Analysis of existing databases at the logical level: the DBA companion project

Fabien De Marchi[1], Stéphane Lopes[2], Jean-Marc Petit[1], Farouk Toumani[1]

[1] Laboratoire LIMOS, CNRS UMR 6158
Université Blaise Pascal - Clermont-Ferrand II
24, avenue des Landais, 63 177 Aubière cedex, France

[2] Laboratoire PRISM, CNRS FRE 2510
45, avenue des Etats-Unis, 78035 Versailles Cedex, France

January 15, 2003

Abstract

Whereas physical database tuning has received a lot of attention over the last decade, logical database tuning seems to be under-studied. We have developed a project called *DBA Companion* devoted to the understanding of logical database constraints from which logical database tuning can be achieved.
In this setting, two main data mining issues need to be addressed: the first one is the design of efficient algorithms for functional dependencies and inclusion dependencies inference and the second one is about the interestingness of the discovered knowledge. In this paper, we point out some relationships between database analysis and data mining. In this setting, we sketch the underlying themes of our approach. Some database applications that could benefit from our project are also described, including logical database tuning.

1 Introduction and motivations

Today's Relational DataBase Management Systems (RDBMS) require DataBase Administrators (DBA) to tune more and more parameters for an optimal use of their databases. Due to the difficulty of such a task and since a large number of companies cannot justify a full-time DBA presence, simplifying administration of RDBMS is becoming a new challenge for the database community: The idea is to have databases adjusting themselves to the characteristics of their applications [2]. For example, in the context of the *AutoAdmin* project [22], physical database tuning is investigated to improve performances of the database, e.g. index definitions or automatic statistic gathering from SQL workloads.

In the same spirit, existing logical database constraints should be fully understood. For example, providing the DBA with functional dependencies (FDs) and inclusion dependencies (INDs) holding in her database is particularly critical not only for improving data consistency but also for ensuring application performances.

Understanding existing databases is a necessary step when the database has to evolve for instance to better match user's requirements or when hardware/software evolution has to be performed. Many database applications, such as database reverse engineering, data interoperability or semantic query optimization to mention a few, assume the availability of data semantics, mainly conveyed by FDs and INDs over existing databases. However, there is no guar-

antee at all such kind of knowledge is a priori known.

We have developed a project called DBA Companion devoted to the understanding of logical database constraints from which logical database tuning can be achieved [16, 15, 4]. A prototype has been developed [14]: its objective is to be able to connect any database (independently of the underlying DBMS) in order to give some insights to DBA/analyst such as:

- the FDs and INDs satisfied in her database,
- small examples of her database, thanks to Informative Armstrong Databases. The same benefits when the design by example were introduced [23, 17] are also expected in this slightly different context (database maintenance vs database design).

Paper organization Section 2 introduces the main underlying problems to achieve logical database tuning such as functional and inclusion dependency discovery. Some applications which could benefit from these dependencies are sketched in Section 3. We conclude in Section 4.

2 Data mining issues

Since data semantics is mainly conveyed by FDs and INDs in relational databases, it gives rise to two data mining problems. In this setting, two main issues need to be addressed: the former is about the design of efficient algorithms for FDs and INDs inference and the latter is about the interestingness of the discovered knowledge.

Example 1 Let us consider a database describing orders of products of a company. A running example is given in table 1 and will be reused throughout the paper.

Table 1: A running example

Product

pid	name	price
1	Chai	$ 18.00
2	Tofu	$ 18.00
3	Paté	$ 12.30
4	Paté	$ 50.00

Order

oid	date	shipVia	pid	qty
10509	11-10-1998	Federal Shipping	1	2
10509	11-10-1998	Federal Shipping	2	1
10510	1-5-2000	United Package	3	1
10511	12-12-1998	Speedy Express	2	2

2.1 FD inference

The problem of discovering FDs is stated as follows: "given a relation r, find a small cover of the FDs that hold in r". Discovering functional dependencies satisfied by a database has been addressed by various approaches, among which we quote [18, 11, 15, 20].

In [15], a general framework has been proposed based upon the concept of *agree set* [1, 18, 7]. Given a relation, an agree set is an attribute set having the same values in at least a couple of tuples in this relation. Informally, such a set conveys information about FDs not satisfied by a couple of tuples. More precicely, it is a closed set with respect to FD closure. From agree sets, maximal sets (also called meet-irreducible sets) can be easily derived and then, many problems related to FD inference can be achieved *without accessing the database anymore*. Examples of related problems are key discovery, approximate FD inference, normal form tests and Armstrong relation generation from FDs that hold in the relation being analyzed [9].

