Attribute Cardinality Maps: New Query Result Size Estimation Techniques for Database Systems

By

Murali Thiyagarajah

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

May 1999

Abstract

Current database systems utilize histograms to approximate the frequency distributions of attribute values. These are used to efficiently estimate query result sizes and access plan costs. Even though they have been in use for nearly two decades, there has been no significant mathematical techniques (other than those used in statistics for traditional histogram approximations) to study them.

The major contributions of this thesis are the two novel histogram-like query result-size estimation techniques, namely, the Rectangular Attribute Cardinality Map (R-ACM) and the Trapezoidal Attribute Cardinality Map (T-ACM), that aim to approximate the density of the underlying attribute values using the philosophies of numerical integration. By deriving the probability density function within the sectors of these structures and proving that the frequencies of the attribute values within the sectors are Binomially distributed, we provide a fairly extensive mathematical analysis for their variances, and the average and worst case errors for result size estimations. This enables us to make a fair comparison with the current state-of-the-art estimation methods, and to prove the superiority of our new techniques. We verify our theoretical results using an extensive set of experiments, including both synthetic and real-world data, and the Transaction Processing Performance Council's TPC-D benchmarking environment.

Finally we investigate a few strategies to improve the estimation accuracy of the ACMs by finding appropriate build-parameters. In the case of the R-ACM, estimation accuracy can be arbitrarily increased by reducing the tolerance value, τ. But this is, of course, limited by the available storage space. As opposed to this, for a given storage space, the accuracy of the T-ACM can be improved by finding the suitable slope for the trapezoidal sectors, and thus we devote some attention to determining the suitable slopes. We anticipate that due to their high accuracy and low construction costs, the attribute cardinality maps could prove to be standard tools for query optimization in future database systems.

To My Precious Wife, Kala

and

Lovely Daughter, Anupa

Acknowledgments

First and foremost, I owe my greatest intellectual debt to Prof. John Oommen, my thesis supervisor. It is a pleasure for me to acknowledge this. I cannot express in words my gratitude to John for his unwavering support and guidance throughout my research work. The kernel of the thesis was conceived at John's house when he had invited my family for an evening. After a stimulating exchange of arguments and ideas the concepts involved in the ACMs and their underlying distribution became clear. Since then the ideas took form, and culminated in this research work, which has finally led to my Ph.D. degree. Without him I could **not** have finished and written the entire thesis in less than ten months.

John has been a constant source of inspiration and encouragement for me, ever since I joined Carleton. I am deeply indebted to him for his observations and ideas which are full of deep insight, and to his exceptional conceptual and organizational abilities that have helped me in many ways.

I would like to express my sincere and profound thanks to John for putting my Ph.D. on the fast track. I will always be grateful to him.

My special thanks are also due to my dissertation committee members, Prof. Sivarama Dandamudi, Prof. Stan Matwin and Prof. Dorina Petriu, for their many valuable and constructive suggestions. My sincere thanks to Prof. Dandamudi for his being an informal advisor and friend over the entire course of my Ph.D. degree. I would also like to take this opportunity to especially thank Prof. Rajjan Shinghal, my external examiner, for agreeing to be on my examining committee and, for patiently reviewing my research work.

It is a pleasure for me to acknowledge and thank Prof. Tim Merrett, McGill University, for reviewing an earlier draft of my thesis and providing a number of valuable comments. The research project on the "Interoperability of Heterogeneous Database Systems", in which I worked for nearly 18 months under his valuable guidance, gave

me many insights into the complexities of real-world heterogeneous database systems and provided with me an opportunity to master Oracle.

I am grateful to Prof. Evangelos Kranakis and Prof. Frank Fiala for the numerous conversations I have had with them and for their friendly guidance throughout my stay at Carleton. Most importantly, I would like to thank Prof. Kranakis for granting me a leave of absence to enable me to complete my studies.

My thanks are also due to the many wonderful staff at the School of Computer Science, who helped make things happen. A big "thank you" to Rosemary Carter, Barbara Coleman, Linda Pfeiffer, Maureen Sherman and, of course, Marlene Wilson!

Finally, but most importantly, I am *greatly* indebted to Kala, my precious wife, and my dearest and best friend. I owe her a great deal for standing with me steadfastly during the numerous difficult periods I had, when I probably reached the limits of her patience. Now that my Ph.D. thesis is finally done, may be the best way of saying thank you is by promising: "I will always be home when you are on call, dear!" To our wonderful daughter Anupa, I am equally indebted.

I would also like to take this opportunity to express my sincere gratitude and appreciation to my parents, sister and the rest of my family, who helped me in more ways than I can say in words.

It is quite likely that I am omitting many others who have been of great help to me over the years, but I hope that they will forgive my oversight and accept my heartfelt thanks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the main uses of computers is storing and retrieving large amount of data efficiently. The computer systems used for this purpose are known as *database systems* and the software that manages them are known as *database management systems* (DBMS). The DBMS facilitates the efficient management of data by (i) allowing multiple users concurrent access to a single database, (ii) restricting access to data to authorized users only, and (iii) providing recovery from system failures without loss of data integrity. The DBMS usually provides an easy to use high-level query/data manipulation language such as the Structured Query Language (SQL) as the primary interface to access the underlying data.

SQL, the most commonly used language in modern-day DBMSs, is a declarative language. Thus, it shields users from the often complex procedural details of accessing and manipulating data. Statements or commands expressed in SQL are generally issued by the user directly, using a command-line interface. The advantage of the declarative SQL is that the statements only need to specify *what* answer is expected, and not *how* it should be computed. The actual sequence by which an SQL command is computed is known as the procedural Query Evaluation Plan (QEP). The procedural QEP for a given non-procedural SQL statement is generated by the DBMS and executed to produce the query result. Typically, for a given query, there are many (often billions) alternative procedural QEPs that all compute the result required. Each QEP, however, has its own *cost* in terms of resource use and response time. The cost

is usually expressed in terms of the I/O operations such as the number of disk reads and writes, and the amount of CPU work to execute a given QEP. The problem of devising the *best* procedural QEP for a (possibly SQL) query so as to minimize the cost is termed *query optimization.*

In all brevity, given a declarative SQL query, the DBMS's *Query Optimizer* module determines the best possible procedural QEP to answer it. In order to do this, the query optimizer uses a model of the underlying system to select (from a large set of candidate plans) an efficient plan as quickly as possible. Efficiency of the QEP is measured in terms of resource utilization and response time.

