Mobile Information Systems 1 (2005) 225–252
IOS Press

Research in mobile database query optimization and processing

Agustinus Borgy Waluyo[a,*], Bala Srinivasan[a] and David Taniar[b]
[a]*School of Computer Science and Software Engineering, Monash University, Australia*
E-mail: {Agustinus.Borgy.Waluyo, Bala.Srinivasan}@infotech.monash.edu.au
[b]*School of Business Systems, Monash University, Australia*
E-mail: David.Taniar@infotech.monash.edu.au

Abstract. The emergence of mobile computing provides the ability to access information at any time and place. However, as mobile computing environments have inherent factors like power, storage, asymmetric communication cost, and bandwidth limitations, efficient query processing and minimum query response time are definitely of great interest. This survey groups a variety of query optimization and processing mechanisms in mobile databases into two main categories, namely: (*i*) query processing strategy, and (*ii*) caching management strategy. Query processing includes both pull and push operations (broadcast mechanisms). We further classify push operation into on-demand broadcast and periodic broadcast. Push operation (on-demand broadcast) relates to designing techniques that enable the server to accommodate multiple requests so that the request can be processed efficiently. Push operation (periodic broadcast) corresponds to data dissemination strategies. In this scheme, several techniques to improve the query performance by broadcasting data to a population of mobile users are described. A caching management strategy defines a number of methods for maintaining cached data items in clients' local storage. This strategy considers critical caching issues such as caching granularity, caching coherence strategy and caching replacement policy. Finally, this survey concludes with several open issues relating to mobile query optimization and processing strategy.

Keywords: Mobile database, mobile computing, query optimisation and processing, data dissemination, data broadcasting, data caching, wireless database information retrieval

1. Introduction

The development of wireless technology has led to mobile computing, a new era in data communication and processing [6,50,74]. With this technology, people can now access information any time and anywhere using a portable size wireless computer powered by battery (e.g. [21,65]. These portable computers communicate with a central stationary server via a wireless channel. Mobile computing provides *database applications* with useful aspects of wireless technology, and a subset of mobile computing that focuses on query to a central database server is referred to as mobile databases [6,49]. Examples of such applications include weather information or weather forecast services, news, stock quotes, and airline schedules.

Although this new technology has a very promising market, mobile database is facing a number of resource constraints, which include severe power and storage restriction; frequency of disconnection that is much greater than in a traditional network; bandwidth capacity; and asymmetric communications

*Corresponding author.

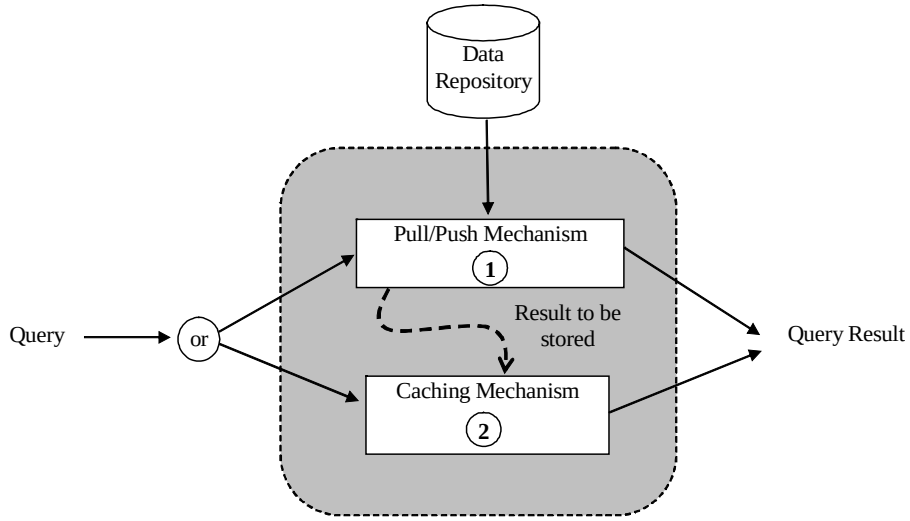


Fig. 1. Query Processing in Mobile Databases.

costs. These constraints are not a consideration in traditional wired database technology. Thus, query processing in a mobile environment requires entirely different approaches.

The fact that wireless radio transmission usually requires a greater amount of power compared with the reception operation [61,76], and the life expectancy of a battery (e.g. nickel-cadmium, lithium ion) was estimated to increase time of effective use by only another 15% [53], the need to use power efficiently and effectively is a crucial issue. As most applications in a mobile environment involve read operation rather than write operation [27], query optimization in mobile databases is definitely of great interest. It is the aim of this paper to present a comprehensive survey of mobile database query optimization and processing strategies.

In mobile databases, query can be served using two different mechanisms namely: (i) the pull/push mechanism (or also called pull/push operation), and (ii) the caching mechanism, which is illustrated in Fig. 1. The data items are obtained from the data repository at the remote central stationary server. Push mechanism is further classified into two schemes: On-demand broadcast and Periodic broadcast. Strategies in Push operation (on-demand broadcast) refer to the optimization method used at the server side to serve an on-demand request or a request that is sent to the server for processing. Periodic broadcast strategy relates to determining a method to disseminate the database item to mobile client so that the *response time* and *tuning time* of retrieving database items are minimized. Response time relates to the total elapsed time of waiting for the data of interest to arrive and the download time (Response Time $=$ Waiting Time $+$ Download Time). Whilst, tuning time is the amount of time a client must listen to the channel, which is used to indicate its energy consumption.

Caching mechanism is used to cache frequently accessed database items. This mechanism is expected to further enhance the query operation and assist mobile clients, especially when there is an unexpected situation such as server failure, channel distortion, and disconnection. The caching mechanism involves a variety of issues which require thorough investigations. A caching management strategy, which include caching granularity, caching replacement policy and caching coherence, has been proposed in order to maintain its effectiveness.

Query optimization and processing strategies in mobile databases are very much centered around these two most important mechanisms. Figure 2 corresponds to the detailed contents of this survey.

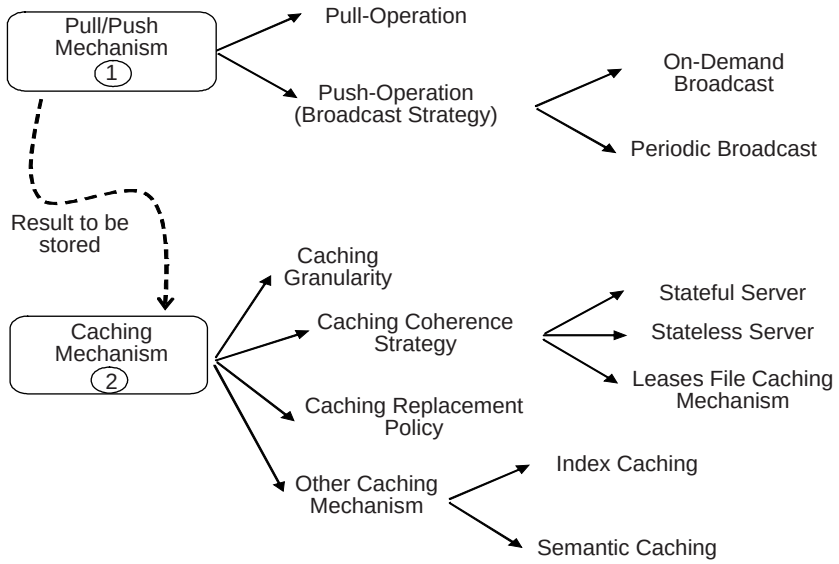


Fig. 2. Structure of the paper.

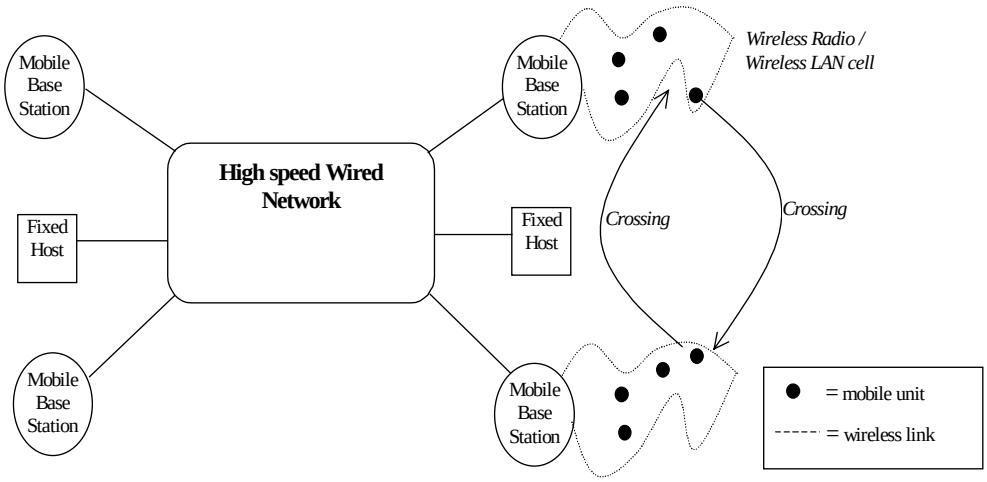


Fig. 3. Mobile Environment Architecture.

1.1. Background

In general, each mobile user communicates with a Mobile Base Station (MBS) in order to carry out any activities such as transaction and information retrieval. MBS has a wireless interface to establish communication with mobile client and it serves a large number of mobile users in a specific region called *cell*. In the mobile environment architecture, each MBS is connected to a fixed network as illustrated in Fig. 3.

Mobile clients can move between cells while being active and the intercell movement is known as a handoff process [29,59,63]. Each client in a cell can connect to the fixed network via wireless radio, wireless Local Area Network (LAN), wireless cellular, or satellite. Each of the wireless networks

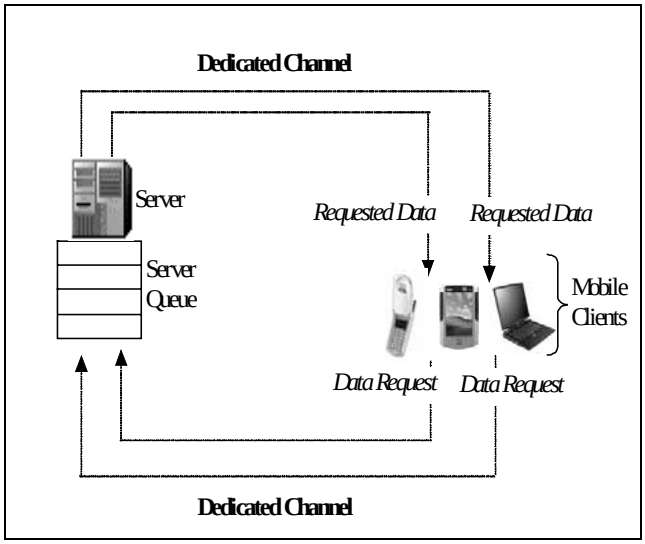


Fig. 4. The architecture of pull operation.