The most important point of this framework is that *a relation is queried only once*, i.e. when agree sets are computed.

A data mining approach and a database approach using SQL have been proposed to compute agree sets from a relation [15]. Both approaches performed equally well but the latter is fully integrated in the DBMS, which is by far more realistic in our context (no pre-treatment is necessary).

Example 2 A cover of FDs satisfied in `Product` and `Order` are: { `Product:{pid}` → `{name,price}`, `Product:{name,price}` → `{pid}`, `Order: {oid}` → `{date,shipVia}`, `Order: {date}` → `{oid}`, `Order: {shipVia}` → `{oid}` , `Order: {pid,qty}` → `{oid}` , `Order: {oid,pid}` → `{qty}` , `Order: {oid,qty}` → `{pid}` }

2.2 IND inference

The problem of inclusion dependency inference in relational databases can be formulated as follows: "given a database d, find a small cover of INDs that hold in d". As far as we know, the discovery of inclusion dependencies has raised little interest in Database and KDD research communities [12, 16]).

In our project, a levelwise algorithm called `Mind` has been devised to discover a cover of INDs satisfied by a database [4]. It extends the well known `AprioriGen` function for association rules discovery to generate new candidate INDs of size $i+1$ from satisfied INDs of size i. It is worth noting that in [4] a new proposition for unary IND inference has been made from which IND discovery becomes more realistic in practical cases.

Example 3 INDs satisfied in the database given in Table 1 are : { `Order[pid]` $\subseteq$ `Product[pid]`, `Order[qty]` $\subseteq$ `Product[pid]`, `Order[qty]` $\subseteq$ `Order[pid]` }

2.3 Interestingness of the discovered knowledge

Once FDs and INDs satisfied by a given database are discovered, they have not the "same weight", i.e. only a subset of them are worth to take into account when for instance a logical database tuning process has to be carried out. Eliciting such subsets can be guided by DBA' expertise or SQL workloads.

We propose two approaches to deal with this important task in a KDD process: the former is based on the intuition that attributes implied in so called *logical navigation*, convey more information than other attributes and the latter is based on the capability of Armstrong relations to represent in a condensed form all the discovered knowledge through small examples.

2.3.1 Heuristics based on logical navigation

The *logical navigation* can be defined as a set of (duplicated) attributes on which join statements are performed in application programs. The basic claim is that duplicated attributes vehicle more semantics than other ones. These attributes can be identified from join statements of SQL workloads representative of database activity. Such SQL workloads are easily gathered in recent RDBMS.

Interestingness of INDs We can define the notion of interestingness for INDs as being those INDs which concern duplicated attributes over relation schemas. In [16], we exploit the *logical navigation* as a guess to automatically find out only interesting INDs, being understood that some of them can be missed. Thus, it can be seen as a pre-treatment of a data mining algorithm to decrease the number of candidate attributes.

Example 4 Continuing our example, it seems quiet reasonable to assume that in a representative workload of the database activity, a join between `Order[pid]` $\bowtie$ `Product[pid]` does exist. Therefore, from example 3 only one IND is considered as interesting: `Order[pid]` $\subseteq$ `Product[pid]`. It is worth noting that other INDs, such as e.g. `Order[qty]` $\subseteq$ `Product[pid]`, do not reflect integrity constraints, rather accidental dependencies.

Interestingness of FDs Basically, the idea which has been developed so far for INDs can be reused for FDs: the idea is to discard FDs whose at least one attribute of the left-hand side is not a duplicated attribute, i.e. does not belong to the logical navigation of the database[1].

Example 5 In addition to the previous example, assume `Order.oid` is also a duplicated attribute (with another schema not listed here). Applying the previous heuristic, it remains only three dependencies from the example 2: { `Product:{pid}` $\rightarrow$ `{name,price}`, `Order: {oid}` $\rightarrow$ `{date,shipVia}`, `Order: {oid,pid}` $\rightarrow$ `{qty}` }

Nevertheless, we cannot argue that the logical navigation always give enough information to select relevant FDs: this is all the more true in case of 'full' denormalization. Indeed, no join could be possible anymore, which could be the case for `Order.oid` in our example.

To cope with such an intrinsic limitation, we have defined a variant of Armstrong databases, so called

[1] In [21], a similar idea was proposed in an informal way.

Informative Armstrong Databases, as an effective alternative representation of the extracted knowledge.

2.3.2 Armstrong relations/databases

Armstrong relations, introduced in [8], are closely related to FDs since they exactly satisfy a set of FDs.

