Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp348-353)

# Review of data Consistency and Integrity Constraints in Spatial Databases

**KALUM PRIYANATH UDAGEPOLA, LI XIANG, YANG XIAOZONG, A.W. WIJERATNE**
School Of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001
CHINA

*Abstract:* - An important activity in the design or enhancement of a particular database application consists of identifying the integrity constraints that must hold on the database, which are used to detect and evaluate inconsistencies to buildup an efficient spatial database. It is also possible to improve data quality by imposing constraints upon data entering the database. The main contribution of this paper is a survey of existing approaches to deal with any inconsistency issue in spatial databases that emphasize the current state of the art and that outline research issues in the context of consistency tolerance to the better spatial database applications.

*Key words:* - Data consistency, Integrity constraint, Spatial database, Database Management System (DBMS), Geographic Information System (GIS), Static constraints, Semantic data, Topological relationship.

## 1 Introduction

Spatial database consistency has become an issue of increasing concern to researchers and practitioners in the field of GIS. At the database level, consistency refers to the lack of any logical contradiction within a model of reality [17]. It is a concept that originates from database systems and it is also called integrity.

It is necessary to update the database needs to apply the integrity constraints to maintain data consistency with time. Integrity constraints seems to be widely utilized and considerable, research efforts have been made to investigate the basic logic, the architecture and the effective implementation of constraints in DBMS. It is said that improved data consistency can be achieved by means of integrity constraints.

## 2 Data consistency in Spatial Databases

Research in the area of consistency in spatial databases has tried to clarify concepts about types of consistency, incorporate integrity constraints at different levels of the database designing, conceptualize consistency problems in generalization and information-integration processes. The research has focused on how to monitor inconsistencies. Although issues about inconsistency tolerance have been addressed for traditional relational databases, spatial databases have not handled explicitly inconsistency tolerance in query answering [18]. The consistency checking is the first step towards high quality database. The terms integrity or consistency are used to characterize these requirements [11]. The integrity constraints can prevent invalid updates to the database and proper actions can be taken whenever such an attempt is made. Data consistency tests are either built in integrity constraints that can be activated by the user, or they are consistency rules and triggers written for specific database transactions or methods. Consistency tests are needed at various stages of spatial data handling. A database fulfilling all consistency constrains that have been imposed is called consistent. In spatial database, data may be logically consistent in every layer, but can be inconsistent among different layers. Data from different sources or different scales covering the same area is often consistent within a particular source or scale, but when they are combined together, the inconsistency occurs. The major challenges of data consistency in spatial databases fall into two categories: Minimizing the positional and attribute error, and Ensuring the logical completeness of the data.

## 3 Database Management Systems: Integrity consistency in Spatial Databases

In spatial database, integrity constraint checking can be performed at two stages: at database creation (upon data entry) and at each spatial object insertion, deletion and update [13].Integrity constraints have been used to describe conditions that must be satisfied with every legal instance of a relation. Enforcing integrity constraints can ensure the logical consistency of the data contained in a database. Integrity refers to the (logical) accuracy or validity of data. Constraints can be categorized based on the levels specified in a database and non-spatial and spatial data. These two classifications are not mutually exclusive but it is hard to draw direct comparisons between them.

## 3.1 Classification of Integrity constraints at different levels

The specification of integrity constraints is part of the database design process. There are three main techniques: (i) Inherent constraints (ii) Implicit constraints (iii) Explicit constraints by which they can be specified in a database [8]. Some constraints can be applied at the database schema level and others have to be applied in the update programs.

**(i) Inherent constraints** are inherent to the data model itself and do not need to be specified in the schema, but are assumed to be hold by the definition of the model constructs (i.e. A character attribute field cannot be used for calculation). Inherent constraints are the rules that define data model constructs.

**(ii) Implicit constraints** are specified using the Data Definition Language (DDL) during the process of creating a database schema. DDL is a language that contains commands to create and alter the databases and tables. These constraints are stored in DBMS catalog so that DMBS software can automatically enforce the constraints each time an update occurs.

