An Integrity Model Based on Knowledge and Belief

David Y. Liu

Department of Computer Science
Pittsburg State University
Pittsburg, KS 66762
liu@ksuvax1.ksu.edu

Abstract

This paper defines a formal model of database integrity using the concept of knowledge and belief. Knowledge are unrefutable facts and rules, while beliefs are refutable rules to be enforced. Transaction based integrity checking is used. Representation of knowledge and belief are given, and related work is discussed.

1 Introduction

Integrity, as defined in Webster's New World Dictionary, is the quality of being complete and sound. In [10], Motro introduced a model in terms of validity and completeness. Philosophically, database designers and users want to capture the integrity of the portion of the real world which they try to model using computer systems completely and precisely. Only in very restricted cases is that possible. For the vast majority of situations, the computer system is used to model a situation not completely understood by system developers. Database integrity is modeled by a set of integrity constraints. As pointed out by Motro, constraints enhance integrity of data, but such constraints cannot ensure it. That is to say, the set of integrity constraints is usually not a complete description of system integrity because of limitation of human understanding. On the other hand, what we would ask is “are all the constraints defined by a DBA or users sound?” Again we recognize that these constraints are not necessarily sound. In many cases, assumptions are made which are thought to be right but later turn out to be wrong. In this paper we address the two aspects of database integrity using the concept of knowledge and belief, which was also used in security in Moser’s paper [9] and later developed as the basis of integrity in Ramanna’s work [12]. In the framework of knowledge and belief, one can model the fundamental truth and existing data as pieces of knowledge, and assumptions made about the database as beliefs. A new piece of knowledge can be added to the model to augmented knowledge about an existing system, and also introduce a belief to make a assumption which may later be refuted. Within our model, instead of defining a set of integrity constraints which make the system complete and sound (impossible in most cases), the user is provided a mechanism to dynamically capture new knowledge and to refute unsound assumptions (beliefs) previously made. This model is more natural and flexible with respect to the evolution of information systems. In the next section, our model is defined based on knowledge and belief, the form constraints expressing knowledge and beliefs may take is examined, and a comparison to the integrity model of Ramanna [12] is presented. In the last section, several related works are reviewed including works of Biba, Clark and Wilson, Lunt, and Moser [2, 4, 7, 9], and future directions for research are identified.

2 Integrity Model

2.1 Model definition

In this section, we define an integrity model for databases based upon the concepts of knowledge and belief. The integrity model consists of the following:
1. A set $S$ of states, where $s_i$ in $S$ is $i$-th state which is represented by $(K_i, B_i)$. Here $K_i$ stands for the set of knowledge of state $s_i$. The knowledge is composed of relation instances, deducible formula based on other knowledge, and basic certified integrity constraints. $B_i$ stands for the set of beliefs of state $s_i$, where beliefs are refutable constraints made about the database.

2. An initial state $s_0$, with empty knowledge and belief sets;

3. A truth value assignment $M$ which checks whether an integrity constraint is satisfied under a certain state $s_i$;

4. A set $T$ of well-defined transactions, which have atomicity and validity.

In a relational database, knowledge corresponds to data stored in a database, and also the system constraints used to enforce things like key integrity, referential integrity, etc. However, our model is not restricted to the relational data model. It is readily applicable to deductive databases, where rules as well as data are modeled as knowledge.

The refutation of existing beliefs occurs when new information which violates a belief enter into the system and the new data is asserted to be true by authority. More concisely, we claim beliefs are only refuted by knowledge. Also, a system authority figure has the power to drop an old belief, and install a new one.

The integrity of databases can be inductively reasoned about as following: Suppose initial state $s_0$ is an empty database, with no knowledge, no belief, thus it has integrity trivially. Now, suppose $s_i$ is valid, i.e., a state that has integrity. A well-defined transaction $t$ is applied, and only if it satisfies all the rules in knowledge and beliefs, otherwise the transaction is not performed. So the resulting state $s_{i+1}$ is also valid and the database has integrity with respect to the currently effective integrity constraints.

