CAT: Correct Answers of Continuous Queries Using Triggers

Goce Trajcevski[1], Peter Scheuermann[1⋆], Ouri Wolfson[2], and Nimesh Nedungadi[2]

[1] Department of Electrical and Computer Engineering, Northwestern University, Evanston, Il 60208,
{goce,peters}@ece.northwestern.edu

[2] Department of Computer Science, University of Illinois at Chicago, Chicago, Il 60607,
{wolfson,nnedunga}@cs.uic.edu

1 Introduction and Motivation

Consider the query *Q1: Retrieve all the motels which will be no further then 1.5 miles from my route, sometime between 7:00PM and 8:30PM*, which a mobile user posed to the Moving Objects Database (MOD). Processing such queries is of interest to wide range of applications (e.g. tourist information systems and context awareness [1,2]). These queries pertain to the *future* of a dynamic world. Since MOD is only a model of the objects moving in the real world, the accuracy of the representation has to be continuously verified and updated, and the answer-set of *Q1* has to be re-evaluated in every clock-tick[1]. However, the re-evaluation of such queries can be avoided if an update to the MOD does not affect the answer-set.

The motion of the moving object is typically represented as a *trajectory* – a sequence of 3D points $(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots (x_n, y_n, t_n)$, and its projection in the X-Y plane is called a *route*. The details of the construction based on electronic maps and the *speed-profiles* of the city blocks are given in [5]. After a trajectory is constructed, a *traffic abnormality* may occur at a future time-point, due to an accident, road-work, etc..., and once it occurs, we need to: *identify* the trajectories that are affected, and *update* them properly (c.f. [5]). In the sequel, we focus on the impacts of the abnormalities to the continuous queries.

Figure 1 shows three trajectories – Tr_1, Tr_2 and Tr_3 and their respective routes R_1, R_2 and R_3. If a road-work starts at 4:30PM on the segment between A and B which will last 5 hours, slow down the speed between 4:30PM and 9:30PM. Tr_1 enters that segment <u>after</u> 4:30PM, and its future portion will need to be modified. As illustrated by the thicker portion of Tr_1, instead of being at the point B at 4:50, the object will be there at 5:05. A key observation is that if the object, say o_1, whose trajectory is Tr_1, issued the query $Q1$, we have to re-evaluate the answer.

⋆ Research partially supported by NSF grant EIA-000 0536

[1] Hence the name *continuous queries* – formally defined for MOD in [3].

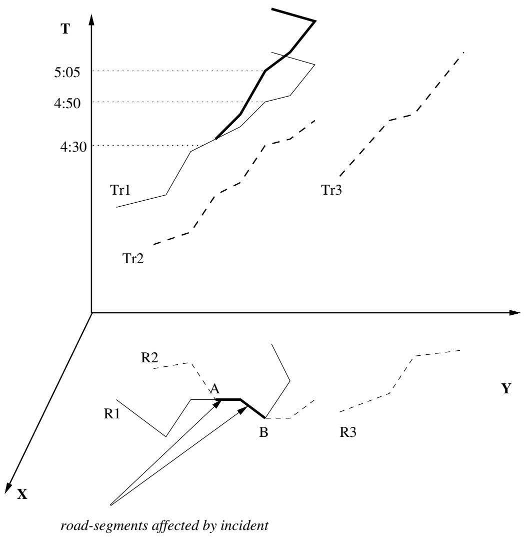
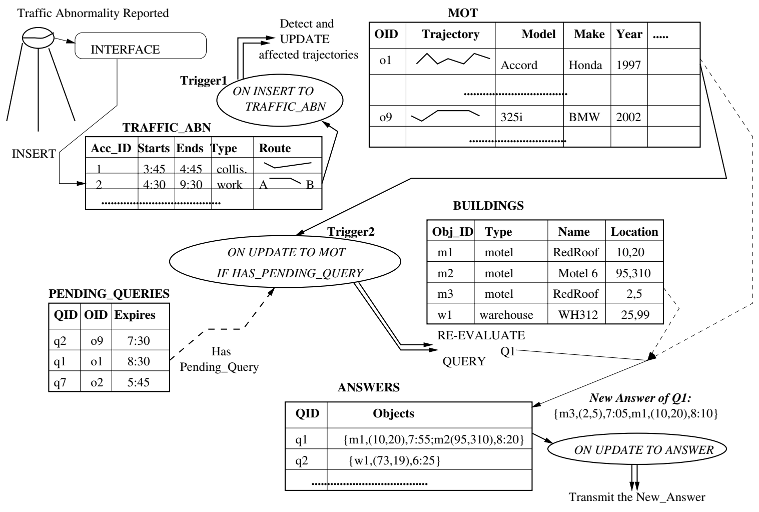


Fig. 1. Trajectories, Routes and Updates

2 System Architecture

Figure 2 illustrates the main components and behavioral aspects of the CAT system:
• There are tables which: store the trajectories (MOT) and landmarks of interest (BUILDINGS); keep track of the queries posed and their answers (PENDING_QUERIES and ANSWERS); and store the information about the traffic abnormalities (TRAFFIC_ABN). The trajectories and the landmarks were obtained using the real maps of Cook County, Chicago.

