy identifiers incomplete, and therefore leaves incomplete the decision on whether two policies are the same. However, a query that asks if Marge Wood is a patient can be answered with certainty. We can use certain query answering semantics [3] to reason about whether an instance of a business schema is sufficient to answer with certainty a given set of queries.

Definition 1. *A BI setting* $(\mathtt{DB}, \mathtt{BS}, \mathcal{M}, \mathcal{A}, \mathcal{Q})$ *consists of a database schema* $\mathtt{DB}$*, a business schema* $\mathtt{BS}$*, a set* $\mathcal{M}$ *of business mapping rules, a set* $\mathcal{A}$ *of vivification assumptions, and a set* $\mathcal{Q}$ *of business queries. Let* I *be a source instance over* $\mathtt{DB}$ *and* J *a target instance over* $\mathtt{BS}$ *such that* $\langle I, J\rangle$ *satisfy* $\mathcal{M}$ *and* J *satisfies* $\mathcal{A}$*.*

- *We say that* J *is* vivid *if any tuple in* J *is null-free and all constants in* J *are justified by either* I *(the source instance at hand) or by* $\mathcal{A}$*.*
- *We say that* J *is* vivid for a set of queries $\mathcal{Q}$ *if for every query* $q \in \mathcal{Q}$*, the sets of possible and certain answers for* $q(J)$ *coincide.*

It follows from this definition that a vivid business schema instance needs to be *discovered*. This needs to be done in the light of concepts in the business schema, any given vivification assumptions, and a set of business queries of interest. To do this, our approach is to develop a refinement process for schema mappings that step-by-step leads to more vivid target instances. We use the business queries to understand what vivification is necessary to ensure business users receive certain answers that are based on documented assumptions.

A First Solution. We now provide a brief overview of our initial solution to the vivification problem. In Arocena et al. [5], we discuss specific strategies for vivifying underspecified attribute values arising from existential quantification in business mapping rules. In particular, as illustrated in Figure 2 Part (a), we consider how to remove *incompleteness* and complement this with a consideration of how to remove *uncertainty* that arises due to inconsistency. Incompleteness may arise when a mapping rule fails to specify what value should be given to a

target attribute or when the database I fails to provide a value, thus rendering an incomplete `NULL` value during mapping execution. Uncertainty may arise due to having more than one way of populating a given target relation or due to a database that does not enforce business-level requirements (for example, a business rule saying every patient must provide a single health insurance policy and the database provides multiple ones).

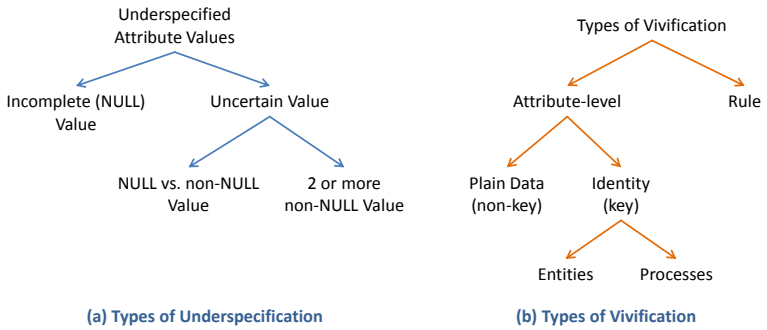

Fig. 2. Types of Underspecification and Vivification in Business Mapping Rules

Our approach uses formal reasoning about a business mapping and a set of assumptions to determine when they are sufficient to produce a vivid instance or an instance that is vivid with respect to a set of queries. To do this, we first consider the attribute positions at which incompleteness or uncertainty occurs in mapping rules, and then we consider the set of business queries and their requirements. In Example 1, for a query that counts patients, we need to make a decision (which we record in our vivification assumptions $\mathcal{A}$) on whether `NULL` values over the target attribute represent the same or different values), but we do not need to map each patient to their policy number. Of course, if such a mapping can be found, this is a valid way to vivify the incompleteness caused by the existential variable in the mapping rule. In general, when applying *attribute-level* vivification strategies, we distinguish between non-key and key attribute positions (shown in Figure 2 Part (b)). We refer to these as *data vivification* and *identity vivification*, respectively. The first is concerned with supplying one or more certain data values to a target business concept. The second one attempts to bring operational data to life in terms of business entities and processes by correctly associating data with specific instances of real-world business concepts. In the next section, we overview our proposed process.