2.2 Constraint Format

Based on our model, we examine the form a constraint should take. Suppose a transaction $t$ takes $r_1(f_{1,1}, f_{1,2}, \ldots, f_{1,k_1}), r_2(f_{2,1}, f_{2,2}, \ldots, f_{2,k_2}), \ldots, r_n(f_{n,1}, f_{n,2}, \ldots, f_{n,k_n})$ as arguments denoted as $arg$, where $n$ is the number of relations involved, and $f_{i,j}$ is the $j$-th field value for $i$-th relation, and $k_i$ is the number of fields in the $i$-th relation. For example, update(oldtuple, newtuple), where oldtuple = $emp('David', 334)$, newtuple = $emp('David', 266)$.

Constraints in our model are composed of two parts: rules in knowledge, and beliefs. The short-lived periods of integrity constraints in a database leads us to give constraints an effective finite or infinite time interval. So in general all constraints have the form: $\{\{s_{\text{starttime}}, s_{\text{endtime}}\}, \text{wff}\}$, where $\text{wff}$ is a well-formed formula. $s_{\text{starttime}}$ and $s_{\text{endtime}}$ may be specified or unspecified. Unspecified $s_{\text{starttime}}$ means currently the constraint is being enforced. Unspecified $s_{\text{endtime}}$ means the constraint would be in effect as long as it is not dropped. If the interval part is omitted, it is equivalent to having both $s_{\text{starttime}}$ and $s_{\text{endtime}}$ unspecified. The unit of time representation can vary depending on application requirements. The checking of a time interval is straight-forward. We maintain an system clock and a daemon process updating the constraint sets at certain intervals. If the $s_{\text{endtime}}$ of a constraint becomes larger than the system clock, the constraint is dropped automatically.

To define the $\text{wff}$, we consider two cases, static constraints and transitional constraints:

Static constraints Static constraints can be specified in the first order predicate logic. In a static constraint one is concerned with the question of whether the current state meets the defined integrity constraints. Static constraints can be checked only using the existing data in the database, without concern about the transaction or its arguments. The form of these constraints is $p(s_i), \text{where } p \text{ stands for any formula in first order predicate logic, } s_i \text{ is the current state}.

Transitional constraints Transitional constraints are concerned with which updates can be performed. Current and/or previous information is needed as well as the new data to be added into the database. For example, Update of marital status, it is desirable to exclude updates such as 'never-married' with 'divorced'. To enforce such restrictions, we need to check the current field of marital status, and new status before doing the actual update. These constraints can take the form: $p(s_1, s_2, \ldots, s_i, arg)$, where $arg$ is the argument of the applied transaction $t$.

There are a lot of research effort on static constraints as in [8]. Some authors [6, 12] also define dynamic constraints which may be expressed as a temporal logic.
formula in future-tense with certain restrictions. However, that may lead to many unsolved problems which are described in the next subsection. In Chomicki's work[3], constraints are represented as past-tense temporal logic formula. Using his approach, problems with future-tense temporal logic are avoided, also the system automatically generates auxiliary relations to keep necessary historical data. However, this approach implicitly excludes a run-time constraint definition. For example, ‘Fired employee cannot be rehired’. If this constraint is to be added at run time, and no status information about whether an employee had been fired before was kept, the system has no way to enforce this constraint. Such a constraint can only be defined at compilation time. Suppose we have very good planning, and all needed historical information are designed into the relational schema at the database design phase, it suffices to check constraints just based on current state. Therefore a transitional constraint should have form $p(s_i, arg)$.

A transaction may either be a single insert/delete operation, or a complicated procedure written in a description language to manipulate database contents. The consistency of a transaction should be checked at run-time because integrity constraints are allowed to be added and dropped at run-time.

2.3 Comparison of our model with Ramanna’s Model

Ramanna’s integrity model [12] is based on both the concept of knowledge and belief, and temporal logic. There are three major points of comparison of our model with Ramanna’s model, namely transaction concept, temporal logic formula, and concept of knowledge.

Transaction concept In [12], the database modification operations are limited to ‘insert’, ‘delete’, and ‘update’. Without the concept of atomic transaction concept, even constraints for very simple operation combinations become very hard to specify. For example, in accounting systems, double entry insertion involves two insertions, with the constraint that before and after the insertion the balance be maintained. It is impossible to tell whether an insertion is part of a double entry operation or just a single database insertion even using temporal logic formula. As pointed out by Qian [11], transactions are not expressible in temporal logic. In our model, the transaction concept is a basic component.

