CHAPTER 5
LOCATION DEPENDENT QUERY PROCESSING
TECHNIQUES

5.1 LOCATION DEPENDENT QUERY PROCESSING

In a mobile cloud database system environment, a user providing queries to a database is allowed to move from one location to another. Therefore, spatial and temporal query-processing techniques cannot be applied directly in mobile cloud databases for effective querying. Naturally, location-dependent query processing techniques intrinsic to mobile cloud databases can be developed by extending the techniques used for processing location-dependent queries in mobile database systems. A location-specific query such as “Find the restaurant nearest to Anna University, Chennai” needs to ascertain the location first before zeroing in on restaurants in the area. Since the university’s location is static, data stored in relational databases with location information is sufficient to process the query. If spatial queries are used, the query can be answered faster, requiring the use of spatial data structures to store data in the database. On the other hand, if new restaurants surface frequently, temporal constraints are also necessary, in addition to spatial database features.

Processing a query such as “Find me/the user the nearest restaurant” requires location information, mobility speed, direction of movements, spatial constraints, temporal constraints and storage structures for successful query processing. Such a query can be better answered by a mobile
cloud database system than a mobile database system in which the cache size is miniscule. The main memory and secondary memory also have limited capabilities. Therefore, processing location-dependent queries requires continuous power supply, spatial features, temporal features and functional storage structures. On the other hand, cloud database systems provide large quanta of memory and elastic processing power. Suitable methods must be developed to process location-dependent queries in mobile cloud database systems.

In this work, a new algorithm for performing location-dependent query processing is proposed for the mobile cloud. In this model, data is stored in the cloud but the user allowed to move from one place to another during query processing. The location-dependent query processing algorithm uses the spatio-temporal tree data structure created during data indexing to enhance the performance of query processing. It considers a range of costs – in terms of the I/O, network, encryption, decryption, cloud resources and mobility - to perform query optimization. The major contribution of this algorithm is that it effectively takes into consideration both security aspects and large storage requirements with temporal analysis for fruitful decision making.

5.2 PROPOSED LOCATION DEPENDENT QUERY PROCESSING ALGORITHM

In this thesis, a new location-dependent query processing algorithm called the Agent-based Location-dependent Query Processing Algorithm (ALDQPA) is proposed. Intelligent agents play roles, performing tasks such as sending and receiving messages, coordinating with sender agents to send messages, and managing locations during query processing in location-dependent query processing.
Agent-based Location-Dependent Query Processing algorithm

Input: Location-dependent query

Output: Tuples obtained by query execution at various sites of the computer network.

Step 1: The coordinate agent distributes the user query to the sender agent which makes a broadcast of the query who is within the specified range of distance from the home location of coordinate agent.

Step 2: Receiver agents receive the user query at all network areas.

Step 3: The receiver agent receives the query in each area.

Step 4: The receiver agent calls the location management agent and checks the distance between sender and receiver nodes in each network area.

Step 5: If (distance < Base station range), check the cache at its location for information.

Step 6: If found, send it to the coordinator agent.

Step 7: If not found in the cache, read the information from the database and send it to the coordinate agents.

Step 8: Upon receiving responses from all areas, the coordinator merges them by performing a union operation.

Step 9: The coordinate agent provides the result to the user.

The proposed Agent-based Location-Dependent Query Processing (ALDQP) algorithm is developed in this thesis to provide efficient location-dependent services with the help of intelligent agents. The proposed ALDQP algorithm is used to search the cache memory so as to improve the
performance of the query process. The searching process is carried out by all participants simultaneously at various network areas and hence the performance of the query processing is enhanced.

5.3 MOBILE TRANSACTION MODEL

A new cluster-based transaction model is proposed in this work for handling mobile transactions. The cluster heads are used for communication and act as local coordinators for executing transactions in the proposed transaction model. The steps of the transaction processing algorithms are as follows:

Transaction Processing Algorithm

Step 1: The coordinator starts the transaction by issuing a begin transaction primitive, after starting a timer.

Step 2: The coordinator performs clustering using the K-Means clustering algorithm (Chi-Ho Tsang & Sam Kwong 2005).

Step 3: The coordinator initiates the cluster head election process.

Step 4: The coordinator asks for commit or rollback messages from all cluster heads.

Step 5: At each cluster, the cluster head asks for ready messages from participants.