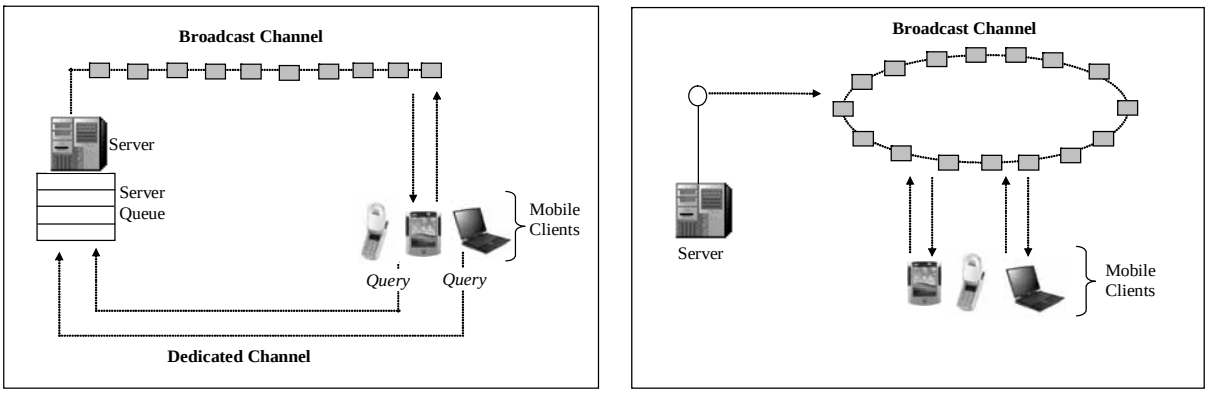
provides a different bandwidth capacity. However, this wireless bandwidth is too small compared with the fixed network such as ATM (Asynchronous Transfer Mode) that can provide speed up to 155 Mbps [16]. In this paper, the term data item corresponds to database record or tuples, and data segment contains a set of data items. It should be noted that a complete broadcast file is referred as a broadcast cycle. The terms mobile client, mobile computer, mobile unit, mobile user and client are used interchangeably, and the server refers to MBS.

2. Query processing

Query processing is classified into two categories: (i) pull operation and (ii) push operation. Push Operation is further classified into On-Demand Broadcast and Periodic Broadcast. Figure 4 illustrates pull operation, and Fig. 5 depicts push operation. The fundamental concept of push operation has been introduced for sometimes and applied in different projects, such as TeleText [5,71], and DataCycle [9]. These projects are concerned with a fast/fixed network rather than a wireless network. Push operation offers a number of advantages that can also be realised in a mobile environment [29]. As such, several new issues need to be addressed that require entirely different approaches as compared with wired network environment.

2.1. Pull operation

Pull operation relates to a traditional point-to-point scheduling scenario, which is similar to client-server interaction. As illustrated in Fig. 4, mobile clients establish a connection with the server via a dedicated link or channel. A request is sent to the server; when necessary, the request can be placed in a queue. The server picks the request, processes it, and sends the result back to the client via another dedicated channel. This scenario is not desirable in a mobile environment where resource preservation is of the utmost importance, as it satisfies only an individual client. This scenario will show its limitation



(a) On-Demand Broadcast

(b) Periodic Broadcast

Fig. 5. The architecture of push operation.

when it comes to large scale systems. Therefore, a more scalable and robust paradigm is of much interest. Push operation, which is described in the next sections, is designed to overcome this issue. Push operation is known as a scalable paradigm, where many clients can be satisfied simultaneously so that resource utilization is maximized [3,49,78]. Pull operation is more appropriate when privacy is of concern.

2.2. Push operation (*on-demand broadcast*)

Push based (On-Demand Broadcast) is where mobile clients send queries to the server through dedicated or point-to-point channel, and the client monitors to the broadcast channel to retrieve the query result. In this mechanism, the downlink channel (or broadcast channel) used to send the query result from the server is shared by all clients, which is the opposite of pull operation where the channel is reserved for an individual client. This operation is shown in Fig. 5(a).

In a wireless environment, the number of requests initiated by clients in a cell can be very large. Thus, it is necessary for the server to apply a certain technique in order to optimize the processing of the request in the server. This section studies different strategies that enable the server to optimize the processing of queries. Two types of algorithms are used in a wireless environment for this purpose, namely the push based and the disk scheduling algorithms. The push based scheduling algorithm is utilized to determine how queries are to be served efficiently in the server considering a number of factors such as the length of the query, the wait time and the popularity of the items. Disk scheduling is related to how data is placed on the disk that improves the query response time.

2.2.1. Push based and disk scheduling algorithms

Push based scheduling algorithms include $R \times W$ algorithm, *MAX*, and exhaustive $R \times W$ algorithm. The $R \times W$ scheduling algorithm is the product of the number of outstanding Requests (R) and the wait time (W) of the oldest outstanding request for all data items in the broadcast server queue [4]. The data item with the highest product value is chosen for broadcast. This mechanism provides a balance treatment between hot and cold items since the highest product value is derived from either the popularity of the data item, which creates a high R-value, or the longest outstanding request (W). *MAX* is a scheduling algorithm for data requests of different sizes [2]. It introduces an alternative to the response time of a

request metric or stretch of a request, which is considered as a good alternative for varying data size. Acharya and Muthukrishnan [2], also presented the criteria of optimizing the worst-case stretch of each request. The exhaustive $R \times W$ algorithm involves a clustering strategy to group a set of requests that require the same set of data items; the $R \times W$ algorithm is then applied [64]. However, the algorithm assumes a fixed length for all data items, and mobile clients need to continuously listen to the channel after the request has been sent to the server in order to eliminate the chance of transmission error.

A disk scheduling algorithm, such as the C-LOOK algorithm, processes each request in the cylinder position on the disk, which has been sorted previously in an ascending order [70].

The possibility of a joint algorithm comprising of the exhaustive $R \times W$ algorithm and the *C-LOOK* algorithm has also been investigated and analyzed in Triantafillou, Harpantidou and Paterakis [64]. This joint algorithm is classified into two schemes: one combines separate push based and disk scheduling algorithms; and the other amalgamates push based and disk scheduling algorithms into a single operation. The details of the joint algorithm are as follows:

a. **Combining separate push based and disk scheduling algorithms**
Two different algorithms under this category: (a) The AdoRe algorithm, and (b) The Flush algorithm. AdoRe algorithm incorporates the $R \times W$ scheduling algorithm and disk scheduling algorithm with designed mechanism, while the Flush algorithm employs a first come-first served basis followed by the *C-LOOK* algorithm.

– *The AdoRe algorithm: Active Disk on Requests*
This algorithm sends a number of requests (K) from the broadcast server queue to the disk scheduler queue for processing. This is done when the disk becomes idle. In a situation when the number of requests in the broadcast server is greater than K, the $R \times W$ algorithm is applied and a group of K requests with the highest $R \times W$ value is transmitted to the disk queue. The advantage of this algorithm is the ability to keep the disk system highly utilized while improving the efficiency. The efficiency is achieved by reducing the average disk service time based on forwarding a batch of request to be processed. For instance, a *C-LOOK* algorithm will take less time to serve a batch of requests like $K = 20$ than running the $R \times W$ algorithm 20 times to pick up one request at a time and forward it to the disk. In the case of the number of requests being less than K, the *AdoRe* algorithm picks one request after another to the disk scheduler queue.
– *The FLUSH algorithm*
This algorithm is not concerned with the number of requests in the broadcast server queue. All requests in the broadcast server are flushed to the disk server straight after the disk completes the processing of a single request. When there is more than one request, they are queued in the *C-LOOK* wait list. Subsequently, the *C-LOOK* algorithm is applied to process each request in the disk queue.

b. **Amalgamating push based and disk scheduling algorithms**
Two algorithms fall into this category namely, The Oweist algorithm, and the $R \times W/S$ algorithm.

– *The Oweist algorithm: Optimal Weighted Service Time*
As in the *AdoRe* algorithm, a parameter K is used for the formation of groups of requests. Whenever the disk becomes idle, K or fewer requests are picked out from the broadcast queue and transmitted to the disk queue. However, this algorithm does not consider the use of the $R \times W$ algorithm, but rather, it incorporates a new mechanism to form a group of K requests or less. The algorithm maintains a graph of K requests. The edge connecting two requests r_i and r_j is associated with $R_j \times S_j$. R_j corresponding to the number of requests in the broadcast

queue for item j, and S_j denotes the disk access cost to access item j. In this case, it is assumed that item i is the previous item retrieved from the disk queue. A number of factors are taken into account when calculating the disk access cost. These factors include the search time from the cylinder of item i to the cylinder of item j, the access item to find the desired item j once the disk head is on j's cylinder, and the time to retrieve item j from the disk. This algorithm calculates all possible permutations for the K requests and picks the optimal permutation that provides the smallest total weighted cost. It is noted that when K equals 1, *Oweist* is identical to *AdoRe* with the same K.

– *The RxW/S algorithm*
This algorithm combines the push based scheduling algorithm ($R \times W$) and disk service time (S). It is not concerned with grouping requests because requests from the broadcast server are sent to the disk server one at a time. When the disk is idle, the $R \times W/S$ algorithm is executed to find the request that obtains a highest value of the $\frac{R \times W}{S}$ to be transmitted to the disk server for processing.

A multiple query optimization scheme for on-demand broadcast has been investigated by Malladi [49]. In this scheme, the queries are processed in a group and common expressions among queries are analyzed. Subsequently, it allows the result to be broadcast only once. The mechanism of this scheme is as follows: mobile clients send query to the server; the server places the query from the clients in the queue; the requests are collected in a group based on a given query time window; the server process and evaluates each query according to common expressions; the server retrieves the data from the data repository; mobile client is sent information about when and how long to listen to the channel and obtain the result; the server broadcasts the result via the broadcast channel; and mobile client tunes into the channel at predefined time. It was claimed that this scheme would improve the query performance by reducing waiting time as well as bandwidth utilization.

The drawback of on-demand broadcast is the increase of bandwidth when the client sends the request to the server. The query performance is overwhelmed by multiple client requests. The congested channel and server queue may severely affect the query performance and power consumption of mobile clients. The periodic broadcast mechanism in the next category avoids the possibility of congested channel bandwidth and server queue, which is caused by an increase in the number of clients, and request arrival rate.

2.3. Push operation (*periodic broadcast*)

As illustrated in Fig. 5(b), *Push operation* (*periodic broadcast*) refers to periodically broadcast database items to an unbounded number of clients in the cell through one or more broadcast channels and allows a mobile client to capture and select data items whichever data it is interested in. Access to data is sequential as clients need to wait for the desired data to arrive on the broadcast channel. The behavior of the broadcast channel is unidirectional which means the server disseminates a set of data periodically to a multiple number of users. With this mechanism, the request in not known *a priori*, which is different from on-demand broadcast. Furthermore, this mechanism can be utilized to overcome the resource limitations in mobile databases. In this way, a mobile client is able to retrieve information without wasting power to transmit a request to the server. Other characteristics include scalability as it supports a large number of queries; query performance is not affected by the number of users in a cell as well as the request rate; and, it is effective to a high degree of overlap in user's request.

The main challenge in periodic broadcast is to minimize query response time and tuning time of retrieving database items. In some cases, the response time is equal to the tuning time. As the data

size increases over time, the required number of data items to be broadcast also increases accordingly. This situation may cause some mobile clients to wait for a substantial amount of time before receiving a desired data item. Consequently, the advantages of periodic broadcast strategy will be diminished. Alternatively, they can send a request via point-to-point channel or dedicated channel to the server (pull operation). In this case, the server processes the query and sends the result back to the client. This situation severely affects the query response time and power consumption of mobile clients, as it involves queuing and cannot scale over the capacity of the server or network.

We classify data dissemination schemes into two categories: one is *minimizing query response time*, and the other is *minimizing tuning time*. These schemes are designed while considering the inherent limitations in mobile computing environment.

