A UNIFIED VERSION-BASED SPATIO-TEMPORAL DATA MODEL

Y.D. Li[a, b, *], X.H. Tong[a], M.L. Liu[a]

[a]Department of Surveying and Geo-informatics, Tongji University, No.1239 Siping Road, Shanghai, 200092, China
[b]College of Marine Science & Technology, Shanghai Ocean University, No.334 Jungong Road, Shanghai, 200090, China

WgS–PS: WG II/1

KEY WORDS: Spatio-temporal Data Model, Temporal GIS, Version, Time-varying, Spatio-temporal Modelling, Database Design

ABSTRACT:

Temporal Geographic Information System (TGIS), which aims to collect, manage and analyze dynamic geographic phenomena, is a hot scientific research field. The all-important task of developing a TGIS is to build the spatio-temporal database. Since data model is the core of a spatio-temporal database, many GIS researchers and experts of database community have made a lot of efforts about the spatio-temporal data model and many literatures and works have been published. Although some substantial data models have been put out, there are few feasible and acceptable ones. This paper attempts to address this problem. The advantages and disadvantages of some of the typical and important existing data models were briefly analyzed firstly in this paper. A novel unified version-based spatio-temporal data model (UVSTDM) is then proposed. Based on the conception of version, the evolution of geographic phenomena is represented by a series of object-versions and attribute-versions which are associated with temporal information. UVSTDM organizes the temporal data more effectively and treats spatial data and thematic or non-spatial data in a uniform way. With this model, the object states can be retrieved conveniently and the evolution history of an object or its attributes can be traced easily.

1. INTRODUCTION

The real world is changeful. Although geographic information system aims at representing, managing and analyzing those dynamic phenomena, existing GISs can maintain one old state about the reality only or forget the past information if new information enters. Therefore, current traditional static GISs can't reserve the various history states and say nothing of analyzing, evaluating and forecasting the time-involving process about some dynamic spatio-temporal phenomena such as environment monitoring, transportation, social economy, land change, and so on.

Change is coupled with time, so it is necessary to introduce temporal information into the traditional GIS and to build the temporal geographic information system (TGIS for short) to address the questions stated above. As a key part of TGIS, spatio-temporal database relies on a good spatio-temporal model designed. In past three decades, many spatio-temporal models were described in some literatures, such as sequential snapshot model (Langran & Chirisman 1988), base state with amendment model (Langran 1992), space-time composites model (Langran & Chirisman 1988), spatio-temporal object model (Worboys 1992), and so on. Sequential snapshot model, which treats the dynamic phenomena as a successive snapshot, is one of simplest models and is used widely. It has many disadvantages and the obvious one is that it stores abundant redundant data (Langran 1992). Compared with Sequential snapshot model, base state with amendment model reduces the duplicated data largely. It stores an initiative state as a base map at the first time. When the change happens, only the difference, which is produced when compared with the previous state, will be saved. Because this model just saves the difference, it is very easy to get the difference between two consecutive states but will appear poor performance when getting the non-initial state. The Space-time composites model treats the real-world as a set of spatially homogenous and temporally uniform objects in a 2D space. Every space-time composite has its unique temporal course of changes in attribute and it conceptually describes the change of a spatial object through a period of time. It is a pity that this model links information to spatial object and it can't handle the temporality of a thematic or non-spatial object. The spatio-temporal object model represents the reality as a set of discrete objects consisting of spatio-temporal atoms by incorporating a temporal dimension orthogonal to the 2D space. Spatio-temporal atoms are the largest homogeneous units in which certain properties hold in both space and time. Apparently, the spatio-temporal object model is object-based and can make use of the object-oriented approach. Just as the space-time composites model, however, the spatio-temporal object model is heavily dependent of spatiality and can't represent the change of those thematic or non-spatial objects. Thus it can be seen that there exist few feasible solutions that are proposed to manage spatio-temporal data. This paper aims to address it. With the suggestions of these models, we propose a novel model to resolve the efficient management of spatio-temporal data, which is based on version, avoiding the shortages that appeared in those existing models. In addition, we confine our focus here to the management of the evolution of spatio-temporal data in the field of GIS, regardless of the management of the alternatives in the field of engineering design or others.

