AN ARCHITECTURE FOR LOCATION BASED SERVICES

Priyanka Goel[1], Neha Gupta[2], Shweta Agarwal[3]

[1,2,3]Assistant Professor, CS & IT Department, MIT Moradabad, (India)

ABSTRACT

In this paper we discuss the various challenges in a mobile environment where the querying unit and the object being queried are in motion. After this we focus on the classification of different types of location dependent queries and various solutions to solve these complex location dependent queries. Additionally this paper also focuses on various methods of LDQP and its applications. The basic aim of this paper is to study various LDQP along with data management problems involved in evaluation of location based queries and its classification.

Keywords: Content Provider (CP), Location Based Query (LBQ), Location Based Services (LBS), Location Dependent Query Processing (LDQP), Mobile User (MU).

I INTRODUCTION

The main aim is facilitating data access to the user anywhere and at any instant of time. Nowadays, people are always on the move and hence, mobile environments hold key aspects to information retrieval. The need to access daily information about the weather, restaurant locations, stock exchange and so on, is unavoidable. This kind of information can be accessed at any time anywhere, because people connect their device to a server in a wireless fashion without any limitations of distance boundaries. Queries are sent to servers while users are moving. These queries are called mobile queries. The queries are accepted by a stationary host, called the Base Station (BS). A base station is a stationary entity which acts as a mediator to forward the message to wireless and wired networks for a certain area. The particular area which is served by a single base station is called a Cell. From the mobile users' point of view, these areas are transparent. In other words, the users do not know if they move within a single cell or multiple cells. Based on this, we categorize queries into two types, Single-cell and Multi-cell cells queries. A single cell query is one that asks for information about objects where the objects are located in the same cell as the mobile user. On the other hand, the multi-cell query is one that asks for the current cell and its neighbouring cells.

If we talk about mobile query processing, it can be done either in the client side or the server side or both. Client side query processing can be achieved by providing a cache for the mobile device. At the server side, the process involves a single or multiple servers depending on the query type.

A cache is a temporary space to store most requested data items in the local storage. Client devices accommodate these query results in a cache upon receiving the query results. When the client asks similar queries, the query results will be answered from the cache if they are available on the cache. Hence, communication to a server can be avoided.

II LOCATION BASED QUERIES

A location based queries are processed on the basis of location dependent data, and its result depends on the location criteria explicitly or implicitly specified. Location-based queries or location-dependent queries provide support for Location-based services (LBS). Location-based applications (LBA) offer location-based services (LBS) by using queries called the Location-based Queries (LDQ); the result of execution of these queries is based on the location of the mobile user (MU).

2.1 Classification of LDQs

2.1.1. Traditional Classification

Traditional query is the most widely known query used in a database. The query types of traditional query can be classified as: Spatial, Temporal, Spatio-Temporal (Hybrid) and others.

(a) A **Spatial query** performs operations which include spatial searches and map overlay, as well as distance-related operations. A spatial query always requests for spatial data information. Spatial data means that the requested data have a complex structure, are often dynamic and no standard algebra are defined.

(b) A **Temporal query** specifies a validity or deadline for the query results to be returned. Example: A person retrieves a calendar for current year". This calendar will not be valid for the past or future year.

(c) A **Spatial-Temporal (Spatio-temporal) query** requests for a spatial search and specifies the validity or deadline for the query results to be received. Example: Retrieve the five ambulances that were nearest to the location of the accident between 4-5pm."

(d) The last category is **Other**. It implies that the other remaining queries do not belong to one of the classifications above. Examples: A tourist requests restaurant information, Students request their academic records or contact details.

2.1.2. Classification according to the evaluation time and past states

(a) **Instantaneous queries (snapshot queries):** the answer for which is computed only once and forwarded to the user at once only.

(b) **Continuous queries:** are evaluated until the user decides to terminate them by itself.

(c) **Persistent queries:** are continuous queries that consider both the current state and the past states of the moving objects. For example, a user in a car can launch “retrieve the motels within 5 miles of my position” as a continuous query, so that the answer to this query is automatically refreshed until he/she finds a satisfactory motel. An example of persistent query is “retrieve the objects whose speed duplicates within 10 seconds.”

III SERVICES ARCHITECTURE

The figure below is a LBS architecture that is used to serve the mobile users requesting both location-dependent and location-independent data. The location-based architecture in the figure1 below explains the services provided from the point when the mobile user initiates the request to a particular location based service to the response that is broadcasted to the user in the form of response.

