

# Query Optimization in Resource Constrained Environment

M. Tech Project First Stage Report

Submitted in partial fulfillment of the requirements  
for the degree of

Master of Technology

by

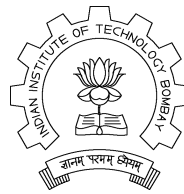
**Prajakta Kalekar**

**Roll No: 04329008**

under the guidance of

**Prof. Krithi Ramamritham**

**Prof. Sudarshan**



Kanwal Rekhi School of Information Technology Indian Institute of Technology,  
Bombay  
Mumbai

# Acknowledgment

I am extremely thankful to *Prof. Krithi Ramamritham* for his able guidance and constant encouragement.

I would also like to thank *Ashwini Rao*, *Aruna Adil* and *Jagrati Agarwal* for their insightful views on the topic.

Many thanks to *Aditee Badge* for invaluable feedback at every stage.

Prajakta Kalekar  
IIT Bombay

# Abstract

A seamless infrastructure for information access and data processing is the backbone for the successful development and deployment of the envisioned ubiquitous/mobile applications of the near future. The development of such an infrastructure is a challenge due to the *resource constrained nature* of the mobile devices, in terms of the *computational power, storage capacities, wireless connectivity and battery energy*.

This report discusses the factors to be taken into consideration for Query Optimization in Resource Constrained Environments - in particular memory and energy constrained environments. Models for prediction of energy consumption of queries on hand-helds have been discussed. Preliminary experiments were conducted to study the energy consumption of different queries in a hand-held. The results and insights gained are discussed.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Need for Query Optimization on hand-helds . . . . .	3
1.2	DELite . . . . .	4
1.3	Simputer . . . . .	4
1.4	Road-map . . . . .	4
<b>2</b>	<b>Query Optimization in Energy constrained environments</b>	<b>5</b>
2.1	Overview . . . . .	5
2.2	Literature Survey . . . . .	5
2.3	Work Partitioning between Server and Client for Query Optimization . . . . .	6
2.3.1	Future Work . . . . .	6
2.3.2	Application to Simputer/DELite . . . . .	7
2.4	Energy behaviour of memory resident data . . . . .	7
2.4.1	Future work . . . . .	7
2.4.2	Application to DELite . . . . .	7
2.5	Energy consumption prediction model . . . . .	7
2.5.1	With CPU computations and I/O as the units of energy consumption . . . . .	8
2.5.2	With hardware components as units of energy consumption . . . . .	8
2.5.3	Using query parameters for estimating the energy consumption . . . . .	9
2.6	Experimental Results . . . . .	9
2.6.1	Technique to measure the energy consumed by a query . . . . .	10
2.6.2	Assumptions . . . . .	10
2.6.3	Queries . . . . .	10
2.6.4	Results . . . . .	11
2.6.5	Observations and Conclusions . . . . .	11
2.6.6	Future Work . . . . .	11
2.7	Summary . . . . .	12
<b>3</b>	<b>Query Optimization in Memory Constrained Environments</b>	<b>13</b>
3.1	Overview . . . . .	13
3.2	Literature Survey . . . . .	13
3.3	Compression in a Memory Constrained Environment . . . . .	13
3.3.1	Motivation . . . . .	13
3.3.2	Issues in Compressed Databases . . . . .	14
3.3.3	Design of a Compressed Database . . . . .	15
3.4	Summary . . . . .	15
<b>4</b>	<b>Multi-Query Optimization</b>	<b>16</b>
4.1	Materialization in Multi-query Optimization . . . . .	16
4.1.1	Generation of a search space of the views to materialize . . . . .	16
4.1.2	Selection and Maintenance of the materialized views and Indexes . . . . .	16
4.2	Pipelining in Multi-query Optimization . . . . .	16



# Chapter 1

## Introduction

Computing is becoming a pervasive and ubiquitous part of everyday life. The traditional modus-operandi of sitting at a desk to interact with a computer system is gradually going out of style, with users demanding access to computational resources and information whenever and wherever they choose. Consequently, hand-held and mobile computing devices are much in demand, bringing into focus *resource-constrained computing*.

An example hand-held system we have used for experiments is the Simputer, developed by IISc, Bangalore and Encore Software (described in Section 1.3). An example DBMS we have used is DELite, developed at IITB (described in Section 1.2).

### 1.1 Need for Query Optimization on hand-helds

There is a mismatch between the kind of resources hand-helds are equipped with and the kind of queries they are expected to service. Though these devices are required to service queries that go well beyond simple Select-Project-Join queries, they have limited memory, computing power and communication band-width. A possible solution to this problem is to delegate the query execution to the server. But this imposes severe communication overheads. The communication costs, coupled with the unreliable connectivity motivates the execution of queries at the client-end itself.

