Universität Passau
Fakultät für Informatik und Mathematik

type of work

Implementierung einer verteilten Umgebung von medizinischen Datenquellen

David Goeth

24. November 2015

type of work
am Lehrstuhl für verteilte Informationssysteme
der Fakultät für Informatik und Mathematik
der Universität Passau

Erstgutachter: Prof. Dr. Harald Kosch
Betreuer: Armelle Natacha Ndjafa

Abstract

This is an abstract.

Contents

List of Tables

List of Figures

1 Summaries of read papers

1.1 Towards a Model for Multimedia Dataspaces

In the paper 'Towards a Model for Multimedia Dataspaces'[1] the authors present a representation model for dataspaces, which is an approach based on the dataspace and dataspace-view paradigm for uniformly represent structured and semi structured data, ontologies and other similar knowledge representation models. With that model, the authors address the current explosion of digital information and data sources. Many branches (i.a. in the medicine) need now large amounts of distributed, heterogeneous data.

However, traditional data integration methods are barely applicable to formulate search queries on a distributed, heterogeneous data set, containing files with many different file formats, as traditional data integration systems are designed for complete structured data. Thus, these systems need a global schema for calculating search queries. Addressing this issue, the concept of dataspaces was developed as an abstraction of wide-area, heterogeneous and distributed data management. For dataspaces there is no need of upfront efforts to semantically integrate data before basic services such as keyword search can be provided.

Dataspaces support uncertainties in schema-mapping and consider that schema mapping from sources to mediated schema may be incorrect. A further interesting feature of dataspaces is successive data integration. So, the system is able to integrate data in iterations as the time goes on depending on user needs.

Research in the field of dataspaces is currently very active, but despite of the deep interest, existing models suffer on a number of shortcomings limiting their applicability. These include the tendency of overlooking the different types of relations that can exist between data items, which restricts the amount of information they can generally integrate. Second, they focus on classical text data ignoring the specifics of multimedia data. Third they don't provide the fine-granularity in the persistence of integrated data that is needed in certain domains.

To address these issues, the authors developed a dataspace model, which sees the dataspace as a set of classes, objects (instances of classes) and relations. In the latter case there exist relations between classes (CRC), relations between objects and classes (ORC) and relations between objects (ORO). Furthermore relations can be internal or external defining whether the anticipating relation objects are in the same data source or in different ones.

A design goal of the model is the maximization of its expression in terms of the types of relations that it can represent, enabling it to deal with information originating from structured data, semi-structured data, ontologies and other forms of knowledge representation as well as from canonical knowledge.

Import to note is the fact that the model includes similarity relations in the type of relations. Similarity functions are a feature of multimedia data that defines a measurement for non-exact matching between objects. This kind of relations can be used to derive relations between other objects in the dataspace. Additionally the model introduces the concept of dataspace-view. This makes it possible to store query results of an existing dataspace into a new sub-dataspace with different modes of persistence(virtualized view, materialized view, mode of synchronization with the content of the original sources,...).

1.2 Databases to Dataspaces - A New Abstraction for Information Management

Dataspaces describes a new abstraction of data management [2]. In current scenarios it is rarely the case, that data to be managed is solely stored in a convenient relational Database Management System (DBMS) or in another, single data model. Therefore, developers often face the challenge to deal with heterogeneous data on a low level base. Challenges are to provide search and query capabilities, enforcing rules, integrity constraints, naming conventions etc.; tracking lineage; providing availability, recovery and access control; and managing evolution of data and metadata. That issues are ubiquitous and arise in enterprises, government agencies and even on one's PC desktop.

As a response to this problem, the authors postulate the concept of dataspaces and corresponding to this, the development of DataSpace Support Platforms (DSSPs). Shortly said, the latter provides an environment of cooperating Services and guaranties, that enables software developers to concentrate on their specific application problem rather than taking care of returning issues in consistency and efficiency of huge, linked but heterogeneous data sets. The remarkable properties of a dataspace system are defined as follows:

A DSSP must deal with data and applications in a variety of file formats, that are accessible through many systems with different interfaces. A DSSP has to support all kinds of data in the dataspace rather than only a few (as DBMSs do). Although a DSSP provides a integrated possibility for searching, querying, updating and administration, the data often can only be accessible and modifiable through native interfaces. Therefore DSSPs haven't full control over their data. Queries on a DSSP may offer varying levels of services. In some cases the answers can be approximated resp. best-effort. An example: If some data sources are unavailable for some reasons the DSSP is able to return the best result as possible. Therefore it uses the data that are available at the time of the query. A DSSP has to provide the tools that allow a tighter data integration process in the dataspace as necessary.

