DATA MODEL AND REAL TIME IN SPATIAL APPLICATIONS: WHAT DATA AND WHEN?

Michel Mainguenaud
Institut National des Sciences Appliquées (Rouen) – LITIS Lab
Avenue de l'Université
F-76801 Saint Etienne du Rouvray, France
E-mail: michel.mainguenaud@insa-rouen.fr

KEYWORDS
Database model, Real time applications, Mobility, Data Definition Language

ABSTRACT

A real time application provides some time constraints in the delivering of database query results. We consider here the database model level to handle such constraints. We rely on the time sensitive object concept and we increase these notions with database properties such as extended functional dependencies, characterized aggregate functions and the definition of an order on a lattice. Depending on the available time to respond to a query, relevant data are provided in accordance to the perception of the data model defined for a given application. The extended functional dependencies involve a 1 to 1 correspondence between several entities of the data model. The order on the lattice defines the priorities. Characterized aggregate functions introduce the N to 1 correspondence and therefore require a more precise definition rather than a single trigger definition. The definition takes into account the relationships of the attributes involved in the aggregate function.

INTRODUCTION

The development of technologies, such as communication and positioning, increases real-time data traffic (Kennedy, 2002, Bossler, 2002). Intelligent Transportation Systems, Location Based Services require more and more Points of Interest within a navigation application. Therefore a database is required to store alphanumeric and spatial data. We define a geographic object as a set of couples (semantic property, value). An important point is to define the data to be provided as the response of a request. Real-time applications are based on the fact that whatever value an object has at any point in time, within a few milliseconds or seconds that value will cease to be valid within a brief time. Models for real-time systems traditionally focus on system processing and its timing characteristics. Timing constraints for real-time systems are typically expressed with respect to processing. In this article, we prefer considering geographic objects in the sense of time-sensitive objects as defined in (Callison, 1995). This model focuses primarily on the time intervals over which object values may ideally be used, rather than the timing of process execution to describe the evolution of a system. The validity intervals are associated with quality indices upon data. To be able to choose the relevant data within a time constraint, a conventional data model should be improved. We introduce the strong dependency of the time sensitive objects constraints, the extension of a conventional database (DB) notion, functional dependency, as an extended functional dependency and an order on a lattice to define a real time application DB schema. Using these extra-meta-data with the data model, a query resolution mechanism is able to define a priority in the delivering of the schema to a mobile application.
The first part presents the problems in the real time environment. The second part presents the add-on in the data model. The third part presents the propositions to take advantage of the new add-on in the data model. The last part presents the conclusion and future works.

PROBLEMS

At the opposite sides in the classification of time validity are single-interval transient and immutable objects. Single-interval transient objects are created, remain in the system for a short time and disappear without changing. At the other extreme, are immutable objects, which persist for the life of the system execution without changing. Most objects in real-time applications fall between these two opposite sides. They persist for some finite time and may change during this lifetime. New values with associated validity intervals are defined (i.e., multi-interval object). We focus on the associated semantic alphanumeric data model to handle a real time application with a DB.
Managing a real time application with a DB is an old requirement (Taina and Son, 1997). Several formats or data models can be used for navigation services such as for example ISO-GDF, SDAL (Shared Data Access Library – NavTech). We are not concerned in this article by the spatial representation of the route nor the data model associated with the representation of such as route (Liu et al. 2005). Several DB transaction models are provided to manage the serialization in a multi-user application to support or not the predictions (Bodlaender and Stok, 1998). Others are designed to support routing in heterogeneous networks (Sun et al. 2007). We are not concerned in this article by the system transaction model or the data propagation within the communication network. Nevertheless we rely on the performances of such systems.
To handle time in a spatial context, several propositions are available in the literature (Guting et al. 2006, Parent et al. 1999, Gregersen, 2006). These propositions are based on an extension of a conventional definition for a DB schema, i.e., a pair (attribute name, domain of validity). The data model is given as a collection of data types and operations, which can

