

Implementation of Integrity Constraint Checker for Healthcare Database System

Kyawt Kyawt San
University of Computer Studies, Kyaing Tong
kyawtkyawts@gmail.com

Abstract

Checking integrity constraints and enforcement is by now firmly established as a key functionality of a DBMS. DBMS must therefore help prevent the entry of incorrect information by enforcing integrity constraints. This paper presents integrity constraint checker for hospital database. The checker will also guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency by using an efficient algorithm, Constraint Planning Algorithm (CPA). Constraints to be enforced are also stored in the meta database. CPA takes as input an update statement and checks violation with the constraints stored in the meta database. In case of violation, the update statement is rejected and the violated constraint is output to the user. This algorithm is efficient since the algorithm does not require the update statement to be executed before the constraint check is carried out.

1. Introduction

With the constant rise of computers abilities and trustworthiness of computers in general, the utilization of systems for data storage has risen as well. Moreover, the success of an organization depends on its ability to acquire of an organization depends on its ability to acquire accurate and timely data about its operations, manage this data effectively and to use it to analyze and guide its activities. As these reasons, integrity theory plays an important role in the relational model, which has obtained a great success in both theory and system.

Using a relational database, it can specify what kind of data a database column is allowed to contain and it can also set data fields, numeric fields, text fields, etc. this gives us control over data integrity. Data integrity can increase the reliability of the data by setting field properties, linking tables and by applying data integrity rules. The consistency of the database is preserved by imposing integrity constraints on such interrelated data.

Integrity checking and maintenance are central issues, as without any guarantee of data consistency, the answers to queries become unreliable. When database system operates, there is a very large

likelihood of constraints to be violated. An update statement issued on a database might cause a constraint to be violated essentially endangering the consistency of the database. Frequent changes in data causes frequent constraint violations causing the system to rollback frequently. Such systems are inherently inefficient as they consume lot of resources for rolling back the database state. Hence, a complete, standalone system that enables efficient and speedy checking of constraint violations is needed [2].

One of the major modules in this system is integrity constraint checker which is responsible for interfacing with the meta database. This system will check hospital administrative functions such as admit, discharge, and treatment and so on. Data all about the patients is stored in tables such as Registration, Admit, Case, Treatment, and so on. To fulfill these tasks, whenever an event of update statement to these tables in the database, consistency checking is first carried. Only if it is a non-constraint violator the update statement is carried out. Otherwise, the update statement is rejected. This paper presents integrity constraint checker architecture by using Constraint Planning Algorithm (CPA).

The rest of the paper is organized as follows : Section 2 presents importance of data accuracy and validity in relational database, classifications of integrity constraints applied in this system, defining critical roles of consistency checking in relational database and . The constraint checker internal architecture is explained in Section 3. Constraint Planning Algorithm is also discussed in this section. In Section 4, implementation results with processing steps applying CPA are presented. Conclusions can be found in Section 5.

2. Importance of Data Accuracy and Validity in Relational Database

The need for storage of collection of records or data gave rise to special branch of computer systems commonly called database management systems. Today computer systems are used for storage of important data in various areas of human activities. Properties like safety, dependability and integrity of such a systems are necessary in order to establish a reasonable degree of confidence in them. These

systems are now very reliable and are therefore used also in areas like medicine, financial transactions, trade, etc., where the data are extremely important and their damage would cause major difficulties and severe damages. Moreover, the success of an organization depends on its ability to acquire accurate and timely data about its operations, manage this data effectively and to use it to analyze and guide its activities.

As these reasons, the accuracy of the data managed by database systems is vital to any application utilizing data for various purposes. Hence, Consistency checking is an important problem in the area of database system.

