DOMINO: Databases fOr MovINg Objects tracking

Ouri Wolfson Prasad Sistla Bo Xu Jutai Zhou Sam Chamberlain

EECS Department, University of Illinois at Chicago Army Research Laboratories

{wolfson,sistla,bxu,jzhou}@eecs.uic.edu wildman@arl.mil

1. BACKGROUND

Consider a database that represents information about moving objects and their location. For example, for a database representing the location of taxi-cabs a typical query may be: retrieve the free cabs that are currently within 1 mile of 33 N. Michigan Ave., Chicago (to pick-up a customer); or for a trucking company database a typical query may be: retrieve the trucks that are currently within 1 mile of truck ABT312 (which needs assistance); or for a database representing the current location of objects in a battlefield a typical query may be: retrieve the friendly helicopters that are in a given region, or, retrieve the friendly helicopters that are expected to enter the region within the next 10 minutes. The queries may originate from the moving objects, or from stationary users. We will refer to applications with the above characteristics as moving-objects-database (MOD) applications, and to queries as the ones mentioned above as MOD queries.

In the military MOD applications arise in the context of the digital battlefield (see [1]), and in the civilian industry they arise in transportation systems. For example, Omnitracs developed by Qualcomm (see [2]) is a commercial system used by the transportation industry, which enables MOD functionality. It provides location management by connecting vehicles (e.g. trucks), via satellites, to company databases. The vehicles are equipped with a Global Positioning System (GPS), and they automatically and periodically report their location.

2. RESEARCH ISSUES

Currently, MOD applications are being developed in an ad hoc fashion. Database Management System (DBMS) technology provides a potential foundation upon which to develop MOD applications, however, DBMS's are currently not used for this purpose. The reason is that there is a critical set of capabilities that are needed by MOD applications and are lacking in existing DBMS's. The following is a discussion of the needed capabilities.

(A) Location Modeling.

Existing DBMS's are not well equipped to handle continuously changing data, such as the location of moving objects. The reason for this is that in databases, data is assumed to be constant unless it is explicitly modified. For example, if the salary field is 30K, then this salary is assumed to hold (i.e. 30K is returned in response to queries) until explicitly updated. Thus, in order to represent moving objects (e.g. vehicles) in a database and answer queries about their location, the vehicle's location has to be continuously updated. This is unsatisfactory since either the location is updated very frequently (which would impose a serious performance overhead), or, the answer to queries is outdated. Furthermore, assuming that the location updates are generated by the moving objects themselves and transmitted via wireless networks, frequent updating would also impose a serious wireless bandwidth overhead.

(B) Linguistic Issues.

Generally, a query in MOD applications involves spatial objects (e.g. points, lines, regions, polygons) and temporal constraints. Consider for example the query: ※Retrieve the objects that will intersect the polygon *P* within the next 3 minutes§ . This is a spatial and temporal range query. The spatial range is the polygon *P*, and the temporal range is the time interval between now and 3 minutes from now. Similarly, there are spatio-temporal join queries such as: ※Retrieve the pairs of friendly and enemy aircraft that will come within 10 miles of each other, and the time when this will happen. § Traditional query languages such as SQL are inadequate for expressing such queries. Although spatial and temporal languages have been studied in the database research community, the two types of languages have been studied independently, whereas for MOD databases they have to be integrated. Furthermore, spatial and temporal languages have been developed for data models that are inappropriate for MOD applications (due, for example, to the modeling problem mentioned above).

(C) Indexing.

Observe that the number of moving objects in the database may be very large (e.g., in big cities with millions of inhabitants). Thus, for performance considerations, in answering MOD queries we would like to avoid examining the location of each moving object in the database. In other words, we would like to index the location attribute. The problem with a straight-forward use of spatial indexing for this purpose is that the continuous change of the locations implies that the spatial index has to be continuously updated. This is clearly an unacceptable solution.

(D) Uncertainty/Imprecision.

The location of a moving object is inherently imprecise because, regardless of the policy used to update the database location of the object (i.e. the object-location stored in the database), the database location cannot always be identical to the actual location of the object. This inherent uncertainty has various implications for database modeling, querying, and indexing. For example, for range queries there can be two different kinds of answers, i.e. the set of objects that "may" satisfy the query, and the set that "must" satisfy the query. Thus, different semantics should be provided for queries. Another approach would be to compute the probability that an object satisfies the query. Although uncertainty in databases has been studied extensively, the new modeling and spatio-temporal capabilities needed for moving objects necessitate revisiting existing solutions.

Additionally, existing approaches to deal with uncertainty assume that some uncertainty information is associated with the raw data stored in the database. How is this initial uncertainty obtained? For MOD applications the question becomes how to quantify the location uncertainty? How to quantify the tradeoff between the updating overhead and the uncertainty/imprecision penalty, and how frequently should a moving object update its location. How to handle the possibility that a moving object becomes disconnected and cannot send location updates?