4 Towards an Iterative Vivification Process

Figure 3 illustrates our proposed approach to vivification. We envision a process where steps can be iteratively applied as real-time information becomes available or changes and, moreover, as technologists and business users further their own understanding and knowledge of the deployed business solution.

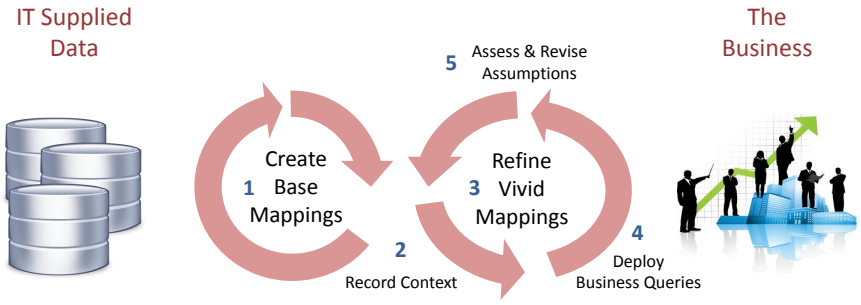

Fig. 3. An Iterative Vivification Process

Step 1: Create Base Mappings. Our vivification process starts by considering a *source database schema* and *target business schema*. The database and business schemas can be very different as they are often developed and evolve independently. The output of this step is an initial set of *business mapping rules* specifying the relationship between concepts in these two disparate schemas, as currently understood by users. These mapping rules may be derived using semi-automatic schema mapping systems, such as Clio [10] and ++Spicy [15], or extensions of these approaches designed for mapping into a conceptual schema [4].

Step 2: Record Context. Before attempting to map any data, we look to the business schema, the business process model, and importantly, to business queries to understand implicit contextual assumptions. In establishing a proper *context of interpretation*, we may also consider user-defined assumptions. This step helps to identify which incomplete or uncertain attribute values need to be vivified to instantiate the business schema with complete high-quality data. Recording the context of interpretation is crucial for understanding what types of vivification strategies may be applied and, later on, for understanding the rationale behind query responses.

Step 3: Refine Vivid Mappings. In this step, we use business context (i.e., the context of interpretation created in Step 2) to guide mapping vivification. We view this task as one of removing incompleteness or uncertainty from business mapping rules. This involves systematically choosing and applying vivification strategies. These include finding mapping tables or finding default values from the context to resolve existential variables and using business rules to resolve inconsistencies [5]. The output (after iteration) is a *refined vivified business mapping*, which chooses among all possible interpretations of the data, one that is consistent with the context of interpretation. We also record the *provenance* of these decisions.

Step 4: Deploy Queries. We use the refined vivified mapping of Step 3 to instantiate the target business schema with complete high-quality data. Once this happens, it can be queried by business users. Notably, most *business queries*

require the computation of information across business concepts. This computation often relies on *grouping* instances of a given business concept by common attributes (e.g., date of occurrence, location, participating entities, etc.) and, optionally, performing *aggregation* of these or some other attributes of interest to create business relevant performance indicators. Grouping and aggregation place strong requirements on the completeness of data when the ultimate goal is to obtain query responses that enable effective decision making.

Step 5: Assess and Revise Assumptions. A decisive step in our vivification process is that of assessing and revising the *assumptions* used during the interpretation of data. A new business mandate, a change in business process models, or even an improved business user's understanding of the application domain may all cause certain assumptions to not hold true. In such cases, we need to revise our assumptions and possibly *retract* any vivification done based on changed or deleted assumptions. Real-time change is ever-present in a business context and must be dealt with to ensure up-to-date consistency and completeness guarantees, as required by the business queries.