**(iii)Explicit constraints** are managed by the application programs, which are associated with the databases which are specified in either procedural approach or declarative approach. Procedural approach involves writing statements to check for the updates that will be applied to the database. Whereas declarative approach is more formal technique in which the constraints are stored in a suitably encoded form.

## 3.2 Non-Spatial and Spatial integrity constraints

### 3.2.1 Non-spatial integrity constraints

Elmasri [8] notes that non-spatial integrity constraints could be subdivided into (a) Static or state constrains (b) Transitional constraints (c) Dynamic constraints.

**(a) Static or state constraints** must be satisfied at every single state of the database that they express which database states are correct and which are not. Database state refers to the data in a database at a single moment of time. Static constraints can be further classified into domains, entity integrity rules, attribute structural constraints, and referential integrities [8]. Static constraints should be checked whenever the database state is change by an update transaction.

**Domains:** This defines the valid values for attributes which are also called value constraints. These values that constitute a domain are often specified via data type.

**Entity integrity rules:** Each instance of entity type must have a unique identifier or primary key value that is not null.

**Attribute structural constraints:** Whether attribute is single valued or multi-valued, and whether it is `null' or not `null' allowed for the attribute.

**Referential integrity constraints:** This guarantees that each foreign key in detail table has valid primary key in the master table. In Relational Database Management Systems (RDBMS) foreign keys are used to establish relationship among tables.

**(b) Transition constraints** are the ones that restrict the possible transaction form one database state to another [8]. In general, this occur less frequently than static constraints and they are mainly specified as explicit constraints.

**(c)Dynamic constraints** are the ones that restrict the possible sequence of state transactions of the database and mainly related to implementation domain [12].

### 3.2.2 Spatial integrity constraints

In general, inconsistencies may relate to both attribute data and spatial data and there are number of ways classifying spatial integrity constraints. The first classification hierarchy classifies it into static or state, transitional and dynamic constrains. The second classifies it into topological, semantic and user defined integrity constraints. In addition to traditional integrity constraints concerning static, transition, and transactional aspects of databases systems [9], rules about spatial data must ensure consistent updating of spatial information. The first classification was similar to non-spatial constraints and the second classification is based on the distinction between topological, semantic and user rules [3]. Static spatial integrity constraints may check the allowed intervals of co-ordinates or the plausibility of geometric and topological relationships between geological objects. [1]

**(a) Topological integrity constraints:** Topology is a mathematical procedure for defining spatial relationships between points, lines and polygons. This refers to asset the rules that govern the geometric and topological consistency in a special domain. There has been some theoretical research into principles of defining these relationships. The issue of defining topological integrity constrains has also been investigated [10]. Servige et al [19] states that the topological integrity constraints are defined as topological errors, and are based on the topological relations defined by the 9-intersections model (9IM) [9]. The present GIS systems are capable of handling many topological connections.

**(b) Semantic integrity constraints:** These differ from topological integrity constraints in that they are concerned with the meaning of geographical features. In this category, an example would be a rule that states that a road may not run through any water body except part of road contains the bridge. If the user attempts to enter, for example a road would not normally run through the water body. These types of semantics are derived from the real world description. In this category the rules to define the constraints are fit into its own data sets and the semantic data is takes into account. The rules would develop from the base of the meaning behind the topological relationship with reality.

**(c) User defined integrity constraints** allow database consistency to be maintained according to user defined constraints. For example for external or legal reasons it may be desirable to locate a nuclear power station away from the residential areas at certain distance. When attempting to enter the power station within the distance, a user-defined rule would be activated.