While there has been extensive prior research on Query optimization in resource-rich environments, these traditional query optimization techniques (for disk-based DBMS) cannot directly be applied to resource constrained environments. This is due to the stark differences between the DBMS in the two environments:

1. The database in the hand-held device would be stored in flash memory. The characteristics of flash memory are very different from that of magnetic disks. Flash memory read time is very small compared to disk read time. It is at par with main memory read time. The write time, however is at par with that of disks.
2. Techniques used in disk based systems assume data to be primarily disk resident. The main optimization criterion for query optimization algorithms, buffer pool management, indexed retrieval techniques, transaction management and other techniques is minimizing number of disk accesses.

Since the techniques designed for high-end systems do not take into account the constrained environment of hand-helds, new techniques that are cognizant of the available constraints on resources need to be used devised.

## 1.2 DELite

DELite [SR05] is an Open Source Light Weight Database Management System developed by Department of Computer Science, IIT Bombay. It is developed for hand-held devices like Simputer. It uses Flash memory as read buffer and RAM as write buffer. Its query processing technique based on the memory allocation algorithms generates only fully pipelined, left deep tree query execution plans. The Index structures are generated in memory during query execution. Queries supported are Select, Project, Join, update and aggregate queries. DELite supports Pointer based storage models like ID-based and Domain based.

## 1.3 Simputer

The Simputer [Sim], developed by IISc, Bangalore and Encore Software. It has 32-bit Intel-Strong Arm processor running at 206MHz, 16-64 MB SDRAM, 08-32 MB Flash Memory for permanent storage, a 320 X 240 monochrome or LCD display panel, a stylus for input and Linux operating system. It has support for wireless communication, Internet access and smart cards. Its cost ranges from Rs 10000 to Rs 20000. HP iPaq and PalmOne are other examples.

## 1.4 Road-map

Chapter 2 starts off by stressing on the need for Query Optimization in an energy constrained environment, followed by details of the Literature Survey done in this area. Query Optimization in such an environment would first require the estimate of the energy consumption of a query evaluation plan. Section 2.5 discusses possible models for estimation of energy consumption. This is followed by the Experimental Results of some preliminary experiments carried out. Chapter 3 explains the motivation for Query Optimization in Memory Constrained environments. Section 3.3 explains the use of Compression as an approach to performing Query Optimization on hand-helds. Chapter 4 explains the use of materialization and pipelining as approaches to multi-query optimization. Chapter 5 presents the Conclusions and the possible areas for Future Work.

## Chapter 2

# Query Optimization in Energy constrained environments

### 2.1 Overview

Energy management has become one of the great challenges in portable computing. This is the result of the increasing energy requirements of hand-held devices without a corresponding increase in battery technology. Energy-conscious design is important at all levels of the system architecture, and the software has a key role to play in conserving the battery energy on these devices.

### 2.2 Literature Survey

The first step towards query optimization in hand-helds is to establish a cost-factor to optimize. While selecting the most energy efficient plan is mostly a matter of modifying the cost function employed by the optimizer to take energy usage into account, this is only an initial step towards solving the problem. [AG92] discusses Energy Consumption Models for hand-helds in the stand-alone mode as well as for Client-server. The amount of power consumed by different components such as CPU, Disk and Display is mentioned. The product of the time spent on a component multiplied by its power consumption gives the total energy consumed by the component. For the execution of a query, several of these components are used. A sum total of the energy consumed by all these components gives the total energy consumed by the query. The equation for the same is given as:

$$energy(p) = p_{cpu} * t_{cpu} + p_{disk} * t_{disk} + p_{response} * t_{response} \quad (2.1)$$

where  $p_{component}$  is the power consumed by *component* and  $t_{component}$  is the time for which it is used. It is this energy consumed that is used as a factor to be optimized by the optimizer. The paper also discusses a dynamic algorithm to search for plan with the least energy among all plans. This concept can be used for building the energy consumption model for hand-helds. The components to be considered in this case would be *flash memory*, *cpu*, *display* and *main memory*. This will be discussed in detail in Section 2.5.

[GAS<sup>+</sup>03] discusses use of work partitioning between client and server to reduce energy consumption at the client-end. Their work is focussed on spatial queries. This is discussed in detail in Section 2.3.

[ASV<sup>+</sup>01] discusses the effect of various Spatial Index organizations on the energy consumption of queries using these Index structures. The application of this concept to DELite is discussed in detail in Section 2.4.

[CY03] outlines the various factors of an mpeg video that affect energy consumption during its playback on hand-helds. The application of this concept to DELite is discussed in Section 2.5.