The cost incurred in evaluating a QEP is proportional to the number of operations (including disk reads, writes and CPU work) required to compute the final answer from the base relations. The size of the *final result* of a query (as well as the sizes of the *base relations*) will be the same regardless of which QEP, from among many possible candidate QEPs, is chosen by the query optimizer. Hence obviously the cost of a QEP depends on the size of the *intermediate* relations generated during the computation of the query, as this is the single most important factor responsible for the difference in the costs of various QEPs of the given query. Hence by choosing a QEP that has smaller intermediate relations than other QEPs, we can minimize the cost involved in computing the final result of the given query. Although this is easy to explain, due to the large number of possible alternative QEPs, computing the sizes of the intermediate relations accurately for each possible QEP is virtually an impossible task. Hence, one approach is to approximately estimating the sizes of the intermediate relations.

The subject of this thesis is the development and application of new histogram-like techniques for approximate query result-size estimation that can be used by the Query Optimizer module.

It is assumed that the reader of this thesis is familar with the basic database terminologies such as relations, attribute values, tuples etc., used in the context of a relational database system. A complete set of glossary can be found in many undergraduate database text books, including [33], pp 137-177.

In section 1.1, we give a brief overview of query optimization. In section 1.2, we

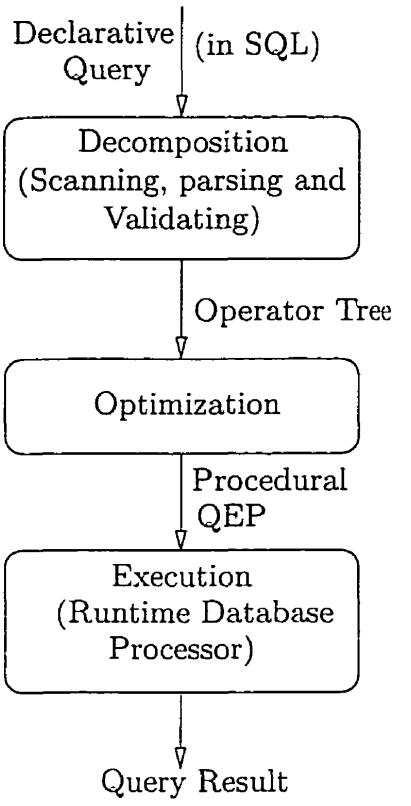


Figure 1.1: Query Processing Architecture

discuss the problem in detail and provide the research objectives. Finally, in section 1.3 we give an overview of the thesis.

1.1 Query Optimization: An Overview

Query optimization for relational database systems is a combinatorial optimization problem, which makes exhaustive search unacceptable as the number of relations in the query increases. The query optimization process can be generally divided into three distinct phases, namely *query decomposition*, *query optimization* and *query execution* as shown in Figure 1.1.

In the query decomposition module, the declarative SQL query is first scanned, parsed and validated. The scanner sub-module identifies the language components

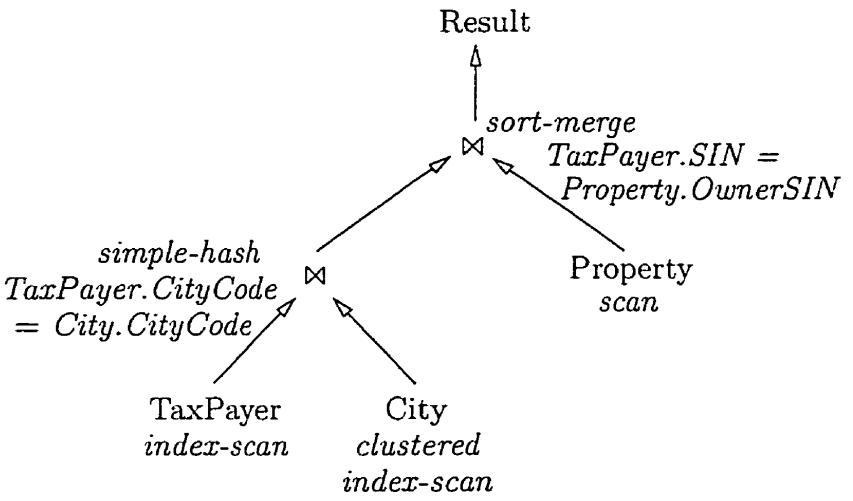


Figure 1.2: An Annotated Query Tree

in the text of the query, while the parser sub-module checks the query syntax. The validator checks that all attribute and relation names are valid and semantically meaningful. The query is then translated into an internal format expressed in relational algebra in the form of a query *Operator Tree*. Using this operator tree as its input, the query optimizer module searches for a procedural plan with an optimal ordering of the algebraic operators. This optimal procedural plan is represented by an annotated query tree. Such trees encode procedural choices such as the order in which operators are evaluated and the method for computing each operator. Each tree node represents one (or several) relational operators. Annotations on the node represent the details of how it is to be executed. For example, a join node may be annotated as being executed by a hash-join, and a base relation may be annotated as being accessed by an index-scan. The choices of the execution algorithms are based on the database and system characteristics, for example, the size of the relations, available memory, type of indexes on a given attribute etc.

This fully annotated operator tree with the optimal QEP is then passed on to the query execution engine where all the low level database operations are carried out and the answer to the query is computed. An example annotated query tree is shown in Figure 1.2.

1.1.1 An Example

Figure 1.3 shows an example relational database from a Property Tax Assessor's office. This database consists of three relations, namely,

- TaxPayer(SIN, CityCode, Name, DOB, Balance),
- Property(Type, OwnerSIN, Tax) and
- City(CityCode, CityName, Population).

Assume that these relations have the following cardinalities:

$|TaxPayer| = 12 \times 10^6$ tuples,

$|Property| = 3 \times 10^6$ tuples, and

$|City| = 4000$ tuples.

TaxPayer:

SIN	CityCode	Name	DOB	Balance
501-112-347	4679	John Smith	09/22/70	$3,218.00
123-456-091	4518	John Doe	12/30/58	$1,093.15
⋮	⋮	⋮	⋮	⋮
318-231-089	1023	Sally White	04/15/73	$4,125.00

Property:

Type	OwnerSIN	Tax
Condo	318-231-089	$6,312
Single	435-010-987	$1,459
⋮	⋮	⋮
Cottage	010-345-180	$2,010
Single	123-456-091	$4,590

City:

CityCode	CityName	Population
4518	Ottawa	990,310
4679	Nepean	725,910
⋮	⋮	⋮
2310	Kingston	980,000
8927	Hull	378,124

Figure 1.3: Example Relations from Property Tax Assessor's Office

Given these pieces of information, we intend to answer the following declarative query:

Query: *Find the name, city and tax information of all property owners.*

To illustrate the large difference in the execution cost that exists among various QEPs, we compare two QEP's for this query using a simple *cost model* based on the number of I/O operations. Since the disk I/O is usually the dominant factor[1] in the query response time, we assume the cost of evaluating a QEP is given by the total number of all tuples read and written to generate the final answer. In addition, we also consider the cost of joining the information of two relations that consists of reading the two input relations and then writing the resulting relation back on to the disk.