2.1 Integrity Constraints

Integrity constraints have been studied from various aspects since the introduction of the relational data model. Integrity of data ensures that the value of the data is meaningful and valid. Integrity of data is achieved by placing restrictions on data values and keeping the relational link valid at all times. These restrictions and linking are expressed as integrity constraints. To maintain the integrity of the data within the database, the followings need to be considered:

Entity integrity is normally enforced through the use of a primary key or unique index to ensure that every row in a table is unique.

Domain integrity ensures that the data entered into a table is not only correct, but also appropriate for the columns into which it is entered. The validity of a domain may be as broad as specifying only a data type (text, numeric, etc.) or as narrow as specifying just a few available values.

Referential Integrity ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation. *Referential integrity* constraints ensure that all foreign keys are maintained. A foreign key is a value in one table that references, or points to, a related row in another table.

DELETE statement's option for Foreign Key Constraints:

- *CASCADE* – when rows with parent key values are deleted, causing all rows in child tables with dependent foreign key values to also be deleted.
- *NO ACTION* – If any row in the child table does not have a corresponding parent key, the deletion is rejected when the *NO ACTION* is used in the DELETE statement. *NO ACTION* means that a non null delete value of a foreign key must match some value of the parent key of the parent table when the DELETE statement is completed.
- *RESTRICT* – If any row in the child table matches the original value of the key, the

deletion is rejected when the *RESTRICT* option is applied.

UPDATE statement's option for Foreign Key Constraints:

- *CASCADE* – when rows with parent key values are updated, causing all rows in child tables with dependent foreign key values to also be updated.
- *NO ACTION* – If any row in the child table does not have a corresponding parent key, the update is rejected when the *NO ACTION* is used in the UPDATE statement. *NO ACTION* means that a non null delete value of a foreign key must match some value of the parent key of the parent table when the UPDATE statement is completed.
- *RESTRICT* – If any row in the child table matches the original value of the key, the update is rejected when the *RESTRICT* option is applied.

State Transition Constraints: deal with two consecutive database states.

State sequence (temporal constraints): These constraints refer to more than two database states (not necessarily consecutive database states).

2.2 Relational Database and Consistency Checking

Databases play a pivotal role in almost every organization in today's information-based society. A Relational Database is typically made up of many linked tables of rows and columns that is created and managed by a relational database management system (RDBMS). In addition to specifying the attributes (column names and associated data types) of each table, it may specify integrity constraints that the data must satisfy. Integrity rules are part of the database and are enforced by the RDBMS. The major feature of a relational model is that each record in the relational database contains information related to a single subject and only that subject.

Database applications play a critical role in almost every modern organization. In RDBMS, the system manages all data in tables. Information is joined on related values from multiple tables or queries. The database must also be assured that all update statements can execute successfully and that the resulting database state satisfies the integrity constraints before permanently saving in the new database state. In order to protect the database from corruption due to a variety of causes, apply constraints, or rules, to the structure of the database and its contents. Integrity checking may be performed at the time of on input, on deletion and on update.

This paper presents integrity check on update operations before saving in the database. Update

operations can be insert or delete or a modify statements. In order to provide integrity checking, constraints are implemented by a mechanism that detects violations and then discards the modifications that caused the violations. Therefore, this system also ensures that the database avoids unnecessary rollback operations. In each occurrence of an update operation, integrity guard must be carried out. Overview of integrity guard is shown in figure 1.

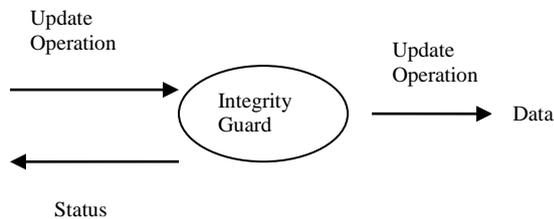


Figure1. Overview of Integrity Guard

2.3 Important Roles of Integrity Constraints in Relational Database

Integrity control is a key feature of systems that provide access to large and shared data resources. Especially when such resources are updatable, integrity constraints have, as their prime use, the role of defining the system's criteria for update validity. The purpose of integrity constraints in database is to prevent semantic inconsistencies in data. Moreover, they are predicates on the database and they must always be true and checked whenever database gets updated. Although database system cannot protect against all accidental errors such as a date being entered incorrectly and mistyping a name, database system should protect against unreasonable entries, updates, deletions etc.