**(d) Combination of classified constraints:** The first classification of static and transitional constraints are combined with the second classification and summarized in table 1.

| **Spatial integrity constraints** | **State** | **Example** |
|---|---|---|
| Topological rules | Static | All polygons must close. |
| | Transition | If a new line or lines are added to make a new polygon the tables must be updated |
| Semantic rules | Static | The buildings cannot intersect the road |
| | Transition | Height of a mountain may not decrease or increase |
| User defined rules (Business rules) | Static | All streets wider than seven meters must be classified as main roads |
| | Transition | In Cadastre database, the date of subdivision should not be before the registration date of land parcel |

**Table 1 Summary of first and second classification**

Constraints at a conceptual and logical level in spatial databases are inherited by the implementation or physical level. These are translated into a proprietary scripting language or into explicit constraints coded in application programs [9]. At a logical level, Hadzilacos and Tryfona [10] describe a logical model with definitions of constraints based on topological relations. They state that it is possible but cumbersome to define topological constraints based on absolute positions. Therefore, they use a formal framework for defining topological relations [5][6] upon which integrity constraints are specified. This framework defines topological relations between subsets of a classical topological space by the emptiness or non-emptiness of the two-by-two intersections of the subsets' interiors (Θ) and boundaries (O). Table 1 summarizes the resulting eight possible topological relations between two polygons. Within Hadzilacos and Tryfona's framework [10], spatial relations and integrity constraints are expressed using first-order logic. Atomic topological formulae in combination create topological sentences. Atomic topological formulae include geometric operators over objects, elementary topological relations between objects, and comparison between objects' attributes.

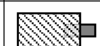
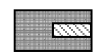
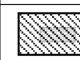
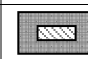
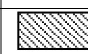
| Operations | OO | ΘΘ | OΘ | ΘO | Relation |
|---|---|---|---|---|---|
| disjoint | ∅ | ∅ | ∅ | ∅ |  |
| meet | ¬∅ | ∅ | ∅ | ∅ |  |
| overlap | ¬∅ | ¬∅ | ¬∅ | ¬∅ |  |
| cover | ¬∅ | ¬∅ | ¬∅ | ∅ |  |
| covered by | ¬∅ | ¬∅ | ∅ | ¬∅ |  |
| contain | ∅ | ¬∅ | ¬∅ | ∅ |  |
| inside | ∅ | ¬∅ | ∅ | ¬∅ |  |
| equal | ¬∅ | ¬∅ | ∅ | ∅ |  |

**Table 2 Definition of topological relations between regions**

The formal specification of this constraint for land parcels *lp* andbuilding blocks *bl* based on the topological relations defined in Table 2 is:

$$\forall\ (lp, bl)[\neg inside(lp, bl) \wedge \neg covered\ by(lp, bl)] \quad (1)$$

Some topological constraints define geometric primitives or some spatial dependence of composite objects. Consider, e.g. partitions of a space. To define a partition rule in first-order logic, one needs to consider predicates of the type $P_i(x)$, with $x$ being an interior point of an object $P_i$. The spatial aggregation of partitions $P_0() \ldots P_n()$ into $W()$, assuming that partitions can only be meet or disjoint, where meet and disjoint were defined in Table 2:

$$\forall\ (P_i, P_j)\ [meet(P_i, P_j)\ \_\ disjoint(P_i, P_j)] \quad (2)$$

is then defined by the statement that a point x in the aggregation must belong to one partition Pi():

$$\forall (x)[W(x) \equiv (P_0(x) \_ P_1(x) \_ . . . \_ P_n(x))] \qquad (3)$$

A graph-based model of maps has also been used to establish topological integrity constraints of objects and their aggregations as a map [17]. This model makes it possible to guarantee the consistency of a map through database updates with respect to a set of topological constraints over vertices, edges and faces on the map graph. Theses integrity constraints are equivalent to the mathematical axioms of maps that are defined by a graph that is plane, connected, non separable and formed by edges that are straight lines bounding internal faces. Some attempts have been made to provide end users with easy mechanisms that hide the logic in specifying constraints [4][19]. In several cases spatial constraints are applied, and object pairs that are retrieved must also satisfy these constraints [15]. Another study allows users to define constraints in English -like fashion. Basic components of the language are entity classes, relations, and qualifiers (e.g., forbidden, at least n times, at least most n times, or exactly n times) [19].