Many of the services a dataspace provide, data integration and exchange systems provide, too. The main difference between these systems is that data integration systems need a semantic integration process before they can provide any services on the data. But dataspace is not kind of a classic data integration approach. In order to avoid semantic integration, a dataspace uses the concept of data coexistence. The idea is to provide base functionality over all data sources, regardless of their specific integration constraints. E.g. a DSSP is able to provide a keyword search similar to a desktop file search. If more sophisticated operations are required such as relational queries, data mining or monitoring of specific data sources, additional effort can be done to integrate the sources tighter. This incremental process is also called as "pay-as-you-go" fashion.

In chapter 3 of the paper the authors pottered at the logical components and services a DSSP should have:

Logical components

A Dataspace should contain all information being relevant for a specific organization/task regardless of their file format or storage location, and it should model a col-

lection of relationships between the data repositories. Therefore the authors define the dataspace as a set of participants and relationships. Participants of a dataspace are individual data sources . Some participants support expressive query languages for querying while others only provide limited access interfaces. Participants can reach from structured, semi-structured right up to unstructured data sources. Some sources provide traditional updates, some are only be appendable, while others are immutable. Further, dataspaces can be nested within each other, which means that a dataspace should be able to be a part of another dataspace. Thus, a dataspace has to provide methods and rules for accessing its sub dataspaces.

Services of dataspaces

The most basic services a dataspace should contain is the cataloging of data elements of all participants. A catalog is an inventory of data resources containing all important information about every element (source, name, storage location inside the source, size, creation date, owner, etc.)of the dataspace. It is the infrastructure for the most other dataspace services. Search and query are two main services a DSSP must provide. A user should be able to state a search query and iteratively precise it, when appropriate, to a database-style query. For the dataspace approach it is a key tenet that search should be applicable to all of the contents, regardless of their formats. The search should include both data and meta-data. The user should be enabled to discover relevant data sources and inquire about the completeness, correctness and freshness. A DSSP in fact should be aware of the gaps in its coverage of the domain. A DSSP should also support updating data. Of course, the mutability of the relevant data sources determines the effects of updates. Other key services would be monitoring, event detection and the support for complex workflows (e.g. it is desired that a calculation is done if new data arrives and that the result will be distributed over a set of data sources). On a similar way a DSSP should support various forms of data mining and analysis. Not every participant will provide all necessary interfaces for being able to support all DSSP features. Hence it is necessary, that data sources can be extended on various ways. E.g. a source don't store its own meta-data, so there has to be an external meta-data repository for it.

Components and architecture of a dataspace system

Catalog and Browse Information about all participants and the relationships among them are stored in the catalog. It must also deal with a large variety of sources and supports to provide different levels of information about their structure and capabilities. It is important, that the catalog includes the schema of the source, statistics, rates of change, accuracy, completeness query answering capabilities, ownership, and access and privacy policies for each participant. Relationships may be stored as query transformations, dependency graphs or even textual descriptions. If possible, the catalog should include a basic inventory of the data elements at each participant: identifier, type, creation data and so forth. Then, it can support basic browsing over the participant's inventories. It isn't a very scalable interface, but it can be used to response to questions about the presence or absence of a data element, or determine which participants have documents of a specific type. On top of the catalog, the DSSP should have a model-management environment allowing the creation of new relationships and manipulation of existing ones.

Search and Query The component has to offer the following capabilities:

(1) *Query everything:* Any data item should be queryable by the user regardless of the file format or data model. Keyword queries should be supported, initially. When more information about a participant is collected, it should be possible to gradually support more sophisticated queries. The transition between keyword query, browsing and structured querying should be gracefully. And when answers are given to keyword (or structured) queries, the user should be able to refine the query through additional query interfaces.

(2)*Structured query:* Queries similar to database ones should be supported on common interfaces (i.e. mediated schemas) that provide access to multiple sources or can be posed on a specific data source (using its own schema). The intention is, that answers will be obtained from other sources (as in peer-data management systems), too. Queries can be posed in a variety of languages (and underlying data models) and should be translated into other data models and schemas as best as possible with the use of exact and approximate semantic mappings.