Attempts to alter a database in ways which violate integrity must be prevented. Responses to these events include reject the attempted operation or request or prompt for alternative values or abort the current transaction. To make integrity control techniques usable for database practice, attention should be paid both to issues of functionality and semantics, and to issues of feasibility and performance.

3. Constraint Checker Overview

In this section, overview of the system, constraint checking procedure and constraint checker architecture is presented. Using the database description of database objects, meta database is constructed. Constraints to be enforced are also stored in the meta database. Information of all tables and constraints are also stored in the meta database.

A module, constraint checker that accepts insert/update/delete request from user and considers constraint from meta database and decides if any constraint is violated. If one of the constraints is violated, the update statement is rejected. In order to perform these tasks, an efficient algorithm, constraint planning algorithm (CPA) is presented for checking any violations caused by the update statement input by the user. CPA forms the algorithmic backbone for the constraint checker.

3.1 Example Database

As it increases in the number of patients, inevitable mistakes happen more and more often in hospital. The most effective way of reducing or eliminating these is to use computer based database system efficiently managing the administrative operations of patients in the hospital. In this paper, healthcare database system maintains seven tables: Doctor, Registration, Admit, Case, Treatment, Visit and OPD. The healthcare database enables to add new patient record, to edit or delete the existing records in all seven tables without violating the database. In each occurrence of an update operation on a table, integrity check is performed before saving in the database. Only it is a non-constraint violator, the update statement is carried out. Here is a typical healthcare database system.

Doctor: Doctor relation with attribute names (DrID, DrName, Dob, Admitted Date, Degree, NRC No,...) are maintained.

Registration: A patient is whether admitted or not, he or she must register once. Patient relation with attribute names (RegNo, Name, RegDate, NRC No, ...) are also recorded.

Admit: Only admitted patients are stored in this table with their attributes such as AdmitNo, Admitted Date, Discharge Date, DrName, Disease,etc., are recorded.

Case: A patient can have one or more cases during their admission periods uniquely identified by their CaseIDs. Case relation with attribute names (CaseId, AdmitNo, RegNo, Disease,DrName,..) are recorded.

Treatment: There must be no patients without a treatment. A patient must have at least one treatment. Treatment relation with attribute names (TreatmentNo, AdmitNo, RegNo,TreatmentDate, Disease,..) are also recorded.

OPD: Only outside patients are stored in this table. Patient relation with attribute names (RegNo, Name, Date, NRC No, ...) are also recorded.

3.2 Constraint Checker Internal Architecture

The internal architecture of the constraint checker and the overall procedure of constraint checking are explained using Figure 2. The integrity constraint

checker has three major modules: meta database extractor, constraint checker, and constraint executor.

Meta database extractor: extracts all the constraints being affected by the update statement.

Constraint checker: makes a decision whether constraint violation occurs or not upon the constraints input by the meta database extractor for efficient constraint checking.

Constraint executor: is responsible to output violated constraints to the user and if not violated saving in the database by interacting with the data source.

