Proceedings of the *International Conference on Theory and Application of Mathematics and Informatics ICTAMI 2005* - Alba Iulia, Romania

INTEGRITY CONSTRAINTS IN DISTRIBUTED DATABASES

Grigor Moldovan, Madalina Valeanu

Abstract. Due to some given integrity constraints for a medical distributed database we shall now consider the check of the global. In many domains, one cannot suppose to have a full availability of data in a distributed database. Given some integrity constraints over a distributed database we consider the problem of incrementally checking global consistency in response to updates made to the base relation but without accessing all these base relation. We were concerned with the issue of preserving global consistency when updating a distributed database and when not all relationships are available and updates relate to possible relations.

2000 *Mathematics Subject Classification*: 91A80, 68U99.

1. Introduction

The distributed database systems have come up at the border of two fields that apparently are in opposition in so far the data processing is concerned: database systems and computer networks. The distributed databases are defined as a *collection of logically interconnected databases, distributed in a computer network*. Hence, the distributed databases are not a mere collection of tables to be stored in each network node individually. In order to form the database the tables should be logically interconnected and the access to the tables in question should be achieved though a common interface.

An important aspect related to databases is **integrity**. Preserving the data integrity is a much more complicated issue in the heterogeneous distributed databases than in homogeneous databases. If the nodes in the distributed database are heterogeneous, there may come up troubles that could threaten the integrity of the distributed data, among which we can mention:

- inconsistencies among local integrity integration constraints;
- difficulties in specifying global integrity constraints;
- inconsistencies among local and global integrity constraints.

Local integrity constraints differ in the case of heterogeneous distributed databases. Inconsistencies can bring about troubles, and this is mainly valid in the case of complex queries based upon more than one database. Developing global integrity constraints can avoid conflicts among individual databases, associated to more organizations. Such a development is not always easy to implement, as it is not easy and practically acceptable to alter the organization structure in order to turn the distributed database into a consistent one. This can lead to inconsistencies among the local and global constraints. Conflicts, then, depend upon the level of the central control. If the control coming from the center is powerful, priority is given to global integrity constraints. In case of a poorer central control, local integrity constraints are given priority.

2. The semantic model for integrity constraints

The semantic model provides a way for the formal definition of *the meaning* of a logical database. An *interpretation* for a logical language is an attribution (evaluation) of true/false to each (defined) base atom or event. It can be represented as a set: any atom in the set is seen as *true* and any atom construction not in the set is *false*. Any sentence built can receive an evaluation related to interpretation, maintaining the true value, beyond logical connectors and quantifiers.

A *model* M in a collection of sentences P(a database DATALOG is such a collection) is an interpretation I for the language P so that each sentence in P is assessed as *true*. A Herbrand model of P is a model for P that contains only predicates and constants (and function symbols in logical programming) that are part of P. Now, we shall refer to a Herbrand model.

A model of P is a *minimal model* if there is no subset to run as belonging to P. Let us consider a database DATALOG **DB.** As any clause is a rule or an event (no clauses with empty head) database consistency is always assured (as it is a *model).* Moreover, in this case the database is a minimal and unique model. The semantic generally accepted model to define DB is this minimal model, noted with M_{DB}.

There are two meanings attributed to integrity constraints more recently. Be $DB = E_{DB} \cup I_{DB}$. With the consistency definition (Kowalski, 1978), IC is checked iff $DB \cup IC$ is consistent, i.e. it has a model. With the inheritance definition (Reiter, 1984), IC is checked iff $DB| = IC.$, i.e. DB logically inherits IC.

Another classical way of reconciling these two approaches consists in saying that IC must be imposed by the database minimal model, M_{DB}:

$$M_{DB}| = IC$$

3. Rationale for using integrity constraints

The explicit representation of integrity constraints in the database and of the rules and events of a database makes them much easier controllable. Thus, it is clear which the constraints in the database are. If these integrity constraints remain implicit, it is not very clear what is the meaning (semantics) of the database and the semantic information available for constraints is no more available for the applications.