The LBS system is composed of three major units: mobile user, central database server and local database server that comprises of components - location services & profile manager, query generator, query processor, query optimizer, remote data manager, local cache manager, query cache manager and a data integrator that performs the function of keeping the overall data in place together.

There is a local database server database server and mobile user. The central database server is used to create the service site for data collected. A push unit is used to store prefetched data as required by the server. This process reduces cost of uplink communication thus utilizing the wireless channel effectively.

In the widely accepted mobile computing model, the wireless interface of the Base Station or Mobile Support Station supports the wireless communication via cellular networks, satellites, or wireless LAN technology [1], [2].

For mobile computing applications, if needed, the location of the mobile user may be estimated by an LCS in collaboration with the location management provided by the wireless operator. We include the LCS box in traditional mobile computing architecture and assume the location of the user is either provided by the network or by the device (GPS).We also assume the primary copy of *LDD* resides on some fixed node in the network but a cached copy of it may exist on the mobile device.

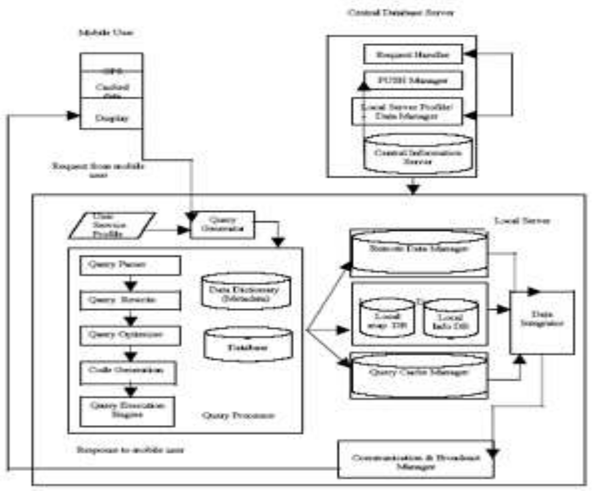


Fig1: LBQ Services Manager Architecture

IV SERVICES FRAMEWORK

A detailed view of *LDS* framework is given in Figure 2. It includes the Mobile Unit, Wireless Network, LDSM and the Fixed Network Partner Services. Mobile devices may access an enterprise network or Internet, and the connection to fixed networks is through a wireless operator's network. *LDS* applications are implemented by the service providers or they are the software adapted for legacy/existing applications. We refer to any end user services which support *LDQ*s as *Fixed Network Partner Services*.

The *Wireless Network* provides the wireless interface and the operator has the responsibility of supplying the location of the *Mobile Unit* to authorized parties by the help of an LCS.

In the *LDS* framework, we envision that *Mobile Unit*s have two types of software components: an *LDS Interface* to provide the basic interface for *LDS* applications, and *User Service Agents - (USAs)*. The mobile user prepares the query, sends it to wireless network, sets user preferences (such as caching, prefetching, and complex query filtering), and views the results with the help of these two interfaces.

The *LDS Interface* is a general interface for *LDS* applications

which does the query communication between the user and the *LDSM*. Queries may originate from the *USA*s or may be the basic queries provided by the interface itself. In the interface, basic facilities are provided to sign on with an *LDSM* service, to state user preferences for *LDSM* services, to view results of previously requested queries (perhaps interrupted by disconnect).

User Service Agents - *USAs* are the mobile user interfaces for the corresponding Fixed Network *LDS* applications. These interfaces may be pre-installed or may be obtained by subscribing to a service or a group of services. A mobile browser can also be considered as a *USA* which provides an interface for Internet applications. The middleware can be passed without any processing in this case.

Each application or a group of applications has to register related properties and arguments to the *LDSM* to start the service. In this way, a service provider informs *LDSM* about the application needs, such as the location granularity type needed for the queries, whether the location is defined in the database schema or if data partitioning is used in implementation. These should be specified in order for the queries sent by the mobile user to be transformed to an appropriate form to get the answers from the service. The legacy systems should be able to process *LDQ*s by registering with the *LDSM*.