2.3.1. Minimizing query response time

There are several periodic broadcasting schemes, which include: (*i*) Selection of Data Items to be Broadcast, (*ii*) Non-Uniform Frequency Distribution of Broadcast Data Items, (*iii*) Distribution of Data Items over Multiple Channels, and (*iv*) Organization of Data items. These schemes aim to minimize the query response time by either reducing the waiting time for the desired data to arrive or both waiting and download time in certain cases.

a. **Selection of Data Items to be Broadcast**
A selection mechanism needs to be designed to reduce the broadcast cycle length, which eventually reduces the query response time. During each broadcast cycle, additional items might be qualified as hot and some previously hot items might cool down, and therefore need to be replaced depending on the size of the cycle. A replacement algorithm is required to replace the cold data items with the new hot items. The server determines a set of appropriate database items to be broadcast, using information from queries received. Since hot items are those accessed by most clients, the access pattern of each client on the database items has to be derived from the clients back to the server. Finally, statistics will be compiled and the desired data items can be broadcast appropriately.
There are three replacement algorithms, namely *Mean Algorithm, Window Algorithm,and Exponentially Weighted Moving Average* (*EWMA*) *Algorithm* [45]. These algorithms maintain a score for each database item to estimate the access probability of the next broadcast cycle. The scores are defined by measuring the cumulative access frequencies of each database item over the length of the observation period.
Alternatively, a *log-time* algorithm that is designed based on the *packet fair queuing* can be used to determine a set of appropriate database items as well as the time to broadcast these items [22]. The log-time algorithm is also called O (log M) algorithm. The M of the *log-time* O (log M) algorithm denotes the number of database items in the server. Log M is used to find the average time required per iteration of the scheduling algorithm. This log-time algorithm can also be utilized to incorporate both single and multiple channels [22]. This is also useful to handle a certain factor like transmission errors on scheduling.
However, as the size of the database increases, the number of broadcast items may also increase accordingly. This situation will lead to an increase in response time. The next scheme can be used to improve the response time of a majority of requests by manipulating the frequency of hot items to be broadcast.

b. **Non-Uniform Frequency Distribution of Broadcast Data Items**
The different bandwidth capacity between the downstream communication and upstream communication has created a new environment called *Asymmetric Communication Environment*. In fact,

there are two situations that can lead to communication asymmetry [1]. One is due to the capability of physical devices. For example, servers have powerful broadcast transmitters, while mobile clients have little transmission capability. The other is due to the patterns of information flow in the application. For instance, in the situation where the number of servers is far fewer than the number of clients. It is asymmetric because there is not enough capacity to handle simultaneous requests from multiple clients.

Broadcast Disk is an information system architecture, which utilizes multiple disks of different sizes and speeds on the broadcast medium. This architecture is used to address the above situations [1]. The broadcast consists of chunks of data from different disks on the same broadcast channel. The chunks of each disk are evenly scattered. However, the chunks of the fast disks are broadcast more frequently than the chunks of the slow disks. This is the opposite of a *flat broadcast* where the expected delay before obtaining an item of interest is the same for all broadcasted items. With differing broadcast frequencies for different items, hot items can be broadcast more often than others. The server is assumed to have the indication of the clients' access patterns so that it can determine a broadcast strategy that will give priority to the hot items. It is designed so that the speeds of the disk can be adjusted to the configuration of the broadcast.

c. **Distribution of Data Items over Multiple Channels**

An alternative strategy to improve query response time is to distribute the broadcast data over more than one broadcast channel. In general, data items are broadcast over a single channel with underlying reason that it provides the same capacity (bit rate/bandwidth) as multiple channels, and it is used to avoid problems such data broadcast organization and allocation while having more than one channel [31]. Nevertheless, the use of single channel will show its limitation when there is a very large number of a data item to be broadcast. Figure 6 illustrates the situation when a broadcast cycle is split into two channels. Data items are broadcast repeatedly over a single channel, but none are broadcast over more than one channel.

The allocation of broadcast cycle into multiple numbers of broadcast channels will reduce the chance of long delay before obtaining the desired data items. A simple example, assume the size of the each data items are: 50 bytes. Consider a mobile client who wants to retrieve data item #8. The following two cases analyze the access time of mobile client when the data is broadcast over a single channel as compared to two channels. It is assumed that the client probes into a first data item in a channel.

Case 1: Data items (#1–10) are broadcast in a single channel
With this case, all data items are available in a single channel. Client who wants to retrieve data item #8, has to wait until the desired items are arrived. The total access time required will be $(50 \times 8) = 400$ bytes.

Case 2: Data items (#1–10) are split into two broadcast channels: channel one and channel two with 6 and 4 data items respectively. Since data items are split into two broadcast channel, the access time will also be different. In this case, client can switch to another channel and wait for the data items of interest to arrive. The total access time for client to retrieve data item #8 would be $(50 \times 2) = 100$ bytes. Thus, as compared to the first case, this time client can have a much more efficient data access.

In another case, multiple broadcast channels are necessary since multiple low bandwidth channels can not simply be merged into a single high bandwidth channel [54]. In light to this issue, Prabhakara et al. [54] presented a broadcast ordering scheme, where the hot items or the most

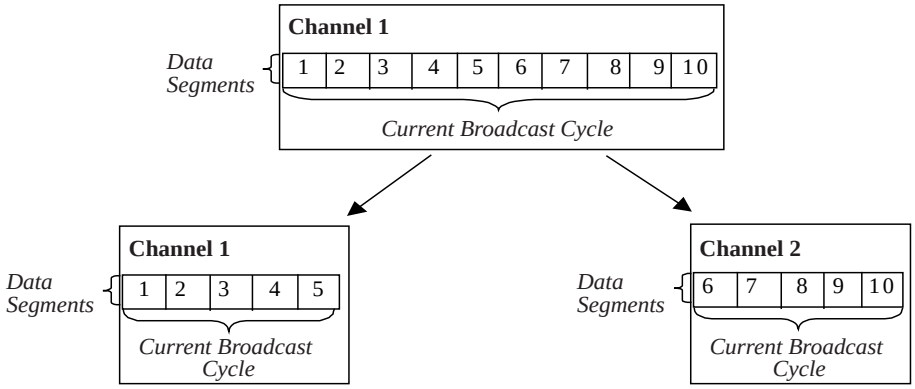
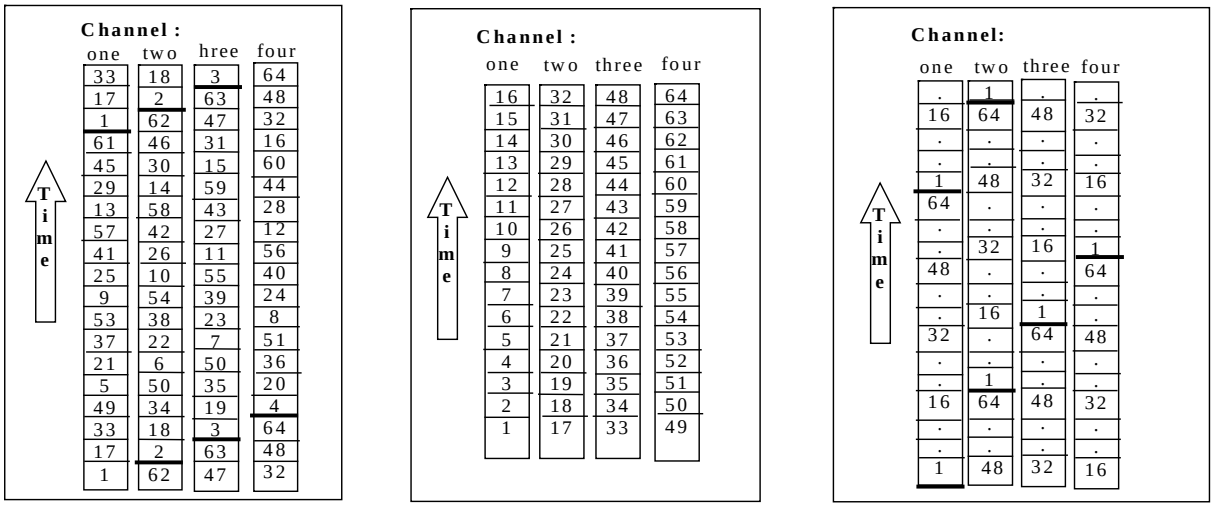


Fig. 6. Multiple Channels Architecture.



(a) Data Stripping Mechanism (b) Data Partitioning Mechanism (c) Data Replication Mechanism

Fig. 7. Data Stripping, Partitioning, and Replication Mechanism.

frequently accessed data items are broadcast more often than cold items. The hot items are allocated in the fastest bandwidth in decreasing order. The number of items in the channel is specified by given a temperature threshold value. Tran, at al. [62] enhanced the work in Prabhakara et al. [54] by considering different bandwidth posed by each channel and the size of each data items is varied. However, these approaches do not consider the relationship of one data item with the other. Therefore, these schemes are not effective for multiple data items retrieval. The next scheme describes the strategy to overcome this issue.

Another advantage of broadcasting the database items over multiple channels is to solve problems such as data distortion and data distribution [44]. The following four broadcasting mechanisms were proposed: Data Stripping, Data Partitioning, Data Replication, and Transformed Data Broadcasting Mechanisms. Figure 7 illustrates the first three mechanisms.

As shown in Fig. 7(a), a Data Stripping mechanism broadcasts consecutive data items over a multiple channel array and the data items are broadcast at certain intervals to let a client have sufficient time

to switch from one channel to another. The Data Partitioning mechanism in Fig. 7(b) allows the database to be partitioned into a number of data segments and each data segment is placed in a different channel. In Fig. 7(c), the database is replicated across all channels, and so is called a Data Replication mechanism.

However, the number of broadcast channels used in these cases is static. When there is a large number of broadcast items, this situation may cause some mobile clients to wait for a substantial amount of time before receiving the desired data item even with the help of a different pattern of data distribution over multiple channels.

This issue can be resolved using a strategy to determine the optimum number of database items to be broadcast in a channel [67]. In this strategy, the query response time over an on-demand channel is used as a threshold point. Once the optimum number is located, the number of broadcast channels should be increased. Subsequently, the length of the broadcast cycle is split and broadcast over multiple channels.

d. **Organization of Data Items**

The previous schemes are concerned with the retrieval of a single data item. However, in most cases multiple data items involve. A number of applications in this category includes a situation when mobile client wants to obtain more than one stock prices information at the same time (i.e. to list the stock price of car companies under General Motors corp.) or when client wants to retrieve weather forecast cities concurrently (i.e. to list the weather forecast of all cities in a certain region). To accommodate this type of request, we need to consider the relationship between one data item and the others. Organizing data items over the broadcast channel can be applied to reduce waiting time as well as download time. The basic idea of this scheme is to allocate related data items in such a way that the client does not have to wait for a substantial amount of time for the relevant data item. The organization of broadcast data is designed to match the query access pattern from mobile clients as closely as possible. As the query initiated by each client varies, this problem is sometimes considered an NP-hard problem [12,48].

The difficulty is to decide the broadcast order in advance, even without much knowledge of any future query. In general, an access graph is needed to identify the optimal organization of database items over a channel. The access graph is used to represent the dependency of data items. Once the access graph is built, a certain algorithm is utilized to determine the best broadcast program [39]. A cost model called the Semantic Ordering Model (SOM) can be used to determine the most efficient access graph [60]. SOM is defined into two models: namely, Simple and Extended SOM. Simple SOM considers only the relationship among entity types while Extended SOM incorporates historical query access patterns. Heuristics algorithm [28,42], randomized algorithm [8], and Genetic algorithm [23,24] are some algorithms that can be used to identify the most effective organization of broadcast data items. The final broadcast program can be distributed over either a single or multiple channels.

2.3.2. Minimizing tuning time

A broadcast indexing scheme is needed to reduce the tuning time by providing accurate information for a client to tune in at the appropriate time for the required data [37,39]. In this scheme, some form of directory is broadcast along with the data, the clients obtaining the index directory from the broadcast and using it in subsequent reads. The information generally also contains the exact time of the data to be broadcast. As a result, mobile clients are able to conserve energy by switching to "*doze*" mode and back to "*active mode*" when the data is about to be broadcast.

