PERGAMON

Information Systems 26 (2001) 477–506

www.elsevier.com/locate/infosys

Functional and embedded dependency inference: a data mining point of view

Noël Novelli[a,b,*], Rosine Cicchetti[a,c]

[a] *LIM, CNRS FRE-2246, Université de la Méditerranée, Case 901, 163 Avenue de Luminy, 13288 Marseille cedex 9, France*
[b] *Université de Provence, CMI, 39 rue Joliot Curie, 13453 Marseille cedex 13, France*
[c] *IUT d'Aix-en-Provence, Avenue Gaston Berger, 13625 Aix-en-Provence cedex 1, France*

Abstract

The issue of discovering functional dependencies from populated databases has received a great deal of attention because it is a key concern in database analysis. Such a capability is strongly required in database administration and design while being of great interest in other application fields such as query folding. Investigated for long years, the issue has been recently addressed in a novel and more efficient way by applying principles of data mining algorithms. The two algorithms fitting in such a trend are TANE and Dep-Miner. They strongly improve previous proposals. In this paper, we propose a new approach adopting a data mining point of view. We define a novel characterization of minimal functional dependencies. This formal framework is sound and simpler than related work. We introduce the new concept of free set for capturing source of functional dependencies. By using the concepts of closure and quasi-closure of attribute sets, targets of such dependencies are characterized. Our approach is enforced through the algorithm FUN which is particularly efficient since it is comparable or improves the two best operational solutions (according to our knowledge): TANE and Dep-Miner. It makes use of various optimization techniques and it can work on very large databases. Applying on real life or synthetic data more or less correlated, comparative experiments are performed in order to assess performance of FUN against TANE and Dep-Miner. Moreover, our approach also exhibits (without significant additional execution time) embedded functional dependencies, i.e. dependencies captured in any subset of the attribute set originally considered. Embedded dependencies capture a knowledge specially relevant in all fields where materialized data sets are managed (e.g. materialized views widely used in data warehouses). © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Data mining; Database design; Functional dependency; Lattices; Algorithms

1. Introduction and motivations

In order to guarantee data consistency, integrity constraints are essential. Various types of integrity constraints have been studied [1–3]. Among them, it is widely recognized that functional dependencies are the most important and common semantic constraints encountered in databases [4–6]. More precisely, a functional dependency between two sets of attributes (X, Y), denoted by $X \rightarrow Y$, holds in a relation if values of the latter set are fully determined by values of the former [7].

Functional dependencies are a key concept in relational theory and the foundation for data

*Corresponding author.
E-mail address: noel.novelli@lim.univ-mrs.fr (N. Novelli).

0306-4379/01/$ - see front matter © 2001 Elsevier Science Ltd. All rights reserved.
PII: S0306-4379(01)00032-1

organization when designing relational databases [8–10]. However, their existence within data sets is independent of the relational model, they are therefore essential as soon as large data sets must be stored, handled, and updated whatever the underlying data model is (object oriented, N1NF, multi-dimensional, etc.). Let us quote the normalization theory, proposed in [11], for object oriented models, and the extension of functional dependencies in the context of nested relational models [12]. This remark being done, the background in which the rest of the paper fits is the relational model [13], besides equivalent schemas are provided [14–17].

Discovering functional dependencies from existing databases is an important issue, investigated for long years [18–20,10], and recently addressed with a data mining viewpoint, in a novel and much more efficient way [21–23].

The approach presented in this paper fits in such a trend and aims to discover minimal functional dependencies, and embedded dependencies which are valid over a subset of the original data source. Before giving an overview of these recent contributions and ours, we describe in more depth the application fields, more than ever up to date, for which extracting functional dependencies from populated databases is critical and the ones for which carrying such dependencies into views, or more generally materialized data sets, is strongly required.

1.1. Application fields

Motivations behind addressing the present issue are originated by various application fields: database administration and design, reverse-engineering and query optimization [4,3,24]. Let us underline that in any case and like for other data mining issues, the objective when discovering new knowledge is to provide users with relevant information for making decisions. However, when extracting functional dependencies, the interested users are database specialists (database administrators or designers, application programmers or system developers, depending on what application field is concerned).

Actually extracting such a knowledge is a key capability for database administration and design tools [4], and makes it possible to assess that keys are minimal, control normalization and detect denormalized relations. The latter situation could be desired for optimization reasons or because data are scarcely ever updated, but it could also result from design errors, or schema evolutions not controlled over time. In such cases, the database administrator is provided with a relevant knowledge for making reorganization decisions [23]. Apart from logical reorganizations, physical tuning is also concerned when clustering or replicating data and for improving the physical independence which is a key feature of modern database systems [25,26].

In a reverse-engineering objective, elements of the conceptual schema of data (such as abstract entities and the links relating them) can be obtained from the database schema, but much more details (additional constraints, candidate keys, etc.) can be acquired when knowing the functional dependencies holding in a database [27–29]. Moreover, such a knowledge enables to relax the assumption under which the relational schema being dealt is well designed and normalized [29]. This expands the practical scope of reverse engineering because denormalization is frequently observed in operational databases (for the reasons above mentioned).

Query folding can also take benefit from the dependency knowledge. Query folding addresses the following issue: provided with a given set of resources, how, if possible, a query can best be answered. It applies in query optimization in centralized databases, query processing in distributed databases and query answering in federated databases [30,6]. Query folding in the presence of inclusion and functional dependencies results in more complete solutions because "foldings which might otherwise be overlooked are generated" [6]. Close to the latter issue, the problem of information gathering by agents includes the computation of query plans in order to optimize cost of queries when remote information sources are requested [31,32]. Such sources are increasingly available on line via the web and for an efficient information retrieval, aided tools are strongly required for end users.

Finally, redundant data are increasingly used through materialized views, data caches or