# 4 Integrity constraints in Oracle Relational Database Management Systems

Integrity constraints are important to guarantee that data adhere to predefined set of rules, as determined by the database administrators or application developers. The following are different types of integrity constraints that are supported in Oracle relational database Management System [14].

## 4.1 Static constraints

The integrity Constraints such as NOT NULL, UNIQUE Key, PRIMARY KEY, Referential and Check integrity Constraints are called static integrity constraints.

**NOT NULL integrity constraints:** In many database applications, missing attribute value is allowed. However, the attribute value must be known then the NOT NULL constraints will guarantee that no null values are assigned to these attributes.To define an integrity constraint in oracle, include a CONSTRAINT clause in a CREATE TABLE or ALTER TABLE statement. The CONSTRAINT clause has been applied in two forms table constraint and column constraint. The first is defined with syntax rules on any columns in the table and the second is a part of a column definition and can impose rules only on the column in which it is defined.

**UNIQUE integrity Constraints** designates a column or combination of columns as a unique key. To satisfy a UNIQUE constraint, no two rows in the table can have the same value for the unique key. However, the unique key made up of a single column can contain nulls. Oracle enforces unique integrity constraints with indexes. Oracle enforces the UNIQUE key constraint by implicitly creating a unique index on the composite unique key. Therefore, composite UNIQUE key constraints have the same limitations imposed on composite indexes: up to 32 columns can constitute a composite unique key [14].

**PRIMARY KEY integrity Constraints** designate a column or combination of columns as the table's primary key. A PRIMARY KEY, by definition, is both unique and not null. The Oracle implementation of the PRIMARY KEY integrity constraint guarantees that both of the following are true: No two rows of a table have duplicate values in the specified column or set of columns and the Primary key columns do not allow nulls. That is, a value must exist for the primary key columns in each row. Composite primary key constraints are limited to 32 columns, which is the same limitation imposed on composite indexes. The name of the index is the same as the name of the constraint. Also, you can specify the storage options for the index by including the ENABLE clause in the CREATE TABLE or ALTER TABLE statement used to create the constraint. If a usable index exists when a primary key constraint is created, then the primary key constraint uses that index rather than implicitly creating a new one [14].

**REFERENTIAL integrity Constraints** designate a column or combination of columns as a foreign key and establishes a relationship between that foreign key and a specified primary or unique key, known as the referenced key. In this relationship, the table containing the foreign key is called the child table and the table containing the referenced key is called the parent table. Different tables in a relational database can be related to common columns, and the rules that govern the relationship of the columns must be maintained. Referential integrity rules guarantee that these relationships are preserved. The following terms are associated with referential integrity constraints. A referential integrity constraint requires that for each row of a table, the value in the foreign key matches a value in a parent key [14].

**CHECK integrity Constraints** explicitly define a condition. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null). Check constraints violate only if the condition is false.

## 4.2 Database triggers

A trigger is defined as a predefined database procedure, conditionally or unconditionally preceding other database operations automatically. A database trigger is store as PL/SQL (Procedural Language/Structural Query Language) program unit associated with a specific database table and executed automatically when the application or user performs specific operations on the database. However, trigger on the tables is fired if an INSERT, UPDATE, or DELETE statement is issued against a table [14]. Database trigger is used in variety of ways for the information management of the database. However, triggers should be used only when necessary. The excessive use of triggers can result in complex interdependencies. Triggers can be used only in the following situations such as; when a required referential integrity rule cannot be implemented in the previous section, when child and parent tables are on different nodes of a distributed database and when the enforce complex business rules not definable using integrity constraints. Trigger contains three parts: triggering event or statement, trigger restriction and trigger action.

**Triggering event or statement** is the SQL statement that causes a trigger to be fired. Triggering event in UPDATE statement can include a list of columns, but we cannot specify the columns in INSERT or DELETE operations.

