Implementation of the ‘Information Bearing Capability Model’ for Databases

Kaibo Xu
Business College of Beijing Union University
Beijing, China
Kaibo.xu@bcbuu.edu.cn

Junkang Feng
School of Computing, University of the West of Scotland
United Kingdom
Junkang.feng@uws.ac.uk

Abstract—To tackle the capability of a data schema of bearing information, a model for the definition and exploitation of what we call the ‘Information Bearing Capability’ of a database (IBC Model for short) model has been proposed. In this paper, we present the notion of IBC with a brief justification. We then describe the IBC model with a diagram and present a set of inference rules for reasoning about information content in terms of whether a given state of affairs is included in the information content of another state of affairs. These rules are a means of deriving nested information. In order to experiment the ideas reflected by the IBC model, we implemented a simplified IBC model by using Prolog. This experimentation shows how domain independent and nested information may be derived from known information and how the IBC model may be implemented and used with and for a database.

Keywords: information bearing capability; inference rules; information content

I. INTRODUCTION

The capability of a database’s answering queries is essential for a database in particular and for a data intensive information system in general. The literature on database design, for example, [1] and [2], does not seem to have addressed this problem adequately. To formulate such a capability, we suggested the notion of ‘classes of a path in an ER schema’ [3]. To put briefly, in relation to a query, namely a piece of information that is required, any path in an ER schema is one of seven possible classes, and we found that only certain classes are useful for answering queries.

However the work that has been done thus far on the problem of the ‘classes’ does not cover how to systematically classify individual paths in an ER schema, which would seem to be the next step of research in this direction. To this end, it would appear that to accurately define the semantics of a path and formally reason about it is highly desirable and possibly also leads to avenues to solutions.

In the literature, systematic investigations into the capability of a database’s in answering a query can seldom be found. It would appear that such a capability is seen as equivalent to the simple question of whether data elements (tuples, attributes, etc) and their connections that are required by a query exist in the database or not. Only some works on the quality of an ER schema were reported, such as those in [4] and in [1], which can be seen related to this problem. Another topic that is also relevant is that of ‘connection traps’.

[5] and [6] address this topic respectively among many other topics. These relatively earlier works are largely based upon an intuitive approach, and no promising tool was used.

In this paper, we propose the notion of ‘Information Bearing Capability Model’ and show how this model may be implemented and used with and for a database.

II. IBC MODEL

Based upon these four IBC conditions, an IBC model for conceptual data schema is developed, which is illustrated as Fig. 1.

The kernel IBC of the data schema is the IBC of the data structure per se, namely without any consideration of information nesting on the structure. Fig. 1 shows that with the development of information nesting or derivation in terms of the Kernel IBC, more and more information can be derived. Such derived information can be reasoned about by importing the implication of the business requirements, pragmatics or some business rules. There are two types of derived information, i.e., domain-dependent information and domain-independent information. Domain-dependent information is the information logically implied by the Kernel IBC by means of inference rules. Domain-independent information is the information derived by using some reasoning mechanisms such as those in Description Logics. Some research work [7] [8] has been done on this.

The notion of IBC is concerned with the capability of a data schema of bearing information, and thus is helpful to approach the question of what information content is and how we may use information content. In the next section, some basic definitions towards the notion of information content of data will be introduced. Furthermore, another notion called information content inclusion relation, which formulates and helps apply the notion of information content, is also proposed.

This paper is sponsored by Funding Project for Academic Human Resources Development in Institutions of Higher Learning under the Jurisdiction of Beijing Municipality
III. INFERENCE RULES FOR REASONING

Given two random events, say X and Y, there might be a special type of relation between them, i.e., ‘the particulars of random event Y are in the information content of the particulars of random event X’. For brevity, we will also call such a relation ‘random event Y is in the information content of random event X’ [9]. We suggested calling such a relation an ‘information content inclusion relation’ (IIR) [10].

Given a set of IIR, generally there are other IIR that are nested within (i.e., logically implied by) them. Sometimes we wish to derive such implied IIR logically from those that we have already known somehow. To this end, in the next section, we will introduce a set of domain independent inference rules, which can be called IIR Rules.

To facilitate the reader to understand the rules, we reiterate the following.

a) What is meant by ‘I(X) ⊃ Y’?