The remainder of this paper is structured as follows. In section 2, the abstract uniform representation of the temporal object is

[*] Corresponding author: lyd911@163.com; phone +86-(0)21-65988851.

given first, and then the conceptual version-based spatio-temporal data model is presented after defining the concepts of object-version and attribute-verison. Section 3 details the object-relational approach on a commercial DBMS for the conceptual model. An illustrative example is given in section 4. Finally, section 5 draws the conclusions and the perspectives of this work.

2. CONCEPTUAL FRAMEWORK OF VERSION-BASE SPATIO-TEMPORAL DATA MODEL

2.1 The uniform presentation for spatio-temporal object and thematic-temporal object

The entities or geographical phenomena of the real world are represented by objects in the information world. They can be physical things, e.g., buildings, poles, etc., but also conceptual ones, e.g., the world economic centre, government, etc. An object can be described by a set of attributes, whose value can be observed or measured by means of surveying. Those attributes can be spatial, such as geometric shape, size, position, etc., or non-spatial, that is, thematic, such as color, height, material, and so on. Particularly, to a geographic feature, its properties often include both spatial attributes and thematic attributes. From the viewpoint of an information system, every object has another special attribute, i.e., object identifier (Bonfatti & Pazzi, 1995). The object identifier uniquely represents one object and is different from that of any other object. Therefore, one object class O can be represented by

O = (oid, ThematicAttributes, SpatialAttributes, ls).

Where

- oid is the object identifier of object instance;
- ThematicAttributes is the collections of thematic attributes that one object instance owns;
- SpatialAttributes is the collections of spatial attributes that one object owns. Spatial attribute is abstracted by a spatial object type.
- ls defines the lifespan of the object instance.

In order to make a generic description of an object, in this paper, the attributes of an object are processed in a generic manner that we don't distinguish they are spatial or non-spatial. Thus, one general object can be represented by O = (oid, attributes, ls), the meaning of oid and ls is the same as those stated just before, and attributes is the collections of an object's attributes as {ai}(i = 0, 1, …, n), where ai is the attribute and n is the amount of the object's attributes. If some object o owns any geometric object attribute ai, the object is a spatial one (it should be noted that, the types of spatial geometric object attributes conform to the spatial types adopted by Open GIS simple feature specification). If none of the attributes is a geometric object attribute, this object is a traditional thematic or non-spatial one. Thinking on the time-varying characteristic of an object, an object involves a series of various states during its lifespan. A state of an object at time t is the corresponding values of all attributes at that time. So a temporal object can described by a series of states.

2.2 Concept of object-version and attribute-version

The concept and research about version started in the fields like CAD (Computer Aided Design), CASE (Computer Aided Software Engineering) and SCM (Software Configuration Management). Nowadays, the version concept is widely used in other fields, e.g. cocurrent engineering, schema evolution, document management, etc. In the database community, there exists respectable research focused on data schema evolution with combination of time and version. Rodriguez et. al. (1999) put forward a model called TVOO (Temporal Versioned Object Oriented) by introducing the temporal and versioning schema features to an object-oriented environment. Afterward, the TVOO model was extended to manage the evolution of object instance and object data schema (Rodriguez et. al. 1999). The TVM (Temporal Versions Model) model brought by Moro et. al (2002) can store the versions of objects and the history for those values of dynamic attributes in each version, but this model focuses mainly on the management of the design alternatives and is not quite suitable for the temporal evolution management of general objects. In the field of GIS, the version concept is also involved, for example, Medeiros and Jomier (1994) considered that version was the means to store different states of entities and allowed controlling the alternative or the temporal data evolution, and suggested that using database versioning mechanism to manage the evolution of geo-referenced data in GIS. The Geodatabase model of ESRI's ArcGIS uses versioning to support the multi-user editing and simple historical data management. As far as we know, there are quite rare examples that using versioning technique to manage the evolution of the spatio-temporal data successfully.

