Spatial Data Integrity Ensuring Mechanism in SDBMS

Xing Lin, Yi Zhang, Yu Liu, Yong Gao
Institute of Remote Sensing and Geographic Information System
School of Earth and Spatial Science, Peking University
Peking 100871, China

Abstract—Data quality and integrity is a critical issue of creating and maintaining a spatial database. It is possible to improve data quality by imposing constraints upon data entered into the database. Due to the complexity of spatial data and spatial relationships, existing data integrity technologies are inadequate and insufficient for spatial data. Introducing a spatial data integrity ensuring mechanism to SDBMS will bring much convenience to GIS applications, especially in department of data manufacture and management. This paper is aimed to seek for a proper solution for building such a spatial data integrity ensuring mechanism into SDBMS. From the result of our experiments and comparisons, a SDBMS supporting spatial integrity constraints has been proved to be much more intelligent and efficient in managing the spatial data.

Keywords-SDBMS, GIS, spatial data integrity, rules, constraints

I. INTRODUCTION

In many GIS applications, Spatial Database Management System (SDBMS) plays an important role as an efficient data management approach. With the explosion of spatial data in quantity, spatial data quality has become an issue of increasing concern to researchers and practitioners in the field of Geographical Information System (GIS) [4]. One approach to improving data quality is the imposition of constraints upon data entered into the database [4]. However, existing data integrity constraints are inadequate for spatial data, because spatial data are much richer than traditional, alphanumeric data [7]. A thoroughly review of data quality and integrity with respect to spatial data is discussed in [4], which also includes a taxonomy of spatial data integrity constraints. These constraints are suggested to be identified and recorded at the database design level, which requires data models that are more specific and capable of capturing the semantics of geographic data [6]. [10] gives such a data model called OMT-G with specific mechanisms for modeling structural, geometric and semantic spatial integrity constraints in data model level. Meanwhile, other efforts also focus in modeling spatial data integrity constraints at the metadata level [8]. Some prototype systems and real-world applications have also been carried out [5] [9]. [12] presents a good attempt to seek for a universal routine for enhancing spatial integrity constraints in SDBMS.

However, those definitions and classifications of spatial data integrity constraints cited above come from different views upon spatial data, including data modeling fields, error survey theories, spatial reasoning areas and database designing practices. They are somewhat misleading for common users, and inconvenient while taking them in implementation. Ability and extensibility of present SDBMS hasn’t been paid enough attention yet. Most existing systems use a standalone repository for storing and validating data integrity rules, while applying all of this mechanism into SDBMS could form an integrated system of higher efficiency.

Our research work in this paper is aimed to seeking for a proper solution for building such spatial data integrity ensuring mechanism into SDBMS. The whole article is organized as follows. Section II discusses relative topics with respect to spatial data quality and integrity, resulting in a commonly acceptable definition and classification of spatial data integrity. Our formal definition language for spatial data integrity rules is presented in section III. In section IV, a comparison over three typical styles of implementation architecture is shown first. Then our prototype system is presented with more details about implementation, including the storing schema and the routine for building a validator of the integrity rules. Section V is the conclusion of our research work in this paper, as well as some remaining problems for further study in the future.

II. TAXONOMY OF SPATIAL DATA INTEGRITY

A. Spatial Data Quality, Integrity Constraints and Rules

Seven components of spatial data quality were defined by the ICA Commission of Spatial Data Quality: lineage, positional accuracy, attributes accuracy, completeness, logical, semantic accuracy and temporal information. Nowadays, GIS are more and more involved in processes in which reasoning is based on spatial features [9]. Hidden Errors that are not apparent when using a GIS for visualization become more critical when used in this way [8].

Integrity constraint is the mechanism in database which is set up in response to a user defined integrity rule, relating to the way they wish the database to respond to a given event and keep in a consistent state. Improvement of data quality is one of the key objectives of establishing integrity constraints in spatial databases [4]. Entity integrity, referential integrity, domain integrity are the most commonly supported and used integrity constraints in DBMSs. User-defined spatial integrity constraints have their origin at the system design stage. There

(C) British Crown Copyright 2005
are plenty of research work and thorough discussion about computational geometry, spatial relationships and spatial integrity constraints in [1], [2], [4], and [9].

B. A Taxonomy of Spatial Data Integrity

As shown in Fig.1, three kinds of integrity constraints are taken into accounted in this paper: the topological characteristic constraint, the spatial relations constraint and the geometric network integrity constraints.

![Taxonomy of spatial integrity constraints](image)

The topological characteristic constraint refers to the topological properties of a single geometry that should be satisfied within spatial databases, such as "A line should not be self-intersected", "A polygon should be closed" and so on.

The spatial relationship constraint deals with the constraints in terms of spatial relationships among two or more geometries that are also connected semantically, e.g. "Bus routes must be on top of roads". Spatial relationship integrity constraints are divided into three sub-categories: topological relation, metric relation and spatial order relation according to the classification of spatial relationship in [2].