From the set of many alternatives, we analyze only two QEPs shown in Figure 1.4. The QEPs execute the answer to the query as described below:

Query Evaluation Plan 1: In the first QEP, the relations `TaxPayer` and `City` are joined to determine the city information for each person where he or she lives. This join generates an intermediate relation. This intermediate relation is then joined with the relation `Property` to compute the final results.

The first join requires the relations `TaxPayer` and `City` to be read. This results in $12 \times 10^6 + 4000$ reads. Assuming the result of this join is written back to the disk, it would require 12×10^6 writes. Note that the size of the intermediate result is 12×10^6. The second join requires the above intermediate relation and the relation `Property` to be read. This involves an additional $12 \times 10^6 + 3 \times 10^6 = 15 \times 10^6$ reads. Since the final result contains 3×10^6 tuples, it requires that many write operations. Hence the total number of operations required for the first QEP is $N_1 = (12 \times 10^6 + 4000) + (12 \times 10^6) + (15 \times 10^6) + (3 \times 10^6) = 42,004,000$.

Query Evaluation Plan 2: Joining the relations `TaxPayer` and `Property`, we obtain an intermediate relation with all the property owners. This intermediate relation

[1] Actually, a realistic cost model should include many other factors such as various join algorithms, availability of indexes and other auxiliary access methods, effects of caching and available memory, data skew etc.

is then joined to the relation City to determine the information of the city for each tax payer.

In the second QEP, the first join requires $12 \times 10^6 + 3 \times 10^6$ reads and 3×10^6 writes. The second join requires another $3 \times 10^6 + 4000$ reads and 3×10^6 writes. Hence the total number of operations required for the second QEP is $N_2 = (12 \times 10^6 + 3 \times 10^6) + (3 \times 10^6) + (3 \times 10^6 + 4000) + (3 \times 10^6) = 24,004,000$, which is almost half of the first QEP.

Using a simple cost model, this short example with two different QEPs illustrates that the cost of one plan can be half of the other. Most of the real-world queries are complex queries with many relations, and with a more sophisticated realistic cost model, one can generate QEPs with substantially different costs. The task of a query optimizer is to judiciously analyze the possible QEPs and choose the one with the minimal cost.

1.1.2 Goals of Query Optimization

The main goals of a query optimizer are that of minimizing both the *response time* and *resource consumption*. These optimization goals are often conflicting. For example, a QEP which computes the result of a query quickly but requires all available memory and CPU resources is probably not desirable because it would virtually deprive other users from accessing the database.

Finding good solutions for a query is resource (time) intensive, but can reduce the

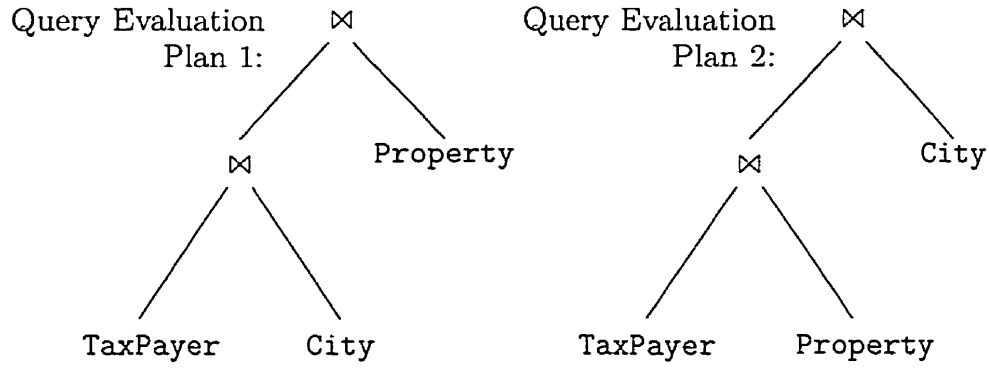


Figure 1.4: Two Alternative Query Evaluation Plans

actual evaluation cost considerably. For example, let T_o be the time taken to find an optimal QEP and let T_c be the time taken to execute the optimal QEP and obtain the results. If the average time taken to execute a random QEP is T_{avg}, then ideally we want $T_o + T_c \ll T_{avg}$. Obviously, there are trade offs to be made, but clearly, the *quality of the solution* and the *optimization time*, T_o are critical.

As queries get more complex (in terms of the number of relations involved or alternative algorithms for computing an operator) the number of potential alternative QEPs that should be considered explodes. The number of alternatives quickly increases with the number of relations etc. into the order of millions, while the differences between the cheapest and most expensive QEP can easily be several orders of magnitude. Even with simple string queries with n relations, there are $(2(n-1))!/(n-1)!$ different join orders. For joins involving small number of relations, this number is acceptable; for example, with $n = 5$, the number is 1680. However, as n increases, this number rises quickly. With $n = 10$, the number of different join orders is greater than 176 billion!

As a rule of thumb, for small queries with no more than four or five relations all QEPs can be generated within a few seconds. In this case the optimization time, T_o, is often a fraction of the response time improvement gained.

Apart from the fact that the execution space of QEPs is usually very large, the computation of a single join operation itself is one of the most time consuming tasks. To compute the join of N relations, $(N-1)$ dyadic join operations have to be performed. Since the size of the relations joined determines the cost of a single join operation, the order in which all N relations are joined should be chosen in such a way so as to minimize the overall cost of computing the join of N relations.

Unfortunately, finding the optimal join order has been proven to be an NP-hard problem [24, 61, 119], while, at the same time, it is one area where considerable cost benefits can be derived. Only for specific join ordering cases, exact and optimal solutions have been obtained [17, 61]. In addition, most of the optimization techniques proposed in the literature cannot be applied to large queries. For example, two popular early day database systems, namely, System R and Ingres use algorithms that essentially perform an exhaustive search over the entire QEP domain. This is

probably adequate in their implementation because they do not allow large queries with more than 15 joins.

With the above as a background we shall present the research problem which we intend to study, and the objectives of this research.