Generally speaking, in GIS study, a version represents the unchanged state of one object in a certain period of time (Medeiros & Jomier 1994, Gong 1997, Moro et.al. 2002). Obviously, the version concept here indicates the version of object or object-version for short. By the definition of version, we can see that it actually defines a static unchanged state in a certain time period, and we call this kind of version static object-version. Static version is only suitable for describing discrete changes, but not suitable for describing continuous change. In fact, the temporal changes include not only discrete changes (e.g., land Change), but also continuous changes (e.g., moving object), some of which have some regularity and can be described by some time-related functions (e.g., the position of an object that moves at a uniform speed along a straight line in a certain period of time). So, for those regular continuous changes, it is unnecessary to sample them discretely, and the states within a certain time period can be described just by using some regular function expressions. Therefore, in order to represent continuous change, we introduce a new concept, i.e., dynamic object-version. Dynamic object-version represents the regularly variable state of an object in a certain period of time. From a lower-level point of view, the object state represented by object-version is an integrated representation of the states of various attributes of the object. By the preceding definition of object-version, we can define two other new concepts, that is, static attribute-version and attribute-version. Static attribute-version defines the unchangeable state of an object attribute in a certain period of time, and the attribute-version defines the regularly changeable state of an object attribute in a certain period of time. Thus, an object-version is composed of the attribute-versions (the attribute-version can be either static or dynamic) which correspond to all attributes of the corresponding object. That's to say, attribute-versions are component parts of an object-version. Therefore, the four conceptions, i.e., the static object-version, dynamic object-version, static attribute-version and dynamic attribute-version, can be used to describe the discrete and continuous changes happened to the geographical features.

2.3 Framework of unified version-based spatio-temporal data model (UVSTDM)

According to the object-oriented thinking, the unified version-based spatio-temporal data model can be represented by the hierarchy framework using a UML diagram as shown in Figure 1. When modelling the change of an object in spatiality and temporality, a temporal object (TemoralObject) (either a Spatio-temoral object (SpatialTemporalObject) or a thematic-temoral object (ThematicTemoralObject)), which owns the temporal characteristic, will involve some different states since the values of its' attributes change in its lifespan. And each such state relates with an object-version (ObjectVersion). Those temporal object-versions may be either static or dynamic. An object-version is the comprehensive representation of the values of the attributes of one object. And each attribute of one object has a number of different versions of the attribute (AttributeVersion). So an object-version can be seen as a composite which consists of a set of attribute-versions. An attribute-version can be shared by multiple object-versions. The class ObjectVersion and AttributeVersion derive from the class Version, which provides the basic information about a version. The class ObjectVersionControl and AttributeVersionControl inherit VersionControl class and provide some version controlling information, such as the first version, the final version, the current version, version quantity, etc., to one object or one attribute of an object respectively.

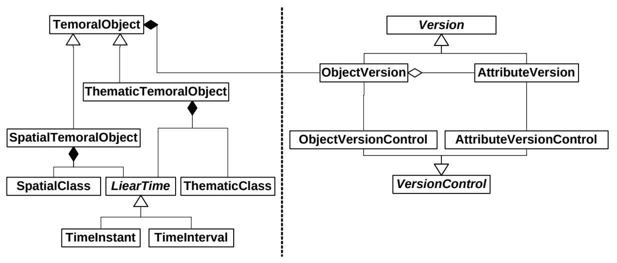

Figure 1. Framework of UVSTDM

3. OBJECT-RELATIONAL APPROACH FOR VERSION-BASED MODEL

3.1 Design of core classes

The version-based spatio-temporal data model mainly involves the following classes: Version, ObjectVersion, AttributeVersion, VersionControl, ObjectVersionControl and AttributeVersion-Control. Their main attributes and methods are shown in Figure 2.

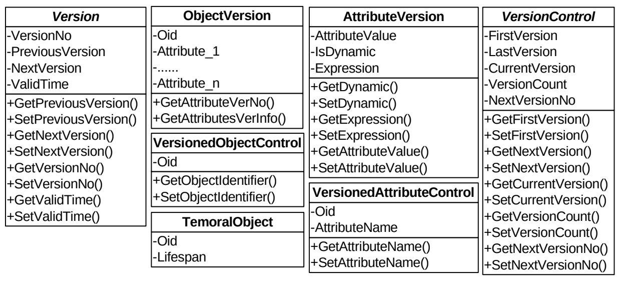

Figure 2. Core classes design of UVSTDM

Class Version is an abstract class, and it has attributes such as version number, previous version, next version, etc. Besides, there is another attribute, i.e., ValidTime, the datatype of which is TimeInterval, which is composed of two time-points (TimeInstant) (one for starting time and another for end time) and represents the period that the fact of version is true in reality.