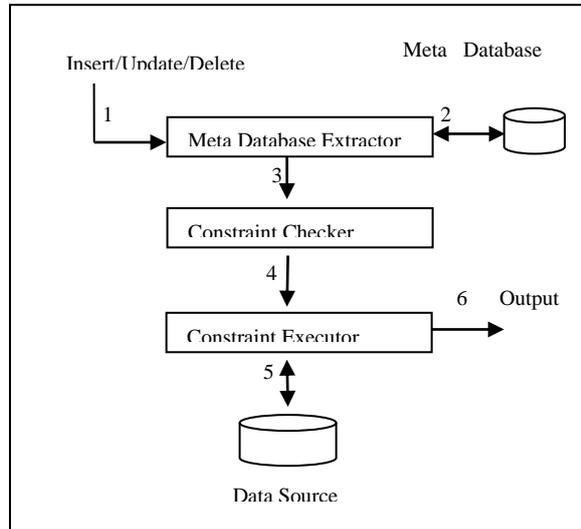


Figure2. Overview of Integrity Constraint Checker

Figure 2 shows the overall procedure of constraint checking in following six steps. An update statement is issued on a table. The meta database extractor computes the list of constraints being affected by the update statement and also returns this list to the constraint checker. The constraint checker takes as input that constraint list and makes a decision if a constraint is violated. The value of each constraint is either 0 or 1 and if the value of constraint is 1, the constraint is violated and rejected update statement.

Step 1

When the user issues an update statement U to the database, the database management system identifies database objects being modified. The output from this step is the database object list (DOL).

Step 2(Meta Database Extractor)

The meta database extractor takes as input database object list. It contacts the meta database and gets the list of constraints being affected by the update statement.

Step 3

The meta database extractor sends affected constraints extracted from the meta database to the constraint checker.

Step 4 (Constraint Checker)

The constraint checker takes as input the affected constraints and constructs the violated constraint list and a decision is made.

$CL(C_i) = \langle C_i \rangle$ where

C_i is the constraint identifier

Table 1. Constraint List

| C_i | Description |
|-------|---|
| C25 | Constraint C25 states that Discharge Date must be greater than Admit Date |

Step 5 (Constraint Executor)

The constraint executor outputs error message to the user if violated. Otherwise, the update statement is saved by interacting with the data source.

Step 6

The results are output to the user.

3.3 Constraint Planning Algorithm (CPA)

Algorithm CPA (Constraint Planning Algorithm) shown in Figure 3 gives efficient constraint checking. Algorithm CPA takes as input the update statement U and outputs the list of constraints (C_{ij}) for each C_i being affected by U . An update U can be and update involving an insert or a delete or a modify statement. The update statement is carried out only if it is a non-constraint violator. The approach of the constraint planning algorithm (CPA) is to scan through the constraint C_i , update statement U and then generate the affected constraints. The value of each C_{ij} is either 0 or 1 and if the value is 1, the constraint is violated, otherwise not.

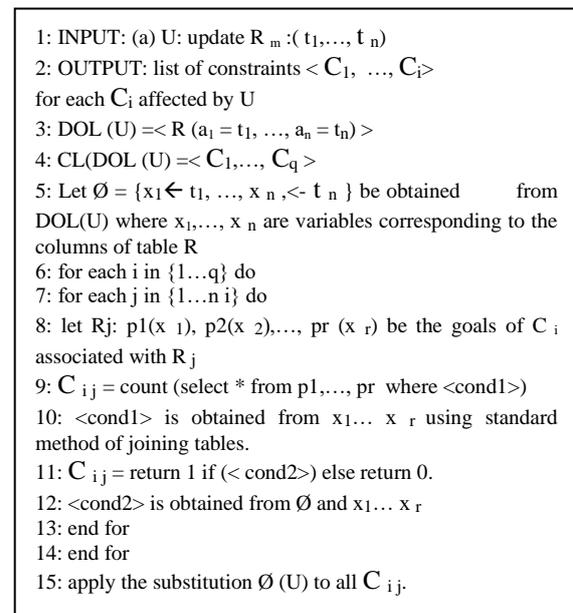


Figure 3. CPA Algorithm

Database Object List (DOL) identifies the database objects being modified by the update statement, U. DOL(line3) identifies the table R with attributes (column names) $a_1 \dots a_n$ inserted with values $t_1 \dots t_n$. CL (line4) gives the list of constraints being affected by the update statement. The outer for loop variable i (line6) loop through all the constraints $C_1 \dots C_q$ affected by the update U. The inner for loop variable j (line 7) loops through each table ($\langle R_{11}, \dots, R_{1n} \rangle, \dots, \langle R_{q1}, \dots, R_{qn} \rangle$) for each constraint i. Inside the for loop (line6-15), all the constraints C_{ij} 's are generated.