[XLWN03] discusses the use of compression as an approach for reducing energy consumption in wireless/networked hand-helds. The paper proposes a model for estimation of energy consumption for compressed downloading. The energy expended by the hand-held is the sum of the energy for downloading and the energy for decompressing. Hence the compression should be such that the net energy expended is minimum. The energy expended is estimated as:

$$E = m * s_c + c_s + t_i * p_i + t_d * p_d \quad (2.2)$$

where  $m$  is energy to receive one unit data,  $s_c$  is the size of compressed file to download,  $t_d$  is the total time taken for decompression,  $p_d$  is the average power required for decompression,  $t_i$  is the idle time between receipt of packets while downloading and  $p_i$  is the power required in the idle mode. One approach used in the paper is interleaving of download of compressed data with decompression - to reduce the idle time.

## 2.3 Work Partitioning between Server and Client for Query Optimization

Resource-constraints on a mobile device can bias an application developer to off-load most of the work to a server whenever possible (through a wireless network). For instance, the hand-held could serve as a very thin client by immediately directing any user query to a server which may be much more powerful and where energy may not be a problem. The hand-held does not need to store any data locally or perform any complicated operations on it in this case. However, it is not clear whether that strategy is the best in terms of the performance and energy viewpoints.

1. Sending the query to the server and receiving the data over the wireless network exercises the network interface, which is a significant power consumer. If this dominates over the power consumed by the computing components on the hand-held, then it is better to perform the entire operation on the client side as long as the hand-held can hold all the data.
2. Issues in accessing the server (unreachability/disconnectivity from remote locations) may necessitate the computations to be performed at the client
3. Privacy issues (the user may not want others, including the server, to be aware of his/her location or queries) may preclude the delegation of work.

There are several options between these extremes - performing all operations at the client or everything at the server - that are worth a closer look. Partitioning the work appropriately between the client and server in a mobile setting can have important ramifications from the performance and energy consumption viewpoints. [GAS<sup>+</sup>03] examines different ways of implementing/partitioning spatial queries between the mobile client and server, and examining energy and performance benefits from these approaches. Their work is specific to *spatial queries*, i.e. identifying those parts of spatial queries that can be partitioned between the client and server. However, this is applicable to any complex query that is to be serviced by the hand-held.

### 2.3.1 Future Work

During the period of time that the server-related part of the query is shipped to the server, processed by it and the results returned to the client, the client is idle. A possible extension would

be to identify those parts of the query that are parallel/pipelineable so that the client may execute in parallel with the server - without wasting energy in the idle-mode.

Another area of work is the dynamic partitioning of work - based on the availability of resources.

### 2.3.2 Application to Simputer/DELite

The client-server model is not considered in case of Simputer - it is assumed that the data is downloaded to the Simputer and queries are executed in the offline mode. Hence the problem of work partitioning for query execution does not arise.

However, instead of partitioning query-execution related task its possible to partition some other tasks. For instance - compression is an energy-consuming task and it may not be possible for the compression to be performed at the client-end. However the low-end client can execute queries on compressed data. Hence partitioning can be done in such a way that compression related tasks are offloaded to the server and query processing over the compressed data may be carried out at the client. This notion can be applied in those scenarios where the client-server connectivity is not an issue and compression/decompression of data is to be carried out very frequently.

## 2.4 Energy behaviour of memory resident data

[ASV<sup>+</sup>01] discusses the impact of various Index Structures of Spatial DBMS on the energy consumption. The energy consumption of *R-Trees*, *Quad-Trees* and *Buddy Trees* for typical Spatial queries such as *Point query*, *Range query* and *Nearest Neighbour Queries* have been analysed. The experimental results reveal that the Index-organization, indeed has an effect on the energy consumption of the queries. These results maybe used in selection of what Index structures to use under depending on the availability of resources, and on the type of queries that are expected to be serviced most frequently.

[ASV<sup>+</sup>01] also details out certain parameters of each Index organization, which when fine-tuned reduce the energy consumption of queries making use of that organization. For instance - the fan-out of R-Trees has an impact on the energy consumption of queries using it.

### 2.4.1 Future work

The energy consumption of these Index structures for other spatial queries can be analysed.

### 2.4.2 Application to DELite

The concept used in [ASV<sup>+</sup>01] can be applied to DELite. Energy behaviour of data and index organizations in DELite can be analysed. - i.e. Index-based, Domain based, Ring-based and Flat storage. Also, the various parameters that affect energy consumption (such as the number of bits used for ID-based storage) can be investigated.