1.2 Research Problem and Objectives

1.2.1 Research Problem

The problem of optimizing a query can be formally stated as follows:

Problem 1 *Given a relational query* $\mathcal{Q}$, *an execution space* $\mathcal{E}$ *and a cost function* $\mathcal{C}(\mathcal{QEP})$ *over elements of* $\mathcal{E}$, *find the query evaluation plan* $\mathcal{QEP} \in \mathcal{E}$ *such that,*

(1) $\mathcal{QEP}$ *computes* $\mathcal{Q}$

(2) There does not exist $\mathcal{QEP}' \in \mathcal{E}$ *such that* $\mathcal{QEP}'$ *also computes* $\mathcal{Q}$ *and* $\mathcal{C}(\mathcal{QEP}') < \mathcal{C}(\mathcal{QEP})$.

where the execution space $\mathcal{E}$ *is the space of QEPs considered by the optimizer.*

As discussed earlier, finding an optimal QEP is a computationally intractable problem, especially when the query involves more than, say, ten relations. One obvious approach is to decrease the size of the execution space $\mathcal{E}$, but although it would reduce the time and/or space requirement, it has the tradeoff of reducing the chances of finding good plans. Another approach consists of *estimating* the costs of QEPs using the standard statistical information available in the database catalog. In fact, most commercial DBMSs use some form of statistics on the underlying data information about the available resources in order to estimate the cost of a query plan approximately. Since these statistics are used to provide approximate estimates of query costs, the validity of the optimizer's decisions may be affected. On the other hand, since optimizers use these costs only for comparison purposes, approximate estimates for the costs of the QEPs are usually sufficient as long as these estimates are reasonably close to the actual cost of executing the QEP. For example, if the actual

Notation	Explanation
N_R	Number of tuples in relation R
b_R	Number of disk blocks storing relation R
s_R	Size of a tuple of relation R in bytes
bf_R	Blocking factor of relation R. This is the number of tuples of relation R that fit into one disk block
$\delta(A, R)$	Number of distinct values of attribute A in relation R.
$\varphi(A, R)$	Selection cardinality of attribute A in relation R. Given a relation R and an attribute A of the relation, $\varphi(A, R)$ is the average number of records that satisfy an equality condition on attribute A.

Table 1.1: Statistical Information found in a typical DBMS Catalogue

time units of execution of two plans P_1 and P_2 are 10 and 20 respectively, and the estimated times are 5 and 15, the optimizer will still pick P_1 as the less expensive plan, and this will be done correctly. On the other hand, if the estimated times are 5 and 4, the optimizer will pick the suboptimal plan P_2. This is an extremely important issue because costs of plans can differ considerably, and choosing a suboptimal plan can result in severe performance degradation, defeating the very purpose of query optimization.

1.2.2 Information in DBMS Catalogue

Query optimizers make use of the statistical information stored in the DBMS catalogue to estimate the cost of a QEP. Table 1.1 lists some of the commonly used statistical information utilized in most of the current commercial DBMSs.

In addition to these pieces of information, most DBMSs also maintain information about the indices[2] in their catalogues. These pieces of information include values such as the average fan-out, number of levels in an index etc.

[2]Indices refer to the actual data structures used to store and retrieve the physical records of base and intermediate relations. Although indices play an important role in how a QEP is evaluated, we will not deal with them in this work.

1.2.3 Approximately Estimating the Cost of a QEP

Since the actual evaluations of all possible QEPs in space $\mathcal{E}$ is very difficult, like all other investigators in this area, we also resort to approximately estimating the costs of QEPs.

To initiate the discussion we consider the cost of a join, which we know is one of the most commonly occurring and expensive relational operations. We consider the cost $\mathcal{C}$ of joining two relations R_1 and R_2 on a common attribute X, using the nested-loop join algorithm [122]. If b_{R_1} and b_{R_2} are the number of disk blocks required for relations R_1 and R_2 respectively, then the cost formula in terms of number of disk accesses is,

$$\mathcal{C} = b_{R_1} + b_{R_1} \times b_{R_2} + \left(\frac{js \times |R_1| \times |R_2|}{bf_{R_1 R_2}} \right)$$

where js is the join selectivity[3] of these other operators and $bf_{R_1 R_2}$ is the blocking factor for the resulting relation.

From the above cost formula, it is apparent that the following quantities are crucial for the query optimizer in order to estimate the cost of such an operation:

(a) *Query Result Sizes:* The cost formula for the above join algorithm depends on the sizes of the input relations, which is obviously true for nearly all relational operators. In a query with several operators, an input to an operator may itself be the result of another operator(s). This shows the importance of developing techniques to estimate the result sizes of these other operators.

(b) *Query Result Distributions:* The result size of a join (or any other relational operator) depends mainly on the data distribution(s) of its input relation(s). Again the input to this operator may itself be the relation which results from invoking another operator(s). This means that one also needs techniques to estimate the distribution of a query result.

(c) *Disk Access Costs:* This is the cost of searching for, reading, and writing data blocks that reside on secondary storage, mainly on disk. The cost of searching

[3]Selectivity of an operator is the ratio of the result size and the product of its input sizes. For the join operator, the join selectivity $js = |(R_1 \bowtie_c R_2)|/(|R_1| \times |R_2|)$

for records in a file depends on the type of access structures on that file, such as its ordering, hashing, and primary or secondary indexes. In addition, factors such as whether the file blocks are allocated contiguously on the same disk cylinder or scattered on the disk affect the access cost.

1.2.4 Research Objectives

In Section 1.2.3 we argued that *estimating the query result sizes*, among others, is crucial for calculating the cost of a QEP. Hence the design of a good query optimizer should involve developing techniques to efficiently and accurately estimate the various query result sizes. *The objective of this thesis is to develop new techniques that would provide more accurate query result size estimation results than the state-of-the-art methods used in the current database systems.* There are other topics of concern in query optimization such as implementation algorithms for various relational operations, physical access plans for retrieving records etc., but they are not addressed in this thesis.

In order to completely solve the above estimation problem, it is necessary to develop result size estimation techniques for each of the relational operators. Most of the current commercial DBMSs employ methods based on one or more of the following techniques to estimate the above quantities:

1. Sampling based techniques
2. Parametric techniques
3. Probabilistic counting techniques and
4. Non-parametric or histogram based techniques.

These techniques are briefly reviewed in the next chapter. The accuracy of these estimates is often of critical importance since the selection of an optimal QEP from an exponential number of alternatives solely depends on these estimates. Despite the widespread usage of the above techniques in many commercial database systems, their accuracy has not been studied extensively.

Since the accuracy of the query result size estimations, in turn, depends on the accuracy of the approximation to the underlying data distributions, we focus in this thesis on developing new techniques that would provide more accurate approximations to the true data distributions than the above methods. Specifically we introduce two new histogram-like techniques that provide more accurate approximation to the data distributions than the traditional histogram methods. These techniques are, namely, the *Rectangular Attribute Cardinality Map* (R-ACM) and the *Trapezoidal Attribute Cardinality Map* (T-ACM). The R-ACM approximates the data distribution using a user-specified tolerance, whereas the T-ACM approximates the data distribution using the well-known trapezoidal-rule of numerical integration.