Class ObjectVersion deriving from class Version and application-dependent, has the attributes and methods of class version. A specific object instance has a number of ObjectVersion instances (object-versions). The corresponding object-version of each class instance has an object identifier, Oid, and the corresponding attribute-versions of the temporal attributes of each application-object class. In addition, class ObjectVersion have functions of getting the version number of the version of the specified attribute and acquiring all the attributes of version information.

Class AttributeVersion deriving from class Version and application-dependent. One temporal attribute of one object can owned multiple attribute-version instances during the evolution of the object. In addition to those attributes and methods derived from class version, Class AttributeVersion also owns the value attribute, which specify the value of attribute, and correlative attributes specifying that whether the version is a dynamic version or not and methods to get or set attribute value.

Class VersionControl is an abstract class, and it owns attributes such as first version, last version, current version, version number and next version number, etc. Additionally, it owns the methods to set and get those attributes.

Class ObjectVersionControl, which is a subclass of class VersionControl, is responsible for managing and controlling to the object-versions. In addition to those attributes and methods derived from version class, it owns the object identifier, Oid, as well as the methods to set and get the value of the Oid attribute.

Class AttributeVersionControl, which derives from VersionControl class, is responsible for managing and controlling to the attribute-versions. In addition to those attributes and methods derived from version class, it owns the attribute-name attribute (AttributeName) and the methods to access it.

3.2 Object-relational approach for UVSTDM

Here, an object-relational approach is chosen to implement the version-based spatio-temporal model because the object-relational databases own some object-oriented features while retaining the good performance of relational databases. Here the database Oracle 11g DBMS is recommended, because it owns a spatial option, i.e. Spatial, which provide the spatial datatype, i.e. SDO_GEOMETRY, and corresponding spatial functions for the storage, management and manipulation of spatial data.

To realize the version-based model on an object-relational database, the classes should be transformed to the equivalent for interaction with the underlying DBMS. There are two ways to map a class into a target DBMS. The first one is called relation-mapping, which maps a class to a relation directly. The second one is called object-relation-mapping, which maps a class to an object type first, and then create a table according to the object type. Both of the above two mapping methods include two steps, i.e. mapping the attributes of the classes and mapping the methods of the classes. Due to space limitations, the routines for the relation-mapping method are discussed in this work. Mapping methods can be realized via store functions and store procedures (Sheldon 2004). The steps for attributes mapping are listed briefly as follows:

- All the abstract classes won't be transformed to relations, and their attributes will transfer to their subclasses.
- Transforming each instantiable class to a relation, the relation owns all the class' attributes including those derived from its superclass.

The result of the attributes mapping for some of classes shown in Figure 2 is detailed as follows, and most of the fields' names in the corresponding relations are self-explanatory:

1. Class ObjectVersion is transformed to an relation Ro = (oid, VersionNo, AV1, AV2, …, AVn, PreviousVersion, NextVersion, ValidTime), where AVi (i∈{0,…,n}, n is the amount of attributes owned by a specific application object class) refers to the attribute-version number.
2. Class AttributeVersion is transformed to an relation Ra = (oid, VersionNo, AttributeValue, IsDynamic, Expression, PreviousVersion, NextVersion, ValidTime).
3. Class ObjectVersionControl is transformed to an object-version relation Rco =(oid, FirstVersion, LastVersion, CurrentVersion, VersionCount, NextVersionNo)
4. Class AttributeVersionControl is mapped to an object-version relation Rca =(oid, AttributeName, FirstVersion, LastVersion, CurrentVersion, VersionCount, NextVersionNo)

4. ILLUSTRATIVE EXAMPLE

4.1 A example about cadastral change

Figure 3 shows several cadastral changes of a given area, which happened after the initial registration date, i.e., 1980-1-1. Those changes include the non-spatial changes (e.g., owner, usage) and spatial changes. The letters A, B, and C represent the parcel

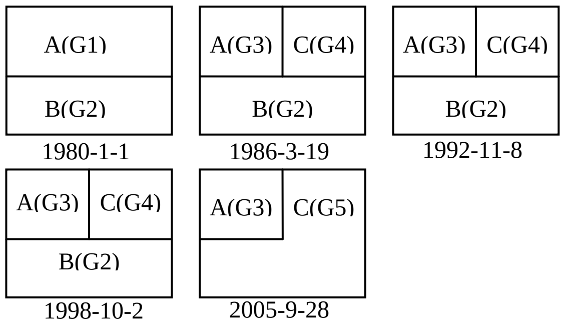

Figure 3 Cadastral change

object identifier, and the letter G together with subsequent number in brackets represents the spatial attribute values of the parcels. The changes are detailed as follows:

- 1986-3-19: Parcel C was divided from Parcel A and the usage of parcel A was changed from "architecture" to "industry";
- 1992-11-8: The owner of Parcel A was changed to "Li jie";
- 1998-10-2: The owner of Parcel A was changed to "Xu haihua";
- 2005-9-28: Parcel B was merged into Parcel C.