To take this study one step further, Multi-query optimization can be carried out. It is possible that energy behaviour of a particular query is most optimal for a specific index organization, but a set of queries when executed together perform better with some other organization.

## 2.5 Energy consumption prediction model

Given a query evaluation plan, we should be able to estimate its energy consumption. It is this estimate that would be used by the query optimizer as a cost factor to minimize. It can be noted that for a given query it is possible to come up with several query evaluation plans. Depending on

the available memory, the JOIN schemes are selected. Different schemes lead to different number of flash/memory accesses. Hence we can say that energy consumption depends on the amount of memory available.

In order to model the energy consumption of a query on a hand-held, we could use different approaches:

### 2.5.1 With CPU computations and I/O as the units of energy consumption

Let

$E_{w_f}$  be the energy required for a write to flash  
 $E_{R_f}$  be the energy required for a read from flash  
 $E_{W_m}$  be the energy required for a write to the main memory  
 $E_{R_m}$  be the energy required for a read from the main memory  
 $E_C$  be the energy required for a unit of computation  
 $n_{w_f}$  be the number of writes to flash  
 $n_{R_f}$  be the number of reads from flash  
 $n_{W_m}$  be the number of writes to the main memory  
 $n_{R_m}$  be the number of reads from the main memory  
 $n_C$  be the number of units of computation

The energy consumption for the QEP will then be:

$$E_{QEP} = n_{W_f} * E_{W_f} + n_{R_f} * E_{R_f} + n_{W_m} * E_{W_m} + n_{R_m} * E_{R_m} + n_C * E_C \quad (2.3)$$

For a query execution plan, using catalog information we can estimate the number of flash reads and writes and number of RAM reads and writes. We also need to come up with some scheme to measure the number of units of computation. A unit could be considered as a single CPU instruction (like incrementing the program counter, jump instruction etc). Now, the problem boils down to finding out the values of energy consumption of flash and RAM reads and writes. To compute the read and write energy requirements of flash, we can execute a program that simply writes to the flash memory in a loop till the battery is exhausted. Hence, given the original energy of the battery and the number of flash writes that took place, we can compute the energy requirement for a flash write. A similar scheme can be employed for computing the energy consumption of a flash read. Schemes for estimating the RAM read/write energy consumption and energy consumption of a unit of computation need to be worked out clearly. An approach to the same has been suggested in [Lee95]. The article discusses methods to model the power consumption of processors from the software point of view. The methods were classified into Instruction Level Power Models and Functional Level Power Models.

To be able to apply the Equation 2.3, we have to assume that the energy required for execution of each instruction (computational unit) is identical.

In order to use this model, each one of the factors appearing in the Equation 2.3 need to be estimated/computed as accurately as possible. Schemes need to be devised to do the same.

### 2.5.2 With hardware components as units of energy consumption

As explained in [AG92], if the power dissipated in each of the hardware components is known, and the time for which it is used is known, we can compute the total energy consumption can be easily calculated.

$$E_{QEP} = p_{cpu} * t_{cpu} + p_{flash} * t_{flash} + p_{RAM} * t_{RAM} + p_{display} * t_{display} \quad (2.4)$$

where  $p_{component}$  is the power consumed by *component* and  $t_{component}$  is the time for which it is used.

In this model, however, it is not clear whether the power dissipated in the flash/RAM for read as well as write is the same. Also, it is not very clear whether given a QEP it is possible to estimate the amount of time for which each of the components would be used.

### 2.5.3 Using query parameters for estimating the energy consumption

[CY03] details out experiments designed to bring forth parameters that can be used to predict the energy requirement for MPEG playback. These parameters in turn are used to predict the energy consumption of mpeg video playback in hand-helds. The experiments reveal that mpeg video parameters such as frame rate, capture size, spatial resolution have an effect on the energy consumption. A similar notion can be applied to DELite on Simputer. For this purpose the parameters of queries that affect energy consumption need to be identified for the various queries. Possible parameters:

- **Point queries** - Number of attributes in the result, number and type of attributes participating in the **WHERE** clause.
- **Range queries** - Apart from the parameters mentioned for Point queries, the size of the result-set.
- **Join query** - Apart from the above mentioned parameters, the number of relations being *JOINED*, the type of the joining attributes etc.

The effect that these parameters have on the energy consumption needs to be carefully analysed such that a model to predict energy consumption of queries maybe built. Given the energy consumption of a query Q1, suppose we need to estimate the energy consumption of a query Q2 that is similar to Q1, differing only in some parameters like number of attributes in result, size of result-set etc. If we are able to study quantitatively the impact that these parameters have on the energy consumption, then we can estimate the energy consumption of Q2 given that of Q1.