In general, the broadcast indexing technique causes a trade-off between optimizing the client tuning time and the query response time. The consequence of minimizing one of them is the increase of the other. For instance, to minimize the response time is to reduce the length of broadcast cycles. In this case, the index can be broadcast once in each cycle; however, it will make the tuning time suffer since the client will have to wait for the index to arrive which happens only once in each broadcast cycle. On the other hand, when the index directory is frequently broadcast in each broadcast cycle to reduce the tuning time, the response time will be greatly affected due to the occupancy of the index in the cycle. Thus, it is necessary to find the optimal balance between these two factors.

Broadcast indexing can be *tree-indexing* based, *signature* based, or *hybrid indexing* that is the combination thereof. The tree-index serves as a pointer to guide the client to the relevant index and each index contains the exact time of the data to be broadcast. Figure 8 illustrates an example of an index tree. Tree-indexing based on $B+$-tree structure such as one presented in [30] and Global Index model in [66, 68] can be utilised to minimise the access time and tuning time. With this strategy, mobile clients have access to selective tuning, and the Global Index guides the client to the right node that may be located in different index channel.

Clustering index, non-clustering index, and multiple index methods are used to determine the index organization over a broadcast channel [31]. Clustering index refers to clustering a record's attribute whenever attribute that belong to the records have the same value consecutively. Non-clustering index defines a method for indexing the non-clustering attributes by partitioning each non-clustered attribute in the broadcast cycle into a number of segments called *meta segments*. In multiple indexes, a second attribute is chosen to cluster the data items within the first clustered attribute.

The tree-indexing based can be broadcast by following two scheduling methods as presented in [31]. These index scheduling methods are; (1,m) indexing and distributed indexing.

- (*1*,m) *Indexing*
  (1,m) indexing is a method to place the index in the data broadcast. The index is broadcast m times in each broadcast cycle. Thus, the entire index directory is broadcast ahead of every fraction $\frac{1}{m}$ of the file. Figure 9 illustrates (1,m) index scheduling method.
- *Distributed Indexing*
  Compared to (1,m) indexing, distributed indexing increases the efficiency by partially replicating the index segments. The index segment is only valid for the subsequent data segment behind it. Only the relevant index is interleaved with data rather than interleaving the entire index with the data as occurs in (1,m) indexing. Figure 10 depicts a distributed index scheduling method.

Unlike the tree-indexing scheme, a signature-based index algorithm is derived by hashing the attribute values into bit strings followed by combining them together to form a bit vector or signature [41]. The signature is broadcast together with the data on every broadcast cycle. The mechanism is also applied to the query initiated by the client. To process the query, mobile clients need to tune into the broadcast channel and verify the query signature with the data signature by performing a certain mathematical operation like 'AND' operation. If the signature is not matched, the client can tune to the "doze" mode while waiting for the next signature to arrive. The main issue with this method is to determine the size of the signature as well as the number of levels of the signature. Three are three different signature-based index algorithms, namely *simple signature, integrated signature* and *multilevel signature* [41].

In simple signature, the signature frame is broadcast ahead of each data frame. This makes the total number of signature frames the same as data frames. An integrated signature is applied to a group of data frames and the signature is calculated accordingly. Figure 11 shows a combination of these two

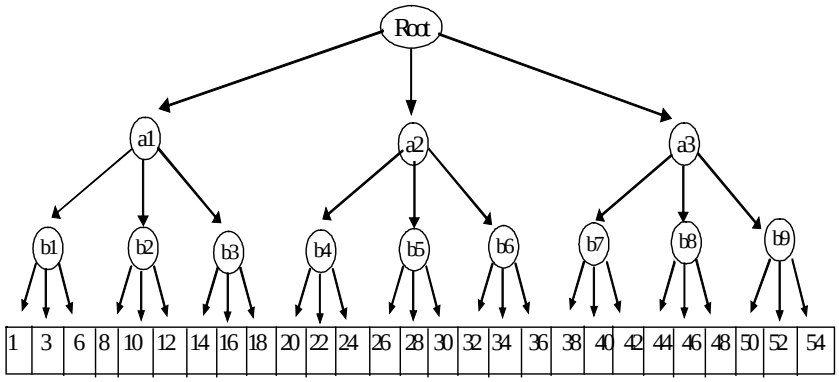


Fig. 8. A Full Index Tree.

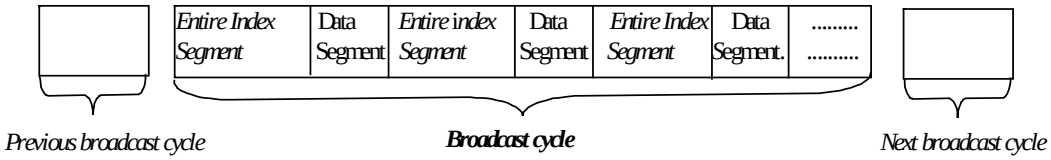


Fig. 9. (1,m) index scheduling.

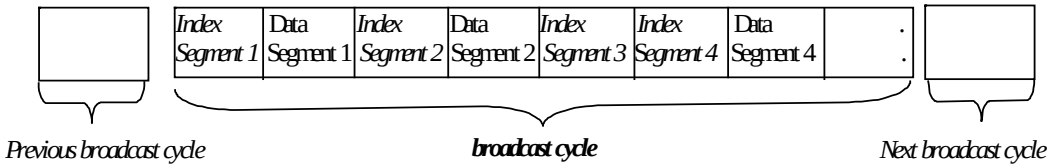


Fig. 10. Distributed index scheduling.

algorithms, forming a multilevel signature. This technique is designed to interleave with data items in a single channel.

A hybrid indexing technique made up of Index tree and Signature is expected to outperform the single indexing techniques by integrating the advantages of the two techniques into a single operation [26,40]. This technique is shown in Fig. 12.

2.4. Hybrid pull and push operation

The hybrid pull and push operation methods try to find the best channel allocation mode or the combination thereof to serve information requests. There are four possibilities of channel allocation methods namely *exclusive on-demand*, *exclusive broadcast*, *static hybrid*, and *dynamic hybrid* [25,26, 43]. The exclusive on-demand method assigns all channels to on-demand or point-to-point mode as opposed to the exclusive broadcast method which assigns all channels to broadcast mode. The static hybrid method allocates the channels into both on-demand mode and broadcast mode. Whilst, the dynamic hybrid method is an evolution of static hybrid, that means the number of channel allocated for on-demand mode and broadcast mode is dynamically changed depending on the server workload and query access patterns.

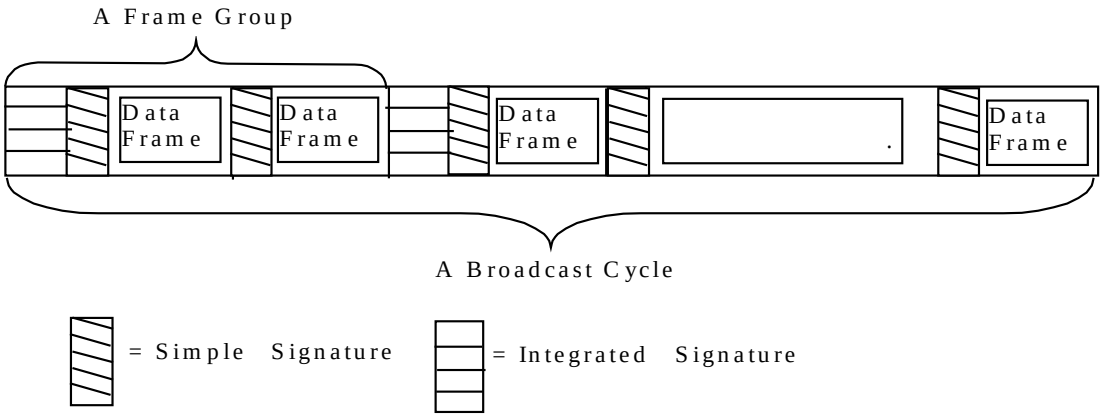


Fig. 11. Multi-level Signature.

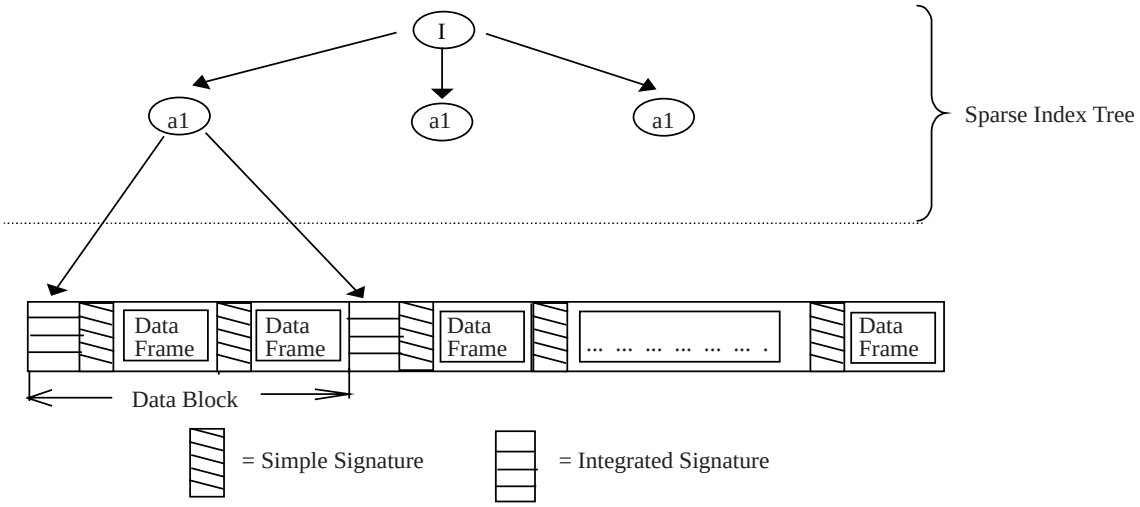


Fig. 12. Hybrid Indexing.

The broadcast channel and the on-demand channel cause different access overheads. The access overheads required for the broadcast channel are determined by the waiting time a client has to wait for the desired data to arrive plus the time to filter the relevant data. As with the on-demand channel, the access overhead is derived from the amount of time a client needs to connect to the server. The time is linearly dependent upon the load of the server. According to the concept of the broadcast channel, the access time will not be affected either by the rate of requests or the number of users in a cell. As opposed to the on demand channel which the performance is fully dependent on these two factors. The hybrid method is considered a better approach overall. Exclusive on-demand channels do not perform well with the variation of the number of channels.

Although this scheme seems promising, it requires a vigorous cost calculation since it introduces a high degree of complexity in the algorithm before it can decide the best channel allocation. Furthermore, the utilization of an on-demand channel forces the client to consume a large amount of energy especially after considering the asymmetric communication cost and limited power resources as indicated earlier.

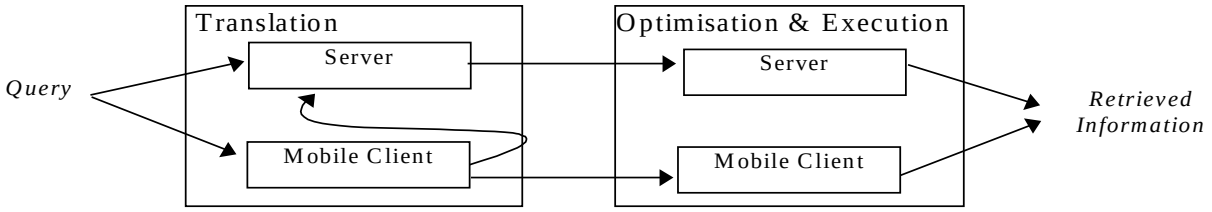


Fig. 13. Query Execution Sites.