Step 6: If all participants from a cluster send ready a message, the cluster head sends a willing-to-commit message to the coordinator.

Step 7: If the coordinator receives a commit from all cluster heads, he performs a global commit.
Step 8: If a cluster head does not respond or send a rollback message, the coordinator checks the mobility speed and current location of all nodes.

Step 9: If a node is in the outlier, then the coordinator gets a direct message on commit or rollback.

Step 10: If all nodes including outlier nodes send commit messages, the coordinator performs a global commit.

Step 11: If a participant or cluster head sends a rollback message or does not respond up to the end of the timer, the coordinator issues an abort message.

Step 12: All participants perform a local abort when they receive an abort message from the coordinator.

Step 13: The coordinator closes the transactions by using an end transaction primitive.

This algorithm is an extension of the two-phase commit protocol using clustering and mobility management. Since the existing two-phase commit protocol fails most frequently when applied to mobile transactions, the challenge is addressed in this work by proposing this particular cluster-based transaction processing algorithm.

5.4 RESULTS AND DISCUSSION

Figure 5.1 shows the query processing cost analysis for mobile database queries between the proposed work ALDGPA and the existing work LDQPA by Vijayalakshmi et al. (2009).
From Figure 5.1, it can be observed that the cost of query processing is reduced much more in the proposed work than in the existing work proposed by Vijalakshmi et al. (2009), owing to the fact that the proposed work uses a cluster-based approach for communication. In both approaches the costs of queries are measured using input / output cost, network cost and mobility cost. Experiments have been conducted with different mobility speeds and the average values plotted in the graph.

Figure 5.2 shows transaction performance in terms of transaction processing time between the proposed model and the two-phase commit protocol by Sekar Ganesh et al. (2007). In this work, different numbers of transactions were executed on the proposed mobile cloud database systems and the existing mobile database system. In the existing work, storage was limited because of the limited memory provided in mobile devices. On the other hand, in the proposed work, additional memory and processors were added for transaction processing. However, the same transactions were executed in both scenarios and the results used for comparison.
From Figure 5.2, it is observed that the number of successful transactions is more in the proposed work than in existing one. Although, several transactions fail in a mobile environment, failures are reduced in the proposed work using clustering and communication. In both cases, a relaxed form of the two-phase commit protocol was utilized in which only consistency within clusters was considered. However, the proposed work performed better with the use of mobile cloud networks rather than simple mobile communication for distributed processing.

5.5 SPATIO - TEMPORAL QUERY PROCESSING ALGORITHMS

Several studies on location-dependent query processing are useful for mobile database systems. Among these, those by Ayse Y Seydim et al. (2001) and Vijayalakshmi et al. (2009) provided algorithms based on spatial constraints. In this work, a location-dependent query processing model is proposed by extending the two models above with temporal constraints, spatial rules, intelligent agents and heuristic search techniques.
5.5.1 Location-Dependent Query Processing

In this work, a new location-dependent query processing algorithm is proposed which uses a modified A* algorithm to perform a search to finding the optimal distance between nodes and the base station. The heuristic function proposed in this work considers spatio-temporal constraints in addition to the distance measure. The heuristic function is defined as follows:

\[ f'(n) = g(n) + h'(n) + t(n) + s(n) + v(n) \]  

(5.1)

Here, \( g(n) \) is the actual cost of getting the current node from the start node. \( h'(n) \) is the heuristic value of the cost of getting the destination or goal node from the current node. The values of \( t(n) \), \( s(n) \) and \( v(n) \) indicate the influence of temporal constraints, spatial constraints and velocity of movement. The Spatial constraints are used to find the direction of movement. Mobility speed can be varied when the user travels from one location to another. Future temporal logic is used to predict the location of the user at the time a query answer is given to the user.