3.1. Eliminating integrity constraints

Kowalski and Sadri's transformation not only that omits integrity constraints, but it also makes the modified database be consistent with respect to them. When the original theory is inconsistent, the new theory can be a new way of returning to consistency. The transformation is made thus:

Be $\Leftarrow a\langle\vec{x}\rangle, L_1, ..., L_n$ an integrity constraint expressed as a negation rule that must be omitted, where a $\langle\vec{x}\rangle$ is an atom, and $L_1, ..., L_n$ are atoms or initially negated atoms. At least three new rules are used to replace:

$$a\langle\vec{y}\rangle \quad \leftarrow K_1, .., K_m, \tag{1}$$

where $K_1...K_m$ are literals for which a $\langle\vec{x}\rangle$ and a $\langle\vec{y}\rangle$ unite at the most general unifier θ. Intuitively, the first new rule deals with the case when the integrity constraint applies to rule (1). The following two rules potentially added treat of the case when this is not applicable because X and Y do not unite. The first rule is:

$a\langle\vec{y}\rangle\theta \quad \leftarrow K_1\theta, .., K_m\theta, not(L_1, \ldots, L_n)\theta.$

The following rules are needed only if $a\langle\vec{y}\rangle\theta$ is not a non-ordinary case of $a\langle\vec{y}\rangle$, hence not a variant of $a\langle\vec{y}\rangle$. The first of them is:

$a\langle\vec{y}\rangle \quad \leftarrow K_1, .., K_m,$ nott $\langle\vec{y}\rangle$,.

where t is the new predicate symbol for which there is only one rule defined as:

$t\langle\vec{y}\rangle \leftarrow (\vec{y} = \vec{x}).$

($\vec{y} = \vec{x}$ represent conjunction equality. Given $\vec{y} = Y_1, ..., Y_n$ and $\vec{x} = X_1, ..., X_n$, the meaning is $(Y_1 = X_1), ..., (Yn = X_n)$)

4. Integrity preservation in BDD

In information systems technology, there is a clear tendency to distribute ,,related" data in different (locations) sites that can be personal computers or forming a zonal (federal) network. In all cases, a benefit from data distribution lies in that each location (site) processes its own data with a certain degree of autonomy. At the same time as data in various locations can be dependent, it is necessary to preserve their coherence and integrity. In this way we have some *global integrity constraints for distributed data.*

To preserve the integrity of a distributed database means to redistribute the components of a distributed in the computer network and only then to apply local checks.

Due to some given integrity constraints for a distributed database we shall now consider the check of the global. In many domains, one cannot suppose to have a full availability of data in a distributed database [2]. Even if all data are available, some can be so costly that they may become the last resort. If it is known that a constraint is fulfilled before performing a modification, the available relations state can principally be used to deduce some information on unavailable relationships [1]. This remark constitutes the basis of the global consistency tests preserved in a given update without taking into account all the data in the database. Practically, for consistency, we shall try to find the data that are most general and *efficient* in generation and execution. These tests will be named *Complete local tests.*

Figure 1: Information system for prescribing cures

tcurent(P,T) tant(P,T‘)
pretrat(T,T‘) specialistOK(P,T)
tcurent(P,T) – patient P follows treatment T
pretrat(T,T‘) – treatment T is preceded by treatment T‘

tant(P,T') – patient P followed treatment T'

specialistOK(P,T) – patient P was given the approval of the specialist doctor to follow treatment T

EXAMPLE1. Be there a medical database information system, distributed in four different locations, according to fig. 1. Thought he databases are independent of own location administered, they are subject of some global integrity constraints stating that a patient can follow a certain treatment only if he had followed another cure previously (deemed as cheaper or with potential better outcomes etc.) or if that patient receives a recommendation from a specialist doctor to follow that cure [1].

Treatment prescription policy can be expressed with the integrity constraint below:

$$(IC1)(\forall P,T,T')tcurent(P,T) \wedge pretrat\ (T,\ T') \Rightarrow tant(P,\ T') \vee specialistOK(P,\ T) \quad (2)$$

Databases can be disconnected, for example, because of defects in the networks (a node failure), updates being allowed as long as the system preserves consistency globally [9]. Let us imagine that we want to insert a new tuple *tcurent (Pop, tr187)*. Let us see how we can guarantee that he insertion does not violate the *global constraint* in each scenario below:

- Node *a* fails and relationship *pretrat* becomes inaccessible. Consistency preserves when patient *Pop* has a recommendation from a specialist doctor to follow treatment T. Alternatively, let us consider the patients following treatment *tr187* without the approval of the specialist doctor (if there is one). All the patients must satisfy all the requirements for *tr187*. All these requirements must be in $\Im$. $\Im$ is shown in *fig. 2*. If *Pop* has followed all the cures in $\Im$, he must have fulfilled all the requirements of *tr187*.