2.5. Other strategies

Other strategies include the utilization of a certain technique to examine the most efficient plan to execute a query. Initially, the query is compiled into a sequence of candidate plans and the optimal one is taken from these candidate plans after considering the battery level of the client as well as the load of the server [18].

Search algorithms for determining a set of candidate plans include the use of optimization techniques such as partial order dynamic programming [19], and linear set algorithm and linear combinations algorithm [18]. Some factors are considered like the query cost model and optimization criterion to obtain the plans. The optimal plan corresponds to optimal energy utilization at a particular stage.

A similar strategy divides the query processing into three phases: translation, optimization and execution [36]. As shown in Fig. 13, the optimization and execution phases are combined into a single phase to avoid excessive communication between server and client. These phases can be executed on different sites, either at the server site or the client site depending on the resources situation. Three possible processing strategies can be applied when choosing the query execution site. The first alternative is to send the query to the server and the server processes the query all the way to the end. The second alternative is to do the translation phase at the client site to the point where the analysis indicates a high energy consumption for the next phase due to its complexity. The query is subsequently sent to the server for processing until all phases are completed and the data are retrieved. The last alternative is to do the query processing in the mobile unit from the start of processing to the end.

3. Caching management strategy

Data caching in mobile databases enables clients to obtain a high computing speed as server, by involving a much smaller volume of data items. A caching management strategy relates to how the client manipulates and maintains the data in its *cache* efficiently and effectively. In order to achieve this, a number of issues have to be considered and these issues require investigation.

A wireless communication channel in general suffers from narrow bandwidth while it is also inherently asymmetric communication, in which the downstream communication bandwidth from a server to the mobile client is much larger than the upstream communication bandwidth from clients back to server. Consequently, *caching* of frequently accessed data items in a client's local storage becomes important for enhancing the performance and data availability of data access queries. Another advantage of caching is the ability to handle fault tolerance. This is related to the characteristics of mobile computing in which each mobile client connects or disconnects from the network frequently. In some situation, the MBS may not be accessible due to problems such as signal distortion but a query can still be partially processed from caches and at least some of the query results can be returned to the user [20].

The mobile client generally performs caching of data at two different locations especially local disk and main memory. Local disk is used to cache data file and main memory is used to store the file or a portion of a file from its local disk [45].

The query processing mechanism determines the data item to be cached. As stated earlier, in order to obtain data items, one can either utilize pull mechanism or push mechanism:

– *Pull mechanism*
  This mechanism employs a conventional query processing methodology. The client initiates a query, which is subsequently sent to the server for processing via dedicated channels. The server then sends the outcome of the query back to the client. The inefficiency of this technique has been realized such as extensive response time especially when the wireless traffic is high, a waste of power consumption since a client has to keep tuned in while waiting for the query result, and a mobile client has to resend the query if it misses the result from the server due to some interferences like disconnection or transmission failure.
  Therefore, one way to overcome this problem is to cache the hottest data (i.e. data that are most frequently accessed). This technique is to improve the query response time, which subsequently reduces the power consumption and the query traffic to the server. A similar situation also applies to push mechanism (on-demand broadcast).
– *Push mechanism* (*periodic broadcast*)
  As the data items broadcast periodically, mobile clients store data or pages, in which the local probability of access is significantly greater than its frequency of broadcast [7]. The reason for caching data in this mechanism is the poor performance that a mobile client experiences when the server broadcast distribution is poorly matched with the client's page or data access order.

There are three main issues that characterize caching mechanism. These issues include (*i*) caching granularity, (*ii*) caching coherence strategy, and (*iii*) caching replacement policy.

3.1. Caching granularity

Caching granularity relates to determining a physical form of cached data items. The form of cached data item is determined by the type of database in which the data is stored. This can be either a relational database or an object-oriented database.

In a relational database, the cached data item can be in the form of *tuple-caching* or *page-caching*. There has been a conflicting discussion between *tuple-caching* and *page-caching* architecture. In page caching, the overhead for transmitting one item or a page is similar to that in a conventional client-server environment. Once a database item is retrieved from the server-disk, an additional overhead for transmitting the remaining database item residing on the same page will be small [17]. The problem with page caching is for the server to decide a physical organization that favors the locality of queries initiated by different mobile clients, rather than locality exhibited by one client as it might result in poor locality for another [11]. Furthermore, since each mobile client is powered by the battery's limited lifetime, the continuing tuning into the wireless channel in order to receive a page of database items for each initiated query will just consume too much energy, even with the help of an index [45]. Therefore, a smaller caching granularity is considered as a better alternative.

Three different levels of caching granularities in an object-oriented database are: *attribute caching*, *object caching* and *hybrid caching* [11]. Attribute caching refers to frequently accessed attributes of a database object in the client's local storage. Similarly, in object caching the object is cached rather than the attribute. The combination of both granularities is the hybrid caching, which caches only those attributes in frequently accessed objects and it provides a better approach to them.

3.2. Caching coherence strategy

Caching coherence strategy or invalidation strategy involves cache invalidation and update schemes to invalidate and update an out-dated cached item [10,11]. A cached data may no longer be valid for a certain period, thus a mobile client should obtain a new version of the cache before the data is retrieved. Several strategies have been proposed to overcome this issue. These strategies fall into three categories: (*i*) Stateful server, (*ii*) Stateless server, and (*iii*) Leases File Caching Mechanism.

a. **Stateful server**
The server has an obligation to its clients [7]. For example, the server will notify a client about changes to item x. The format of notification may vary which can be in the form of individual messages or a refresh message (with the item value) or a broadcast report. Also, the notification could deliver two types of information; one is the information of the status of a particular cache, which can be determined using a known algorithm, and the other is the information about a particular item.
This strategy requires the client to register its presence and the server keeps the information of every cache in the cell. The impact of this strategy is that a server may no longer know which clients are currently located under its cell and which of them are in active mode in the case when its connection is disrupted by disconnection and mobility of clients. Since disconnected clients cannot be reached, when they do re-connect, they have to send a message to the server to synchronize the version of cached data items with the server. The invalid status of a cache will still be sent even though the client does not access a particular cache. This mechanism causes a waste of bandwidth as well as energy from the client. The Andrew File System [57] and Coda [58] in client-server architecture have been using this strategy.

b. **Stateless Server**
A stateless server refers to the server that is not aware of the state of the client's caches [7]. The server broadcasts a report *asynchronously* or *synchronously*, in which only the updated database items are included. In an asynchronous broadcast, the server broadcasts an invalidation message for given data item as soon as the item changes its value. This is opposed to synchronous broadcast, where the server periodically broadcasting invalidation report. Asynchronous broadcast can also include other data items and the timestamp of the most recent updates, which corresponds to an *invalidation report*. However, an asynchronous broadcast does not specify the time when the report will be broadcast, thereby giving no indication of the waiting time.
Since clients may have caches of different age, so these reports have to be well defined by giving a time window of reference for the updates or update's time stamps. There are at least three strategies to synchronously/periodically broadcast an invalidation report, which include: (1) Broadcasting Timestamps, (2) Amnesic Terminal and (3) Signatures.
Broadcasting Timestamps: In this strategy, the server broadcasts notification of data items that have changed in the last x seconds (x is a parameter) [7]. The invalidation report consists of the timestamps of the latest change of these items. Upon receiving notification, mobile clients can update the status of its cache or retain the status, or eliminate the item from the cache. When the item is not mentioned in the report, it means that no update has occurred; then the mobile client renews the timestamp in the cache according to the timestamp of the report. The cache is discarded when the reported timestamp is larger that the timestamp in the cache. In a situation where the reported timestamp is smaller than the cache, the status remains the same.

```
If (current report timestamp – last report timestamp received by mobile client) is > x
        Clean the whole cache
Else
        Determine whether any broadcast items are in the mobile client's cache
        Compare the broadcast timestamps with the timestamps in the cache
           If the timestamp in the cache is less than the broadcast time stamp
                   Eliminate the related data from the cache
           Else
                   Update the status of the cache according to the timestamp of the report
           End if
 End if

For each item
     If the item is available in the cache
          Retrieve the relevant value to answer the query
      Else
         Send the query to the server for processing then cache the query result for later use
     End if
End for
     Update the last report timestamp according to current report timestamp
```

Fig. 14. Broadcasting Timestamp Algorithm.

The server monitors each of the data items that have changed in the last x seconds and retain the timestamps. The updated data items are stored in a list to be broadcast. Mobile clients keep the timestamps of the cached data item, determine from the report whether the cached data has been updated, and compare the timestamps with the broadcast timestamps from the server. A list of cached items that have been used to answer a query at each interval is available in the mobile client. Other than that, mobile clients record the last time a notification report has been received. The time difference between the record and the current report timestamps has to be less than x seconds. When the difference is bigger than x, it means that mobile client has missed one or more reports. Thus, the whole cache is not valid and needs to be discarded. The algorithm in Fig. 14 illustrates the mechanism of this method.

Amnesic Terminal: In this method, the server broadcasts only the identifiers of the items that have changed since the last invalidation report, without the timestamp attachment that happens in the broadcast timestamp algorithm [7]. The mobile client compares the items in the cache with those in the report. If a cached item is reported, then the item is dropped from the cache or else, it considers the cached item valid. The client keeps a timestamp that indicates the timestamp of the last report received. If the gap between timestamp in the client and the current timestamp is larger than a specified report interval (x), it drops the entire cache and starts re-building its cache from scratch. This situation usually corresponds to the disconnected client.

The server monitors each of the data items that have changed since the last broadcast report. The identifiers of changed data items are stored in a list to be broadcast. From the report, mobile clients determine whether the cached item has been changed, and eliminate the cache if the cached item is on the list. A list of cached items that have been used to answer a query at each interval is stored in the mobile client. Also, mobile clients record the last time a notification report has been received in order to keep track of the report. If the time difference between the record and the current report is bigger than the specified interval, then the entire cache is cleaned because it may no longer be valid. The algorithm is shown in Fig. 15.

Signatures: Signatures behave as checksums computed over the value of items. It computes a signature *sig* (*i*) per item i in the database and a set combined signatures that are the Exclusive OR

```
If current report timestamp – last report timestamp received by mobile client is > x
        Clean the whole cache
Else
        Determine whether the current report contains any cached data item
            If the cached item is in the report
                Eliminate the related data from the cache
            End if
End if

For each item
    If the item is available in the cache
        Retrieve the relevant value to answer the query
    Else
        Send the query to the server for processing then cache the query result for later use
    End if
End for
    Update the last report timestamp according to current report timestamp
```

Fig. 15. Amnesic Terminal Algorithm.

of the individual checksums [7]. The server periodically broadcasts a set of combines' signatures in which the composition of the subsets of each combined signature is known and agreed to. Mobile clients compare these signatures with the ones broadcast by the server. An item is declared invalid if it has "too many" unmatching signatures, and is then regarded as being out of date. The consequence of this technique is the client may falsely diagnose some of the valid caches as invalid.

c. **Leases File Caching mechanism**
This mechanism, which is also called the *lazy invalidation approach* [38], assigns each mobile client to be responsible for invalidating its cached items. Each file in the local storage is associated with a pre-specified refresh time (x). The time specifies the duration within which the cached file could be regarded as valid in the client's local storage. The estimation depends on its update probability. If an object is updated frequently, then its refresh time will be shorter. The data is accessed only when required. When it is found to be invalid, the client will send a message to the server and the server will send the corresponding data back to the client together with its new refresh time (x').

The refresh time for an item can be updated dynamically whenever the item is sent to a client from the server, and this technique is called adaptive caching [11]. A *statistical estimation method* is used to estimate the refresh time. This does not require a client to keep tuned into the channel in order to invalidate its cache. If a particular cached item is never accessed anymore, its refresh time will never be renewed even after its refresh time is expired.