## 2.6 Experimental Results

In order analyse the energy behaviour of Data and Index organizations of DELite, certain basic experiments were carried out. The experiments were carried out on Simputer using 1.2V 700mAh Ever-ready rechargeable batteries. The schema used for testing is the hospital schema.

Relation	Size
DOCTOR(DOCID int, NAME char[20])	91
PRES(VISITID int, DRUGID int)	2155
VISIT(VISITID int, DOCID int, DATE int)	830
DRUG(DRUGID int, TYPE char[20])	77

Table 2.1: Hospital Schema used for experiments

### 2.6.1 Technique to measure the energy consumed by a query

The rechargeable battery was charged for 1 hour duration. A script was written that fires a particular query continuously in a loop, till the battery is completely discharged. The number of times the query has executed is output on the console. Given that a query executes  $n$  number of times for a 1 hour charged battery, we can assume that the query requires an average of  $1/n$  hours worth energy.

### 2.6.2 Assumptions

1. Communication overhead between Simputer and the desktop hosting *minicom* has not been accounted for.
2. The script used for executing the queries outputs the results to the console. This console-output may affect the energy consumption.
3. The charging of the rechargeable batteries varies linearly with time.
4. As mentioned in Section 2.5, the energy consumed depends on the memory available. For the experiments, the memory has been kept constant - by setting the value in the memval file to 65335. The join schemes that resulted were:
  - VISIT - Indexed Nested Loop Join
  - DOCTOR - Hash Join
  - PRES - Indexed Nested Loop Join
  - DRUG - Buffered Aggregate

### 2.6.3 Queries

The following queries were used for:

1. Testing the effect of number of attributes on the energy consumption-
  - $Q1$  : SELECT DOCTOR.NAME FROM DOCTOR, PRES, DRUG, VISIT WHERE VISIT.DATE = 1998 AND DOCTOR.DOCID = VISIT.DOCID AND DRUG.DRUGID = PRES.DRUGID AND PRES.VISITID = VISIT.VISITID AND DRUG.TYPE = 'Antibiotic'
  - $Q2$  : SELECT \* FROM DOCTOR, PRES, DRUG, VISIT WHERE VISIT.DATE = 1998 AND DOCTOR.DOCID = VISIT.VISITID AND DRUG.DRUGID = PRES.DRUGID AND PRES.VISITID = VISIT.VISITID AND DRUG.TYPE = 'Antibiotic'

The number of attributes in the result set for  $Q2$  is more than that for  $Q1$ .

2. Testing the effect of number of relations participating in the join-
  - $Q3$  : SELECT DRUG.TYPE FROM DRUG,PRES WHERE DRUG.DRUGID = PRES.DRUGID;
  - $Q4$  : SELECT DOCTOR.NAME FROM DOCTOR, PRES, DRUG, VISIT WHERE DOCTOR.DOCID = VISIT.DOCID AND DRUG.DRUGID = PRES.DRUGID AND PRES.VISITID = VISIT.VISITID ;

The number of relations participating in the join are more in  $Q4$  than in  $Q3$

3. Testing the effect of condition evaluation-

- $Q_5$  : SELECT DOCTOR.NAME FROM DOCTOR,PRES,DRUG,VISIT WHERE DOCTOR.DOCID = VISIT.DOCID AND DRUG.DRUGID = PRES.DRUGID AND PRES.VISITID = VISIT.VISITID AND DRUG.TYPE = 'Antibiotic'
- $Q_6(Q_1)$  : SELECT DOCTOR.NAME FROM DOCTOR, PRES, DRUG, VISIT WHERE VISIT.DATE = 1998 AND DOCTOR.DOCID = VISIT.DOCID AND DRUG.DRUGID = PRES.DRUGID AND PRES.VISITID = VISIT.VISITID AND DRUG.TYPE = 'Antibiotic'

Q6 has an additional condition (test for date).

#### 2.6.4 Results

Query	Number of executions	Time(min)
idle	none	10
Q1	75	6
Q2	74	6
Q3	1321	6
Q4	34	5
Q5	106	5
Q6	75	6

Table 2.2: Experimental Results

#### 2.6.5 Observations and Conclusions

- As can be observed from the outcomes of queries 1 and 2, the number of executions of both are approximately the same. This leads us to believe that the *number of attributes in the result set has little effect on the energy consumption*.
- Q3, containing fewer number of relations participating in the join as compared with Q4, has executed more number of times than Q4. From this we can conclude that the more the number of relations participating in the join, the more the energy consumption. This result is intuitive, since more number of relations means more I/O and more computations. What's left is to determine by what factor the energy consumption varies - whether it depends on the size of the additional relations, or on some other factors.
- The number of executions of Q5 is greater than that of Q6. This result too is intuitive, since Q5 has to test for fewer conditions than Q6. Consequently it requires less computations.