Table 1 lists the states of the parcels in different time period.

Oid	Shape	Owner	Usage	VTs	VTe
A	G1	Zhang rutian	architecture	80-1-1	86-3-18
A	G3	Zhang rutian	industry	86-3-19	92-11-7
A	G3	Li jie	industry	92-11-8	98-10-1
A	G3	Xu haihua	industry	98-10-2	05-9-27
A	G3	Xu haihua	industry	05-9-28	-
B	G2	Lu feng	agriculture	80-1-1	86-3-18
B	G2	Lu feng	agriculture	86-3-19	92-11-7
B	G2	Lu feng	agriculture	92-11-8	98-10-1
B	G2	Lu feng	agriculture	98-10-2	05-9-27
C	G4	Sun xiangyi	industry	86-3-19	92-11-7
C	G4	Sun xiangyi	industry	92-11-8	98-10-1
C	G4	Sun xiangyi	industry	98-10-2	05-9-27
C	G5	Sun xiangyi	industry	05-9-28	-

Table 1. States of parcels

4.2 Implementation based on version

Using UVSTDM to implement the management of the time-varying cadastral information in one relational database, we need to build the following tables: parcel object main table (table 2), attribute-version table for all time-varying attributes of parcel objects (table 3~table 5), object-version table for parcels (table 6), object-version control table (table 7) and attribute-version control table (table 8). It should be noted that the dynamic attribute-version information is omitted due to the discrete changes of parcels. In those tables, VTs is short for the start of valid time, VTe for the end of valid time, Ver for VersionNo, NV for NextVersion, PV for PreviousVersion, FirstV for FirstVersion, LastV for LastVersion, CurV for CurrentVersion, NextVNo for NextVerionNo, and VCnt for versionCount.

Oid	**VTs**	**VTe**
A	1980-1-1	-
B	1980-1-1	2005-9-27
C	1986-3-19	-

Table 2. Object table of parcels (CPARCEL)

OID	Ver	Value	**VTs**	**VTe**	NV	PV
A	S0	G1	80-1-1	86-3-18	S1	-
A	S1	G3	86-3-19	-	-	S0
B	S0	G2	80-1-1	05-9-27	-	-
C	S0	G4	86-3-19	05-9-27	S1	-
C	S1	G5	05-9-28	-	-	S0

Table 3. Attribute-version table of shape (AV_USAGE)

OID	Ver	Value	**VTs**	**VTe**	NV	PV
A	O0	Zhang rutian	80-1-1	92-11-7	O1	-
A	O1	Li jie	92-11-8	98-10-1	O2	O0
A	O2	Xu haihua	98-10-2	-	-	O1
B	O0	Lu feng	80-1-1	05-9-27	-	-
C	O0	Sun xiangyi	86-3-19	-	-	-

Table 4. Attribute-version table of owner (AV_OWER)

OID	Ver	Value	**VTs**	**VTe**	NV	PV
A	U0	architecture	80-1-1	86-3-18	U1	-
A	U1	industry	86-3-19	-	-	U0
B	U0	agriculture	80-1-1	05-9-27	-	-
C	U0	agriculture	86-3-19	-	-	-

Table 5. Attribute-version table of usage (AV_USAGE)

OID	Ver	shape	owner	usage	**VTs**	**VTe**	NV	PV
A	A0	S0	O0	U0	80-1-1	86-3-18	A1	-
A	A1	S1	O0	U1	86-3-19	92-11-7	A2	A0
A	A2	S1	O1	U1	92-11-8	98-10-1	A3	A1
A	A3	S1	O2	U1	98-10-2	-	-	A2
B	B0	S0	O0	U0	80-1-1	05-9-27	-	-
C	C0	S0	O0	U0	86-3-19	05-9-27	C1	-
C	C1	S1	O0	U0	05-9-28	-	-	C0