3.3. Caching replacement policy

A caching replacement policy is required in order to retain the frequently accessed database items in a client's local storage. The more effective a caching replacement policy is in keeping the frequently accessed items, the better a query will perform and the more queries could be served during disconnection situation. There is a large variety of caching replacement policies, and most policies utilize access probability as the primary factor to determine the data items to be replaced.

a. **Mean Interoperation arrival duration, Window and Exponentially Weighted Moving Average**
This scheme manipulates the so-called *a replacement score*, which indicates the prediction of access probability of each item [11]. With the Mean Interoperation arrival duration, the cached

item with the highest mean arrival duration is replaced. However, this scheme is not able to adapt well to variations in access patterns since it considers the entire access history. To have a statistical measure, a *window* scheme is employed. It associates each object with a *window* of size w, which corresponds to the access time of w's most recent operations. The highest mean arrival duration of the cached item within the window is discarded.

Unfortunately, the effectiveness of this window scheme depends on the window size. Another technique called *Exponentially Weighted Moving Average* is used to avoid the need for a moving window and to adapt quickly in access patterns by assigning a weight to each arrival duration, which is defined as a parameter (α). The parameter is ranged from 0 to 1 with current duration as 1, previous duration as α, and the second previous duration as α^2. The recent duration will have a higher weight and the weights decreased linearly as the duration gets older.

b. **Cost-based page replacement**

This strategy follows page-caching architecture [1]. The cache is replaced if it is having the lowest ratio between its probability of access (P) and its frequency of broadcast (X). This ratio (P/X) corresponds to PIX (P Inverse X). An example of PIX, considering two pages in which one page is accessed 3 percent of the time at a particular client and is broadcast 3 percent of the time. The other page is accessed 1.5 percent of the time but is broadcast 0.5 percent of the time. In this example, the first page obtains PIX equivalent to 1 and therefore it should be replaced instead of the second page with PIX ratio of 3, although the first page is accessed twice as often.

c. **Worst, Least Recently Used (LRU), LRU-k, Most Recently Used (MRU), Least Reference Density (LRD), Clock.**

The cache replacement policies in this section consider page caching architecture. The algorithm "*Worst*" is designed to find out the page in the cache that is physically the farthest away from the current requested page [15]. Since this page only increases the buffer fault rate, the page will be replaced with another page that has the shortest forward distance.

A *Least Recently Used* (LRU) algorithm is used to replace a page in the buffer, which is considered as the oldest page for being referenced [15].

LRU has been extended into LRU-k. LRU-k records the times of a page in the last k references [51]. This algorithm is capable of determining the frequently and infrequently used page. Furthermore, the ability to monitor the recent and past reference frequency of a page, has made LRU-k a better approach. For example, LRU-2 means the last two references of a page and the page that has a longer reference inter-arrival time is selected for elimination from the buffer.

The *Most Recently Used* (MRU) strategy sometimes is used instead of LRU [35]. MRU is different from LRU in that it replaces the most recently used page rather than the least recently used one. This happens in a situation that involves a number of sequential steps to be performed on an operation and each step results in a predictable reference page. The referenced page will not be re-referenced until all the steps have been performed. Thus, in order to free some spaces, the most recently used page is replaced with the least recently used page, as it will be referenced in the near future.

The *Least Reference Density* (LRD) algorithm is another type of replacement policy. This algorithm replaces the page which has the lowest probability of being referenced [15]. Generally, the age measurement of this algorithm starts from the first time the page is being referenced.

The *Clock* algorithm uses bit to every page to monitor the reference of the page [15]. This enables the Clock algorithm to replace a page that contains 0-bit. Each page that is referenced by the selection pointer in current processing will be allocated to 1 bit. The stepwise examination, which behaves like a tick of the clock, will check the bit of pages that are encountered. If the examination

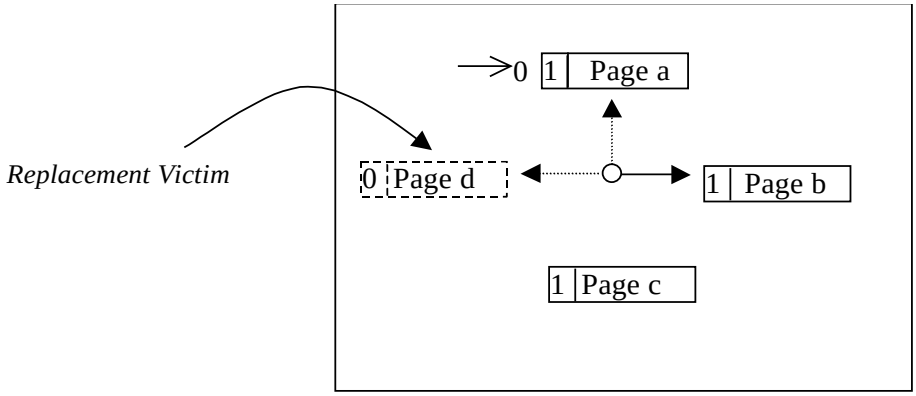


Fig. 16. Clock Algorithm.

encounters a 1-bit page, it will be reset to 0 and the first page with 0-bit will be replaced. Figure 16 illustrates how the Clock algorithm works.

d. **Stretch, Access rate, Inverse Update-frequency (SAIU) replacement policy**
Most of the previous policies use access probability as the primary factor to determine a cache replacement policy. The SAIU policy [72] employs what is called a "gain" replacement policy. The "gain" policy adds three additional factors to be considered which include *data retrieval delay, data update frequency*, and *data item size*. The SAIU policy calculates the "gain" of each data item and chooses the one with the most minimum "gain" value to be discarded from the buffer. This process continues until it gets enough space to accommodate a new cached item. The "gain" value is determined using the Eq. (1) as follows:

$$\text{gain}(i) = \frac{Li \times Pi}{si \times Ui}, \; i \text{ represents an item number} \tag{1}$$

The gain function is defined for each item i. It divides the product of data retrieval delay (L) and the data access probability (P), by the product of data item size (s) and update frequency of data item (U). If the equation is multiplied by the broadcast bandwidth (which is a constant B), the equation then becomes Eq. (2). It was claimed that this policy performs better as compared to LRU policy. However, this policy is only designed to accommodate push-mechanism (on-demand broadcast).

$$\text{gain}(i) = \frac{Li \times Pi}{si/B \times Ui} \tag{2}$$

3.4. Other caching mechanism

In this section, we describe other caching mechanisms used in a mobile computing environment such as, semantic query caching and index caching.

a. **Semantic Query Caching**
Semantic query caching stores the semantic descriptions and associated answers of previous queries in the mobile client. The use of semantic query caching in a client-server environment is similar to the data-caching scheme. The main reason for the use of semantic caching is due to its content-based reasoning ability. The fact that only the required data, which happens to be the contrary of a file or pages of data, is transmitted over the wireless channel is another reason [55,56]. Traditional

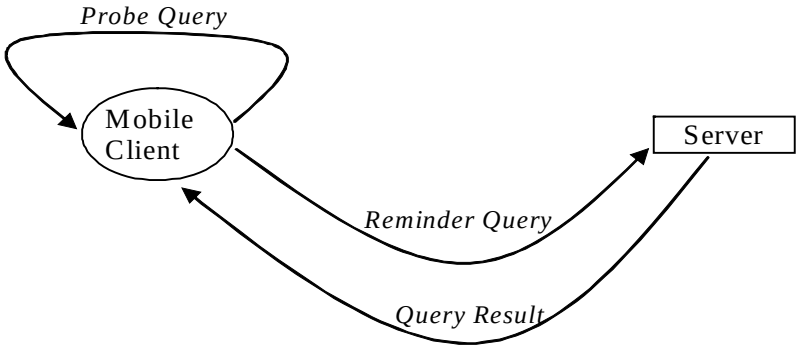


Fig. 17. Trimmed Query Mechanism.

caching schemes such as *page* and *tuple* caching do not maintain the semantic information about cached data, so it is hard to support content-based reasoning.

The advantage of using semantic query caching is the ability of the client to trim the query. The concept of trimmed queries is introduced by Godfrey and Gryz [20]. When a new query is specified, the mobile client can decide whether it can be totally answered from the cache, how much it can be answered and what data are missing. If a query can be partially answered by a semantic segment, it is split into two disjoint parts, namely, a *probe* query when it can be answered from the cache and a *remainder* query when the answer is missing and send the remainder query to the server [55]. This technique intensely reduces the wireless traffic since only the missing data are transferred. Figure 17 depicts the trimmed query mechanism.

The execution of a query is interpreted as a retrieval of a set of chunks, not as an individual tuple. To obtain the exact items, additional filtering on the retrieved chunk is necessary. Through reasoning with the semantics of a chunk, clients can deduce the precise nature of data items that are missing. The basic semantic cache model is presented by Keller and Basu [34]. To reduce the time to answer queries, a clustering technique of a semantic cache can be employed [55]. In this technique, the semantic cache is made of a set of semantic segments and each of the segments has different semantics description. The query results that are semantically related are grouped together and formed a cluster. The varying sized clusters can be simplified into fixed sized blocks defined in range of values, in which the query and the relation are modeled as multidimensional units called *chunks* [14].

The semantic locality can be utilized to predict the future use patterns for the segments by examining the semantic relationships with the current query. Two-Level Least recently used (LRU) on the clustering structure exploiting both semantic and temporal locality was designed as a cache replacement policy under this scheme [**?**]. Another semantic caching replacement policy is to measure the distance function of each semantic region. For each cached semantic region, a replacement value is assigned which is the negative of the "Manhattan distance" between the *center of gravity* of that region and the *center of gravity* of the most recent answered queries. Semantic regions that are semantically closer to the current query are less likely to be replaced [13].

b. **Index caching**

Index caching is another alternative to improve the retrieval of data items in the broadcast paradigm. The index generally contains information of the broadcast data organization. Since the size of index-data is probably small and relatively invariant, mobile clients could cache the index of any data item of interest in its local storage. Thus, the client needs only to listen to the channel when the

client first enters a wireless cell or when the cached index needs to be updated [60]. This technique should consider a few constraints; for instance, the data access patterns of mobile clients do not change, otherwise the server should be able to send a message to the client when the index needs to be updated. Index caching means that clients do not have to continuously listen to the channel. Consequently, the power consumption can be reduced. However, this solution has raised a number issues such as [31]:

- When a client leaves its cell and enters a new cell, the directory or index cached may need to be updated.
- New clients in the cell need to obtain an index directory from the broadcast channel. This includes mobile unit, which was turned off and turned on again.
- Since the broadcast data can change its content and longer or shorter in broadcast length between successive broadcast, the client will need to refresh its cache frequently to keep up with the data variations. This may generate excessive traffic between the clients and server.
- When a large number of different files are broadcast over many channels, clients need to be able to accommodate all the directories of all files being broadcast and this will create problems since mobile clients have a limited storage capacity.

4. Open issues

This section discusses some possible investigations that can be done in order to enhance the existing query optimization and processing techniques that have been outlined in this survey.