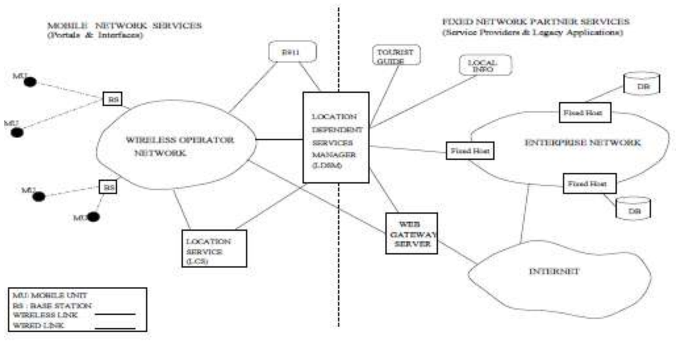


Fig2: Location Dependent Services Framework

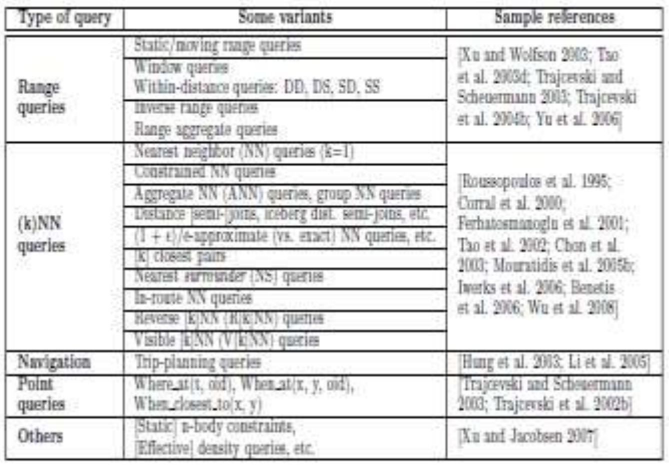
V QUERY CLASSIFICATION

In general, two types of queries can be distinguished in a mobile environment:
A **traditional query**, is a query whose answer does not depend on locations or retrieve location data; for example, “retrieve the names of the employees of a company”.

A **location-dependent query**, on the contrary, is a query whose answer depends on the locations of objects; that is, the location is an attribute that determines whether an object is part of the answer or not. For example, “find hotels within 5 miles” depends on the location of the user that poses the query, and “find taxi cabs near Peter” depends on both the location of the taxi cabs and the location of Peter; in works such as [3], the term location-dependent query refers to queries that depend only on “the query issue location” (i.e., the location of the mobile user).
In the following, location-dependent queries,they are classified according to their purpose -
In the specialized literature, different types of location-dependent queries have attracted attention (see Table I), among which the following could be highlighted (some examples will be presented in relation to the sample scenario presented in Figure 3):

Range queries[4] They retrieve the objects located within a certain range/region. Range queries can be static range queries or moving/mobile range queries, depending on whether the interesting region is fixed or moves.

Table1: Types of LDQ according to their purpose.



Type of query	Some variants	Sample references
Range queries	Static/moving range queries	[Xu and Wolfson 2003; Tao et al. 2003d; Trajcevski and Scheuermann 2003; Trajcevski et al. 2004b; Yu et al. 2006]
	Window queries Within-distance queries: DD, DS, SD, SS	
	Inverse range queries Range aggregate queries	
(k)NN queries	Nearest neighbor (NN) queries (k=1)	[Roussopoulos et al. 1995; Corral et al. 2000; Ferhatosmanoglu et al. 2001; Tao et al. 2002; Chon et al. 2003; Mouratidis et al. 2005b; Iwerks et al. 2006; Benetis et al. 2006; Wu et al. 2008]
	Constrained NN queries	
	Aggregate NN (ANN) queries, group NN queries	
	Distance [semi-]joins, iceberg dist. semi-joins, etc.	
	$(1+\epsilon)/\epsilon$-approximate (vs. exact) NN queries, etc.	
	[k] closest pairs	
	Nearest surrounder (NS) queries	
	In-route NN queries	
	Reverse [k]NN (R[k]NN) queries	
	Visible [k]NN (V[k]NN) queries	
Navigation	Trip-planning queries	[Hung et al. 2003; Li et al. 2005]
Point queries	Where_at(t, oid), When_at(x, y, oid), When_closest_to(x, y)	[Trajcevski and Scheuermann 2003; Trajcevski et al. 2002b]
Others	[Static] n-body constraints, [Effective] density queries, etc.	[Xu and Jacobsen 2007]

Nearest neighbor (NN) queries [5]. They retrieve the object of a certain class which is the closest to a certain object or location.
Navigation queries retrieve the best path for a mobile client to get to his/her destination, based on the underlying road network and the current traffic conditions [6].