#### 2.6.6 Future Work

1. Experiments need to be conducted for testing the effect of other parameters (such as those mentioned in Section 2.5.3) on energy consumption.
2. Suppose the energy required for a query is known, and we need to use this information for computing the energy consumed by another query that is similar to the first one - but that

differs in terms of some parameters like result-set size, number of joining conditions etc. It may be possible to estimate this if we can find the amount by which the energy consumption varies depending on the query parameters.

3. Devising other methods for measuring the energy consumption that overcome the drawbacks mentioned in the Section 2.6.2 on Assumptions
4. To study the relation between time and energy consumption - whether the time spent on execution of a query is directly proportional to the energy consumed by it.

## 2.7 Summary

For performing Query Optimization in the Energy Constrained Environment in DELite (that does not follow the Client-Server architecture), we need to first come up with an Energy Consumption Model that can be used for estimating the energy consumption of a Query Execution Plan. As pointed out in Section 2.5, this Plan depends upon the amount of available memory. The query optimizer can make use of this estimate as the cost factor to minimize.

## Chapter 3

# Query Optimization in Memory Constrained Environments

### 3.1 Overview

Hand-held devices have limited memory. If the query optimization algorithm is not aware of this constraint, it may lead to severe degradation in performance. Hence, the query optimizer should be aware of the available memory while evaluating which plan is the best.

### 3.2 Literature Survey

[NA03] proposes a query execution model that reaches a lower bound in terms of RAM consumption. It devises a new form of optimization, called iteration filter, that drastically reduces the prohibitive cost incurred by the preceding model, without hurting the RAM lower bound. It also analyses how the preceding techniques can benefit from an incremental growth of RAM. However the paper does not explain how exactly a memory cognizant optimizer can be designed.

[HSS00] explains the design of a query optimizer that is cognizant of the available memory. Given a query evaluation plan, there are several ways in which the available memory maybe distributed amongst the operators. The paper shows how to optimize a query given a cost vs memory allocation function of each operator. There are two approaches to solve this problem. The first one, a 2 phase optimizer, makes use of a traditional optimizer for generation of a query evaluation plan. Then, the memory is divided between the various operators of the generated query evaluation plan in the second phase. The second approach, a 1 phase optimizer, makes use of the memory availability while selecting the optimal Query Evaluation Plan. To determine the optimal memory allocation between operators in a pipeline, sharing of memory between 2 operators is explained, followed by 3 operators, followed by  $n$  operators.

### 3.3 Compression in a Memory Constrained Environment

#### 3.3.1 Motivation

Over the last few decades, improvements in CPU speed have outpaced improvements in disk access rates by orders of magnitude, motivating the use of data compression techniques in database systems to trade reduced disk I/O against additional CPU overhead for compression and decompression of data. In a compressed database system, data are stored in compressed format on disk and

are decompressed during query processing. Compression can reduce the amount of space required for a given data set. This has several implications on the I/O:

1. The reduced data space fits into a smaller physical disk area; therefore, the *seek distances and seek times are reduced*.
2. More data fit into each disk page, track, and cylinder, allowing *more intelligent clustering of related objects into physically near locations*.
3. The *unused disk space can be used for disk shadowing to increase reliability, availability, and I/O performance*.
4. Compressed data can be transferred faster to and from disk. In other words, data compression is an effective means to *increase disk bandwidth* and to relieve the I/O bottleneck found in many high-performance database management systems.
5. In distributed database systems and in client-server situations, compressed data can be *transferred faster across the network* than uncompressed data.
6. Retaining data in compressed form in the I/O buffer allows more records to remain in the buffer, thus *increasing the buffer hit rate and reducing the number of I/Os*.
7. *I/O to log devices should decrease* since the log records can become shorter.

Apart from the above, the join-processing also benefits from compression:

1. Materializing join output records is faster because records are shorter and less copying is required.
2. For join inputs larger than memory, more records fit into memory. In hybrid hash join, for instance, the fraction of the file that can be retained in the hash table and thus be joined without any I/O is larger.

### 3.3.2 Issues in Compressed Databases

Some issues make it non-trivial to apply compression in database systems:

1. Compression and decompression are CPU intensive operations. The cost is especially important at low-end clients.
2. Database applications often need random access to data in small pieces (tuples, attributes, etc). The popular compression methods such as LZ77 only work well for large chunks of results, and a large chunk of results must be decompressed even if only a small piece is needed. Therefore, these methods have prohibitive decompression costs for low-end clients that work on batteries.
3. A DBMS holds domain specific knowledge of the query results, which can be used to enhance the effect of compression. For instance, values of the same attribute have many similarities while values of different attributes usually have very different characteristics. This implies that a single method may not be appropriate for all attributes. Instead, a combination of compression methods that compress each attribute separately based on the domain knowledge usually can achieve better effect. The choice of an appropriate combination is not obvious.