The geometric network constraint could be applied to describe the constraints of connectivity and flows among the edges and junctions in a geometric network graph.

Within a geographic context, topological relations and other spatial relationships are important in the definition of spatial integrity rules. Topological relations between objects can be described using the 9-intersection model [1]. The validity of a given topological relation is based on the semantics (the meaning) of both entities. The term topological relational integrity constraint adopted here could be regarded as the counter-part of topo-semantic integrity constraint presented in [9]. It is also the main spatial integrity constraint to be discussed within this paper.

III. FORMAL DEFINITION LANGUAGE FOR SPATIAL DATA INTEGRITY RULES

Each atomic spatial data integrity rule concerns less than two tables, while every table could participate in several individual integrity rules.

DEFINITION 1. An atomic spatial integrity rule (ASIR) is defined as a tuple $ASIR = \{ R_1, R_2, F, J, P, M \}$:

- $R$ stands for a table or a logical-view of table in spatial database. $R_1$ and $R_2$ are the two tables or views concerned in an integrity rule. No more than one of them could be NULL.
- $F$ is a spatial or non-spatial predicate for the each relation to express some semantic restrictions. Both of $F_1$ and $F_2$ could be NULL.
- $J$ is a joint condition that could be used to establish a semantic connection between these two relations. $J$ could be NULL to cover the whole set of tuples of both relations filtered by $F$.
- $P$ is a spatial or non-spatial predicate presenting the constraint within each tuple-couple consisted of those from $R_1$ and $R_2$, both of which are filtered by $F_1$ or $F_2$ respectively. $P$ could be a predicate with one or two operands. With $J$ and $F$, $P$ is capable of expressing various kinds of spatial or non-spatial integrity constraints concerning two relations semantically connected by alphanumeric or spatial joint conditions.
- $M$ is a modifier of the predicate $P$ about how many times $P$ ought to be satisfied within this rule. $M$ could be in the form of "AT LEAST n TIMES", "AT MOST n TIMES", "EXACTLY n TIMES", "FORBIDDEN" or "ALLOW ANY TIMES".
- The value of a $SSIR$ is determined by the satisfaction of predicate $P$ under the restriction of $M$, returning "TRUE" or "FALSE".

An example of an atomic spatial integrity rule looks like \{city, null, state, null, "city.state = state.name", "within", 1\}. It means that any city must be located within the boundary of the only one state that it belongs to.

Several individual integrity rules can be combined together as a compound one to describe a more complex spatial integrity constraint in real-world applications.

DEFINITION 2. A compound spatial integrity rule (CSIR) is defined as a tuple $CSIR = \{ C_1, LP, C_2 \}$:

- $C$ could be an atomic integrity rule or another compound integrity rule. "NOT" could be put before integrity rule $C$ as an inverse modifier to $C$, which means $C$ should not be satisfied. Then $NOT C$ is also a legal element in this definition tuple.
- $LP$ (Logical Operator) which could be "AND", "OR" or "NOR", is the connector between the two atomic or compound spatial integrity rules. It has been implied that only "AND" and "NOT" are necessary while the other logical operators could be made from the formers.
- The value of a compound integrity rule $CSIR$ is determined by values of the two subordinate integrity rules $C_1$ and $C_2$, and the Logical Operator.

By such an iterative way, many complex spatial integrity constraints could be described using compound integrity rules.
Together with the atomic example cited ahead, a compound integrity rule could be build as follow: \{city, null, state, null, "city.state = state.name", "within", 1\} AND \{city, null, city, null, "city.name ! = city.name", "without_distance(1000m)", ALLOW ANY TIMES\}, which enhances another constraints that cities should stand 1000 meters out of one another.

IV. SCHEMA OF DATA TABLE FOR INTEGRITY RULES AND IMPLEMENTATION FRAMEWORK OF RULE VALIDATOR

A. Storage Schema for Integrity Rules in Spatial Database

Integrity rules are stored in a special data table. Each atomic integrity rule is corresponding to one record, while a compound integrity rule is consisted of several records logically connected by "AND". The schema of such data table is presented in Tab.1. The validator can then retrieve these spatial integrity rules by parsing certain tuples of this table and do the validating jobs.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table1</td>
<td>Text</td>
<td>any existing data table or view</td>
</tr>
<tr>
<td>Filter1</td>
<td>Text</td>
<td>/</td>
</tr>
<tr>
<td>Table2</td>
<td>Text</td>
<td>any existing data table or view</td>
</tr>
<tr>
<td>Filter2</td>
<td>Text</td>
<td>/</td>
</tr>
<tr>
<td>Joint</td>
<td>Text</td>
<td>/</td>
</tr>
<tr>
<td>Predicate</td>
<td>Text</td>
<td>/</td>
</tr>
<tr>
<td>Modifier</td>
<td>Integer</td>
<td>0, -1 or other positive integer</td>
</tr>
</tbody>
</table>

B. Implementation of Integrity Validator

1) Comparison of Implementation Architectures

In DBMS implementations, many approaches, such as assertions, rule system, triggers and so on, are available for building such a validator. Some ORDBMSs support customized data type and its integrity constraints [17]. A standalone toolkit has also been proved to be a flexible and efficient way [14]. As shown in Fig.2, these approaches can be sorted into three types according to their architectural differences: the layered architecture, the monolithic architecture and the extensible architecture [13].