(3) *Meta-data queries:* It is essential, that the system supports a huge variety of meta-data queries. These include (a) source inclusion of an answer or how it was derived or computed, (b) timesteps provision of the data items that are included in the computation of an answer, (c) specification of whether other data items may depend on on a particular data item and the ability to support hypothetical queries. A hypothetical question would be 'What would change if I removed data item X?. (d) Querying the sources and degree of uncertainty about the answers. Queries locating data, where the answers are data sources rather than specific data items, should be supported, too.

(4) *Monitoring:* All stated Search and Query services should also be supported in an incremental form which is also applicable in real-time to streaming or modified data sources. It can be done either as a stateless process, in which data items are considered individually, or as a statefull process. In the latter multiple data items are considered.

Local store and index A DSSP has a store and index component to achieve the following goals: to create efficiently queryable associations between data items in different participants. Important is here, that the inde should identify information across participants when certain tokens appear in multiple ones (in a sense, a generalization of a join index) to improve accesses to data items with limited access patterns. Here, the index has to be robust in the face of multiple references to real-world objects, e.g. different ways to refer to a company or person. to answer certain queries without accessing actual data sources. Thus the query load is reduced on participants which cannot allow ad-hoc external queries. to support high availability and recovery

The index has to be highly adaptive to heterogeneous environments. It should take as input any token appearing in the dataspace and return the location at which the token appears and the roles at each occurrence. Occurrences could be a string in a text file, element in file path, a value in a database, element in a schema or tag in a XML file.

Discovery Component This components locates participants in a dataspace, creates relationships between them, and helps administrators to refine and tighten these

relationships. For each participant the component should perform an initial classification according to the participant's type and content. The system should provide an environment for semi-automatically creating relationships between existing participants and refining and maintaining existing ones. This involves both finding which pairs of participants are likely to be related, and then proposing relationships which a human can verify and refine. The discovery component should also monitor the content in order to propose additional relationships in the dataspace over time.

Source Extension Component Some participants may don't provide significant data management functions. For example, a participant might be no more than departmental document repository, perhaps with no services than weekly backups. A DSSP should support to enrich such a participant with additional capabilities, such as a schema, a catalog, keyword search and update monitoring. It may be necessary to provide these extensions locally as there can be existing applications or workflows that assume the current formats or directory structures. This component also supports "value-added" - information held by the DSSP, but not present in the initial participants. Such information can include "lexical crosswalks" between vocabularies, translation tables for coded values, classifications and ratings of documents, and annotations or linked attached data set or document contents. Such information must be able to span participants in order to link related data items.

1.3 Principles of Dataspace Systems

The document "Principles of Dataspace System" of Alon Haley, Michael Franklin and David Meier[3] is based on their previous work "From Databases to Dataspaces - A New Abstraction for Information Management"[2] where the concept of dataspaces was firstly presented, and poses specific technical challenges realizing Dataspace Support Platforms (DSSPs) relating to query answering, introspection and the benefits of human attention for improving semantic relationships within a dataspace. In the following are stated the results of the above mentioned challenges:

1. Query answering

Queries are posed in a wide range of different languages. Most of the activities will properly begin with a keyword search, but it will also be common to see queries as a result of a form (which results to queries with several selectable predicates). It will come to more complex queries when a user interacts deeper with a certain data source. If it isn't explicitly stated, it it usual, that a user is likely to believe that a query considers all relevant data in a dataspace, regardless of the used data model or schema. Even if a query is posed to a data source, it is implicitly expected that the system considers the data of other sources, as well. If additional answers are desired, one has to do transformations on the schema and the data model.

1.1.Challenges of answer querying

Answers corresponded to queries of a dataspace are different from traditional queries in several ways. The challenges to it are analyzed more deeply in the third chapter:

Ranking: Queries are typically sorted by their relevance, similar to a web search engine. Ranking is necessary not only for keyword search, but also for structured queries, when transitions to other data sources should be approximated.

Heterogenity: Answers will come from many sources and will differ in their used data model and schema. The Ranking has to manage heterogeneity, too.

Sources as answers: In addition to base elements (e.g. documents or tuples), a DSSP should be able to provide sources, as well. This means that it returns links to locations where additional answers can be found.

Iterative queries: Normally, the interaction with a dataspace can't be reduced to the process of posing a sole query and getting an answer to it. Instead, a user is involved in an information finding task that requires a sequence of queries, each being a refinement or modification on the previous ones.

Reflection: It is expected, that a DSSP reflects on the completeness of its coverage and the accuracy of its answers.

2. Dealing with challenges

The first step to face these challenges is to build a formal model. Therefore, the authors propose five directives:

Challenge 3.1. The development of a formal model for analyzing the query answering in a dataspace.