Constraints and temporal logic formula

Constraints in Ramanna’s model use a temporal logic formula as an important part. Using temporal logic provide more expressive power to express during a period of time something will happen. In [12], we can also say something will eventually happen. However, are these really constraints? Firstly, if a constraint states that something will eventually happen, one can always assume that it is potentially consistent, and defer the checking to a next step. Sometimes such constraints may never be satisfied. In this sense, such unlimited constraints do not impose any restriction on the database. Secondly, if a constraint says that something will happen before a certain time point, we check the constraint until the specified time. If the event does happen, then the constraint is satisfied. But what about if it does not happen. We may well have gone through lots of states to that point. Merely knowing that we have arrived at a inconsistent state is not enough, and usually we cannot undo or discarding all the previously possibly consistent states. Based on the above two arguments, in our model we don’t allow the using of temporal formula except for the effective time interval to limit the life span of a first order predicate. This appears not to be a severe restriction because one can use transactions to do complicated operations.

Extended knowledge concept In [12], knowledge is limited to be constraints and does not include data in databases. Since the natural meaning of ‘knowledge’ is everything one knows, which certainly includes data in databases, in our model we extend the concept of knowledge to include data. Ramanna introduced the idea that “a belief should be refuted by acquired knowledge”. This idea was not consistent if the data is not part of the knowledge. That implies that every time a belief is refuted, there must be an introduction of a new knowledge constraint. In reality, new input data if certified to be true should be able to refute an inadequate belief. In this way, the idea of the refutation of a belief by knowledge becomes more significant and consistent.
3 Related work and future effort

Quite a few authors have discussed about database integrity models, Biba [2] defined an integrity model as an supplement to the Bell and LaPadula security model [1]. One of the Bell and LaPadula's rule is: subjects of higher security levels are not allowed to write down into lower level data items to prevent disclosure of sensitive information. In Biba's model, data items are assigned not only security levels, but also integrity levels, which form a lattice. Items of lower integrity level are not allowed to write into the items of higher integrity level to prevent corrupting the higher level. This restriction compartmentalizes data and restricts user to one compartment which is not feasible in some cases.

In Clark and Wilson integrity model[4] focus is on management considerations and the setting is more biased toward commercial databases. It is required that all operations allowed should be certified well-formed transactions. Other requirements are separation of duty, authentication, and audit logging. The principle of separation of duty requires identifying roles involved in transactions. This work is the first to distinguish the differences in the commercial and military environments and it has stimulated formal model development of user level integrity constraints.

Compared to the Biba and Clark & Wilson models, currently our model assumes only one level of integrity (i.e., every object in the system should satisfy a same level of integrity constraints). Another assumption we made is that integrity constraint does not specify role or user information. We simplify our model to make clear the fundamental requirement for an integrity model without concerning classification and user information. One of our contributions is to establish a clear formal framework for integrity modeling, identifying the necessary restriction of the form of integrity constraints, and the use of knowledge and belief to dynamically capture the changing real world. This should lead to proofs that the system is sound and complete.

The SeaView model [7] is a security model for a multilevel secure relational database system. It also addresses the database integrity problem in the context of a multilevel system. It considered application-independent and application-specific integrity constraints. Application-independent constraints include entity integrity, referential integrity and polyinstantiation integrity constraints. Application-specific constraints include value constraints and class constraints. SeaView supports a similar transaction concept as our model. The difference of the two models lies in that we are addressing a DBMS without special considerations for high security. The application-independent constraints in SeaView are approximately comparable to the rules in the knowledge set of our model. Application-specific constraints which are more subject to changes and refutations are comparable to the belief set. However by using the concept of knowledge and belief, our model provides the refutation ability of beliefs which SeaView model does not support.

Modal logic is also used in literature as a basis of database security and integrity [5, 9]. Glasgow and MacEwen uses logic of obligation to model integrity. Integrity is defined as an obligation to know. In Moser's work, logic of knowledge and belief are used in reasoning about security. In our work, we only cope with an universal world since we do not distinguish users according to their roles or classification. So for the time being, we do not need modal logic. As we said, we want to extend our model to address roles and classifications, and so our model may later also be based on modal logic.

In conclusion, we have defined a small and solid integrity model which uses the concept of knowledge and belief. Based on this model we will investigate an extended integrity model with multiple levels of classifications, and the new model should support different roles for transactions and beliefs defined for different users in databases.

References