**Trigger restriction** specifies a Boolean expression that must be TRUE for the trigger to fire. The trigger action is not executed if the trigger restriction is evaluated to be false or unknown.

**Trigger action** is the procedure that contains the SQL statements and PL/SQL code to be executed when a triggering statement is issued and the trigger restriction is evaluated to TRUE.

### 4.2.1 Types of triggers and their execution

The different types of triggers are Row and statement, BEFORE and AFTER, INSTEAD OF and Triggers on System Events and User Events

**Row and statement triggers** are fired each time when the table is affected by the triggering statement. For example if an UPDATE statement updates multiple rows the row trigger fires once for each row. If a triggering statement affects no row, a row triggers is not executed.

**BEFORE trigger** action is executed before the triggering statement is executed. These triggers are commonly used, whenever some processes has to be completed before the updating.

**AFTER trigger** is an opposite operation of the before trigger. These triggers have executed after the trigger statement is executed. It is used multiple triggers of the same type or combination of these types for any given table.

**INSTEAD OF trigger** provide a transparent way of modifying views that cannot be modified directly through SQL Data Manipulation Language statements (INSERT, DELETE, & UPDATE).

Data dictionary in a database contains all the information about the triggers and integrity constraints of the database. A trigger cannot be explicitly altered; it must be replaced with a new definition. The ALTER TRIGGER command is used only to recompile, enable or disable a trigger. When replacing a trigger, we must include the OR REPLACE option in the CREATE TRIGGER statement. The trigger can be dropped using the DROP TRIGGER command.

## 5 Approaches to manage Integrity constraints in Spatial Databases

The three ways in which constraints can be integrated into database systems [2] are, (1) Queries can be interpreted as a kind of constraint [7], (2) Integrity constraints can be defined on the data in order to define valid database states and (3) Constraints can be used to represent database objects or sets of objects. Hadzilacos et al [10] define a Geographic Relational Data Model that incorporates topological integrity constraints. It highlights the static topological integrity constraints conditions on the allowable values of spatial attributes of the database. Then the queries can be made on topological relationships among geographic objects. In that model the integrity constraints have been implemented as queries on the data. However, in this model a virtual layer is constructed and then combining the rules with the topological relationships. In this approach the constraint rules at the data entry would not be handled. In the second approach [2] quotes that firms describe the spatial integrity constraints that could be created at database design level. He has represented the constraints in Spatially Extended Entity Relationship (SEER) model. Any pair of entity-sets which is related to a sub-type of a node entity is potentially spatially related to one another [2]. This is called schema of instance attribute to facilitate the spatial features within a thematic layer. The last approach is the object oriented approach, as a means of overcoming the traditional data models in the management of spatial data. In this method a database is a set of objects of the real world entities that are abstracted and held as objects. All objects belong to object class which is characterized by a set of attributes and methods. The object data model allows defining the constraint rules as methods in the database. This means that the database will enforce the rules as the data entered to the database or whenever they are modified.

**Use of repository in spatial database schemas**

[2] has discussed about the use of repository to control data

quality and defining metadata. At the simplest level metadata is additional information that is necessary for data to be useful. This repository is also used to store the semantic and user rules of the integrity constraints. The advantages of using the repository are faster application development, reduced the maintenance effort, faster response to business changes, and promote ease of use of database. The repository will be able to store constraints on topological relationship and attribute values which are then imposed at data entry. This research adopts the repository approach designed and developed this repository on Relational Database management system. Facilities for the storage and analysis of large quantities of spatial data are important to many applications, and are central to geographic information systems. This has given rise to a range of proposals for spatial data models and software architectures that allow database systems to be used cleanly and efficiently with spatial data [16].

# 6 Conclusion

Constraints can be classified into two categories based on different levels, non–spatial and spatial data. For constraints based on levels, there are three main techniques: inherent constraints, implicit constraints, explicit constraints can be specified in a database. Non–spatial integrity constraints can be subdivided into static or state constrains, transitional and dynamic constraints. Spatial integrity constraints can be classified as same as non–spatial integrity constraints, but it can be also classified based on the distinction between topological, semantic and user rules. Since oracle relational database is one of the most important databases, the paper also talks about integrity constraints in oracle database. There are two kinds of integrity constraints supported in oracle relational database management system: static constraints, database triggers. At last, approaches to manage integrity constraints in spatial database are emphasized in the paper.