3. THE DOMINO APPROACH

Therefore, there is a critical set of capabilities that have to be integrated, adapted, and built on top of existing DBMS's in order to support moving objects databases. The objective of our Databases fOr MovINg Objects tracking (DOMINO) project is to build an envelope containing these capabilities on top of existing DBMS's. The key features of our approach are the following.

(1) Dynamic Attributes.

In our opinion, the key to overcoming the location modeling problem is to enable the DBMS to <u>predict</u> the future location of a moving object. Thus, when the moving object updates the database, it provides not only its current location, but its expected future locations. For example, if the DBMS knows the speed and the route of a moving object, then it can compute its location at any point in time without additional updates.

Thus, we proposed a data model called the Moving Objects Spatio-Temporal (or MOST for short) model. Its novelty is the concept of a dynamic attribute, i.e. an attribute whose value changes continuously as time progresses, without being explicitly updated. So, for example, the location of a vehicle is given by its dynamic attribute which consists of motion plan (e.g., north on route 481, at 60 miles/hour). In other words, we devise a higher level of data abstraction where an object's motion plan (rather than its location) is represented as an attribute of the object. Obviously, the motion plan of an object can change (thus the dynamic attribute needs to be updated), but in most cases it does so less frequently than the location of the object. We devised mechanisms to incorporate dynamic attributes in existing data models and capabilities to be added to existing query processing systems to deal with dynamic attributes.

(2) Spatial and Temporal Query Language.

We introduced a query language called Future Temporal Logic (FTL) for query and trigger specifications in moving objects databases. The language is natural and intuitive to use in formulating MOD queries, and it is basically SQL augmented with temporal operators (e.g. SOMETIME-DURING, UNTIL, LATE) and spatial operators (e.g. INSIDE-REGION).

(3) Indexing Dynamic Attributes.

We propose the following paradigm for indexing dynamic attributes. The indexing problem is decomposed into two subproblems; first is the geometric representation of a dynamic attribute value (i.e. a moving object's speed, initial location, and starting time) in multidimensional time-space, and second is the spatial indexing of the geometric representation. The geometric representation subproblem concerns the question: how to construct the multidimensional space, and how to map an object (more precisely, a dynamic attribute value) into a region (or a line, or a point) in that space, and how to map a query into another region in that space, so that the result of the query are the objects whose regions intersect the query region. The object region is updated only when the dynamic attribute is explicitly updated (e.g. when the speed of the object changes) rather than continuously. The spatial indexing subproblem concerns the question how to find the intersection-of-regions mentioned above in an efficient way. The latter subproblem can be solved by an existing spatial indexing method, but it is an open problem which method is most appropriate for a particular geometric representation and dynamic attribute values distribution. We have devised several solutions to the geometric representation subproblem.

(4) Uncertainty/Imprecision Management.

We extended our data model, query language, and indexing method to address the uncertainty problem. The data model was extended by enabling the provision of an uncertainty interval in the dynamic attribute. More specifically, at any point in time the location of a moving object is a point in some uncertainty interval, and this interval is computable by the DBMS. Thus, the DBMS replies to a query requesting the location of a moving object *m* with the following answer A: "m is on route 698 at location (*x*,*y*), with an error (or deviation) of at most 2 miles". The bound *b* on the deviation (2 miles in the above answer) is provided by the moving object, i.e. the object commits to send a location update when the deviation reaches the bound.

The FTL language is also extended. We devised two extensions, a qualitative one and a quantitative one. In the qualitative extension, two kinds of semantics, namely MAY and MUST semantics, are incorporated, and the processing algorithms are adapted for these semantics. The indexing method is also extended to enable the retrieval of both, moving objects that "must be" in a particular region, and moving objects that "may be" in it. In the quantitative extension, the location of the moving object is a random variable, and the uncertainty interval, the network reliability and other factors are used to determine a density function for this variable. An algorithm was developed to associate with each object retrieved in response to a range query, the probability that the object satisfies the query.

We also addressed the question of determining the uncertainty associated with a dynamic attribute, i.e. the bound *b* mentioned above. We proposed a cost based approach which captures the tradeoff between the update overhead and the imprecision. The location imprecision encompasses two related but different concepts, namely deviation and uncertainty. The deviation of a moving object *m* at a particular point in time *t* is the distance between m's actual location at time t, and its database location at time t. For the answer A above, the deviation is the distance between the actual location of *m* and (*x*,*y*). On the other hand, the uncertainty of a moving object *m* at a particular point in time *t* is the size of the interval in which the object can possibly be.

For the answer A above, the uncertainty is 4 miles. The deviation has a cost (or penalty) in terms of incorrect decision making, and so does the uncertainty. The deviation (uncertainty) cost is proportional to the size of the deviation (uncertainty). The tradeoff between imprecision and update overhead is captured by the relative costs of an uncertainty-unit, a deviation-unit, and an update-overhead unit. Using the cost model we propose update policies that establish the uncertainty bound *b* in a way that minimizes the expected total cost. Furthermore, we propose an update policy that detects disconnection of the moving object at no additional cost.