### 3.3.3 Design of a Compressed Database

The design of a compressed database involves several aspects:

1. *Use of effective compression methods for the data*, in particular for strings. One such compression method for strings, as explained in [?] is **Hierarchical Dictionary Encoding strategy** that intelligently selects the most effective compression method for string-valued attributes, and achieves significant performance improvement over standard compression techniques such as Lempel-Ziv.
2. *Efficient running of queries over the compressed data*. This can be achieved by making use of **Compression-aware query optimization**. [?] details out such query optimization algorithms.
3. *Automatic selection of compression methods based on a query workload*, which is crucial for the performance and usability of compressed database systems. However [?] does not consider caching of intermediate (decompressed) results that can reduce the overhead of transient decompression. Also, the effect of updates on the Compression techniques have not been considered.
4. *Analysis of how to compress results shipped to clients* that have limited bandwidth connections or severe storage restrictions. [?] describes query-result compression techniques that achieve a high compression ratio and are also conveniently to decompressed.

## 3.4 Summary

[HSS00] explains the design of a memory cognizant query optimizer, which means that given the memory availability it is possible for the optimizer to select the best query evaluation plan. Another approach to query optimization is to use Compression of data. It can be assumed that Compression, being a computationally expensive operation, cannot be performed on the hand-held that has limited resources. However, data can be stored on the hand-held in the compressed form, and queries may be executed over this compressed data.

## Chapter 4

# Multi-Query Optimization

Often, a sequence of interdependent queries may be posed simultaneously to the DBMS. The optimization of such sequences is called multi-query optimization, and it attempts to exploit dependencies between the queries in the derivation of a query evaluation plan. The approaches for sharing of data in multi-query optimization are *Materialization* and *Pipelining*.

### 4.1 Materialization in Multi-query Optimization

Multi-query Optimization traditionally determines the common sub-expressions between multiple queries and materializes these sub-expressions so that they need not be recomputed.

#### 4.1.1 Generation of a search space of the views to materialize

[TX04] addresses the problem of constructing search space for materialized view selection by using an approach which is based on adding Closest Common Derivators (CCDs) to the alternative plans of the queries, and on rewriting the queries using the CCDs. CCDs are a more generalized form of common-subexpressions and are independent of the Query Evaluation Plan. Adding CCDs to the alternative evaluation plans of the queries generates a search space which is expected to comprise all the interesting views for materialization since the CCD of two queries is as close to these queries as possible. The generation of this search space does not take into account memory/energy constraints. This could be treated as a topic for further work.

#### 4.1.2 Selection and Maintenance of the materialized views and Indexes

Materialized views are a common way to improve performances in the execution of queries against views. However, a materialized view has to be updated when the base tables on which it is defined are updated. The paper [MRSR01] explains how to efficiently select expressions and indices that can be effectively shared, by transient materialization, and also additional expressions and indices for permanent materialization. For every view it proposes the best maintenance plan - incremental or re-computation. However, the paper does not take into account the amount of space available for materialization. [MS01] presents an algorithm for incremental view maintenance in data warehouse environment with multiple independent data sources.

### 4.2 Pipelining in Multi-query Optimization

Significant performance benefits can be reaped if common subexpressions are pipelined to their uses, without being materialized. [DSRS01] presents a general model for schedules with pipelining,

and presents a necessary and sufficient condition for determining validity of a schedule under the proposed model. Given a query, it explains how to determine the least cost pipeline schedule and also explains that this problem is NP complete. The implementation of their algorithm demonstrated that pipelining can be added to existing multi-query optimizers without any increase in time complexity. The drawback of their approach, however, is that they use 2-phase optimization. This means that the multi-query optimizer does not take into account pipelining. As a part of the future work, it can be analysed whether a 1-phase optimizer is feasible and whether it achieves any better performance.