The result size estimation techniques studied in this research work involve the result estimation of two of the most important and commonly occurring relational operators[4], namely, *joins* and *selections* using histogram-like techniques.

It is important to note that the estimation of intermediate result sizes of any query is prone to error propagation which "travels" upward along the query operator tree. Thus this error propagation can result in a final value for the query result-size that can be significantly different from the true result size.

1.3 Outline of the Thesis

This thesis is structured as follows. Chapter 2 describes the related work in result size estimation. Although this survey is by no means a complete collection of every research work in this area, it is fairly comprehensive. Here we discuss most of the relevant work in the area of query result size estimation, including *sampling based techniques*, *parametric techniques*, *probabilistic counting techniques* and non-parametric or *histogram-based techniques*. In Chapter 3, we describe the *Rectangular Attribute Cardinality Map* (R-ACM), which is the first major contribution of this thesis. Chapter 4 continues along the same vein, and we introduce the *Trapezoidal Attribute Cardinality Map* (T-ACM), which is the second major contribution of this

[4]The result size estimation of the *project* operation is not addressed here, since histogram based methods are not suitable for such operations.

thesis. In these chapters, we also provide a fairly extensive mathematical analysis for these new structures, and provide some preliminary experimental results for query result size estimation based on synthetic data.

Chapter 5 is a prototype validation of the ACMs on real-world databases. Here we use real-world databases such as the U.S. Government CENSUS database and the NBA Performance Statistics Database for our experiments. We shall also use mathematical distributions such as the Zipf distribution and multi-fractal distribution to generate a variety of potential distributions for the value domains from these real-world databases. These powerful distributions provide us with unlimited possibilities of data distributions, enabling us to carry out an exhaustive set of experiments, and to thus make intelligent comparisons between the various techniques.

Our experimental study in Chapter 6 involves the industry-standard TPC-D benchmark tests. Here we use the TPC-D database and queries supplied by the Transaction Processing Performance Council to benchmark the newly discussed ACMs. We also compare the performance of our new techniques to that of the current state-of-the-art methods, such as the equi-width and equi-depth histograms. Unlike the prototype validating experiments of Chapter 5, where we tested the schemes with *simple* synthetic queries against real-world databases, the benchmarking experiments in this chapter include *complex* simulated real-world query types against a scalable simulated real-world database. We use the results of these experiments to verify the theoretical results developed in Chapter 3 and Chapter 4, and also to demonstrate the superiority of our new techniques over the current state-of-the-art histogram based estimation methods.

The estimation accuracy of the R-ACM and T-ACM depends on their build-parameters, namely the tolerance value, τ and the slope of the trapezoid respectively. As we shall see, the estimation accuracy of the R-ACM can be arbitrarily increased by reducing the tolerance value, τ. Hence building an optimal R-ACM entirely depends on the available storage. As opposed to this, given a storage space, the estimation accuracy of a T-ACM can be improved by finding the optimal slopes of the trapezoidal sectors. In Chapter 7, we discuss two different methods for determining suitable trapezoidal sectors.

Finally in Chapter 8, we provide a summary of our results and identify the directions for future research work.

As a final note, we would like to emphasize that this thesis work only deals with the problem of evaluating the size of the relational operations. This is, in fact, as we earlier mentioned, a small, but important, part of the much bigger problem known as query optimization. The problem of query optimization requires comparison of an exponential number of potential alternative QEPs of a given relational query and choosing the least expensive one. Thus query optimization involves many inter-related tasks such as constructing operator trees for every possible QEP of the given query, evaluating the cost of each QEP in terms of query result-size, disk access time, etc., executing the selected query and so on. Due to the enormity and complexity of the problem, we will not address any of these latter issues in this research work. Instead, we conclude this introduction by noting that the problem of evaluating the size of the relational operations (as addressed in this thesis) is probably the most fundamental and crucial one for query optimization. Consequently finding more accurate strategies for solving this kernel problem is the single most important issue in the design of superior query optimizers for any relational database system.

Some of the theoretical results presented in Chapters 3 will appear in [106]. Some of the results from Chapter 4 are currently being reviewed for publication. They can also be found as Carleton University Technical Reports in [107, 108]. The research results on the prototype validation and testing of the R-ACM and T-ACM presented in Chapter 5 have been published in [132, 133] respectively. Some of the results from the TPC-D benchmarking experiments presented in Chapter 6 are currently being reviewed for publication. They can also be found as a Carleton University Technical Report in [105]. We are presently compiling the results obtained from Chapter 7 for potential publication.

Chapter 2

Related Work

Query optimization has been a very active research field in the database community for the last two decades. In this chapter we shall briefly discuss some of the important results that have been developed in this field. It is obviously impossible to mention every single research result available. However the intention of this chapter is to present to the reader a reasonable view of the state of the art, and also to prepare the stage for the new schemes we intend to present.

Apart from the area itself being vast, the techniques used to optimize a query search itself are varied. At the top most level, the query optimizer has to distinguish the various query evaluation plans. This is a problem in its own right. The question of pruning these query evaluation plans, and discarding the less promising ones without actually processing them in any depth, is itself, a huge area of research. Additionally, for any *given* query evaluation plan, the question of estimating its efficiency without actually processing the query is a problem that has captivated much research. Over the past two decades, we estimate there are probably thousands of important and less important results claimed for all of these respective areas. Since it is impossible to survey or mention all of them in this chapter, we shall go into greater detail only in the case of results which are pertinent to our central thesis. In papers which are of peripheral interest, a few sentences highlighting the contributions of the author is all that we will include.

We divide this chapter into four sections, each dealing with a distinct area in

query optimization. In the first section, we follow a top-down approach to query optimization, utilizing the general evaluation procedure that follows as a framework for the specific techniques developed in query optimization research. In the second section, we review the major techniques developed in join tree re-ordering for query optimization. The third section of this chapter is devoted to the review of the work done in parallel and distributed query optimization. The final section, which has a much closer relationship with this thesis, deals with the statistical estimation and result-size approximation schemes used in database systems.

2.1 A Top-down Approach to Query Optimization

Early work in query optimization essentially followed two distinct tracks, namely, the bottom up and top down. Database researchers found the query optimization problem, when posed in a general context, to be both difficult and complex - and indeed, intractable with regard to optimality. They, in the early days, mainly undertook a bottom-up approach, considering only special cases. This included, for example, finding "optimal" strategies to implement major relational operations and finding "optimal" methods to execute simple subclasses of queries. As better strategies emerged with time, researchers attempted to obtain "optimal" strategies for relatively larger and more complex queries by composing the results from the smaller building blocks.