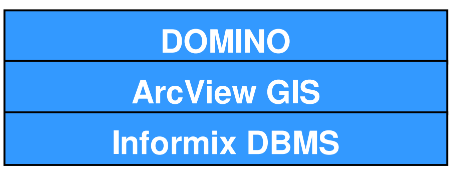
4. THE DEMONSTRATION

We will demonstrate the following features of Domino:

4.1 System Architecture

Our Domino system is the third in a three-layer architecture (see Figure 1). The first layer is an Object Relational DBMS. The database stores the information about each moving object, including its plan of motion. The second layer is a GIS that adds capabilities and user interface primitives for storing, querying, and manipulating geographic information. The third layer, Domino, adds temporal capabilities, capabilities of managing the uncertainty that is inherent in expected future motion plans, and a simulation testbed. Currently, Domino uses the Informix DBMS and the Arc-View GIS.

Figure 1: System architecture



4.2 Motion Plan Specification

The motion plan of a moving object is a sequence of location-time points, (*p1*,*t1*), (*p2*,*t2*),...(*pn*,*tn*), indicating that the object will be at geographic point *p1* at time *t1*, at geographic point *p2* (closer to the destination than *p1*) at time *t2* (later than *t1*), etc. The plan is interactively specified by the user on a GIS on a map. The moving object updates the database whenever the deviation from the plan exceeds a prespecified bound given in terms of distance or time. The update includes a revised plan and possibly a new bound on the deviation. Maintaining plan information enables queries pertaining to both, the current and future locations of the moving object, for example:

Q1 = Retrieve the moving objects that are expected to be in a given region *R* sometime during a given time interval *I*.

Also, queries may pertain to future arrival times, for example:

Q2 = Retrieve the moving objects that are expected to be late at their destination by more than one hour.

4.3 Spatio-temporal Capabilities

We will demonstrate the spatial and temporal primitives of the FTL query language and its answer-display screen. The primitives are given in graphical format, and they can be combined with textual SQL in a natural and intuitive way. For example, in the query Q1 above the region *R* may be drawn with a mouse on a real GIS map, and the time interval *I* may be specified on a graphical timeline. Then *I* and *R* can be incorporated in the textual part of an FTL query. Clearly, since FTL is an extension of SQL, the query can also include regular literals, e.g., WEIGHT > 5000. Information about the moving objects that satisfy the query is displayed in textual form, and the location of each such moving object is displayed as a point on the map.

4.4 Uncertainty

We will demonstrate the capabilities of the FTL query language and its answer-display screen in dealing with uncertainty. These include MAY and MUST semantics for queries. In other words, the query Q1 above can be specified with MAY or MUST semantics. Under the MAY semantics, an object will be retrieved if its uncertainty interval intersects the region *R* sometime during the interval *I*. Under the MUST semantics, an object will be retrieved if its uncertainty interval is wholly contained in the region *R* sometime during the interval *I*. The location of each moving object retrieved is displayed on the map, along with the uncertainty interval currently associated with the location.

4.5 Simulation Testbed

We will demonstrate a simulation testbed in which the performance of a moving objects database application can be evaluated. The input to the simulation system is a set of moving objects, their motion plans, their speed variations over time, the cost of deviation, the cost of uncertainty, the cost of communication, the wireless bandwidth distribution over the geographic area, and the location update policy used by each moving object. The objective is to determine the performance of MOD queries, as well as to answer questions such as: How many objects can be supported for an average imprecision that is bounded by *x*, and a wireless bandwidth allocated to location updates that is bounded by *y*? Or, given n moving objects and a bound of 10% on the imprecision, what percentage of the bandwidth is used for location updates?

5. REFERENCES

[1] S. Chamberlain, Model-Based Battle Command: A Paradigm Whose Time Has Come, 1995 Symp. on C2 Research and Technology, June 1995.

[2] OmniTRACS, Communicating Without Limits, http://www.qualcomm.com/ProdTech/Omni/prodtech/omnisys.html.

[3] P. Sistla, O. Wolfson, S. Chamberlain, S. Dao, Modeling and Querying Moving Objects, Proceedings of the Thirteenth International Conference on Data Engineering (ICDE13), Birmingham, UK, Apr. 1997.

[4] O. Wolfson, L. Jiang, P. Sistla, S. Chamberlain, N. Rishe, M. Deng, Databases for Tracking Mobile Units in Real Time, Springer-Verlag Proceedings of the Seventh International Conference on Database Theory (ICDT), Jerusalem, Israel, Jan. 10-12, 1999.

[5] O. Wolfson, B. Xu, S. Chamberlain, L. Jiang, Moving Objects Databases: Issues and Solutions, Proceedings of the 10th International Conference on Scientific and Statistical

Database Management (SSDBM98), Capri, Italy, July 1-3, 1998, pp. 111-122.