- Node *b* fails and relationship *pretrat* becomes inaccessible. Again, the approval given to patient *Pop* by the specialist doctor is a way to preserve the consistency of the distributed database. Alternatively, we cannot be sure that *Pop* followed all the cures required for *tr187*. Evidently, we cannot check directly this condition, as relation *tant* is not available. Fortunately, there is an indirect method that considers all the requirements for all previous cures taken by *Pop* without using the approval of

the specialist doctor. If these requirements include all the requirements of *tr187*, we can conclude that Pop satisfied all requirements in *tr187*. *Fig. 2* illustrates this test [10].

- Both nodes fail and no relation *pretrat*, *tant* and *specialistOK is available*. There is no way to provide that it is legitimate to prescribe treatment T to *Pop*.

Figure 2: Examples for preserving global consistency

While it is clear that data integrity is preserved at a given update in these tests, it is also desirable to have them as general as possible. Really, a degenerative testing procedure resulting always in a potential violation forecast is sure enough, but too conservative to be practical. We shall try to find the most general of the tests, when possible. Such general tests, called *Complete local tests* or CLT can be defined in this way:

- A CLT considers only basic local relations and updates.
- If the test is fulfilled, global consistency is surely preserved, irrespective of the distance relationship.

Now we shall consider zonal information systems providing distributed services. Suppose, generally, a federal database is managed in each zone of the zonal system. The data transfer, suppose it, is performed among different zones.

In this paper, we mainly dealt with the database relational model [3] though it can be applied to other models too. Let us consider a local relation L and some more distant relations. The general situation requires:

- the insertion of a tuple in the local relation L;
- an integrity constraint C implying local relation L and some more distant relations.

We can check this test condition for the local relation L, so that if L satisfies the test, the insertion in L does not affect C. On the other hand, if L does not satisfy the test, a conventional checking method must be used for the integrity constraint.

For a given integrity constraint, the optimization can be used in each location involved in the constraint with an applicable optimization. No competition control raises a problem, as optimization generates conditions only with respect to **local** data [11].

DEFINITION.*Presumption of initial integrity: Given an integrity C and a modification performed in the database, the presumption of initial integrity states that C is not violated in the database before the modification is performed.*

4. PRELIMINARY NOTIONS

4.1. INTEGRITY CONSTRAINTS

We mainly deal with constraints expressed by conjunctive queries with negations CIC$^\neg$ that is Datalog$^\neg$programmes consisting of only one rule of the shape:

inconsistent:- $g_1, ..., g_m, \neg h_1, ..., h_n$,

where g_i and h_i are EDB predicate atoms. In order to ensure safe use of a negation, variables

used in the h_i's must occur among the g_i's. The content (body) of the rule above is called *constraint query*. If it has an answer, the constraint is violated.

As EDB relations can be local or distant, we introduce o notation to distinguish between local and distant. This notation also includes in the meaning the variables from subgoals. Thus, CIC$^\neg$ will be represented by [4]:

inconsistent:-

$$P(\overline{X}, \overline{Y}), \underset{i \in L}{\wedge} \neg Q_i(\overline{X}, \overline{Y}_i, \overline{Z}_i), \underset{j \in M}{\wedge} \neg Q_j(\overline{X}_j, \overline{Y}_j, \overline{Z}_j), R(\overline{Y}, \overline{Z}) \qquad (3)$$

- $P(\overline{X}, \overline{Y})$ specifies a conjunction of zero local subaims, whose majority is positive (i.e. subaims with local predicates),
- $\boldsymbol{R}(\overline{Y}, \overline{Z})$specifies a conjunction of zero distance subaims, whose majority is positive,
- $\overline{X}$denotes an ordered set of variables, used for *positive local subaims*, but not for for positive distance subaims,
- $\overline{Y}$denotes the set of variables used in a certain *positive local subaim* and a certain *positive distance subai*m,
- $\overline{Z}$denotes the set of variables used in *positive distance subaims*, but not for positive local subaims.