As the number and sizes of the commercial databases grew, the need for functional optimization systems resulted in the development of full-scale query evaluation techniques. These techniques provided more general solutions and handled query optimization in a uniform, though heuristic manner [2, 89, 102, 111, 118, 141]. Obviously, generating a query optimization plan out of smaller building blocks, often did not achieve optimal system efficiency. This led the researchers to move towards a top-down approach that incorporated more knowledge about special-case optimization opportunities into the general procedures. This top-down approach also incorporated many general algorithms, which were augmented by combinatorial cost-minimization techniques for choosing between the various potential query evaluation plans.

Using the top-down approach, the general query optimization can be divided into

the following steps:

(1) Use logical transformation techniques to change the given query so that (a) the query is represented in a standardized form, (b) the query is simplified such that the same operation is not computed twice, and (c) the query is ameliorated so as to organize the computation to apply procedures developed for special case scenarios.

(2) Represent the transformed query into alternative sequences of simple elementary relational operations that can be computed using known implementation techniques. This makes it possible to compute the associated execution cost for the evaluation plan. This step generates a set of potential *access plans* that simplify the process of evaluating the original query.

(3) Compute the execution cost for each access plan in the above access plan set.

(4) Execute the cheapest access plan and return the results.

The first step of this procedure involves logical transformation of the query, and thus it is, generally, independent of the underlying data. Hence this step can often be handled at compile time. But in order to properly carry out steps (2) and (3), and to generate optimal query evaluation plans, we need to have sufficient knowledge about the underlying data distribution.

The fact that steps (2) and (3) require some prior knowledge about the underlying data poses a number of difficulties. First of all, if the data distribution is dynamic (i.e: due to frequent updates), steps (2) and (3) can be carried out only at run time. Consequently we need to sacrifice some gain in the overall execution efficiency in order to minimize the optimization cost. Another difficulty is that the DBMS should maintain a catalogue (or a meta-database) about the pertinent statistical information about the underlying data distributions. It is obvious that in order to benefit from such an optimization strategy one must compare the costs of gathering and maintaining such information in the DBMS catalogue against the cost savings in the overall optimization process.

2.1.1 Query Transformation

Any relational algebraic query can be transformed into a number of semantically equivalent expressions. In this section, we discuss the work on transformation of a given expression into an equivalent query by means of well defined transformation rules. The objectives of query transformation can be described as the following:

1. Obtaining a normalized starting point for query optimization (*called standardization*).
2. Eliminating the duplication of effort (*called simplification*) and
3. Generating query expressions that have superior evaluation performance compared to the initial query (*called amelioration*).

A number of works [68, 70, 111, 141] have been done to define a normalized starting point for representing a query for the purpose of query optimization.

We have already mentioned that there might be several semantically equivalent expressions representing a given query. It is obvious that one of the main differences between any two equivalent expressions is their degree of redundancy [51]. In other words executing a query with redundant expression would result in carrying out a number of unnecessary duplicated operations. Hence query simplification step should involve the transformation of a redundant query expression into an equivalent non-redundant one by applying simple logical transformation rules.

A redundant query expression can be simplified by applying various transformation rules listed below.

Transformation Rules

A transformation or equivalence rule says that expressions of two forms are equivalent. This implies that we can transform either expression to the other while preserving equivalence. When two relations have the same attributes ordered in a different manner, and equal number of tuples, the two expressions that generate them are considered to be *preserve equivalent.* Modern-day query optimizers heavily use equivalence rules to transform initial expressions into simpler logically equivalent expressions.

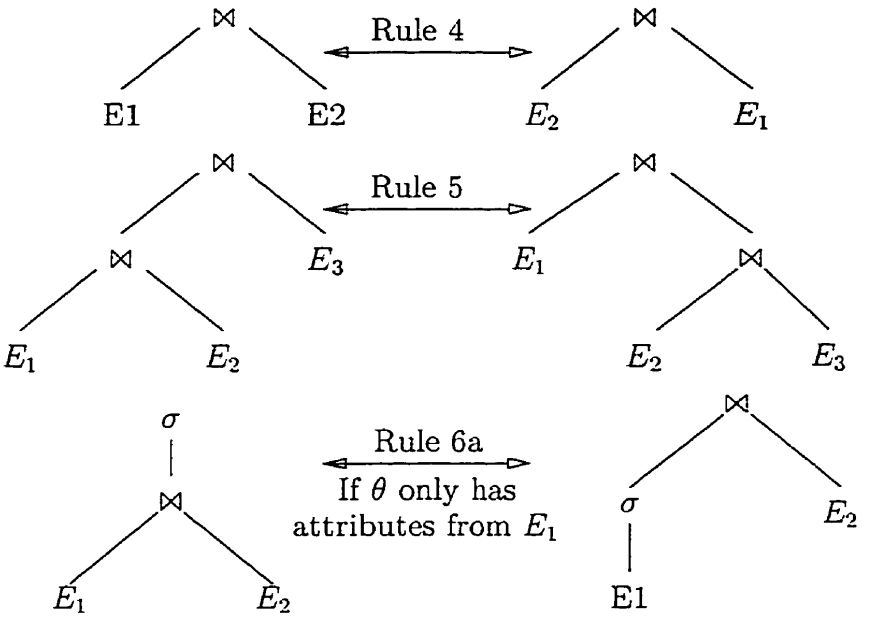


Figure 2.1: Equivalent Transformations.

A number of commonly used general equivalence rules for transforming relational algebraic expressions are given below. Three equivalent transformations are shown in Figure 2.1. The predicates in the query expressions are denoted by θ_i, attributes are denoted by L_i. and the algebraic expression is denoted by E_i. Since the result of a relational algebraic expression itself is a relation, a relation name R can appear anywhere in place of an expression.

1. Conjunctive selection operations can be simplified into a sequence of individual selections. This transformation operation is known to as a cascade of σ.

$$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E)).$$

2. Selection operations applied successively onto a relation are commutative.

$$\sigma_{\theta_1}(\sigma_{\theta_2}(E)) = \sigma_{\theta_2}(\sigma_{\theta_1}(E)).$$

3. In a sequence of projection operations, only the final operation is required to

compute the results. This transformation operation is known as a cascade of π.

$$\pi_{L_1}(\pi_{L_2}(\ldots(\pi_{L_n}(E))\ldots)) = \pi_{L_1}(E).$$

4. Selections can be processed together with Cartesian products and θ—joins (also written as theta-joins).

   (a) $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$.
   This expression is indeed the definition of the theta join.

   (b) $\sigma_{\theta_1}(E_1 \times_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$.