Given a relation as input, Armstrong relations can be computed with respect to the FDs satisfied in that relation. In [5], we have introduced the notion of *informative Armstrong relation* which is both 1) an Armstrong relation with respect to the FDs satisfied in the input relation and 2) a subset of that relation. Experiments show its interest to sample existing relations (up to 1000 times smaller). Moreover their tuples come from the "real world" and thus convey more semantics than other existing proposals.

Inclusion dependencies can also be safely integrated into this framework: they lead to the definition of *informative Armstrong databases* [6].

3 Applications

We present below three examples of database applications which can benefit from a clear understanding of data semantics in existing databases.

3.1 Data integration

Many organizations are using databases implemented over a decade ago and are actually faced with difficulties to modernize or to replace these databases to match the evolution of both the technology and the requirements.

In such applications, it is essential to recover data semantics in order to develop new databases or to change underlying data models. As an example, consider the problem of *data/schema integration* in which the Clio system is developed [19]. It is devoted to the transformation and the integration of heterogeneous data (relational data and XML Data). As far as relational databases are concerned, authors mention the necessity of the discovery of keys and referential constraints in existing databases.

3.2 Semantic query optimization

Semantic query optimization takes advantage of semantic information stored in databases to improve the efficiency of query evaluation [10, 3].

As an example, consider a semantic query optimization technique called *query folding*. It consists in computing query answers using a given set of resources (materialized views, cached results of previous queries or queries answerable by other queries). In [10], the author proposes to use INDs to find folding of queries that would otherwise be overlooked.

3.3 Logical database tuning

We expect the greatest impact of our work for this application. We sketch the usefulness of the discovered dependencies to assist a DBA for tuning an existing database at the logical level [14].

The key tasks of such an activity should be:

- *logical constraints definition*: for instance, definition of keys, foreign keys and triggers,
- *database restructuring*: for instance, normalizing a relation schema with the help of small samples (informative Armstrong relations) of the relation.

Logical DB tuning from FDs A key is a special case of a FD. Keys turn out to be one of the most important integrity constraint in practice. Therefore, a DBA has the opportunity to enforce keys over a relation schema of an existing database.

Example 6 For instance, attributes `pid` in `Product` is a key and should be enforced (if not already done).

Logical DB tuning from INDs A foreign key is a special case of IND, since it represents the left-hand side of an IND whose right-hand side is a key. Within our framework, foreign keys can be derived from both INDs and keys of the relation schema of the right hand side.

As for keys, a DBA has the opportunity to enforce foreign keys over an existing database.

Example 7 For instance, attribute `pid` in `Order` is a foreign key and should be enforced.

Table 2: A restructurated database

Product

pid	name	price
1	Chai	$ 18.00
2	Tofu	$ 18.00
3	Paté	$ 12.30
4	Paté	$ 50.00

OrderDetail

pid	oid	qty
1	10509	2
2	10509	1
2	10511	1
3	10510	2

Order

oid	date	shipVia
10509	11-10-1998	Federal Shipping
10510	1-5-2000	United Package
10511	12-12-1998	Speedy Express

Database restructuring From interesting FDs and INDs, classical algorithms (e.g. synthesis algorithm) could be applied to normalize existing relation schemas (see [13] for non interaction conditions between FDs and INDs).

When a restructuration has to be performed on an operational database, one need to migrate the data which is an easy task once the target defined.

Example 8 From $F =$ { `Product:{pid}` $\rightarrow$ `{name, price}`, `Order: {oid}` $\rightarrow$ `{date, shipVia}`, `Order: {oid,pid}` $\rightarrow$ `{qty}` } and $I =$ {`Order[pid]` $\subseteq$ `Product[pid]` }, we can normalize relation schemas and then migrate the data. We would get the database given in Table 2 (we omit the constraints).

To take into account *application programs* accessing the database schema through existing relations and views, there is no easy solution. One approach is to define a set of views over the new database schema to "simulate" the old database schema. Nevertheless, updating relational views is a difficult problem which is weakly supported by major RDBMS.

4 Conclusion

Most of the time, applications of data mining concern decision-making for humans. In this paper, we show that systems could also take advantage of data mining. More precisely, we have shown that logical database tuning, e.g keys and foreign keys enforcement, is a new application of data mining. Indeed, it can be thought as a decision that an expert (here the database system) have to make to improve her business (here data coherence enforcement) from knowledge hidden in her data.

Since data semantics of existing databases is mainly conveyed by FDs and INDs, two data mining algorithms need to be dealt with: FD inference and IND inference. We have contributed to each of these inference problems and we have presented in this paper how much this knowledge can be relevant and useful for database applications.