Layered architecture enjoys a higher independency and portability upon the underlying DBMS in implementation. As a result, it is possible to make effort for only one implementation that will be suited for almost all the DBMS products by using the most commonly supported APIs. By the use of some special data structures, some complex spatial integrity rules could also be defined and enhanced in a layered architecture. However, system efficiency remains a big problem for layered style because of the high abstraction of data access interfaces. System constructors should even do those jobs used to be done by traditional DBMS, if he wants to add in some advanced features, such as support for transactions and concurrency control.

Monolithic architectural systems could benefit a lot from its host DBMS, such as connections management, transactions support and so on. Less works are needed to build a spatial integrity ensuring mechanism using a monolithic architecture. However, the expressiveness is to a great extent restricted to the host system’s extensibility. Only some simple integrity rules are acceptable to enhance in the monolithic style, while complex ones are nearly mission impossible.

Extensible architecture could be viewed as an eclectic approach between the layered architecture and the monolithic architecture. Extensible architecture gains a relatively high expressiveness comparing with monolithic architecture. It is also possible to take advantage of the abilities of host DBMS. However, all of this should be based on a good understanding of the ability and extensibility of its host DBMS.

ESRI is using the layered architecture to implement its spatial middle-ware ArcSDE and GIS desktop ArcGIS [14] [15]. Assisted by ArcGIS, system users are enabled to define domain-specific integrity rules, including domain integrity rules and topological integrity rules, which are then stored within ArcSDE and shared among all the users that connected. PostGIS [18] is a well-known open source spatial database established by taking advantage of the powerful extensibility of PostgreSQL [17]. PostGIS also supports some simple spatial integrity constraints in an extensible way.

2) A Prototype Implementation of Validator

In our implementation work, a prototype system of the monolithic style has been created in Oracle® 9i Spatial [16] using triggers and stored procedures.

While building the validator for spatial integrity rules, it is possible to set up an individual trigger for each integrity rule. In fact, any table would be concerned with several spatial integrity rules. More than one trigger might conflict with one another, resulting in an unpredictable outcome and inconsistent state. According to [12], it is acceptable to set up two stored procedures separately for each integrity rule:

- **CheckConstraint**: a stored function indicating whether the new update is compatible with the integrity rules.
- **PerformUpdate**: a stored procedure that executes the real updating actions that are defined by the integrity rule according to the return value of CheckConstraint.

These two functions could be automatically built basing on the corresponding prototype function and the definition of the
For each integrity rule between corresponding record in table following Fig.5. The logical procedure to implement such trigger is listed in the containing geometry data, which will be fired before updating.

- **Table1**: the name of table to be updated.
- **Table2**: the name of table which has pre-defined integrity rules with Table1.
- **CheckConstraintName**: the name of CheckConstraint function validating the integrity rule between Table1 and Table2.
- **PerformUpdateName**: the name of PerformUpdate procedure executing the real updating actions.

For each integrity rule between Table1 and Table2, there is a corresponding record in table **USER_CONSTRAINTS**.

Then a before-statement trigger is set up for each data table containing geometry data, which will be fired before updating. The logical procedure to implement such trigger is listed in the following Fig.5.

```
ALLOWED = TRUE
RECORDSET a
SELECT * into a FROM USER_CONSTRAINTS WHERE Table1 EQUALS the name of current table
FOR each record in a DO
   INVOKE the function CheckConstraintName
   IF this function returns FALSE THEN
      ALLOWED = FALSE
      BREAK out of the loop
   ELSE
      CONTINUE
   END IF
END LOOP
IF ALLOWED = TRUE THEN
   FOR each record IN a DO
      INVOKE the procedure PerformUpdateName
   END LOOP
ELSE
   REJECT the updating
```

Figure 3. Logical procedure to implement the trigger as a validator

V. CONCLUSION AND FEATURE WORK

In this paper, a formal definition language that is suitable for describing various kinds of spatial integrity constraints in spatial database was presented at first. Then a generic table schema is also provided for storing spatial integrity rules, which would be used to build the integrity validator. Three types of architectures for building a validator are thoroughly reviewed, resulting in a practical comparison of them. At the end of this paper, a prototype system was also prompted with more details with respect to implementation as a guide for the construction of real-world applications.

However, there are still some challenges related to spatial integrity constraints remained unresolved. The completeness of the formal definition language has not been fully examined. Some kinds of spatial integrity constraints, e.g. "there should not be any gap between adjacent areas", are awkward to define in this language. Subsequent actions after the validation have not been fully discussed yet. Although it might be difficult to abstract the common part of various post-validation tasks, some automatic mending works and other intelligent measures are still possible to pick up and defined formally. More future research works should be devoted to them.

REFERENCES