*References:*

[1] O. Balovnev, T. Bode, M. Breunung, A.B. Cremers, W. Muller, G. Pogodaev, S. Shumilov, J. Sieck, A. Siehil and A. Thomsen, The Story of the GeoToolkit- An Object oriented Geodatabases Kernel System, GeoInformatica, Vol.8, No.1, 2004, pp.5–47.

[2] S. Cockcroft, Towards the Automatic Enforcement of Integrity rules in spatial database systems. In: $8^{th}$ colloqqium of the Spatial Information Research Centre, University of Otago, New Zealand, July 9-11, 1996.

[3] S. Cockcroft, A taxonomy of spatial integrity constraints. GeoInformatica, Vol.1, No.4,1997, pp.327–343.

[4] S. Cockcroft, Modelling spatial data integrity constraints at the metadata level. In D. Pullar, editor, GeoComputation, Brisbane, Australia, 2001.

[5] M. Egenhofer and R. Franzosa, Point-set topological spatial relations. International Journal of Geographical Information Systems, Vol.5, No.2, 1991, pp.161–174.

[6] M. Egenhofer and J. Herring, Categorizing topological spatial relations between point, line, and area objects. Technical Report Report 94-1, National Center for Geographic Information Analysis,1994.

[7] M. Egenhofer, Pre-Processing queries with spatial constraints, In: Photogrammetric Engineering & Remote Sensing, 1994,pp. 783-790.

[8] R. Elmasri and S. Navathe, Fundamentals of Database systems. $2^{nd}$ Edition. Benjamin/Cummings, 1994,pp. 638-643.

[9] R. Elsmari, and S. Navathe, Fundamentals of Database Systems. Addison Wesley, 3ed edition, 2000.

[10] T. Hadzilacos and N. Tryfona, A model in geographic databases. Theoreies and methods of spatio- temporal reasoning in geographic space. Proceedings of international conference GIS, Pisa, Italy, 1992, pp. 252-268.

[11] W. Kainz, Logical consistency In: Guptill and J.L. Morrison, Elements of spstial data quality. Published on behalf of the Internatinal Cartogrphic Conference, Pergamon Press,1995,pp. 109-137.

[12] O. Kufoniyi, Spatial coincidence modeling automated database updating and data consistency in vector GIS. Phd-Thesis,ITC publication number 28,1995.

[13] R. Laurini and D. Thompson, Fundamentals of spatial information systems. Academic Press, 1998,pp. 576-592.

[14] Oracle. Oracle® Database Concepts, 2003,pp. 517-556.

[15]A.N. Papadopoulos, A. Nanopoulos and Y. Manolopouls, Closest Pair Queries with Spatial Constraints , Advances in Informatics: 10th Panhellenic Conference on Informatics, PCI 2005, Volas, Greece, 2005, pp.1–13.

[16] N.W. Paton, M.H. Williams, K. Dietrich, O. Liew A. Dinn and A. Patrick, VESPA: A Benchmark for Vector Spatial Databases, Advances in Databases: 17th British National Conference on Databases, Exeter, UK, 2000, pp.81.

[17] L. Plümer and G. Gröger, Achieving integrity constraints in geographic information systems, GeoInformatica, Vol.1, No.4,1997,pp.345–367.

[18] D.W. Rhind, P.A. Longley, M.F. Goodchild and D.J. Maguire, geographic Information System, Volume 1. $2^{nd}$ Edition, 1999,pp. 182-184.

[19] S. Servige, T. Ubeda, A. Puricelli and R. Laurini,. A methodology for spatial consistency improvement of geographic databases. GeoInformatica, Vol.4,2000, pp. 7–24.