It means that a suitably placed observer could learn that Y (particulars of Y) by consulting the particulars of X. A sufficient condition for ‘I(X) ⊃ Y’:

1) Both X and Y are random events, namely they could be contingently true and contingently untrue, but are neither necessarily true nor necessarily untrue. Mathematically, P(X) ≠ 1 and P(X) ≠ 0, and P(Y) ≠ 1 and P(Y) ≠ 0.

2) Whenever X is true, Y is always true. That is, P(Y|X) = 1. In other words, X ⊂ Y.

Information Inference Rules:

Sum: If Y = X₁ ∪ X₂ … ∪ Xₙ, then I(Xᵢ) ⊃ Y for i = 1, …, n
Product: If X = X₁ ∩ X₂ … ∩ Xₙ, Y = Xᵢ for i = 1, …, n, then I(X) ⊃ Y
Transitivity: If I(X) ⊃ Y, I(Y) ⊃ Z then I(X) ⊃ Z
Union: If I(X) ⊃ Y, I(X) ⊃ Z then I(X) ⊃ Y ∩ Z

Augmentation: If W = W₁ ∩ W₂ … ∩ Wₙ, Z is the product of a subset of {W₁, W₂, …, Wₙ}, I(W) ⊃ Y then I(W ∩ X) ⊃ Z ∩ Y

Decomposition: If I(X) ⊃ Y ∩ Z then I(X) ⊃ Y, I(X) ⊃ Z

In [9], proofs of the soundness and the completeness of these inference rules are given. To explore how the ideas of IIR and the rules for reasoning about IIR may be implemented and made use of for a database, a prototype of a standalone system which is called IIR-Reasoning was developed, which works with a database and a number of other elements such as ontology. More details can be found in that paper in terms of how to exploit the notion of information content as a basis for reasoning about information in databases.

IV. IMPLEMENTATION OF THE IBC MODEL

Here is an example of our implementation. Fig. 2 shows a conceptual schema of a database. Assume that we have the following information requirements.

Information Requirements:

1. Which client has rented from an owner?
2. Does staff have only person?
3. Does staff want to get information from clients?
4. Is there a client who is interested with some properties?
5. Is there a client who wants to own some properties?
Fig. 2 A Conceptual Schema of a Database

We then convert the inference rules, business rules to Prolog clauses which are shown as follows.

Inference Rules:

1. “Client-Property-Owner” ⊆ “Client-Owner” ⇔ I(“Client-Property-Owner”) ⊆ “Client-Owner” (Reflexivity)
2. “A client rented some properties from an owner.” ⊆ “A client rented from an owner.” ⇔ I(“A client rented from an owner.”) ⊆ “A client rented some properties from an owner.” (Reflexivity)
3. I(“Staff-Has-Branch”) ⊆ “Staff has branches.”, I(“Staff has branches.”) ⊆ “Staff has more than 1 person.” ⇔ I(“Staff-Has-Branch”) ⊆ “Staff has more than 1 person.” (Transitivity)
4. I(“Staff-Set-up-Interview”) ⊆ “Staff sets up an interview with clients.”, I(“Staff sets up an interview with clients.”) ⊆ “Staff wants to get some information from clients.” ⇔ I(“Staff-Set-up-Interview”) ⊆ “Staff wants to get some information from clients.” (Transitivity)
5. I(“Client-Views-Property”) ⊆ “A client views properties for rent.”, I(“A client views properties for rent.”) ⊆ “A client is interested with some properties.” ⇔ I(“Client-Views-Property”) ⊆ “A client is interested with some properties.” (Transitivity)
6. I(“Client-Rents-Property”) ⊆ “A client rented properties for rent.”, I(“A client rented properties for rent.”) ⊆ “A client wants to own some properties.” ⇔ I(“Client-Rents-Property”) ⊆ “A client wants to own some properties.” (Transitivity)
7. I(“Client-Property-Staff”) ⊆ “Renting transactions are related to clients.”, I(“Client-Property-Staff”) ⊆ “Renting transactions are related to clients.” ⇔ I(“Client-Property-Staff”) ⊆ “Renting transactions are related to clients.” (Transitivity)
8. I(“Client-Property-Owner”) ⊆ “A client rented some properties.”, I(“Client-Property-Owner”) ⊆ “A client rented some properties.” ⇔ I(“Client-Property-Owner”) ⊆ “A client rented from an owner and a client rented some properties.” (Union)
9. I(STAFF) ⊇ “Staff” ⊆ I(STAFF) ⊇ “full-time staff”, I(STAFF) ⊇ “part-time staff” (Decomposition)
10. “Michael is sitting.”, “Somebody is sitting.”, “A client is working.”, “Somebody is in the office.”, “A client is working in a sitting position.” ⊆ “Somebody is sitting in the office.”