Table 6. Object-version table of parcels (OV_CParcel)

OID	FirstV	LastV	CurV	NextVNo	VCnt
A	A0	A3	A3	A4	4
B	B0	B0	-	B1	1
C	C0	C1	C1	C2	2

Table 7. Object-version control table (OV_CONTROL)

Oid	Attribute	FirstV	LastV	CurV	NextVNo	VCnt
A	shape	S0	S1	S1	S2	2
A	owner	O0	O2	O2	O3	3
A	usage	U0	U1	U1	U2	2
B	shape	S0	S0	-	S1	1
B	owner	O0	O0	-	O1	1
B	usage	U0	U0	-	U1	1
C	shape	S0	S1	S1	S2	2
C	owner	O0	O0	O0	O1	1
C	usage	U0	U0	U0	U1	1

Table 8. Attribute-version control table (AV_CONTROL)

4.3 Several typical spatio-temporal queries

1 Querying state

⑴ Get the state of some specified object-version

Q1: Get the state of A2 object-version of parcel A.

SELECT t.oid, a1.value, a2.value, a3.value FROM OV_CParcel t, AV_SHAPE a1, AV_OWER a2, AV_USAGE a3 WHERE t.oid = A AND t.version = A2 AND a1.oid = t.oid AND t.shape = a1.ver AND a2.oid = t.oid AND t.owner = a2.ver AND a3.oid = t.oid AND t.usage = a3.ver;

Result: A G3 “Li jie” “Industry”

⑵ Get the current / first / last state of a single object or some attribute of one object

Q2: Get the current state of parcel A.

SELECT t.oid, a1.value, a2.value, a3.value FROM OV_CParcel t, AV_SHAPE a1, AV_OWER a2, AV_USAGE a3 WHERE t.ver in (SELECT ovc.curV FROM OV_CONTROL ovc WHERE ovc.oid = A) AND t.oid = A AND a1.oid = t.oid AND a2.oid = t.oid AND a3.oid = t.oid AND t.shape = a1. ver AND t.owner = a2.ver AND t.usage = a3.ver;

Result: A G3 “Xu haihua” “Industry”

Q3: Get the initial owner of parcel A.

SELECT a.value from AV_OWER a WHERE a.oid = A AND a.ver in (select avc.CurV from AV_CONTROL avc WHERE avc.oid = A AND avc.Attribute =’owner’)

Result: “Zhang rutian”

⑶ Get current states of all objects in database

In our model, it is easy to query the current states of some kind of present objects and to query the all current states of all kinds of present objects.

Q4: Query the current states of present parcels.

SELECT t.oid AS NO, a1.value AS SHAPE, a1.value AS OWNER, a3.value AS USAGE FROM OV_CParcel t, OV_CONTROL ovc, AV_SHAPE a1, AV_OWER a2, AV_USAGE a3 WHERE t.oid = ovc.oid AND t.version = ovc.curv AND ovc.curv IS NOT NULL AND a1.oid = t.oid AND a1.ver = t.shape AND a2.oid = t.oid AND a2.ver = t.owner AND a3.oid = t.oid AND a3.ver = t.usage.

Result:
A G3 “Xu haihua” “Industry”
C G5 “Sun xiangyi” “Industry”

⑷ Get the state at a specified time instant

Q5: Get the state of parcel A at 1993-9-1.

SELECT t.oid AS NO, a1.value AS EXTENT, a1.value AS OWNER, a3.value AS USAGE FROM OV_CParcel t, AV_SHAPE a1, AV_OWER a2, AV_USAGE a3 WHERE t.oid =A AND VTs >= ‘1993-9-1’ AND VTe <= ‘1993-9-1’ AND a1.oid = t.oid AND a1.ver = t.shape AND a2.oid = t.oid AND a2.ver = t.owner AND a3.oid = t.oid AND a3.ver = t.usage.

Result: A G3 “Li jie” “Industry”

Q6: Get all states of whole parcels at 1993-9-1.