4. System Implementation

The main goal of this system is to facilitate the checking of updates for violation of database integrity constraints and to ensure integrity and consistency of data in a database.

In this section, a prototype of the system implementation is shown in figure 3 and figure 4. The figure 3 shows that user enters invalid discharged date in Admit table and tries to save in the database. As a result, the violated update statement is rejected and output error message to the user as shown in figure 4.

Whenever user tries to enter the inconsistent data into the healthcare database, integrity checking is first carried out with the pre defined constraints stored in the meta database. By performing integrity checks before saving in the database, expensive rollback operations are avoided. Pushing most of the processing before saving in database, efficiency is gained.

4.1 Processing Steps of the System Implementation Using CPA

The processing steps of the system implementation are traced line by line using CPA algorithm as follows:

Input:

User updates admitted patient record via Admit entry form.

$U = \text{Admit} ('', R0006, 'Ma Zar Zar Oo', '17/Jul/2008', '13/June/2008', 'Daw Hnin Thuzar', 'Stomach Ache')$

Output:

List of Constraints ($C_1 \dots C_i$) for each C_i affected by update statement U.

/ DOL from CPA line (3) */*

$\text{DOL}(U) = \text{Admit} \{ \text{Admit No}='', \text{Reg No}=R0006, \text{Name}='Ma Zar Zar Oo', \text{Admit Date}='17/Jul/2008', \text{Discharge Date}='13/June/2008', \text{Dr Name}='Daw Hnin Thuzar', \text{Disease}='Stomach Ache' \}$

/ Constraint List from CPA line (4) */*

$\text{CL}(\text{DOL}(U)) = \langle C25 \dots \rangle$ where C25 states that Discharge Date must be greater than Admit Date.

One of the affected constraints, only C25 is traced for illustrative purposes.

/ CPA line 5*/*

$\emptyset = \text{Admit} (\text{Admit No}='', \text{Reg No}=R0006, \text{Name}='Ma Zar Zar Oo', \text{Admit Date}='17/Jul/2008', \text{Discharge Date}='13/June/2008', \text{Dr Name}='Daw Hnin Thuzar', \text{Disease}='Stomach Ache')$

The constraint checker loops through affected constraints output by the meta database extractor and corresponding tables. (CPA lines 6-7).

/ Constraint C25 is generated from Algorithm CPA lines (9-12) */*

$C25 = \text{count} (\text{select} * \text{ from Admit where Admit.Admit Date} > \text{Admit.Discharge Date})$

Apply the substitution $\emptyset(U)$ to the affected constraint C25.

$\emptyset(C25) = \text{return } 1 \text{ if } (17/July/2008 > 13/June/2008)$ else return 0

The value of each C_{ij} is either 0 or 1 and if the value of constraint is 1, the constraint violation occurs. Since $\emptyset(C25) = 1$, constraint C25 evaluates to true, hence, the constraint C25 is violated and the result is output to the user as shown in figure 4.