- *Adaptive Broadcast Channels*
  An adaptive broadcast channel is utilized in conjunction with a multiple broadcast channel mechanism. The distribution of data items over multiple channels with different distribution patterns enables mobile client to switch among channels and retrieve the desired data [44]. This technique not only helps mobile client to retrieve data item during signal distortion or channel failure, but also enhances the query response time.
  With an adaptive broadcast channel, the number of broadcast channels is set to be dynamic. This technique is effective in handling an ever-changing environment. For example; when there are a very large number of mobile users in a cell, the number of data items of interest (hot items) increase which subsequently increases the length of the broadcast cycle. This situation may cause some mobile clients to wait for a substantial amount of time before receiving the desired data item even with the help of different patterns of data distribution over multiple channels.
  An adaptive broadcast channel is expected to learn the environment and broadcast the data items over a dynamic number of channels. Consequently, when there is an enormous number of broadcast items, the number of channels can be adjusted accordingly. On the other hand, when the number of broadcast items is small, a dynamic broadcast channel will shrink the number of channels, so the resource utilization is preserved.
  An algorithm to find the optimal number of broadcast channels in a particular point in time depending of the number of users in a cell and query access patterns, needs to be investigated. The algorithm should be able to determine the best organization of data. Subsequently, the data is replicated over dynamic number of broadcast channel with possibly different data organization or different distribution schedule. This technique is expected to improve the response time for the client to retrieve the required data items.

– *Simultaneous data retrieval over multi channels*
One of the advantages of a data broadcasting scheme, which separates the index and the data items over multiple channels is that it will overcome the expected delay that occurs from a broadcasting index directory, since the frequency of broadcasting the index affects the query performance. However, in a normal situation, when more than one channel is involved, the client has to move between channels to obtain the consecutive data items. To prevent a situation where a client needs to move from one channel to another especially when it involves severals channel, a further investigation of a means to receive data simultaneously over multiple channels is necessary.
With a simultaneous data retrieval mechanism, a client can just tune into more than one channel and filter the required data items in a single operation. This situation reduces the delay of missing the data items of interest, and reduces the overhead of switching channels, which results in an improved response time. However, some evaluation needs to be conducted since the operation may cost high power consumption.
– *Hybrid method of broadcasting, caching and pull based request*
The combination of caching, broadcasting and on-demand methods can be an effective technique to enhance the query performance. The caching technique provides speedy data retrieval since the data is available in local storage. However, a mobile unit has a limited capacity to store data items. Thus, only a fraction of the desired data items can be cached. With the complement of a broadcasting technique, the caching problem is removed since clients can tune into the channel to pick up data of interest when the data is not available in the cache. Nevertheless, the quantity of data broadcast is changing based on the query access patterns. Therefore, the response time for clients' data retrieval may be severely affected.
As an alternative, mobile clients may send a direct request to the server when it is made aware of when the desired data items will appear in the broadcast channel. For example, a threshold as proposed in [43] can be employed to make a selection between broadcast and on-demand service. The indexing technique is used to indicate the time of the next desired data items to arrive in the broadcast channel; if the data item appears within the next threshold number of data frames, the client can choose to stay tuned in the channel; otherwise an on-demand service is required.
The possible drawback with this technique is power utilization. This technique enables the speeding up of data retrieval but at some point it may not be energy efficient since a number of checks may be needed before it can decide how the data is to be obtained. Thus, a thorough investigation is required.
– *Caching Management for Mobile Queries*
The caching strategies in this survey deal with traditional queries; that is, queries that we normally invoke in traditional database systems on a non-wireless communication network environment. The only difference is that now these queries are executed over a wireless communication network. However, as mobility is one of the properties of a wireless environment, queries on wireless database systems have much more variety, something which does not exist in traditional non-wireless databases, such as location dependent queries [46,52].
The existing works on caching of mobile data are not sufficient for these new nomadic types of queries. Thus, it is desirable to design caching management strategy that focus on "real" mobile queries – not traditional queries, and therefore, innovative mobile client strategies are clearly critical for mobile database applications [68]. Index caching to save memory caching can be employed, in order to improve query response time as well as utilizing space more efficiently, which is especially significant for location dependent queries. There is only a very few works have been done in this matter which include [75–77].

– *Broadcast Management for Mobile Queries*
In mobile queries where the mobile client's location is relevant to the information requested, or the information requested is based on a particular location, the utilization of a mechanism is very challenging [32,42]. Data item organization for nomadic users' queries will be much more complex, as the calculation may become obsolete as soon as the client moves. Data to be broadcast are also naturally changed, depending on the location of the clients. A mobile client should be informed when the relevant data will arrive in the channel while considering the current location of the client as the parameter.
Since mobile users may spread into many different regions, the central database server must be intelligent enough to organize its data so that mobile clients in different regions can still efficiently retrieve the relevant data; otherwise, mobile clients will waste a lot of unnecessary power during the tuning and listening processes. The problem is exacerbated when the mobile users frequently move from one region to another, requiring frequent tuning as well. Therefore, an efficient data structure for broadcast cycles is critical in saving mobile clients' battery power. So far only a limited number of work has been undertaken on this issue [33,73,74]. As such, this issue opens up many research problems, which it is definitely of great interest to pursue. Our work on identifying research issues with regard to location-dependent queries can be found in [69].

5. Conclusions

The inherent limitations and characteristics of mobile computing such as power, storage, asymmetric communication cost, and bandwidth, have become interesting challenges and have opened up research opportunities in the field.

This paper describes main issues, and a number of approaches in mobile database query optimization and processing that have been derived from literature. We classified these approaches into two schemes: one is a query processing strategy, and the other is a caching management strategy. Query processing includes pull operation and push operation. Push operation is divided into two categories: on-demand broadcast and periodic broadcast. A variety of methods have been explored, which is designed to improve query performance in mobile databases. Caching of frequently accessed data items in a client's local storage enables clients to obtain a high computing speed as server, by involving a much smaller volume of data items. Caching management strategy is used to deal with critical caching issues such as caching granularity, caching coherency, and caching replacement policy. Several strategies with regard to these issues have been outlined. Finally, we identified some open issues that require further investigation.

This paper serves as a valuable starting point as well as state-of-the-art for those who wish to learn about query optimization and processing in a mobile computing environment.

References

[1] S. Acharya, R. Alonso, M. Franklin and S. Zdonik, Broadcast Disks: Data Management for Asymmetric Communication Environments, *In Proceedings of ACM Sigmod* (1995), 199–210.
[2] S. Acharya and S. Muthukrishnan, *Scheduling On-Demand Broadcast New Metrics and Algorithms*, In Proceedings of ACM/IEEE International Conference on Mobile Computing, 1998, 43–54.
[3] D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang and S. Zdonik, *Research in Data Broadcast and Dissemination*, In Proceedings of 1st International Conference on Advanced Multimedia Content Processing, LNCS, 1999, 1554:194–207.

[4] D. Aksoy and M. Franklin, *Scheduling for Large-Scale On-Demand Data Broadcasting*, In Proceedings of IEEE InfoCom Conference, 1998, 651–659.
[5] M. Ammar and J. Wong, The Design of Teletext Broadcast Cycles, *Perform. Eval* **5**(4) (1985), 235–242.
[6] D. Barbara, Mobile Computing and Databases-A Survey, *IEEE Transactions on Knowledge and Data Engineering* **11**(1) (1999), 108–117.
[7] D. Barbara and T. Imielinski, *Sleepers and Workaholics: Caching Strategies in Mobile Environments* (*Extended Version*), In Proceedings of ACM Sigmod International Conference on Management of Data, 1994, 1–34.
[8] A. Bar-Noy, J. Naor and B. Schieber, *Pushing Dependent Data in Clients-Providers-Servers Systems*, In Proceedings of the 6th ACM/IEEE on Mobile Computing and Networking, 2000, 222–230.
[9] T.F. Bowen, G. Gopal, G. Herman, T. Hickey, K.C. Lee, W.H. Mansfield, J. Raitz and A. Weinrib, The Datacycle Architecture, *Communications of the ACM* **35**(12) (1992), 71–81.
[10] G. Cao, A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments, *IEEE Transactions on Knowledge and Data Engineering* **15**(5) (2003), 1251–1265.
[11] B.Y. Chan, A. Si and H.V. Leong, *Cache Management for Mobile Databases: Design and Evaluation*, In Proceedings of the International Conference on Data Engineering (ICDE), 1998, 54–63.
[12] Y.D. Chung and M.H. Kim, Effective Data Placement for Wireless Broadcast, *Distributed and Parallel Databases* **9**(2) (2001), 133–150.
[13] S. Dar, M. Franklin, B. Jonsson, D. Srivastava and M. Tab, *Semantic Data Caching and Replacement*, In Proceedings of 22nd International Conference on Very Large Data Bases, 1996, 330–341.
[14] P.M. Deshpande and K. Ramasamy, *Caching Multidimensional Queries Using Chunks*, In Proceedings of the ACM SIGMOD Conference on Management of Data, 1998, 259–270.
[15] W. Effelsberg and T. Haerder, Principles of Database Buffer Management, *ACM Transactions on Database Systems* **9**(4) (1984), 560–595.
[16] R. Elmasri and S.B. Navathe, *Fundamentals of Database Systems*, Fourth Edition, Addison Wesley, USA, 2003.
[17] M. Franklin, M. Carey and M. Livny, *Global Memory Management on Client-Server Architecture*, In Proceedings on International Conference on Very Large Databases, 1992, 596–609.
[18] S. Ganguly and R. Alonso, *Query Optimization in Mobile Environments*, In Proceedings of Fifth Workshop on Foundation of Models and Languages for Data and Objects, 1993, 1–17.
[19] S. Ganguly, W. Hasan and R. Krishnamurthy, *Query Optimization for Parallel Executions*, In Proceedings of the 1992 ACM SIGMOD, 1992, 9–18.
[20] P. Godfrey and J. Gryz, *Answering queries by semantic caches*, In Proceedings of Database and Expert Systems Applications (DEXA), 1999, 485–498.
[21] J. Häkkilä and J. Mäntyjärvi, Combining Location-Aware Mobile Phone Applications and Multimedia Messaging, *Journal of Mobile Multimedia* **1**(1) (2005), 18–32.
[22] S. Hameed and N.H. Vaidya, Efficient Algorithms for Scheduling Data Broadcast, *ACM/Baltzer journal of Wireless Network* **5**(3) (1999), 183–193.
[23] J.L. Huang and M.-S. Chen, *Dependent Data Broadcasting for Unordered Queries in a Multiple Channel Mobile Environment*, In Proceedings of the IEEE GLOBECOM, 2002, 972–976.
[24] J.L. Huang and M.-S. Chen, *Broadcast Program Generation for Unordered Queries with Data Replication*, In Proceedings of the 8th ACM Symposium on Applied Computing (SAC-03), 2003, 866–870.
[25] Q. Hu, D.L. Lee and W.C. Lee, *Optimal Channel Allocation for Data Dissemination in Mobile Computing Environments*, In Proceedings of 18th International Conference on Distributed Computing Systems, 1998, 480–487.
[26] Q. Hu, W.C. Lee and D.L. Lee, *Indexing Techniques for Wireless Data Broadcast under Data Clustering and Scheduling*, In Proceedings of the 8th ACM International Conference on Information and Knowledge Management, 1999, 351–358.
[27] Y. Huang, P. Sistla and O. Wolfson, *Data Replication for Mobile Computers*, In Proceedings of the ACM SIGMOD, 1994, 13–24.
[28] A.R. Hurson and Y. Jiao, *Data Broadcasting in Mobile Environment, Wireless Information Highways*, D. Katsaros, A. Nanopoulos and Y. Manolopaulos, eds, Chapter 4, IRM Press Pubslisher, London, 2005.
[29] T. Imielinski and B. Badrinath, Mobile Wireless Computing: Challenges in Data Management, *Communications of the ACM* **37**(10) (1994), 18–28.
[30] T. Imielinski, S. Viswanathan and B.R. Badrinath, *Energy Efficient Indexing on Air*, In Proceedings of the ACM Sigmod Conference, 1994, 25–36.
[31] T. Imielinski, S. Viswanathan and B.R. Badrinath, Data on Air: Organisation and Access, *IEEE Transactions on Knowledge and Data Engineering* **9**(3) (1997), 353–371.
[32] J. Jayaputera and D. Taniar, Data retrieval for location-dependent queries in a multi-cell wireless environment, *Mobile Information Systems* **1**(2) (2005), 91–108.