Sub-Challenge 3.2. The development of intuitive semantics to answer a query taking into consideration a sequence of earlier posed queries.

Sub-Challenge 3.3. The development of a model for an information finding task which includes operations on lower levels.

Sub-Challenge 3.4. The development of algorithms sorting the data sources according to how likely they are to contain the answer. These algorithms should start from a keyword query and operate on a large collection of data sources.

Sub-Challenge 3.5. The development of methods for ranking answers retrieved by multiply heterogeneous data sources.

3. Query answering model and challenges

Challenge 3.7. The development of techniques to answer queries based on the following ideas or combinations of it:

- apply several approximate or uncertain mappings and compare the answers obtained by each.

- apply keyword search techniques to obtain some data or some constants that can be used in instantiating mappings.

- examine previous queries and answers obtained from data sources in the dataspace and try to infer mappings between the data sources. Whenever we have access to queries that span multiple data sources, try to infer from them how the sources are related (e.g., the join attributes should provide some hint of common domains).

- Develop a formal model for approximating semantic mappings and for measuring the accuracy of answers obtained with them.

- Develop automatic best-effort methods for translating a query over one data set onto the other.

4. Introspection

In chapter 4 introspection within a dataspace is analyzed. Introspection describes on this occasion the ability of a dataspace to observe assumptions and uncertainties and specifying their origin. In contrast to traditional databases introspection isn't a nice feature but a necessity. Introspection has to be possible over the following highly related parameters: Lineage, Uncertainty and Inconsistency. Thus, one speaks of LUI introspection in the context of dataspaces.

4.1.1.Uncertain databases

Uncertainty arises in applications for data management if the exact state of the (data) world is not known. The goal of a uncertain database is to represent a set of possible states the world can have. Usually, the states are referred to as possible worlds. Every possible world represents a complete valid state of the database. For declaring these states there were developed several formalisms [4, 5, 6, 7, 8]. Examples for formalisms would be a-tuples, x-tuples or c-tables[9].

4.1.2. Inconsistency in databases

The task of an inconsistency database is to handle situations where the database contains conflicting data. A common example would be the existence of two values of a salary of an employee whereby the values are obtained from different data sources. The essence for solving this problem is to consider all possible repairs. A repair is

a minimal change resulting the database to a consistent state. As a rule, there are more one one possibility which is why the database owns several repairs to choose from. That's also the reason for the tight relationship between uncertainty and inconsistency as inconsistency can be interpreted as uncertainty over the knowledge, which of the conflicting value is the best solution.

4.1.3. Modeling of data lineage

The lineage of a tuple reports how the tuple was derived from a certain data set. There is a distinction to internal and external lineage. Internal lineage applies to tuples of a query result – lineage specifies here how the tuple was derived within a database. External lineage relates to tuples inserted into a database, the lineage relates hereby to the external sources or processes by which the tuples were inserted.

4.1.4 LUI Introspection

A DSSP should provide a uniform mechanism for modeling uncertainty, inconsistency and lineage. Thereby the following challenges arise:

Challenge 4.1. Develop a formalism for enabling the modeling of uncertainty, inconsistency and lineage.

Sub-Challenge 4.2. Develop a formalism which captures the uncertainty over general forms of inconsistency in databases.

As inconsistency results to a special type of uncertainty, the formalism for uncertainty should leverage this special structure. The formalism for uncertainty tells us only what the possible states of the world are and sometimes it assigns every possible world its corresponding possibility. But in many cases, it is the only way to resolve uncertainty and to get knowledge about data lineage and how and where the data was derived from.

Web search engines already unify uncertainty and lineage in a simple way. One of the main reasons why the authors want to unify lineage and uncertainty is to reason the relationship between external sources and their impact on answers. Thereby arises the following challenges:

Sub-Challenge 4.3. Develop a formalism which represents the external lineage and reasons it.

To combine uncertainty and lineage, there exists

Sub-Challenge 4.4. Develop a general technique for extending every kind of uncertainty and for investigating the representative and calculative advantages of this procedure.

To achieve this, the object that should be assigned uncertainty, has to match to the object lineage is attributed to.

Uncertainty on Views

There's a general problem with modeling uncertainty as the uncertainty formalism associate uncertainty with a single schematic construct: tuples in the case of x-tuples and attribute values in the case of a-tuples. So the chose of database schema and normalization limits the kinds of uncertainty to express. In the case of using views, the authors propose to link uncertainty with the view's tuples, hence:

Sub-Problem 4.5. Develop formalisms where uncertainty can be attached to tuples in views and view uncertainty can be used to derive uncertainty of other tuples

4.2 Finding the right answer

This chapter address to what has to be done to determine the quality of a query answer (how good is a query?).