References

[1] C. Beeri, M. Dowd, R. Fagin, and R. Statman. On the structure of Armstrong relations for functional dependencies. *JACM*, 31(1):30–46, 1984.

[2] P.A. Bernstein and al. The ASILOMAR report on database research. *ACM Sigmod Record*, 27(4):74–80, 1998.

[3] Q. Cheng, J. Gryz, F. Koo, T.Y.C. Leung, L. Liu, X. Qian, and B. Schiefer. Implementation of two semantic query optimization techniques in DB2 universal database. In *Proc. of the* 25^{th} *VLDB*, pages 687–698, Edinburgh, Scotland, UK, 1999. Morgan Kaufmann.

[4] F. De Marchi, S. Lopes, and J.-M. Petit. Efficient algorithms for mining inclusion dependencies. In *Proc. of the* 7^{th} *EDBT*, volume 2287 of *LNCS*, pages 464–476, Prague, Czech Republic, 2002. Springer.

[5] F. De Marchi, S. Lopes, and J-M. Petit. Samples for understanding data-semantics in relations. In *International Symposium on Methodologies for Intelligent Systems (ISMIS'02)*, vol-

ume 2366 of *LNAI*, pages 565–573, Lyon, France, 2002. Springer-Verlag.

[6] F. De Marchi and J-M. Petit. Construction de petites bases de données d'Armstrong informatives. *Revue d'intelligence artificielle RSTI-RIA (from EGC'03)*, 17:31–42, Jan. 2003.

[7] J. Demetrovics and V.D. Thi. Some remarks on generating Armstrong and inferring functional dependencies relation. *Acta Cybernetica*, 12(2):167–180, 1995.

[8] Ronald Fagin. Armstrong databases. Technical Report 5, IBM Research Laboratory, 1982.

[9] G. Gottlob and L. Libkin. Investigations on Armstrong relations, dependency inference, and excluded functional dependencies. *Acta Cybernetica*, 9(4):385–402, 1990.

[10] J. Gryz. Query Folding with Inclusion Dependencies. In *Proc. of the* 14^{th} *IEEE ICDE*, pages 126–133, Orlando, Florida, Feb. 1998. IEEE CS.

[11] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(3):100–111, 1999.

[12] M. Kantola, H. Mannila, K-J. Räihä, and H. Siirtola. Discovering functional and inclusion dependencies in relational databases. *Int. Journal of Intelligent Systems*, 7:591–607, 1992.

[13] M. Levene and G. Loizou. Guaranteeing no interaction between functional dependencies and tree-like inclusion dependencies. *TCS*, 254(1-2):683–690, 2001.

[14] S. Lopes, F. De Marchi, and J-M. Petit. DBA companion : un outil pour l'analyse de bases de données (demo session). In *BDA'2002 (French database conference)*, pages 523–528, 2002.

[15] S. Lopes, J-M. Petit, and L. Lakhal. Functional and approximate dependencies mining: Databases and FCA point of view. *Special issue of JETAI*, 14(2/3):93–114, 2002.

[16] S. Lopes, J-M. Petit, and F. Toumani. Discovering interesting inclusion dependencies: Application to logical database tuning. *IS*, 17(1):1–19, 2002.

[17] H. Mannila and K-J. Räihä. Design by example: An application of Armstrong relations. *JCSS*, 33(2):126–141, 1986.

[18] H. Mannila and K-J. Räihä. Algorithms for inferring functional dependencies from relations. *DKE*, 12(1):83–99, 1994.

[19] R.J. Miller, M.A. Hernández, L.M. Haas, L-L Yan, C.T.H. Ho, R. Fagin, and Lucian Popa. The Clio project: Managing heterogeneity. *ACM Sigmod Record*, 30(1):78–83, Mar. 2001.

[20] N. Novelli and R. Cicchetti. Functional and embedded dependency inference: a data mining point of view. *IS*, 26(7):477–506, 2001.

[21] J-M. Petit, F. Toumani, J-F. Boulicaut, and J. Kouloumdjian. Towards the reverse engineering of denormalized relational databases. In *Proc. of the* 12^{th} *IEEE ICDE, New Orleans, Louisiana*, pages 218–227, 1996.

[22] AutoAdmin project. Microsoft research, `http://www.research.microsoft.com/dmx/autoadmin`.

[23] A. M. Silva and M. A. Melkanoff. A method for helping discover the dependencies of a relation. In Hervé Gallaire, Jean-Marie Nicolas, and Jack Minker, editors, *Advances in Data Base Theory*, pages 115–133, Toulouse, France, 1979.