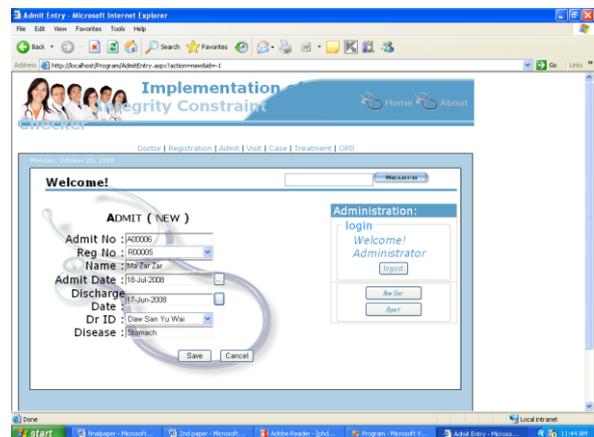


Figure.3. Input Inconsistent Update Statement to the Admit Relation

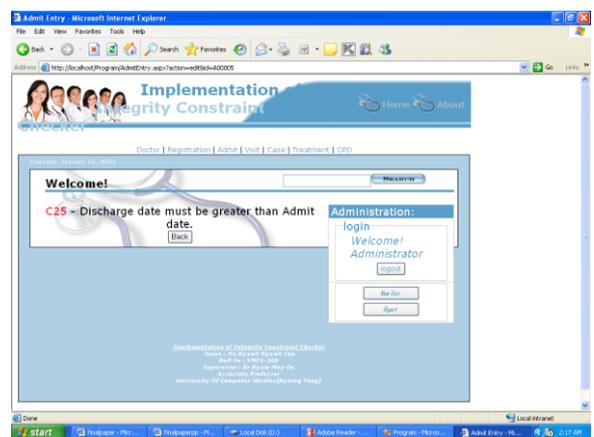


Figure 4. Rejecting Inconsistent Update Statement

5. Conclusion

In other constraint distribution model, an update statement is first carried out and the new database state is checked for constraint violation. If the constraint is violated, the update statement is rolled back. This system differs from others by giving an algorithm that automatically checks a constraint with the constraints stored in the meta database. This approach is much more sophisticated, as the healthcare system checks for constraint violation without actually updating the database. The update is executed only when there are no constraint violations. Hence, CPA algorithm is efficient as there are no problems involved with rollbacks as such. Also, the overhead introduced from this algorithm is very negligible as the only extra overhead is the time required for constraint checking on the relation where update is happening. Constraints are implemented by a mechanism that detects violations and then discards the modifications that cause the violation. The additional time spent on integrity constraint design will eventually pay off in better data quality.

References

- [1] Praveen Madiraju and Rajshekhar Sunderraman, “Efficient Constraint Planning Algorithm for Multidatabases”, Department of Computer Science, Georgia State University ,Atlanta GA 30302
- [2] PRAVEEN MADIRAJU. “Global Semantic Integrity Constraint Checking for a System of Databases” (2005).
- [3] Robert Mach, “Design of Integrity Check and Repair Algorithms for PostgreSQL Data File”, Czech Technical University in Prague, Faculty of Electrical Engineering, January 2008.
- [4] Maedeh Sharif Khodaei, “Case Study: Implementation of Integrity Constraints in Actual Database Systems”, Czech Technical University in Prague, Faculty of Electrical Engineering ,Department of Computer Science and Engineering, 2007/2008.
- [5] Hai Zhuge and Yunpeng Xing, “Integrity Theory for Resource Space Model and Its Application “, China Knowledge Grid Research Group, Key Lab of Intelligent Information Processing Institute of Computing Technology, Chinese Academy of Sciences, 100080, Beijing, China.
- [6] Can Tiirker and Michael Gertz “Semantic Integrity Support in SQL-99 and Commercial (Object) Relational Database Management Systems”
Swiss Federal Institute of Technology (ETH) Ziirich
Institute of Information Systems, ETH Zentrum CH-8092
Ziirich, Switzerland, University of California, Davis,
Department of Computer Science, One Shields Avenue
,Davis, CA 95616, USA.
- [7] Andreas Behrend, Rainer Manthey and Birgit Pieper, “An Amateur's Introduction to Integrity Constraints and Integrity Checking in SQL”, University of Bonn, Institute of Computer Science III, Romerstr. 164, D-53117 Bonn, Germany.
- [8] Michael Benedikt and Glenn Bruns, “On Guard: Producing Run-Time Checks from Integrity Constraints”, Bell Labs, Lucent Technologies.