be plugged as attribute types into any DBMS (Data Base Management System) data model. The idea is to represent the temporal development of spatial entities in certain data types such as moving point, the balloon model, spatial abstract data types.
These models consider that they do not have any time constraints during the query resolution mechanism (obviously every one would like to have a short response time of the system that implements the model, but every relevant data will be provided in the final results).
Our goal is to promote data definition properties to introduce an order of evaluation in a DB query resolution mechanism. In real time application, the DBMS may not have time to compute the complete answer of a query. Nevertheless, at least part of the answer could be provided as a result. The problem is to define which part should be provided first. Starting with a conventional definition of a DB schema based on a pair (attribute name, domain of validity), we must increase this definition with several notions since choices should be performed by the DBMS. The architecture is close to Dynamic Data Driven Real Time Management. We start with an Abstract Model as defined in (Forlizzi et al. 2000). The idea is to represent the temporal development of spatial entities in certain data types such as moving point or moving region. Suitable operations are provided on these types to support querying. Such data types can be embedded as attribute types into object-relational or other data models. It allows one to focus on the essential concepts and not get bogged down by representation details.
In the following, we consider a DB schema defined as a set of pairs (attribute name, domain of validity). The domain may be as complex as we like, e.g., a spatial representation with or without time management labels. The only requirement is that an attribute with a time management label must be documented, i.e., we do not accept a null value for the time management label when an attribute is defined as time-constrained.

MODELING TOOLS

To achieve our goals, we introduce in the DB schema several concepts to help the query resolution engine to define the relevant data to evaluate first: extended functional dependency, strong dependency, the characterized aggregate functions and the lattice.

Extended Functional Dependency

A functional dependency, as defined in conventional relational DBs (Ullman, 1995), indicates an assertion on the real world based on a 1 to 1 relationship.
Definition: Given a relation R of a DB schema, an attribute (or a minimal set of attributes) X in R is said to functionally determine another attribute (or set of attributes) Y, also in R, (written $X \rightarrow Y$) iff each X value is associated with precisely one Y value.
In this case, cycles are allowed. This definition is straightforward in the sense that X and Y must belong to the same relation (e.g., in the same class in UML for example if we want to translate this notion to the conceptual level to simplify).
We extend this definition by an Extended Functional Dependency in the sense that X is in the set of attributes of an entity E and Y may be an attribute (or a set of attribute) in another entity E' of the DB schema. This represents the extension of the 1 to 1 relationship in a conventional Entity-Relationship diagram or the 1-1 Association in a class diagram in UML.
The granule "sub-set" as defined in (Mainguenaud, 1994), applied with the inclusion operator provide the relevant candidates to an extended functional dependency.
Definition: An attribute is defined with a granule "sub-set" iff its semantic is valid for a sub set of its spatial representation.
As an example, the value of the mayor attribute (i.e., the name of the mayor of a town) is a valid data for a sub-set of the spatial representation of a town (i.e., a mayor is also responsible for a sub part of the town). The value of the population is not valid for a sub-set of the spatial representation (i.e., no guaranty is provided that all the inhabitants of the town live in this specific sub-set of the spatial representation of the town).
The application of the inclusion operator means that a park is spatially included into a town. Therefore, an attribute classified as "sub-set" of a town is still valid for the park since the park represents a sub-set of the spatial representation of a town (since it is included into the town).
The graphical representation of such a pair of attributes is defined by a single circle on the attributes and an oriented edge as the edge in the definition. Figure 1 represents such an example (i.e., <idNumber> $\rightarrow$ <Mayor>).

Strong dependency

The strong dependency, as defined in (Callison, 1995), indicates that the value of the dependent object depends critically on the value of the influencing object.
Definition: The value of the strong dependent object is to be updated as quickly as possible in response to any asynchronous change to the value of the influencing object.
To assure termination of the propagation process, cycles are prohibited in the strong-dependency relationship. We do not consider here, the update mechanism. The important point is that an update may occur (e.g., this update may be sporadic, periodic or any other way).
As an example, the speed of a car should be changed as soon as the car arrives in a new speed-limited area. In the DB model the attribute "carSpeed" should be updated as soon as "speedLimit" is changed (carCurrentSpeed --> currentSpeedLimit). carCurrentSpeed is the strong dependent object and currentSpeedLimit is the influencing object.
The graphical representation of the strong dependency between such a pair of attributes (i.e., the strong one with a double circle and the influencing one with a single circle) is defined by an edge between these attributes (i.e., an oriented edge from the dependent one to the influencing one). Figure 1 represents such an example - i.e., <<carCurrentSpeed>> --> < currentSpeedLimit> .
Extended Functional Dependencies may accept cycles since they represent a semantic link (i.e., even without changes). Nevertheless, two attributes may be linked by an Extended Functional Dependency and by a strong dependency.