Hence, $\overline{X}, \overline{Y}$ and $\overline{Z}$ are mutually disjunctive. To stress the fact that predicates in R have distance subaims in the constraint query (2), one uses **bold.**

- The negated subaims are divided in two subsets of indices L and M: indices L are *negated local subaims*, i.e. for $i \in L$, $Q_i(\overline{X}_i, \overline{Y}_i, \overline{Z}_i)$ denotes a subaim with a certain local predicate; indices M are for *negated distance subaims*, i.e. for $j \in M$, $Q_j(\overline{X}_j, \overline{Y}_j, \overline{Z}_j)$ denotes a certain subaim with a certain distance predicate. Obviously, we suppose that L and M are disjunctive. Notice the use of **bold** for distance Qs.
- $\overline{X}_i \subseteq \overline{X}, \overline{Y}_i \subseteq \overline{Y}, \overline{Z}_i \subseteq \overline{Z}$ are supposed for $\forall i \in L$ and $\forall i \in M$, to ensure the safe use of negation. We use$\overline{x}$ (respectively$\overline{y}$, $\overline{z}$) to denote a vector of constants of the same arity as $\overline{X}$ (respectively $\overline{Y}, \overline{Z}$). If $\overline{X}_i \subseteq \overline{X}$, $\overline{x}_i$ is the projection of $\overline{x}$ on the variables in $\overline{X}_i$. If $\overline{x}$ and $\overline{x}$' are two vectors of same arity constants as $\overline{X}$, we shall use $\overline{x} =_M \overline{x}$' as an abbreviation for $\wedge_{j \in M}$ $\overline{x}_j = \overline{x}_j$', showing that two vectors are concordant with the variables in each $\overline{X}_j, j \in M$.
- Now we will use the following terminology for subqueries in the constraint: we call P and Q_i, $i \in L$, for local queries, R and Q_j, $j \in M$, for distance queries.

4.1.2. Complete local tests

Let us give a more accurate definition of the *Complete local tests,* in short CLT.

Definition. *If:*

- *Q is a query representing a constraint testing local databases D_{local} and at distance databases D_{remote} for violations as in (3)*
- *an update A of D_{local},*

then the condition of local test CLT is a condition characterized by:

$$(\forall D_{remote})Q(D_{local}, D_{remote}) = \emptyset \quad \Rightarrow Q(A(D_{local}), D_{remote}) = \emptyset \qquad (4)$$

4.1.3 An alternative for the incorporation approach

The incorporation of the query [3] is an often met implication when working with databases. Given two queries P and Q on the database D, we say that P is incorporated in Q, written $P \subseteq Q$, if the response to P is a subset of the response to Q, for any instance of D.

This approach relies on the generation during compilation of the interrogation tests during execution [1] [8].

4.2. Extended example

We shall now look for a complete local test for our first example. Formal results for the general case in the following chapter will also be presented. We have to find complete tests for constraint (2):

$$(\forall P, T, T')tcurent(P,T) \wedge pretrat\ (T,\ T') \Rightarrow tant(P,\ T') \vee specialistOK(P,\ T) \qquad (5)$$

if *tant* is distant. First a simple insertion and then a simple deletion will be taken.

Consistency for a simple insertion

Let us suppose we want to insert *tcurent (Pop, tr187).* For *Pop* to follow treatment *tr187*, with no constraint violation, we must be sure that he (*Pop*) has got an approval from the specialist doctor or has fulfilled all requirements

(*pretrat*). In other words, a cover for any tuple pair must be found *tcurent (Pop, tr187)* and *pretrat(tr187, T')*. This condition is formally expressed by replacing P by *Pop* and T by *tr187* in (5):

$$[(\forall T')pretrat\ (tr187,\ T')\ \Rightarrow\ tant(Pop,\ T')]\ \vee\ specialistOK(Pop,\ trat187) \tag{6}$$

But relationship *pretrat* is not available and we cannot choose directly the requirements for *tr187*. To avoid the ,,unknown" *pretrat(tr187,*T') in (6), we must be able to "limit" the requirements for *tr187*or otherwise our "rival" could prevent *Pop* from following treatment *tr187*, inventing requirements *Pop* has never fulfilled. We used an adversary argument for the unknown, where our imaginary "rival" tries to determine constraint violations by controlling the unknown.