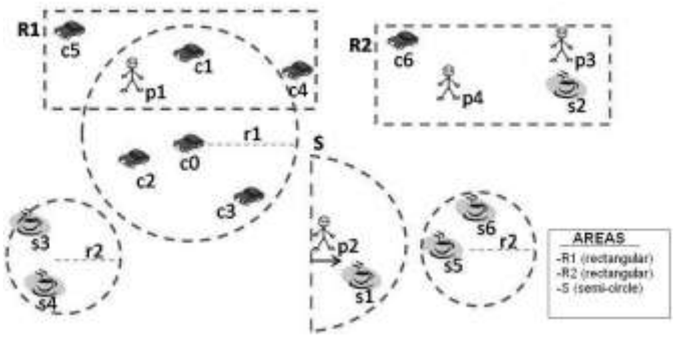


Fig3: Sample scenario for location-dependent queries

Navigation queries retrieve the best path for a mobile client to get to his/her destination, based on the underlying road network and the current traffic conditions [6].

Related to the above classification, location-dependent queries are distinguished in [7] based on different types of predicates: one-to-many queries (a predicate is applied to many objects; e.g., nearest neighbor queries) vs. many-to-many queries (a predicate is checked for every object in relation to every other object; e.g., –constrained– spatial or spatio-temporal joins [8][9] or closest pair queries [10], queries that involve topological/directional/metric predicates, etc.

VI SERVICES MANAGER

The **Location Dependent Services Manager (LDSM)** is shown Figure4. We describe the functionality of the *LDSM* using the following example query: "*Find the closest hotels.*". The area to be searched may be a circle, the user being at the center. This area could be a half circle or rectangle in the direction of the movement of the mobile user. As it is seen, this query uses only one relation and location constraint is implicitly specified. Suppose the area to be searched is a circle, user being at the center, and the relation is "Hotel". The processing of this query by the five basic components of *LDSM* is as follows:

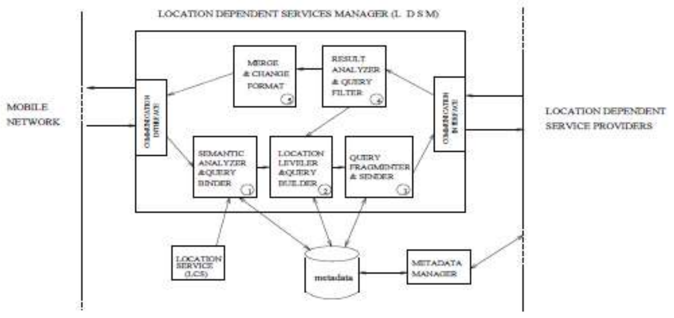

LOCATION DEPENDENT SERVICES MANAGER (L D S M)
MOBILE NETWORK
COMMUNICATION INTERFACE
MERGE & CHANGE FORMAT
RESULT ANALYZER & QUERY FILTER
COMMUNICATION INTERFACE
LOCATION DEPENDENT SERVICE PROVIDERS
SEMANTIC ANALYZER &QUERY BINDER
LOCATION LEVELER &QUERY BUILDER
QUERY FRAGMINTER & SENDER
LOCATION SERVICE (LCS)
metadata
METADATA MANAGER

Fig4: Location Dependent Service Manager

6.1 Semantic Analyzer and Query Binder

The request sent by the user is semantically analyzed, checked to determine whether the user's current location is needed or not and query predicates are identified. The spatial operator "closest" is replaces with a predicate "within 5 miles". The query implies the current position of the user, so the mobile user id is sent to LCS.

6.2 Location Leveler and Query Builder

The query is checked whether the bound location is sufficient for the service. The *metadata* database, the location graphs are examined for the data granularity. When the *LDS* provider for this query has stored the hotels by their zipcodes, there is a Location Granularity Mismatch. By using the location graphs, the Location Leveler translates the query to a range query, or a set of queries which will include the zipcodes. The new location granularities are bound to the query and sent to Query Fragmenter.