5 Validation: A Prospective Case Study

To validate our proposal, we are conducting a case study which considers the flow of patients attending an Emergency Department (ED) in a health care organization in Ontario. In this section, we briefly overview our initial undertaking.

Various health organizations in Ontario are deploying the Daily Activity Report Tool (DART), a performance management toolkit that monitors improvements to patient flow within hospitals. DART collects daily Emergency Department (ED) and General Internal Medicine (GIM) activity data to help illustrate a hospital's patient flow from start to end. In our work, we consider a *target business schema* outlining a basic *ED process model*, based on the schema developed by Barone et al. [7]. A small portion of our schema is depicted in Figure 4. The data collected by DART (described as 'Output Data' in the figure) can be seen as an instantiation of the process model. From this data, DART computes a set of mandatory *business queries* (also known as indicators) that can be used to track hospital performance, such as 'Time to Physician Initial Assessment (PIA)' and 'ED Length of Stay (LOS)' indicating the length of a patient's stay in the ED during one episode of care. The aim of DART is to empower hospital stakeholders in their daily discussions regarding patient flow.

Interpretive Challenges. We use a *source database schema*, which models operational hospital data, based on that of a leading hospital in Ontario. Most relational tables record information about patients and observed events with respect to activities that occur on or on behalf of patients. Interestingly, the concept of patient as an entity does not exist in the database. Instead, the recording of information hinges upon the concept of ED Visit (or episode of care), and various location-based, status and census data items. Moreover, across hospital sites, the hospital may use different chart or medical record identifiers

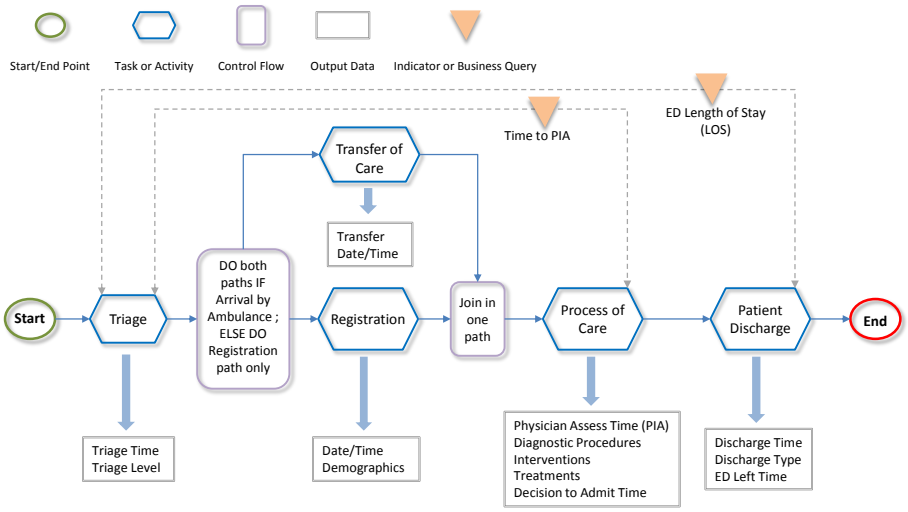

Fig. 4. Emergency Department Patient Workflow

to refer to a person's involvement in an episode of care. As a result, counting ED patients at any given point in time is a challenge (surprisingly, this count happen to be one of the most widely used indicators within DART). Another challenge encountered in our application domain is that of missing and uncertain information. Missing values mostly occur over the following business schema target attributes: Triage Acuity Level, Triage Time, Physician Assessment Time (PIA), Physician Identifier and ED Departure Time. In some cases, it has been estimated that close to 25% of ED visits may have missing data [1]. For example, census times are mostly recorded manually and thus, when not recorded, they may impact the interpretation of business indicators.

Our Approach. As we proposed in Section 4, we are deploying an iterative vivification process over our case study. We briefly illustrate the use of our techniques with two examples.