Characterized Aggregate Function

Aggregate functions may be useful to sum up a set of data and introduce several levels of abstraction. An aggregate function can be provided with several critical precisions. The computational complexity of such a function is very important in a time-constraint environment. We define three levels of complexity: Linear (e.g., find the average value within a set of values), Quadratic (e.g., the number of tuples that satisfy a join operation in SQL), Super (i.e., higher than quadratic).

Managing navigation applications very often requires the definition of predictions (e.g., on the traffic, schedules). The aggregate functions may use a regression technique, an interpolation or any other types of models. The operations do not belong to the kernel of a DBMS. The application developer must provide an extra operator such as for example a trigger or an add-on operator in object-relational DBMS.

In a time constraint environment, the characteristic of an external evaluation of a function is important. The characterization of the function may be introduced with the Data Definition Language. As an example (Deshpande and Madden, 2006) provides a specific data definition order to create a view based on an interpolation technique (in this case for a set of sensors).

Lattice

Lattices offer a natural way to formalize and study the ordering of objects using a general concept known as the poset, partially ordered set (Buckley and Harary, 1990). A lattice as an algebra is equivalent to a lattice as a poset. Two elements of a lattice are incomparable if neither dominates the other.

We rely in this context on the ANSI/SPARC definition of a DB schema. The three levels are: the external level (i.e., views), the conceptual level (i.e., the global information system schema) and the physical representation (e.g., files, data organization or access with the file management primitives of the operating system). Only the external level is concerned by the proposed extensions. The conceptual level is global for an information system. It does not take into account the specificities of applications. Within the external level, we can take the whole semantics of an application. The data set is reduced (i.e., a sub set of the whole available schema in the conceptual level).

Depending on an application, the same entity may be considered as very important or as quite marginal. We take advantage of this difference of semantics to specify an order on the data model. We consider three levels: Essential attributes, Important attributes, Useful attributes.

Within a level the DB designer does not define an order. In this case, we have a partial order involved by the three levels. Let us consider the DB schema defined as a view on a global schema. Let us consider a mobile application using this schema (e.g., the maintenance of parks in a tourist region). Figure 2 presents the associated lattice.

Essential =	{Park.name, Car.carCurrentSpeed, PartRoute.speedLimit}
Useful =	{Town.mayor, Route.averageSpeed}
Important =	{Town.workInProgress}

Figure 2 – Associated lattice

As an example, a query is defined on such a schema and involves several classes. The aim is to help the DBMS to present relevant data depending on the time constraints. To simplify the presentation let us consider the following relational algebraic operators (Ullman, 1995): Π, the projection operator (i.e., the relevant attribute); σ, the selection operator (i.e., the conditions to be verified)

An example of a simplified query, Q, is defined on the toy DB schema using these operators.

Q = Π (σ Route,origin = 'ParkStartPoint' and destination = "ParkArrivalPoint"),
Park.name, carCurrentSpeed, Route.averageSpeed, Park.leisures).

This query requires providing carCurrentSpeed that is involved in a strong dependency. It also can provide the responsible of the initial and arrival Park since mayor is involved in an Extended Functional Dependency with a park (via the idNumber of Park). It also requires the evaluation of a function with a small complexity (i.e., linear), the computation of the average speed (between the departure and the current time). The DBMS is then able to elaborate an execution plan with some specific priorities with respect to the order defined in the lattice. From the user-interface point of view, the presented data is not a unique query but in fact a set of queries. As we can see in Figure 1, the priority is given to the speed of the car versus other data. Complex data such as providing the current works in progress in the area (an aggregate function with a high complexity) would be provided if the computational time is available. One can remark that the lattice is query-independent. It is based on the application (i.e., the view).

Conclusion

Using these modeling tools, three levels are covered: static model with the extended functional dependency, dynamic on update with the strong dependency, and dynamic on resolution with the lattice and the characterized aggregate function.

To be able to evaluate the complexity of a function the overhead must be limited. In a time constraint environment, we must give the priority to a "compiled" approach. By "compiled" approach we mean that the maximum knowledge must be introduced into the data model or into the query. Managing a meta-base is simpler than evaluating information for each query.