## Chapter 5

# Conclusion

This report summarizes the various aspects of query optimization in resource constrained devices, in particular memory and energy constrained devices, in terms of the work done so far along with future work in the area. Wherever needed, the application of concepts to DELite has been discussed. The following can be categorized as future work:

1. To come up with a model for Estimation of Energy Consumption of a query evaluation plan.
2. To come up with an optimizer for an energy constrained environment that makes use of the above mentioned model for determining the costs of the query evaluation plans.
3. To study the various query parameters that affect energy consumption as discussed in section 2.5. These parameters would be used for estimation of energy consumption of a query given the energy consumption of a similar query.
4. To study the energy behaviour of various Index and Data Organizations in DELite, as mentioned in section 2.4.2. The insights gained by this study can be used by the optimizer to determine what organization should be used for storing the data and the query results.
5. To study the inter-relation between energy consumption and memory requirements. It may be observed that a sufficiently large memory may be helpful in generation of query evaluation plans that consume minimum energy. On the other hand, if sufficient energy is available (and performance degradation is tolerable), then a low-memory consuming fully pipelineable plan may be generated. In a hand-held what configuration (memory and energy) is optimal could be analysed
6. To study the relation between the execution time and the energy consumed.

# Bibliography

- [ACN00] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. Automated selection of materialized views and indexes in SQL databases. In *The VLDB Journal*, 2000.
- [AG92] R. Alonso and S. Ganguly. Energy efficient query optimization. Technical report, Technical Report MITL-TR-33-92, Matsushita Infotech Lab, Princeton, NJ, 1992.
- [ASV<sup>+</sup>01] Ning An, Anand Sivasubramaniam, Narayanan Vijaykrishnan, Mahmut T. Kandemir, Mary Jane Irwin, and Sudhanva Gurusurthi. Analyzing energy behavior of spatial access methods for memory-resident data. In *The VLDB Journal*, 2001.
- [BA03] K. BARR and K. ASANOVIC. Energy aware lossless data compression, 2003.
- [Bat] Battery. <http://www.gizmology.net/batteries.htm>.
- [BBPV00] C. Bobineau, L. Bouganim, P. Pucheral, and P. Valduriez. Picodbms: Scaling down database techniques for the smartcard. Technical report, PRiSM Technical Report, 2000.
- [CGK01] Zhiyuan Chen, Johannes Gehrke, and Flip Korn. Query optimization in compressed database systems. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001.
- [CY03] Srijan Chakraborty and David K. Y. Yau. Predicting Energy Consumption of MPEG video playback on handhelds, 2003.
- [DSRS01] Nilesh N. Dalvi, Sumit K. Sanghai, Prasan Roy, and S. Sudarshan. Pipelining in multi-query optimization. In *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2001.
- [GAS<sup>+</sup>03] Sudhanva Gurusurthi, Ning An, Anand Sivasubramaniam, N. Vijaykrishnan, Mahmut Kandemir, and Mary Jane Irwin. Energy and performance considerations in work partitioning for mobile spatial queries. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, 2003.
- [HSS00] Arvind Hulgeri, S. Seshadri, and S. Sudarshan. Memory cognizant query optimization. In *Proc. of COMAD*, 2000.
- [Lee95] M. Lee. Power analysis and low-power scheduling techniques for embedded dsp software, 1995.
- [MRSR01] Hoshi Mistry, Prasan Roy, S. Sudarshan, and Krithi Ramamritham. Materialized view selection and maintenance using multi-query optimization. *SIGMOD Rec.*, 2001.
- [MS01] G. Moro and C. Sartori. Incremental maintenance of multi-source views, 2001.

- [NA03] P. Pucheral N. Anciaux, L. Bouganim. Memory requirements for query execution in highly constrained devices. In *29th International Conference on Very Large Data Bases, VLDB'03, Berlin, 2003*.
- [PCK04] Jayaprakash Pisharath, Alok Choudhary, and Mahmut Kandemir. Reducing energy consumption of queries in memory-resident database systems. In *CASES '04: Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems, 2004*.
- [Rao05] Ashwini G Rao. Memory Constrained DBMSs with Updates, 2005.
- [RSSB00] Prasan Roy, S. Seshadri, S. Sudarshan, and Siddhesh Bhoje. Efficient and extensible algorithms for multi query optimization. In *ACM SIGMOD International Conference on Management of Data, 2000*.
- [Sen04] Rajkumar Sen. An open source DBMS for handheld devices, 2004.
- [Sim] Simputer. <http://www.simputer.org>.
- [SKS02] A. Silberchatz, H. Korth, and S.Sudarshan. *Database System Concepts*. 2002.
- [SR05] Rajkumar Sen and Krithi Ramamritham. Efficient data management on lightweight computing devices. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 419–420, Washington, DC, USA, 2005. IEEE Computer Society.
- [TX04] Dimitri Theodoratos and Wugang Xu. Constructing search spaces for materialized view selection. In *DOLAP '04: Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, 2004*.
- [XLWN03] R. Xu, Z. Li, C. Wang, and P. Ni. A report on impact of data compression on energy consumption of wireless-networked handheld devices, 2003.