In this algorithm, a set of intelligent agents comprising the Initiation Agent (IA), Goal Agent (GA), Coordinator Agent (CA), Heuristic Agent (HA) and Tracker Agent (TA), Decision Making Agent (DA), Temporal Reasoning Agent (TRA) and Spatial Agent (SA) are proposed to perform effective query processing. This model uses a cache memory to improve the performance of query processing. Whenever data is unavailable in the cache memory, the query is incrementally processed by separating the available and non-available in the cache. As a result data records are read from the database in blocks. Every system is connected with a base station using either a wired or wireless mode. Whenever a query is made, the location of the user is sensed and his security credentials checked with the home location from where his services can be validated.
When a query is given by the coordinator agent, initiation agents present in all nodes are given a message to initiate query processing. A timer is started by the base station before the query is processed. Tracker agents present in different nodes are responsible for tracking the current location of the user. The entire setup is modeled as a graph and the coordinator agent node called the start node. The start node initiates the search for service by providing a search message and possible goals. Using this information, participating nodes are informed of the possible goals, the route to be discovered, and query results are passed through the shortest path. For this purpose, the $f'$ value is computed by the heuristic agent and transferred to the decision-making agent. The latter calls the temporal reasoning and spatial agents to apply spatio-temporal constraints and arrive at final decision. The coordinator agent collects information from all participating nodes, consolidates the results and provides the answer to the query, based on the location.

In this model, the search starts with the agents present in the Mobile Unit (MU), the Home Location Base Station (HLBS) and the Next Base Station (NBS) for processing user queries. The base stations, including the home location base station are responsible for handling hand off operations because of their clients’ mobility. The CA initiates the query, checks the start time (ST) and the end time (ET) and employs them to find the value of $t(n)$, used in the equation to compute the value of $f(n)$. The temporal reasoning operators use coordinates $(mx,my)$ for mobile clients and $(bx,by)$ for the base station compute the distance, apply temporal constraints and find the velocity. This helps to monitor the user’s movement in addition to the sensing values and base station reports.
Algorithm for Searching the Cache

Step 1: Formulate a user query Q.
Step 2: Find the current location of the user.
Step 3: Find the shortest distance to the goal node.
Step 4: Check the cache available in the current node.
Step 5: If values are available, return the query answer.
        else
        Find the heuristic value and search the neighboring nodes.
Step 6: If an answer is available in a neighbor’s cache memory, return the result.
        else
        Send a message to the coordinator to retrieve it from the database available in the coordinator node.

Algorithm for Propagating Values

Step 1: Compute the g value.
Step 2: Compute the h’ value for the current node.
Step 3: Call the spatial agent and the temporal reasoning agent and apply spatio-temporal constraints.
Step 4: Compute the radius up to which the query answer is to be considered.
Step 5: Find all base stations within this range.
Step 6: Send a distributed query processing plan to perform database access
Step 7: Collect query answers from all participants.
Step 8: Find the optimum value using the heuristic function
Step 9: Use tracking agents to find the current location of the user
Step 10: If the user’s range is within the current location, provide the answer to the query.
else
Repeat steps 2 to 10 to find the next location.
Step 11: Provide the answer to the query.

Algorithm for Base Station Management

Step 1: Find the location of the user using the spatial agent
Step 2: If it is different from his home location get the mobile node to register with the current base station
Step 3: Make a connection between the home location agent and the current location where the base station is available
Step 4: Handover all the control packets to the current base station
Step 5: Check for user mobility
Step 6: Check whether there is a change in base stations.
Step 7: If there is a change in base stations, apply the handover procedure
else
Continue with the current base station
This algorithm has been developed in this work to provide a location-dependent query processing facility by using the intelligent agents proposed. Further, the system is developed in such that it can handle distributed query processing with mobility and hence the hand off procedure is taken care. This work uses the services of intelligent agents, such as the initiation agent to start the work and the coordinating agent to issue queries to participants, receive query answers from them and consolidate results before they are provided to the mobile client. The advantage of the proposed work is that it uses an extended A* algorithm to perform query processing. The combination of intelligent agents, rules, spatial and temporal constraints with distributed query processing techniques makes this much system more productive with respect to query processing for handling location-dependent queries.

5.6 PROACTIVE LOCATION-DEPENDENT QUERY PROCESSING

Location-Dependent queries can be processed with either proactive or reactive mechanisms. Proactive location-dependent query processing provides mobile users registered services through the proactive query processing mechanism. In this model, the user can provide a query once and his details are registered in the database. Therefore, whenever the service is made available, the user is advised via a mail or message from the database query answering mechanism that the service requested by him, temporarily unavailable earlier, is presently available. On the other hand, in a reactive location-dependent query processing model, the user must provide the query again and again to receive service. As a result, the user is required to send the query every single day to the server, and the coordinator agent in the server replies with whatever is available that day. This process is repeated until the
time the service is provided. In business models, proactive location-dependent query processing guarantees more success than its counterpart.