Example 2. A relevant business query in our case study is that of computing the average ED Time to Physician Initial Assessment (PIA) in a hospital. This permits understanding how long patients wait before being seen by a physician (and thus before any diagnostic procedures or interventions are being prescribed). To provide meaningful query responses, we need a complete business schema instance. An initial course of action may be that of vivifying incomplete or uncertain PIA times by using a default value. Such default value may vary by type of hospital (e.g., community vs. teaching), patient's acuity level and ED volume, and may be input into our process as a vivification assumption. This vivification strategy may be sufficient to interpret the operational data at hand within a local business environment, but it might be deemed unsuitable if later on, the query response is attached to a pay-for-performance funding model. Under

this new scenario, the government may require stricter strategies for imputing missing values, or may deem imputation impermissible. By recording vivification decisions, our framework is able to accommodate real-time changes and revise any decisions made during the interpretation process. This may involve adopting a different vivification strategy that counts in the indicator only patients for whom the database contains complete information.

Example 3. As another example, consider how vivification offers insight into how to tally the ED patient counts in a hospital. Initially, we may attempt to identify patients based on the concept of ED Visit, as suggested by DART guidelines. We may choose to vivify patient identifiers by generating unique identifiers (i.e., Skolem terms) from existing ED Visit identifiers. However, this strategy might introduce a bias and business users might need to understand from the vivified mapping how patients are counted. Nowadays, business users are mostly given numbers but not the intuition or rationale about the formulae used. A frequently ignored fact, which influences patient counts, is that of having patients visiting the ED more than once in a single day. An improved understanding of this situation may then cause a revision of the entity resolution strategy used on patients, and the adoption of one that successfully separates patients from their multiple ED visits. If we add the date of a hospital visit to patient identifiers to indicate a visit, we can distinguish multiple visits, but only on different days. The business user needs to decide if this is sufficient or if multiple visits on a single day need to be counted separately. This type of decision should not be hardcoded in procedural logic decided by the IT department. Our framework actively supports the declarative representation and modification of such decisions by a business user.

6 Related Work

Our work builds upon the foundation of knowledge base vivification [13] and data exchange between disparate databases [11, 17]. The data-to-business interpretation semantics that we propose in this paper adapts the notion of certain answers to a vivification context [3]. Close in spirit to our work is that of Sismanis et al. [19]. This work proposes a resolution-aware query answering model for OLAP applications, where uncertainty about business entities is handled dynamically at query time; unlike ours, the work does not discuss how to deal with incomplete or uncertain attribute values in a general BI context. The problems of duplicate detection (a.k.a entity identification, data deduplication or record linkage) and data fusion have prominent roles in the data integration literature [8, 9]. Numerous techniques have been proposed for vivifying real-world entities (deduplication) and their multiple possibly inconsistent descriptions (data fusion). Most of these cleansing techniques are still applicable in our proposed framework notwithstanding, the main differences between the two appear to be first, the point of intervention at which the techniques are being applied and second, our emphasis on declaratively recording these resolution decisions. In particular,

we view mapping rules as crucial tools for recording and reasoning about incompleteness and uncertainty. Among recently studied information quality criteria, Naumann's completeness of data measures the amount of `non-NULL` values in a database instance (this is done in comparison to a potentially complete universal relation) [16]. This criterion embodies a model that can also be used to calculate the completeness of query answers which result from combining sources through merge operators; as such, this work can be leveraged in our framework to encode specific completeness assumptions or constraints over business interpretations. Our work also relates to both data exchange and semantic web efforts on bridging open and closed world reasoning [14, 18]. Moreover, our work complements that of van der Aalst et al. [20] in Business Process Modeling (BPM). While their emphasis is on discovering and enhancing business process models from event logs in the data, ours is on providing an interpretation from observed events with respect to business models. Last, to the best of our knowledge, current BI/ETL tools offer little to no support for systematically using *adaptive* incompleteness rules within their data analysis and interpretation processes. Indeed, a primary motivation for our work is the desire to put the many stages of BI processing (e.g., data acquisition, consolidation, analysis, and evaluation of queries) into a framework that openly and declaratively records at every BI stage, the assumptions and choices made to deal with incompleteness and uncertainty. The interpretation of `NULL` values must be done in the context of the answers business users seek, i.e., it cannot be done solely in a batch offline mode during the consolidation stage.