In the heart of the CAT system is the set of *Triggers*, part of which we illustrate in the context of the example query $Q1$. If $o1$ posed the query $Q1$ at 3:30PM, its answer contains the motels m_1 and m_2 to which $o1$ should be close at 7:55PM and 8:20PM, respectively. When an abnormality is detected, its relevant parameters are inserted in the TRAFFIC_ABN table. This, however, "awakes" $Trigger_1$, whose event is:
ON INSERT TO TRAFFIC_ABN
$Trigger_1$ checks if the abnormality affects some of the trajectories stored in the MOT and, if so, updates their future part. However, the action part of $Trigger_1$ which updates the MOT, in turn, is the event for $Trigger_2$:
ON UPDATE TO MOT
IF HAS_PENDING_QUERIES ...
$Trigger_2$ checks if the corresponding moving object has posed some queries which are still of interest (i.e. their time has not "expired"). The condition of $Trigger_2$ is satisfied by $o1$ and its action part will re-evaluate the query $Q1$, based on the new future-portion of Tr_1. Due to the delay, o_1's trajectory will



MOT

OID	Trajectory	Model	Make	Year	.....
o1		Accord	Honda	1997	
	....				
o9		325i	BMW	2002	
	....				

TRAFFIC_ABN

Acc_ID	Starts	Ends	Type	Route
1	3:45	4:45	collis.	
2	4:30	9:30	work	A B
....	....	....		

BUILDINGS

Obj_ID	Type	Name	Location
m1	motel	RedRoof	10,20
m2	motel	Motel 6	95,310
m3	motel	RedRoof	2,5
w1	warehouse	WH312	25,99

PENDING_QUERIES

QID	OID	Expires
q2	o9	7:30
q1	o1	8:30
q7	o2	5:45

ANSWERS

QID	Objects
q1	{m1,(10,20),7:55;m2(95,310),8:20}
q2	{w1,(73,19),6:25}
.....	....

Fig. 2. Behavioral Aspects of the CAT

be near m_2 at 8:35PM, which is a bit too late for the user. On the other hand, o_1 will be near the motel m_3 at 7:05PM. Before the traffic incident $m3$ was not part of the answer, (it would have had the desired proximity at 6:50PM).

3 Demonstration

All the back-end components are implemented using Oracle 9i as a server. We used User-Defined Types (UDT) to model the entities and User-Defined Functions (UDF) to implement the processing, exploiting the Oracle Spatial predicates.

• The front-end client, which is the GUI presented to the end-user, is implemented in Java. The GUI gives the options of specifying the queries (i.e. time of submission; relevant time interval; objects of interest; etc...). Once the user clicks the SUBMIT button, the query is evaluated and its answer is displayed. In the server, the query is assigned an *id* number and it is stored in the PENDING_QUERIES table. Clearly, in a real MOD application, the client will be either a wireless (mobile) user of a web browser-based one, properly interfaced to the server.

• To test the execution of the triggers and the updates of the answer(s) to the continuous queries posed, the GUI offers a window for generating a traffic abnormality. The user enters the beginning and the end times of the incident as well as its "type" (which determines the impact on the speed-profile). He also enters the route segments along which the traffic incident is spread. The moment this information is submitted to the server, the affected trajectories are updated AND the new answer(s) to the posed continuous queries are displayed back to the user.

References

1. A. Hinze and A. Voisard. Location-and time-based information delivery in tourism. In *SSTD*, 2003.
2. A. Pashtan, R. Blatter, A. Heusser, and P. Scheuermann. Catis: A context-aware tourist information system. In *IMC*, 2003.
3. A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *ICDE*, 1997.
4. G. Trajcevski and P. Scheuermann. Triggers and continuous queries in moving objects databases. In *MDDS*, 2003.
5. G. Trajcevski, O. Wolfson, B. Xu, and P. Nelson. Real-time traffic updates in moving object databases. In *MDDS*, 2002.