Business rules:

1. I(“Staff has branches.”) ⊇ “Staff has more than 1 person.”
2. I(“Staff sets up an interview with clients.”) ⊇ “Staff wants to get some information from clients.”
3. I(“A client views properties for rent.”) ⊇ “A client wants to own some properties.”
4. I(“A client rented properties for rent.”) ⊇ “A client wants to own some properties.”
5. I(“Client-Property-Owner”) ⊇ “A client rented from an owner.”
6. I(“Client-Property-Owner”) ⊇ “A client rented some properties.”
7. I(“Client-Property-Staff”) ⊇ “Renting transactions are related to clients.”
8. I(“Client-Property-Staff”) ⊇ “Renting transactions are related to staffs.”
9. I(STAFF) ⊇ “full-time staff”
10. I(STAFF) ⊇ “part-time staff”
11. “Michael is sitting.” ⊇ “Somebody is sitting.”
12. I(“Somebody is working.”) ⊇ “Somebody is in the office.”
13. Union(“Michael is sitting.”, “Somebody is working.”, “Michael is working in a sitting position.”)
14. Union(“Somebody is sitting.”, “Somebody is in the office.”, “Somebody is sitting in the office.”)

In our implementation, we use PIE (Prolog Inference Engine) as our inference engine. It uses the standard Prolog syntax and releases the most popular set of predicates.

Implemented by Prolog Inference Engine (PIE)

/* Business rules for Augmentation */
stateSubsume("CPO", "CO").
/* Business rules for Transitivity */
infoSubsume(Staff-Has-Branch", "Staff has Branches.").
infoSubsume("Staff has Branches.", "Staff has more than 1 person.").
/* Business rules for Union */
infoSubsume("CPS", "clients").
infoSubsume("CPS", "staffs").
union("clients", "staffs", "people").
/* Business rules for Decomposition */
infoSubsume("STAFF", "staffs").
union("full", "part", "staffs").
/* Business rules for Augmentation */
infoSubsume("working", "in office").
stateSubsume("Michael is sitting.", "Sb. is sitting.").
union("Michael is sitting.", "working.", "Sb sitting working").
union("Sb. is sitting.", "in office", "Sb sitting working").
*/----------Information Inference Rules--------*/
/* Augmentation */
infoSubsume((A, B)) :-
    union((W, X), A), union((Z, Y), B), stateSubsume(W, Z), infoSubsume(X, Y).
/* Reflexivity */
infoSubsume((X, Y)) :- stateSubsume(X, Y).
/* Union */
infoSubsume((X, S)) :-
    union((Y, Z), S), infoSubsume(X, Y), infoSubsume(X, Z).
/* Decomposition */
infoSubsume((X, A)) :- union((A, B), S), infoSubsume(X, S) .
/* Transitivity */
infoSubsume((X, Z)) :- infoSubsume(X, Y), infoSubsume(Y, Z).

Proposed Query:
/* Reflexivity */
infoSubsume("CPO", "CO")
/* union */
infoSubsume("CPS", "people")
/* Transitivity */
infoSubsume("Staff-Has-Branch", "Staff has more than 1 person.")
/* Decomposition */
infoSubsume("STAFF", "full")
/* Augmentation */
infoSubsume("Michael_sitting_working", "Sb_sitting_working")

V. CONCLUSIONS

The implementation of the ‘Information Bearing Capability Model’ in a database is a complex task, which is difficult to complete successfully. Through analyzing information content inclusion relations of random events, our approach seems simpler and more convenient to use than some other complex knowledge-based approaches, and yet our approach is grounded on sound information theories. This paper only shows our preliminary attempt in this direction in order to explore the possibility and effects of an information theoretic approach to some aspects of IBC implementation. Our future work will focus on the reasoning complexity, derivation of business rules etc.

REFERENCES