6.3 Query Fragmenter and Sender

When the query involves more than one database and one *LDS* application provider, it has to be fragmented and sent to the corresponding site for processing. After the second binding, execution plan of the query can be determined. After the fragmentation of the query, *LDSM* communication interface sends the queries to the related data sources and service providers. If one fragment's execution depends on other's successful ending, then *LDSM* has to keep the query processing records. Assume the "Hotel" database is a centralized database, then the query sent to the service provider as it is.

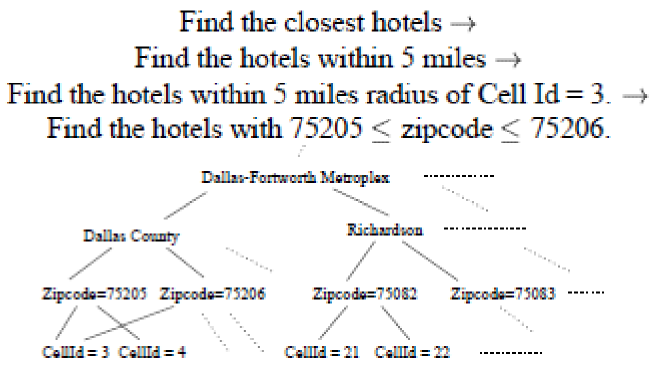
6.4 Result Analyzer and Query Filter (4)

When the query results are ready, they are sent to the *LDSM*. The *LDSM* then combines the individual results and according to the QoS constraints. The results may need to be modified using some filtering mechanisms to ignore the obvious unnecessary results.

6.5Merge and Change Format

The query results may be merged and the location constraints included in the results are ignored. If the *Mobile Unit* is a PDA and needs the result in a tiny HTML format, then the format can be changed. Since there is no fragmentation in this query, only one group of results are expected.

The *LDQ* query goes through several stages prior to being processes as shown in Figure5:



Find the closest hotels →
Find the hotels within 5 miles →
Find the hotels within 5 miles radius of Cell Id = 3. →
Find the hotels with $75205 \leq$ zipcode ≤ 75206.

Fig5: A Sample Location Graph.

VII CONCLUSION

Location Based Query Processing is the building block for the location based services. In this paper, we proposed a model for managing the LDS applications of different types and different requirements. LDSM provides a standard framework for these applications which is flexible, independent and facilitates adaptability.

The future of these location dependent services is highly sensitive for rapid development that requires promising investments, but at the same time the future of well-being of these services is significantly critical.

REFERENCES

[1] M. H. Dunham and A. Helal. Mobile computing and databases: Anything new? *SIGMOD Record*, 24(4):5–9, December 1995.

[2] E. Pitoura and B. Bhargava. Building information systems for mobile environments. In *Third International Conference on Information and Knowledge Management, CIKM'94*, pages 371–378, Gathesburg, MD, USA, November 1994.ACM.

[3] Seydim, A. Y., Dunham, M. H., and Kumar, V. 2001. Location dependent query processing. In 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'01). ACM, New York, NY, USA, 47–53.

[4] Xu, B. and Wolfson, O. 2003. Time-series prediction with applications to traffic and moving objects databases. In 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'03). ACM, New York, NY, USA, 56–60.

[5] Tao, Y., Papadias, D., and Shen, Q. 2002. Continuous nearest neighbor search. In 28th Inter- national Conf. on Very Large Data Bases (VLDB'02). Morgan Kauffman, San Francisco, CA, USA, 287–298.

[6] Hung, D., Lam, K., Chan, E., and Ramamritham, K. 2003. Processing of location-dependent continuous queries on real-time spatial data: The view from RETINA. In 6th International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'03). IEEE Computer Society, Washington, DC, USA, 961–965.

[7] Huang, X. and Jensen, C. S. 2004. Towards a streams-based framework for defining locationbased queries. In 2nd Workshop on Spatio-Temporal Database Management (STDBM'04).73–80.

[8] Tao, Y. and Papadias, D. 2003. Spatial queries in dynamic environments. ACM Trans. Database Syst. 28, 2 (June), 101–139.

[9] Sun, J., Tao, Y., Papadias, D., and Kollios, G. 2006. Spatio-temporal join selectivity. Inf. Syst. 31, 8 (Dec.), 793–813.

[10] Corral, A., Manolopoulos, Y., Theodoridis, Y., and assilakopoulos, M. 2000. Closest pair queries in spatial databases. In ACM SIGMOD International Conf. on Management of Data (SIGMOD'00). ACM, New York, NY, USA, 189–200.