Two main approaches may be defined: the first one is based on the query definition process; the second one is based on the DB schema. Both of them are compatible with the conventional view mechanism defined in DBMS. In both case, the query resolution mechanism must be changed since a transaction may be shortened to respect the time constraints. Therefore, the provided attributes (and the provided values) may be different depending on the context. The same query in different contexts does not provide the same results. This is the price to pay to provide an answer whatever the circumstances are.

With the first approach, the query definition process specifies the relevant relative importance of the required data. The query language must be extended to take into account this new requirement. The DB schema definition is still conventional.

With the second approach, the query definition process is unchanged whatever the used device is (e.g., a mobile phone, a PDA or a micro-computer). The query language is unchanged. The DB schema definition is changed. The data model is increased with new concepts. The main advantage is that an application can be designed as a set of queries and the queries are device-independent (that does not mean that the presentation of the results are device independent). Therefore the current proposition is based on an extension of the DB schema.

PRINCIPLES

A conventional DBMS tries to provide a query result as soon as possible. No time constraint are involved. Whatever the conditions are, the same amount of data will be provided. In a real time constraint environment, the amount of data will be different since the length of a transaction may be different depending on the conditions). We propose to use some heuristics to enhance the accuracy of provided data. Tools defined in the previous section will be used to improve the accuracy of the result.

Extended Functional Dependency

A functional dependency is based on the fact that a value in the left part of the dependency provides a single value in the right part. One can notice that it does not mean that the right value is a single atomic value (e.g., it could be a set). This is not related with the notion of first normal form in relational DB. The sub-set granule property guaranties that the data is relevant in such a context.
Definition H1: In a time-constraint environment, we propose to give priority to attributes that are involved in an extended functional dependency.
It widely depends on the domain associated with the right value. A conventional data type (e.g., integer, real, string) allows taking full advantage of the situation. In a navigation application, providing part of a film, a song or a full text document that popularized the place will change the situation.

Strong dependency

A strong dependency will require that the left value of the dependence is updated as soon as possible. The quality of data is therefore enhanced.
Definition H2: In a time-constraint environment, we propose to give priority to attributes that are involved in a strong dependency.
We do not have the guaranty that the update has been performed. The update has been requested but may not have already been performed. Nevertheless, if the update has been performed, the quality of the answer is enhanced since the value is up to date.

Characterized aggregate function

Conventional relational DBMSs use some heuristics in the execution plan. As an example, selections are performed before joins since the natural complexity of the selection is linear (i.e., no index is used) and since the natural complexity of the join is quadratic. We propose to use a similar heuristic.
Definition H3: In a time-constraint environment, we propose to give priority to characterized aggregate function that are classified as linear, then as quadratic and then as super.
Obviously, this is only a heuristic since we may be in the same situation as the conventional relational DBMS. As an example, in some circumstances a join (formally defined as a selection on a Cartesian product) may be transformed into a (small) set of selections and therefore the complexity is now linear. So an aggregate function classified as quadratic may be transformed into a linear complexity depending on the amount of data during the evaluation of an execution plan.

Lattice

The lattice is defined to provide a better relevance between clients' needs and the answer that could be provided. The lattice is a heuristic by itself since it provides a partial order within the different attributes involved in a query.
This level is the highest level of heuristic. It provides an order between the different attributes that can be involved in the result. The lattice defines a partial order. The lattice defines a partial order. A relative order within a level of the lattice is defined using H1, H2 and H3 otherwise, the order is not important. The order of application between H1, H2, and H3 is external schema dependent (i.e., view dependent).

CONCLUSION

Taking advantage of new applications provided by Location Based Services or GPS data requires introducing the notion of real time constraints in the DB models. Conventional DB models, extended to manage time and/or spatial data, provide an opportunity to manage real applications. A new step has to be defined to handle time constraint in the query resolution mechanism since Location Based Services often involve real-time constraint.
In this article, we proposed to enhance a DB model whatever the underlying formal model is (e.g., UML, object-oriented, object-relational). It takes into account the fact that the response time may vary depending on the context. The introduction of a partial order defines a relative importance between the set of data that may be provided as a result. The aggregate functions involved in a query may be or not computed depending on the available time. The evaluation is based on the computational complexity of the aggregate function. The coherency of a DB, mainly for navigation application is very important. Therefore, the specification of hard coherency maintenance is provided with the strong dependency. The Extended Functional Dependency enhances the relevance of the attributes provided in an answer.
Following the methodology proposed in (Forlizzi et al. 2000), our new step will be the definition of the associated discrete model (i.e., to fix representations). Using these data definition language extensions, a query resolution model can be defined for a real time application context in order to present a sub-set of the data that should be given when no resolution time constraints are applied.