SELECT t.oid, a1.value AS shape, a2.value AS owner, a3.value AS usage FROM OV_CParcel t, AV_SHAPE a1, AV_OWER a2, AV_USAGE a3 WHERE t.VTs >= ‘1993-9-1’ AND t.VTe <= ‘1993-9-1’ AND a1.oid = t.oid AND a1.ver = t.shape AND a2.oid = t.oid AND a2.ver = t.owner AND a3.oid = t.oid AND a3.ver = t.usage;

Result:
A G3 “Li jie” “Industry”
B G2 “Lu feng” “Agriculture”
C G4 “Sun xiangyi” “Industry”

2 Getting the whole evolution process

This model is based on the conceptions of object-version and attribute-version. It can reproduce or trace the evolution process not only of an object but also of some attribute of an object.

Q7: Get the whole change history of parcel A.

SELECT t.oid, a1.value AS shape, a2.value AS owner, a3.value AS usage, t.vts, t.vte FROM OV_CParcel t, AV_SHAPE a1, AV_OWER a2, AV_USAGE a3 WHERE t.oid = A AND a1.oid = t.oid AND a1.ver = t.shape AND a2.oid = t.oid AND a2.ver = t.owner AND a3.oid = t.oid AND a3.ver = t.usage ORDER BY t.vts;

Q8: Get the whole change history of the owner of parcel A.

SELECT oid, value, vts, vte FROM AV_OWER WHERE oid = A ORDER BY vts;

5. CONCLUSIONS

Since traditional static GISs can't no longer cater for the increasing application needs because of their deficiencies in managing dynamic geographic phenomena. A data model named UVSTDM has been proposed to implement the management of the evolution of spatio-temporal data in the field of GIS. This model organizes time-varying spatial data and non-spatial data in a uniform way by extending the concept of version to object-version and attribute-version. With versioning mechanism, object states are organized as object-versions and attribute-versions. In this model, the object states can be retrieved conveniently and the evolution history of an object or its' attributes can be traced easily. Approximatively, this model can be regarded as a combination of both object-state timestamping and attribute-value timestamping while incorporating the advantages of each and eliminating their drawbacks. At the same time, this model can process both spatial objects and non-spatial objects with temporality making up the shortage of the most models which were heavily dependent on spatiality.

Future work lies, for instance, in designing of operations about data manipulation, e.g. insertion, deletion, etc. and exploring indexing technique for spatio-temporal data. To manage the alternatives in the field of engineering design or of urban planning is another research direction.

REFERENCES

Bonfatti, F. and Pazzi, L., 1995. Ontological foundations for state and identity within the object-oriented paradigm. *Human-Computer Studies*, 43(5-6), pp. 891-906.

Gong, J.Y., 1997. An object-oriented spatio-temporal data model in GIS. *Acta Geodaetica et Cartographica Sinica*, 26(4), pp. 289-298.

Langran, G. and Chirisman, N.R., 1988. A Framework for Temporal Geographic Information Systems. *Cartographica*, 25(3), pp. 1-14.

Langran, G., 1992. *Time in Geographic Information Systems.*, Taylor & Francis, London.

Medeiros, C.B. and Jomier, G., 1994. Using versions in GIS. In: *Proc. of Database and Expert Systems Applications Conference (DEXA'94), Lecture Notes in Computer Science 856*, Springer-Verlag, pp. 475-484.

Moro, M. M., Nina Edelweiss, dos Santos, C. S., 2002. Temporal Versions Model. http://citeseer.ist.psu.edu/608209.html (accessed 18 Mar. 2007).

Rodriguez, L., Ogata, H., Yano, Y., 1999. TVOO: A Temporal Versioned Object-Oriented data model. *Information Sciences*, 114(1-4), pp. 281-300.

Rodriguez, L., Ogata, H., Yano, Y., 2001. A temporal versioned object-oriented database schema. *Computers and Mathematics with Applications*, 41(1-2), pp. 177-192.

Sheldon, R., 2004. *SQL: A Beginner's Guide*. Tsinghua University, Beijing, pp. 250-273.

Worboys, M. F., 1992. A model for spatio-temporal information. In: *Proceedings of 5th International Symposium on Spatial Data Handling*, Charleston, South Carolina, USA, pp. 602-611.

ACKNOWLEDGEMENTS

The work described in this paper was substantially supported by the National Natural Science Foundation of China (Project No. 40771174 and No. 40301043), the Program for New Century Excellent Talents in Universities (Project No. NCET-06-0381), the Foundation of Shanghai ShuGuang Scholarship Program (Project No. 07SG24), and the grants from the Doctoral Program of Higher Education of China (Project No. 20070247046).