Sub-Challenge 4.6. Define metrics for comparing the quality of query answers and answer sets over a dataspace, and find efficient techniques to process queries.

A limited version of this issue was already addressed with minimal repairs related to inconsistent databases. Here it will be tried to construct a consistent database as tight as possible on the base of the inconsistent one. For improving the results, it is necessary to do some preferences, which results in the following challenges:

Sub-Challenge 4.7. Develop query-language extensions and their corresponding semantics that enable specifying preferences on answer sets along the dimensions of completeness and precision, certainty and inconsistency, lineage preferences and latency.

Along with the specification of preferences there are needed methods for reasoning query answer sets. This is essential for comparing answer sets. Query containment [10] will be extended in the context to dataspaces. By which the following challenge submits to:

Sub-Challenge 4.8. Define notions of query containment that take into consideration completeness and precision, uncertainty and inconsistency and lineage of answers, and efficient algorithms for computing containment.

The following issues stands in relation to section 4.1. of the paper:

Sub-Challenge 4.9. Develop methods for efficient processing of queries over uncertain and inconsistent data that conserve the external and internal lineage of the answers. Study whether existing query processors can be leveraged for this goal.

1.4 Towards Realization of Dataspaces

In the paper "Towards Realization of Dataspaces", written by Ibrahim Elsayed and Peter Brezany[1], the architecture and functionality of a Dataspace Management System(DMS) is presented. Additionally it is discussed, how grid technology of future implementations is able to support such an architecture. A DMS is a set of software programs, which controls the organization, storage and retrieval of data in a Dataspace. It also handles the security and integrity of a Dataspace. In the following, the architecture and the functionality requirements of a DMS will be described.

1. Requirements

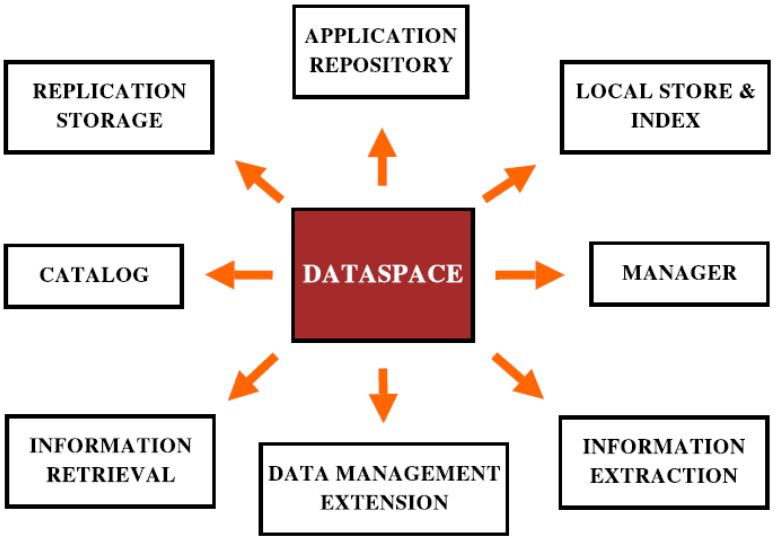

Figure 1.1: Environment of a Dataspace

Information Retrieval: Queries and searches are two different methods of information retrieval and represent together one of the main services a DMS should support. Generally, queries and searches should be supported by all participants of the Dataspace, regardless of their used data model. It shouldn't make a difference for a user to operate on a sole database or on a Dataspace. A good known and simple search operation is keyword searching. The Support of spanning such a search method over all participants, is a challenging research topic. The development of methods for keyword searching on relational and XML databases was done by the data-engineering community [2,3,4], For supporting a global query functionality allowing to formulate uniform queries on all Dataspace participants, intelligent methods for interpreting and translating of queries in several languages are required. Methods for query translation were investigated by a large body of research communities [5,6].

Information Extraction: The world-wide web has become a large data container, by now. For allowing to postprocess data obtained from the web, techniques for web information extraction, extracting relevant information from semi-structured web pages should be supported. For further processing, extracted content has to

be converted to structured data and saved locally. Non structured web documents have to be classified first using text mining techniques, which organize the parsed documents into groups with the help of ontologies[7]. Based on these ontologies, a keyword search is possible to find individual documents. Structured documents allow easier access and integration due to the rich semantically information included in the data representation.