5.6.1 Intelligent Agent based Approach for Location based Services

In this work, a new intelligent agent-based approach is proposed for providing proactive query answers to facilitate location-based services. This is achieved by using a context-aware, location-based, proactive service-providing algorithm which uses intelligent agents to provide valuable services.

5.6.2 Proactive Location and Time Aware Service Provision Algorithm

In this work, a new proactive location-based service algorithm is proposed using intelligent agents, temporal constraints, spatial constraints, distributed query processing and handoff management techniques. This algorithm uses a mobile client who queries the base station where he is located. The query is forwarded by the base station to the next, and thereafter to other and, finally, to a node where the service is stored. The path from the source node called the coordinating node to the service provider node-called the goal node is obtained using the modified A* algorithm proposed in this thesis. The algorithm has been developed by extending the communication using agents and providing facility for using control packets. The algorithm is explained step-wise in this section for each phase.

Algorithm for Client and Coordinator side Processing

Step 1: The client sends a query processing request to the base station to which the client is attached.

Step 2: The base station replies with a yes / no flag.
Step 3: If the flag value is equal to yes then

   Perform next step

else

   Return "fail"

Step 4: Accept the user query

Step 5: Call the initiation agent

Step 6: Invoke all intelligent agents

Step 7: Submit the query to the coordinator agent

Step 8: If the query answer is available in the cache then provide reply

   else

   Call the decision agent, tracking agent and heuristic agent

Step 9: Initiate the search process by calling the query processing algorithm.

Step 10: Display query results.

**Algorithm for Base Station Coordination**

Step 1: Start a timer

Step 2: Create mobile agents for communication

Step 3: Send mobile agents to all neighboring nodes with the current query and provide them power to move to other base stations.

Step 4: At each base station, collect information using query constraints

Step 5: Find the goal node
Step 6: If the goal node is found stop propagation

Step 7: Otherwise, propagate the information on the query to all neighboring base stations.

Step 8: Upon the expiry of the timer or query answering stop processing

Step 9: Start a new timer and go to step 2 if all clients are not serviced

Step 10: If all clients are serviced then make an entry in the query processing register and return.

**Algorithm for Base Station Movement**

Step 1: Start with the base station of the mobile client.

Step 2: If mobile clients home location is different then check the with home location.

Step 3: Transfer all the control packets necessary for movement to the next base station.

Step 4: Repeat steps 2 and 3 until the goal base station is reached.

Step 5: Collect replies from all base stations and send them to the mobile client’s base station.

**5.6.3 Query Processing Challenges**

In this work, query processing challenges with respect to mobility, caching and distributed storage management are addressed using temporal and spatial constraints. A graph structure is developed for representing nodes participating in query processing. A distributed catalogue is used in this work to provide distances and heuristic information to the query processor. The
shortest route from the start node to the goal node is computed by extending the A* algorithm with spatial constraints, mobility and temporal constraints.

5.7 SIMULATION AND RESULTS

This work has been simulated using a JAVA simulator and the query processing algorithms were executed using JSim simulations. The MySQL Database is used as a backend for the Java programming environment. In this model, varying numbers of nodes - 20, 40, 60, 80 and 100 - were used in the simulation, and mobility speed was varied from 10 m/s to 50 m/s. The results obtained are shown in this section. Table 5.1 shows the simulation parameters.

Table 5.1 lists the primary simulation parameters.

**Table 5.1 Description of Simulation Parameters**

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum no of mobile users</td>
<td>100</td>
</tr>
<tr>
<td>Base Stations with Servers</td>
<td>10</td>
</tr>
<tr>
<td>Number of nodes in each base station</td>
<td>10</td>
</tr>
<tr>
<td>Cache size</td>
<td>512 KB</td>
</tr>
<tr>
<td>Local Database size</td>
<td>100000 records</td>
</tr>
<tr>
<td>Record size</td>
<td>1024 bits</td>
</tr>
<tr>
<td>Handoff probability</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Figure 5.3 shows the response time analysis for responses obtained on query processing with and without cache memory in a low mobility scenario.
From Figure 5.3, it is observed that the use of cache memory reduces query response time when compared with a system without cache memory. Moreover, the results shown in the graph are considered for a mobility speed of 5 m/s ensuring low node mobility.