[33] K. Park, M. Song and C.-S. Hwang, *An Efficient Data Dissemination Schemes for Location Dependent Information Services*, In Proceedings of the First International Conference on Distributed Computing and Internet Technology (ICDCIT 2004), Vol. 3347, Springer-Verlag, 2004, 96–105.
[34] A.M. Keller and J. Basu, A Predicate-Based Caching Scheme for Client-Server Database Architectures, *The VLDB Journal* **5**(2) (1996), 35–47.
[35] H.F. Korth and A. Silberschatz, *Database System Concepts*, McGraw-Hill, New York, 1991.
[36] H.-E. Kottkamp and O. Zukunft, *Location-Aware Query Processing in Mobile Database Systems*, In Proceedings of ACM Symposium on Applied Computing, 1998, 416–423.
[37] K.C.K. Lee, H.V. Leong and A. Si, *A Semantic Broadcast Scheme for A Mobile Environment based on dynamic chunking*, In Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), 2000, 522–529.
[38] C.K.K. Lee, H.V. Leong and A. Si, *Adaptive Semantic Data Broadcast in a Mobile Environment*, In Proceedings of the 2001 ACM Symposium on Applied Computing, 2001, 393–400.
[39] K.C.K. Lee, H.V. Leong and A. Si, *Semantic Data Access in an Asymmetric Mobile Environment*, In Proceedings of the 3rd Mobile Data Management, 2002, 94–101.
[40] D.L. Lee, Q. Hu and W.C. Lee, *Indexing Techniques for Data Broadcast on Wireless Channels*, In Proceedings of the 5th Foundations of Data Organization, 1998, 175–182.
[41] W.C. Lee and D.L. Lee, Using Signature techniques for Information Filtering in Wireless and Mobile Environments, *Journal on Distributed and Parallel Databases* **4**(3) (1996), 205–227.
[42] G. Lee, S.-C. Lo and A.L.P. Chen, Data Allocation on Wireless Broadcast Channels for Efficient Query Processing, *IEEE Transactions on Computers* **51**(10) (2002), 1237–1252.
[43] W.C. Lee, Q. Hu and D.L. Lee, *Channel Allocation Methods for Data Dissemination in Mobile Computing Environments*, In Proceedings of the 6th IEEE High Performance Distributed Computing, 1997, 274–281.
[44] H.V. Leong and A. Si, *Data Broadcasting Strategies over Multiple Unreliable Wireless Channels*, In Proceedings of the 4th Information and Knowledge Management, 1995, 96–104.
[45] H.V. Leong and A. Si, Database Caching Over the Air-Storage, *The Computer Journal* **40**(7) (1997), 401–415.
[46] D.-L. Lee, M. Zhu and H. Hu, When location-based services meet databases, *Mobile Information Systems* **1**(2) (2005), 81–90.
[47] D.K. Lee, J. Xu, B. Zheng and W.-C. Lee, Data Management in Location-Dependent Information Services, *IEEE Pervasive Computing* **2**(3) (2002), 65–72.
[48] V. Liberatore, *Multicast Scheduling for List Requests*, In Proceedings of IEEE INFOCOM Conference, 2002, 1129–1137.
[49] R. Malladi and K.C. Davis, *Applying Multiple Query Optimization in Mobile Databases*, In Proceedings of the 36th Hawaii International Conference on System Sciences, 2002, 294–303.
[50] B.A. Myers and M. Beigl, Handheld Computing, *IEEE Computer Magazine* **36**(9) (2003), 27–29.
[51] E. O'Neil, P. O'Neil and G. Weikum, *The LRU-K Page Replacement Algorithm for Database Disk Buffering*, In Proceedings of the ACM SIGMOD, 1993, 297–306.
[52] M. Östergren and O. Juhlin, Road Talk: a Roadside Location-Dependent Audio Message System for Car Drivers, *Journal of Mobile Multimedia* **1**(1) (2005), 47–61.
[53] L.D. Paulson, Will Fuel Cells Replace Batteries in Mobile Devices? *IEEE Computer Magazine* **36**(11) (2003), 10–12.
[54] K. Prabhakara, K.A. Hua and J.-H. Oh, *Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment*, In Proceedings of the IEEE International Conference on Data Engineering (ICDE'00), 2000, 167–176.
[55] Q. Ren, M.H. Dunham and V. Kumar, Semantic Caching and Query Processing, *IEEE Transactions on Knowledge and Data Engineering* **15**(1) (2003), 192–210.
[56] Q. Ren and M.H. Dunham, *Using Semantic Caching to Manage Location Dependent Data in Mobile Computing*, In Proceedings of the 6'th International Conference on Mobile Computing and Networking, 2000, 210–221.
[57] M. Satyanarayanan, J.H. Howard, D.N. Nichols, R.N. Sidebotham, A.Z. Spector and M.J. West, *The ITC Distributed File System: Principles and Design*, In Proceedings on the 10'th ACM Symposium on Operating Systems Principles, 1985, 35–50.
[58] M. Satyanarayanan, J.J. Kistler, P. Kuar, M. Okasaki, E.H. Siegel and D.C. Steere, Coda: A Highly Available File System for a Distributed Workstation Environment, *IEEE Transactions on Computers* **39**(4) (1990), 447–459.
[59] K. Sato, M. Katsumoto and T. Miki, A New VOD Technique to Support Client Mobility, *Journal of Mobile Multimedia* **1**(3) (2005), 198–210.
[60] A. Si and H.V. Leong, Query Optimization for Broadcast Database, *Data and Knowledge Engineering* **29**(3) (1999), 351–380.
[61] M. Song, S.-W. Kang and K. Park, On the design of energy-efficient location tracking mechanism in location-aware computing, *Mobile Information Systems* **1**(2) (2005), 109–127.
[62] D.A. Tran, K.A. Hua and N. Jiang, *A Generalized Design for Broadcasting on Multiple Physical-Channel Air-Cache*, In Proceedings of the ACM SIGAPP Symposium on Applied Computing (SAC'01), 2001, 387–392.

[63] K.S. Trivedi, S. Dharmaraja and X. Ma, Analytic modelling of handoffs in wireless cellular networks, *Information Sciences* **148** (2002), 155–166.
[64] P. Triantafillou, R. Harpantidou and M. Paterakis, *High Performance Data Broadcasting: A Comprehensive Systems Perspective*, In Proceedings of the 2nd International Conference on Mobile Data Management (MDM 2001), 2001, 79–90.
[65] P.K.C. Tse, W.K. Lam, K.W. Ng and C. Chan, An Implementation of Location-Aware Multimedia Information Download to Mobile System, *Journal of Mobile Multimedia* **1**(1) (2005), 33–46.
[66] A.B. Waluyo, B. Srinivasan and D. Taniar, Optimising Query Access Time over Broadcast Channel in Mobile Databases, in: *Proceedings of the Embedded and Ubiquitous Computing, Lecture Notes in Computer Science*, (Vol. 3207), Springer-Verlag, 2004, pp. 439–449.
[67] A.B. Waluyo, B. Srinivasan and D. Taniar, A Taxonomy of Broadcast Indexing Schemes for Multi Channel Data Dissemination in Mobile Databases, in: *Proceedings of the 18th International Conference on Advanced Information Networking and Applications* (*AINA 2004*), (Vol. 1), IEEE Computer Society Press, 2004, pp. 213–218.
[68] A.B. Waluyo, B. Srinivasan and D. Taniar, Research on Location-Dependent Queries in Mobile Databases, *International Journal on Computer Systems: Science and Engineering* **20**(3) (2005), 77–93.
[69] A.B. Waluyo, B. Srinivasan and D. Taniar, Indexing Schemes for Multi Channel Data Broadcasting in Mobile Databases, *International Journal of Wireless and Mobile Computing* **1**(6) (2005).
[70] B.L. Worthington, G.R. Ganger and Y. Patt, *Scheduling Algorithms for Modern Disk Drivers*, In Proceedings of the ACM SIGMETRICS, 1994, 241–251.
[71] J.W. Wong, Broadcast Delivery, *In Proceedings of the IEEE* **76**(12) (1988), 1566–1577.
[72] J. Xu, Q. Hu, D.L. Lee and W.-C. Lee, *SAIU: An Efficient Cache Replacement Policy for Wireless On-demand Broadcasts*, In Proceedings of the 9th International Conference on Information and Knowledge Management, 2000, 46–53.
[73] J. Xu, X. Tang and D.L. Lee, Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments, *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*) **15**(2) (2003), 474–488.
[74] J. Xu, B. Zheng, W.-C. Lee and D.L. Lee, The D-tree: An Index Structure for Planar Point Queries in Location-Based Wireless Services, *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*) **16**(12) (2004), 1526–1542.
[75] J. Xu, B. Zheng, W-C. Lee and D.L. Lee, *Energy Efficient Index for Querying Location-Dependent Data in Mobile Broadcast Environments*, In Proceedings of the 19th IEEE Int. Conf. on Data Engineering (ICDE '03), 2003, 239–250.
[76] J. Xu, Q. Hu, W.-C. Lee and D.L. Lee, Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination, *IEEE Transaction on Knowledge and Data Engineering* (*TKDE*) **16**(1) (2004), 125–139.
[77] B. Zheng, J. Xu and D.L. Lee, Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments, *IEEE Transactions on Computers* **51**(10) (Oct. 2002), 1141–1153.
[78] E. Yajima, T. Hara, M. Tsukamoto and S. Nishio, Scheduling and Caching Strategies for Correlated Data in Push-based Information Systems, *ACM SIGAPP Applied Computing Review* **9**(1) (2001), 22–28.

Agustinus Borgy Waluyo is a Ph.D. candidate at School of Computer Science and Software Engineering (SCSSE), Monash University, Australia. Prior to joining SCSSE, he completed BBusSys (hons) in 2002, also from Monash University. His research interest includes query processing, data management, data broadcasting and caching management in mobile computing environment.

David Taniar has a Bachelor, Masters, and PhD – all in Computer Science and Information Technology. He is currently a Senior Lecturer at the Faculty of Information Technology, Monash University, Australia. He is a Fellow of the Institute for Management of Information Systems (FIMIS). He has been appointed as a founding editor-in-chief of a number of international journals, including Journal of Mobile Multimedia (Rinton Press, USA), and Mobile Information Systems (IOS Press, The Netherlands), and also serves as a member of the editorial board for more than ten international journals. Over the years, he has served as a program committee member of more than 40 international conferences. He is an active and productive researcher and publishes extensively every year. He has published more than 100 research articles. He also has authored three books and edited three books. A newly authored book on Object-Oriented Oracle (IRM Press, USA) has just been released (check the Amazon.com). A CV and complete list of publications are available at http://users.monash.edu.au/∼dtaniar/.

Bala Srinivasan is a Professor of Information Technology in the School of Computer Science and Software Engineering at the Faculty of Information Technology, Monash University, Melbourne, Australia. He was formerly an academic staff member of the Department of Computer Science and Information Systems at the National University of Singapore and the Indian Institute of Technology, Kanpur, India. He has authored and jointly edited 6 technical books and authored and co-authored more than 150 international refereed publications in journals and conferences in the areas of Multimedia Databases, Data Communications, Data Mining and Distributed Systems. He is a founding chairman of the Australasian database conference. He was awarded the Monash Vice-Chancellor medal for post-graduate supervision. He holds a Bachelor of Engineering Honours degree in Electronics and Communication Engineering, a Masters and a Ph.D degree, both in Computer Science.