5. A theta-join operation is commutative.

$$E_1 \bowtie_\theta E_2 = E_2 \bowtie_\theta E_1.$$

   Note that the natural join operation is also commutative, since the natural-join operator is actually a special case of the theta-join operator.

6. (a) Natural join operations are associative.

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3).$$

   (b) Theta joins can be associative as shown below:

$$(E_1 \bowtie_{\theta_1} E_2) \bowtie_{\theta_2 \wedge \theta_3} E_3 = E_1 \bowtie_{\theta_1 \wedge \theta_3} (E_2 \bowtie_{\theta_2} E_3).$$

   where θ_2 involves attributes from only E_2 and E_3. Since any of these predicates can be empty, it can be seen that the Cartesian product ($\times$) is an associative operation. It is important to observe that the properties of commutativity and associativity of join operations are important for join reordering in query optimization.

7. The selection operation and the theta join operation are distributive under the conditions stated below:

(a) Selection operation distributes over the theta join operation when all the attributes in selection predicate θ_0 involve only the attributes of one of the expressions (E_1) being joined.

$$\sigma_{\theta_0}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_0}(E_1)) \bowtie_\theta E_2.$$

(b) Selection operation distributes over the theta distribution when selection condition θ_1 involves only the attributes of E_1 and θ_2 involves only the attributes of E_2.

$$\sigma_{\theta_1 \wedge \theta_2}(E_1 \bowtie_\theta E_2) = (\sigma_{\theta_1}(E_1)) \bowtie_\theta (\sigma_{\theta_2}(E_2)).$$

8. The projection operation and the theta join operation are distributive.

(a) Suppose the attributes L_1 and L_2 belong to the expressions, E_1 and E_2, respectively. Assume that the join predicate θ has only attributes in $L_1 \cup L_2$. This means,

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = (\pi_{L_1}(E_1)) \bowtie_\theta (\pi_{L_2}(E_2)).$$

(b) Consider a join $E_1 \bowtie_\theta E_2$. Assume the attributes, L_1 and L_2 belong to the expressions E_1 and E_2, respectively. Assume the attribute L_3 is from the expression E_1 which are involved in join predicate θ, but are not in $L_1 \cup L_2$, and let attribute L_4 of E_2 that are involved in join condition θ, but are not in $L_1 \cup L_2$. This means,

$$\pi_{L_1 \cup L_2}(E_1 \bowtie_\theta E_2) = \pi_{L_1 \cup L_2}((\pi_{L_1 \cup L_3}(E_1)) \bowtie_\theta (\pi_{L_2 \cup L_4}(E_2))).$$

9. The union operation and the intersection are commutative set operations.

$$E_1 \cup E_2 = E_2 \cup E_1 \text{ and } E_1 \cap E_2 = E_2 \cap E_1.$$

It is important to note that set difference operation is not commutative.

10. The union operation and intersection operation are associative set operations.

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3) \text{ and } (E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3).$$

11. The selection operation distributes with the union, intersection, and set difference operations.

$$\sigma_P(E_1 - E_2) = \sigma_P(E_1) - \sigma_P(E_2).$$

When substituting the set difference $(-)$ with either one of $\cup$ or $\cap$, selection operation is distributive with the union and intersection operations.

12. The projection and union operations are distributive.

$$\pi_L(E_1 \cup E_2) = (\pi_L(E_1)) \cup (\pi_L(E_2)).$$

The proof of the above equivalence is straightforward and can be found in [92]. Also, the preceding list is only a partial list of equivalences. More equivalences involving extended relational operators, such as the outer join and aggregation, can be constructed from the above list.

The process of query simplification does not always produce a unique expression. Many non-redundant expressions can exist that are equivalent to the one generated by a given equivalence transformation. The evaluation of expressions corresponding to a given query may differ significantly with respect to performance parameters, such as the size of the intermediate results and the number of relation elements accessed.

2.1.2 Query Evaluation

In this section we provide a brief overview of the methods for evaluating relational query components of varying complexity. This includes queries with one-variable expressions, two-variable expressions, and multi-variable expressions. These evaluation methods provide us with a set of building blocks for evaluating any general query expression. The execution cost of this evaluation step forms the final part of the overall query optimization cost.

One-Variable Expressions

These are the simplest algebraic expressions used to select a given set of attribute values from a single relation. A simple approach to evaluate such an expression is to read every corresponding attribute value of the relation, and to check whether it satisfies each term of the expression. Obviously this approach is very expensive, especially when the query expression involves large relations. In these situations, there has been many techniques proposed to minimize the number of attribute values accessed and the number of tests applied to an accessed attribute value. These are briefly discussed below.

One of the commonly employed techniques to minimize the number of record accesses is by means of data structures that provide direct access paths, so that an exhaustive sequential search can be avoided. This can be accomplished by maintaining the relation in a sorted form with respect to one or more attributes so that records can be easily accessed. This also facilitates a binary search on the sorted attribute values. One of the obvious uses of such a technique is the evaluation of range queries [13, 27].

Another commonly used technique is hashing. Hashing technique has many variants and its major advantage is that it provides fast direct access to records without requiring that the attribute values are maintained in a sorted order.

In addition to hashing, techniques using *indexes* play an important role in providing direct and ordered access to records. Indexes can be often combined with multi-list structures. The idea of an index is based on a binary relationship between records being accessed. Specifically the index method uses tuple identifiers or keys to establish a binary relationship between records, so that traversal becomes much faster than a sequential search. The indexing method can be either *one-dimensional* or *multidimensional*. A one-dimensional index implements record access based on a single relation attribute, whereas a *multidimensional index* implements record access based on a combination of attributes. Two popular approaches for implementing one-dimensional indexes are namely, ISAM [25] and B-tree [6] structures. A detailed discussion on multidimensional index structures can be found in the work by Bentley and Friedman [8].

The logical transformation rules can often be applied at run time to a query expression to minimize the number of tests applied to an accessed record (or attribute value) of a relation. Changing the order in which individual expression components are evaluated can also result in performance increase. A number of techniques, utilizing *a priori* probabilities of attribute values, to obtain optimal evaluation sequences for specific cases have been discussed in [52].

Two-Variable Expressions

Query expressions that describe conditions for the combination of records from two relations are known as two-variable expressions. Usually two-variable expressions are formed by monadic sub-expressions and dyadic sub-expressions. Monadic expressions restrict single variables independently of each other, whereas dyadic expressions provide the relationship between both variables. In what follows we briefly discuss, the basic strategies for the evaluation of a single dyadic expression, and methods for the evaluation of any two-variable expressions.