Figure 5.4 shows the response time analysis for processing queries obtained from mobile clients, processed by nodes and servers with and without cache memory in a high-mobility scenario.
From Figure 5.4, it is observed that the use of cache memory in nodes and servers reduces query processing time when compared with a system consisting of nodes and servers without cache memory. Also, the results shown in this graph are considered for a mobility speed of 100 m/s, confirming that the high mobility of nodes relatively downsizes the performance of query processing, yet this challenge is addressed successfully in this work using spatio-temporal constraints.

Figure 5.5 shows the query response time analysis for processing location dependent queries obtained from mobile clients, processed by nodes and servers with and without agents in a low-mobility scenario.

![Query Processing Cost Analysis-1](image)

**Figure 5.5 Query Processing Cost Analysis with Low Mobility**

From Figure 5.5, it is observed that the use of agents reduces query response time when compared with a system without agents for processing queries. Further, the results shown in this graph are considered for a mobility speed of 5 m/s, ensuring that low-mobility nodes perform better than high-mobility ones.
Figure 5.6 shows the response time analysis for processing queries obtained from mobile clients, processed by nodes and servers with and without agents in a high-mobility scenario. Here, experiments were conducted using spatial temporal databases with mobility and, additionally, with mobile database systems using the intelligent agents proposed in this work.

From Figure 5.6, it is observed that the use of intelligent agents reduces query processing costs when compared to a system without intelligent agents. Moreover, the results shown in this graph are considered for a mobility speed of 100 m/s. confirming that a node's high mobility diminishes performance slightly, though the use of intelligent agents handles this challenge better.

Figure 5.7 shows the query-answered relevancy analysis for queries provided by mobile clients. Here, queries provided by the client are "find me the cheapest, nearest hotel" and "find the best hotel in my location." Based on these queries, results were obtained and average values shown in the graph.
Figure 5.7 Query Result Relevancy Analysis Using Spatial Constraints

From Figure 5.7, it is observed that the use of spatial constraints reduces query processing costs when compared with a system which does not use spatial constraints. In both cases, spatial indexing techniques were used to store data in the mobile database. However, the application of spatial constraints provided more relevant answers than systems without spatial constraints.

Figure 5.8 shows the query-answered relevancy analysis for queries provided by mobile clients when spatial constraints alone are applied, and when a combination of spatial and temporal constraints is applied.

From Figure 5.8, it is observed that the use of spatial constraints along with temporal constraints increases the accuracy of query relevance when compared with a system using only spatial constraints.
Figure 5.8 Query Result Relevancy Analysis Using Spatio-Temporal Constraints

It is concluded, from Figure 5.8, that spatial constraints are important to process location-dependent queries. However, they are not adequate enough to provide a user the most relevant services. Hence, a combination of spatial and temporal constraints is necessary to increase the relevancy of answers provided to the user by the location-dependent query processor. In all the experiments, heuristic search techniques using an extended form of the A* algorithm are used to locate databases and objects present in them. In all cases, it is assumed that R-tree indexing, in combination with a time-segmented tree-indexing technique, is used in all databases distributed and stored in different servers and nodes present in mobile database systems. The cost of a query is measured in terms of input / output, network, and mobility costs, as well as those incurred for processing constraints. The various experiments conducted for evaluating the proposed algorithm which shows that the query processing cost is optimized and relevancy increased. One of the challenges addressed in this thesis is the provision of better performance, even for a high mobility scenario. Therefore, the proposed location-dependent query processing techniques provide mobile
users more efficacious, proactive services. The hand off mechanism is also addressed by the query processing algorithms proposed in this thesis.

5.8 CONCLUSION

In this thesis, new algorithms are proposed for location-dependent query processing. To this end, a new location-dependent query processing algorithm using intelligent agents for processing distributed and mobile queries is proposed. The algorithm takes care of distributed coordination, concurrency control and recovery mechanisms better. Further, a new transaction model is proposed in this work for productive transaction processing in a mobile database environment. The algorithm uses the services of the intelligent agents proposed in this thesis to carrying out transactions successfully. Finally, a location-dependent query processing model which provides proactive services resourcefully is proposed. This has been accomplished by integrating intelligent agents with a heuristic search by extending the A* algorithm and providing spatio-temporal constraints for query processing as well. The major contributions of this thesis are increase in relevancy, reduction in query processing time and proactive mobile services.

The next chapter discusses new techniques proposed for feature selection and classification of data items stored in mobile cloud databases. Such a classification system is useful to perform analyses on data retrieved using queries, including location-dependent ones.