To find this restriction, a key notice would be that the ,,rival" is not totally free to choose a state for *pretrat* and that he must observe the constraint saying that all previous treatments *tr187* were legally made (by substituting T by *tr187* in (5)):

(∀P)[*tcurent(P,tr187)*

$$\wedge\neg specialistOK(P,tr187)\ \Rightarrow (\forall T')[pretrat\ (tr187,T')\Rightarrow tant(P,T)] \tag{7}$$

Modifying the premise of the two implications in (7), we obtain a limit for the unknown *pretrat*(*tr187,T'*). We call this limit *possible-tr187-pretrat (T representing all* normally attended cures, with no approval on behalf of the specialist doctor, expressed by:

possible-tr187-pretrat(T') $\stackrel{def}{\equiv}$

$$(\forall P)[tcurent(P,tr187)\ \wedge\neg\ specialistOK(P,tr187)\ \Rightarrow tant(P,T') \tag{8}$$

This limit is tight and a complete test with the given insertion is reached by replacing the unknown *pretrat*(*tr187,*T') by *possible-tr187-pretrat*(T') in (5):

$$(\forall T')[possible\text{-}tr187\text{-}pretrat(T')\Rightarrow tant(Pop,T')]\ \vee\ specialistOK(Pop,trat187) \tag{9}$$

4.3. Other optimization strategies

Let us consider example 1 of distributed databases. All the tests discussed require consultation of the entire local relationships. To diminish the local test time, we propose a new strategy to solve this issue.

Let us suppose we want to insert a tuple or modify some tuples in a distributed database and check some integrity constraints tested locally. In this respect, we make *a journal* of the movements (operations of insertion and deletion) presumed by setting up a cache database with all tuples ¡d_1, d_2, ..., d_n¿ inserted or modified, to which an attribute *FA- Accessing Frequency* will be added, with the range of integers incremented any time when the cache is accessed (DBC) initially, the value initialized by 1. In DBC tuples will be of the shape:

$< d_1, d_2, ..., d_n, fa >$,

where fa is from the range of the FA attribute.

DBC is ranked according to *FA* values and its cardinal can be limited to a reasonable value N, by checking conditions for preservation in DBC, *FA*<*N*.

4.4. Conclusions

We were concerned with the issue of preserving global consistency when updating a distributed database and when not all relationships are available and updates relate to possible relations. This problem is solved for integrity constraints of databases when conjunctive queries are negated (CIC¬). In case a query defining a constraint exhibit both recursiveness and negation, the issue becomes very complicated and the decision is difficult to make [7].

More clearly, for CIC¬, insertions and deletions, respectively, from local relations result in insertions, respectively deletions from the local queries of the constraints [5].

With CIC¬ where distance predicates do not come up more than once, the local insertion and deletion consistency can be demonstrated fully in time, as a polynomial magnitude related to local updates and local relations [6]. Moreover, these tests can be generated during compilation as SQL or Datalog¬ queries, safely, non-recursively their magnitude increasing linearly (the best case) or exponentially (the worst case), function of the constraint size.

References

[1]Moldovan, M.: *Problema integritatii în baze de date distribuite*, teza de doctorat, Universitatea"Babes-Bolyai", Cluj-Napoca, 2004.

[2] Moldovan, Gr., Damian, S.: *Sur la redistribution des bases de donnees dans un reseau des ordinateurs*, Univ. Cluj-Napoca, Fac. Math. and Phys., Research Sem., Preprint no. 9, pag 57-60, 1989.

[3] Ullman, J.D., *Principles of Database and Knowledge-Base Systems*, Computer Science Press, 1989.

[4] Huyn, N.: *Testing CQCØ constraints under limited data access*, Technical Report, URL http://wwa-db.stanford.edu/pub/papers/cqcnclt-tr.ps.

[5] Fellegi, I.P.: *On the Question of Statistical Confidentiality*, JASA, 1972.

[6]Tigan, S., Achimas, A., Drugan, T.: *Informatica si statistica aplicate în medicina*, Ed. Srima, 2001.

[7] Blaga P.: *Statistica matematica*, lito. Univ. Babes-Bolyai, Cluj-Napoca, 2000.

[8]Dollinger, R.: *Baze de date si gestiunea tranzactiilor*, Ed. Albastra, 2001.

[9] Huyn, N.: *Testing CQCØ constraints under limited data access*, Technical Report, URL http://wwa-db.stanford.edu/pub/papers/cqcnclt-tr.ps.

[10] Ramakrishnan, R.: *Database Management System*, Mc Garw-Hill, 1998.

[11] Tâmbulea, L.: *Baze de date*, Litografia Universitatii Babes-Bolyai, Cluj-Napoca, 2001

Grigor MOLDOVAN
Department of ComputerScience, University of Babes-Bolyai Cluj-Napoca
email:*moldovan@cs.ubbcluj.ro*

Madalina VALEANU
Department of Medical Informatics and Biostatistics, University of Medicine and Pharmacy "Iuliu Hatieganu" Cluj-Napoca
email:*mvaleanu@umfcluj.ro*