Data Management Extensions: This component provides possibilities to improve low-level working Dataspace components. All these base components have only limited data management capabilities. It is a task of a DMS to provide additional functionality such as backup, recovery and replication.

Catalog: Contains a detailed description of all participants of the Dataspace. Beside basic information about the participants, such as owner, creation date, etc., the description for each participant should also contain semantic information about its data. A user should be able to browse the catalog for getting more detailed information about certain data sources. The catalog may reference a meta-data repository to separate the basic and more detailed descriptions.

Manager: In order to realize the above mentioned functionality, a central component managing the system and interacting with the user is needed. Alongside user authentication, right assignments and other services, this manager component is responsible for communicating with all participants. Thus, this component serves as an interface between the users and the participants of the Dataspace.

Local Store and Index: This component is responsible for caching search and query results, so that certain queries can be answered without the need of accessing the actual data sources. Furthermore it supports the creation of queryable associations between the participants.

Replication Storage: Allows to copy participant data in order to increase its access time. This results in high availability and high recovery is supported.

Application Repository: With this component, the user is able to share data analysis tools, domain specific models, evaluations, etc., which can be applied to the (available) data of the Dataspace.

2. Dataspace Management System

2.1Architecture

Generallly, a Dataspace consists of Dataspace components, the so-called participants, and relationships between the participants. Components are individual data sources, such as relational databases, text databases, XML databases, data stream systems, web services or other data archives.

A Dataspace component includes a description for describing what data it contains, which storage formats are used and which query mechanisms are supported. In addition it contains information about the storage location of the data and whether the data was replicated, equally it contains information about relationships to other participants. The information and data are marked if they are added by the management system, the user or were automatically generated.

Each participant is described the same way as the components and is subsequently registered within the catalog. Thus they are available as data sources for a more

abstracted layer, which provides global query and search methods. This global layer is embedded in the DMS. It supports the user to create new Dataspace components, to add relationships between other participants, to register participants within the catalog, to browse the catalog for available participants, to decide whether to share all or only selected participants within a community including assignments of rights, and to search all or only selected participants of a Dataspace.

As each participants is managed by its own and thus is responsible for its data management such as data update, recovery and replica, it isn't a task of the DMS to change the data represented by the participants. Therefore, the DMS has no administration privileges for writing and changing content of the data of a participant. But the DMS provides possibilities to add missing data management services, such as data replication, transformation, download, upload, etc..

2.2 Managing Sub-Dataspaces

The idea behind a Sub-Dataspace is to create a new Dataspace that is included by a another and thus is a subset of a bigger Dataspace. This functionality is needed for access privileges and security management. An administrator of a Dataspace is thus able to define different data access rights for certain groups.

Bibliography

[1] A. Ndjafa, H. Kosch, D. Coquil, and L. Brunie, "Towards a model for multimedia dataspaces," in *Multimedia on the Web (MMWeb), 2011 Workshop on*, pp. 33–37, Sept 2011.

[2] M. Franklin, A. Halevy, and D. Maier, "From databases to dataspaces: A new abstraction for information management," *SIGMOD Rec.*, vol. 34, pp. 27–33, Dec. 2005.

[3] 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, (New York, NY, USA), pp. 1–9, ACM, 2006.

[4] R. H. S. Abiteboul and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995.

[5] D. Barbará, H. Garcia-Molina, and D. Porter, "The management of probabilistic data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 4, pp. 487–502, Oct. 1992.

[6] A. H. A. Das Sarma, O. Benjelloun and J. Widom., "Working models for uncertain data," *Proc. of ICDE*, April 2006.

[7] G. Grahne, "Dependency satisfaction in databases with incomplete information," in *Proceedings of the 10th International Conference on Very Large Data Bases*, VLDB '84, (San Francisco, CA, USA), pp. 37–45, Morgan Kaufmann Publishers Inc., 1984.

[8] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian, "Probview: A flexible probabilistic database system," *ACM Trans. Database Syst.*, vol. 22, pp. 419–469, Sept. 1997.

[9] G. Grahne, "Conditional tables," in *Encyclopedia of Database Systems*, pp. 446–447, 2009.

[10] A. K. Chandra and P. M. Merlin, "Optimal implementation of conjunctive queries in relational data bases," in *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, (New York, NY, USA), pp. 77–90, ACM, 1977.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

LOCATION, den DATE

David Goeth