7 Concluding Remarks

In bridging the gap between the world of IT-supplied data and the world of business, we are working on a number of techniques for transforming operational data into complete high-quality business information. Current BI solutions tend to adopt a two phase approach to solve this problem, where the mapping of data generally precedes the process of imparting a business semantics to it. In this paper, we have advocated on-demand interpretation of IT-supplied data in terms of business concepts.

As for our future work, in one direction, we are developing a general mapping framework for embracing incompleteness. We are investigating how to determine, given an incomplete business mapping and a set of business queries, when is it possible to create a vivid business schema instance. We study a number of business query types and how rewritten mappings can be used to satisfy these queries. We also propose a declarative mechanism for expressing preference rules that aims at encapsulating default domain assumptions. In another direction, we plan to finalize our case study that implements and evaluates our vivification techniques.

Acknowledgements. The authors wish to acknowledge Alex Borgida for his insightful comments and suggestions, and also the anonymous reviewers for their feedback. This work was partially supported by the NSERC Business Intelligence Network.

References

1. Understanding Emergency Wait Times: Who is Using Emergency Departments and How Long are They Waiting. Canadian Institute for Health Information, 2005.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.
4. Y. An, A. Borgida, R. J. Miller, and J. Mylopoulos. A Semantic Approach to Discovering Schema Mapping Expressions. In *ICDE*, pages 206–215, 2007.
5. P. C. Arocena, R. J. Miller, and J. Mylopoulos. Vivification in Business Intelligence: From Data to Business Entities and Processes. In *Topics in Business Intelligence, Morgan & Claypool*, To Appear in 2012/2013.
6. B. Azvine, Z. Cui, D. Nauck, and B. Majeed. Real-Time Business Intelligence for the Adaptive Enterprise. In *IEEE CEC/EEE*, page 29, 2006.
7. D. Barone, T. Topaloglou, and J. Mylopoulos. Business Intelligence Modeling in Action: A Hospital Case Study. *CAiSE*, pages 502–517, 2012.
8. J. Bleiholder and F. Naumann. Data Fusion. *ACM Comput. Surveys*, 41(1), 2008.
9. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.
10. R. Fagin, L. M. Haas, M. A. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis. Clio: Schema Mapping Creation and Data Exchange. In *Conceptual Modeling: Foundations and Applications*, pages 198–236, 2009.
11. R. Fagin, P. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *Theor. Computer Science*, 336(1):89–124, 2005.
12. L. Jiang, D. Barone, D. Amyot, and J. Mylopoulos. Strategic Models for Business Intelligence. In *ER*, pages 429–439, 2011.
13. H. J. Levesque. Making Believers out of Computers. *Artif. Intell.*, 30(1):81–108, 1986.
14. L. Libkin. Data exchange and Incomplete Information. In *PODS*, pages 60–69, 2006.
15. B. Marnette, G. Mecca, P. Papotti, S. Raunich, and D. Santoro. ++Spicy: an OpenSource Tool for Second-Generation Schema Mapping and Data Exchange. *PVLDB*, 4(12):1438–1441, 2011.
16. F. Naumann. *Quality-Driven Query Answering for Integrated Information Systems*, volume 2261 of *Lecture Notes in Computer Science*. Springer, 2002.
17. L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.
18. K. Sengupta, A. A. Krisnadhi, and P. Hitzler. Local Closed World Semantics: Grounded Circumscription for OWL. In *ISWC*, pages 617–632, 2011.
19. Y. Sismanis, L. Wang, A. Fuxman, P. J. Haas, and B. Reinwald. Resolution-Aware Query Answering for Business Intelligence. In *ICDE*, pages 976–987, 2009.
20. W. M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.