REFERENCES

Bodlaender, M.P. and P.D.V. Stok. 1998. "A transaction-based temporal data model that supports prediction in real-time databases", 10th IEEE Euromicro Workshop on Real Time System, Berlin, Germany, June.

Bossler, J.D. 2002. "Manual of geospatial science and technology", J.D. Bossler, J. R. Jensen, R. B. McMaster, C. Rizos (Eds), Taylor & Francis, London, UK.

Buckley F. and F. Harary. 1990. "Distance in graphs", Addison-Wesley, Redwwod, California.

Callison, H.R. 1995. « A time-sensitive object model for real-time systems », ACM Transactions on Software Engineering and Methodology, 4(3), 287-317.

Deshpande, A. and S. Madden. (2006). "MauveDB: Supporting model-based user views in database systems", 25th SIG Management Of Data (SIGMOD) Conference, Chicago, Illinois, June.

Forlizzi, L.; R.H.Guting; E. Nardelli; and M. Schneider M. 2000. "A data model and data structures for moving objects databases", 19th SIG Management Of Data (SIGMOD) Conference, Dallas, Texas, May.

Gregersen, H. 2006. "The formal semantics of the TimeER model", 3rd Asia-Pacific Conference on the Conceptual Modelling, Hobart, Australia, January.

Guting, R.H.; V.T. de Almeida; and Z. Ding. 2006. "Modeling and querying moving objects in networks", Journal of Very Large Data Base (JVLDB), 15(2), 165-190.

Kennedy, M. 2002. "The Global Positionning System and GIS : an introduction", Taylor & Francis, London, UK.

Liu, Y.; J. Zheng; L. Yan; and Y. Xu. 2005. "Study on the real time navigation data model for dynamic navigation", IEEE GeoScience and Remote Sensing Symposium, Seoul, South Corea, July.

Mainguenaud, M. 1994. "Consistency of Geographical Information System Query Results", Computers, Environment and Urban Systems, No18, 333-342, Pergamon-Elservier.

Parent C.; S. Spaccapietra; and E. Zimanyi. 1999. "Spatio-temporal conceptual models: data structures + space + time", ACM GIS Conference, Kansas City, Missouri, November.

Sun, Y.; E.M. Belding-Royer; X. Gao; and J. Kempf. 2007. "Real-time traffic support in heterogeneous mobile networks", Wireless Network, 13, 431-445, Springer.

Taina, J. and S.H. Son. 1997. "Requirements for real-time object-oriented database models – How much is too much ?", 9th Euromirco Workshop on Real Time System, Toledo, Spain, June.

Ullman, J.D. 1995. "Database and Knowledge-Base Systems – Volume I : Classical Database Systems", Computer Science Press, Rockville, Maryland.

BIOGRAPHY

MICHEL MAINGUENAUD was the head of the Engineering Information System department of the Institut National des Sciences Appliquees de Rouen (INSA-Rouen) and then the Dean for Education of INSA-Rouen. He is currently the director of the CNRS/GDR2340 SIGMA-Cassini research group (www.sigma-cassini.org).

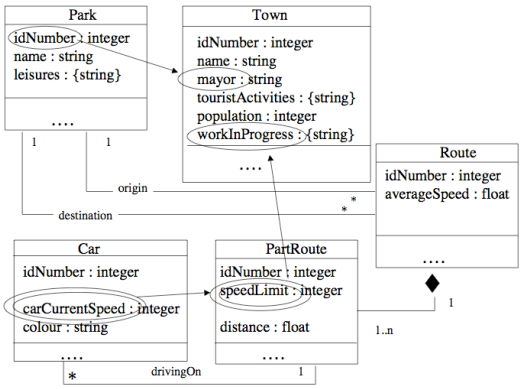

Figure 1 – Toy database schema