Algorithm 2.1 Nested_Iteration_Join

```
for i := 1 to N1 do
   read the i^th element of R1;
   for j := 1 to N2 do
      read j^th element of R2;
      compute the join based on join predicate;
   end;
end;
```

There are two major approaches to implementing the join operation, namely order-dependent and order-independent strategies. The *nested iteration* method [122] is a simple approach that is independent of the order of record access. In this method, every pair of records from the relation is accessed and concatenated if the join predicate is satisfied. A nested iteration based join is listed in Algorithm 2.1.

Let N_1 and N_2 be the number of records of the relations read in the outer and inner loops respectively. It is clear that $N_1 + N_1 * N_2$ disk accesses are required to evaluate the dyadic term, assuming that each record access requires one disk access.

We can improve the nested iteration method by using an index on the join attribute(s) of the relation R_2. This strategy does not require accessing R_2 sequentially for each record of R_1 as the matching R_2 records can be accessed directly [49, 74]. Hence using an index we only need $N_1 + N_1 * N_2 * j_{12}$ record accesses, where j_{12} is a join selectivity factor representing the Cartesian product of R_1 and R_2.

In a paged-memory environment, we can extend the idea of the nested iteration method to the *nested block method* [70]. In this method, we use a main memory buffer to hold one or more pages of both relations, where each page contains a set of records.

The nested block algorithm is essentially similar to the nested iteration method. The only difference is that memory pages are read instead of single records of the relation. Using this approach we reduce the number of disk accesses required to compute the join to $P_1 + (P1/B1) * P_2$, where P_1 and P_2 are the number of pages required to store the outer and inner relations of the algorithm respectively and B_1 is the number of pages of the outer relation occupying the main memory buffer. It is easy to observe that it is always more efficient to read the smaller relation in the outer loop (i.e., make $P_1 < P_2$). If we maintain the smaller relation entirely in the main memory buffer, then we need only $P_1 + P_2$ disk accesses to form the join.

Another approach, based on the order in which records of the relations are retrieved, is known as the *merge method* [12, 122]. In this approach, both relations are scanned in ascending or descending order of join attribute values and merged according to the join predicate. We require almost $N_1 + N_2 + S_1 + S_2$ disk accesses to form the join, where S_1 and S_2 are the number of disk accesses needed to sort the relations being joined. It is obvious that this method is an excellent choice when both relations are already sorted by the same criterion. Of course, when the smaller relation can be completely maintained in the main memory buffer, the nested block method is more efficient. Whenever an indexing mechanism exists, then when one relation is much larger than the other, nested iteration with indexes is preferable.

The strategies developed to evaluate one-variable expressions and computation of dyadic terms can be used to evaluate any *arbitrary two-variable expressions*. These strategies mainly differ in the manner they implement the access paths and methods, namely indexing mechanisms and the order in which the component expressions are

evaluated.

Blasgen and Eswaran [11], Niebuhr and Smith [102], and Yao [142] describe many different approaches and compare their evaluation efficiency. It is their conclusion that usually there is often no *a priori* best algorithm for a general evaluation process and thus the query optimizer must rely on heuristics as the only alternative is to choose the optimal plan through an expensive cost comparison of potentially many alternatives for each query.

Multi-variable Expressions

A query optimizer designed to process any arbitrary query should incorporate strategies for evaluating multi-variable expressions.

One of the popular techniques, known as *parallel processing* of query components evaluates an expression by avoiding repeated access to the same data. This is usually achieved by simultaneous or parallel evaluation of multiple-query components. Usually all monadic terms associated with a given variable are evaluated first and all dyadic terms in which the same variable participates are partially evaluated parallely [111]. This also includes parallel evaluation of logical connectors such as AND existing among the various terms, that always minimizes the size of intermediate query results [68]. A strategy, using a similar approach, where aggregate functions and complex sub-queries are computed concurrently, has been proposed by Klug [74]. Parallel processing of query components require appropriate scheduling strategies for the overall efficient optimization. A few interesting techniques for parallel scheduling for query processing have been proposed by Schmidt [120].

Another strategy that is useful for scheduling parallel optimization is the pipelining of operations that can work on the partial output of preceding operations [125, 142]. One possible example is the operations restriction and projection can be pipelined with a minimal buffer for data exchange, without the need for the creation and subsequent accessing of a potentially big temporary relation.

Another method known as the *feedback* technique combines the tasks of simultaneous evaluation of query components and pipelining, using partial results of a join operation [23].

2.1.3 Access Plans

Having reviewed the various techniques for evaluation of query components that can be used as building blocks of an arbitrary general query evaluation algorithm in the previous section, we shall now discuss the final step of our query evaluation process. This step requires the combination of these building blocks into an efficient and optimal evaluation procedure for an arbitrary standardized, simplified, and ameliorated query expression.

The access plan selection strategies use the transformed and simplified query, the storage structures and access paths, and a simple mathematical cost model to generate an optimal access plan. This is usually achieved by the following procedure:

1. Find all possible logical access plans for evaluating the given query. A logical access plan is defined as a sequence of operations or of intermediate results leading from existing relations to the final result of a query.

2. Obtain details of the physical representation of data such as access paths, statistical information stored in the data-dictionary.

3. Using the mathematical cost model based on the access paths and processing costs, select the cheapest access plan.

Generation of Access Plans

The sequence of operations or intermediate results leading from the existing data structures to a query result is known as an access plan for a query. An ideal query optimizer should generate an optimal access plan with a minimal optimization effort.

Two completely different methods were proposed by Smith and Chang [125] and Yao [142]. Smith and Chang use a fixed set of query transformation rules. This approach usually results in a single access plan, which need not be optimal. Yao's approach generates all non-dominated access plans that are possible in a given scenario. Obviously generating every single possible access plan becomes prohibitively expensive for complex queries.

In addition to the above methods, there are other strategies that resort to methods that lie between heuristic selection and detailed generation of alternative access plans. This includes an SQL nested block method based on a hierarchical technique used in System R [122]. In this strategy, in addition to generating and comparing the evaluation plans for each query block, the sequence in which how query blocks are evaluated is also computed. The disadvantage of this method is that as this approach requires a user-specified block structure, the initial query normalization step is pushed into the actual query processing phase [70]. INGRES also uses a somewhat similar approach [141].

Cost Analysis of Access Plans

The query optimizer chooses a physical access plan for a given query either on the basis of a set of heuristic rules or on the basis of a mathematical cost model involving the underlying data structures and access methods. Merrett discussed a top-down approach for cost analysis based on the storage structures [93]. Though the techniques presented in [93] have been significantly improved on since 1977, it is worth mentioning that Merrett's ideas were seminal and initiated the work in the area of cost analysis in relational database systems. In particular, in [93], Merrett provides general principles that allow a database analyst either to quickly estimate the costs of a wide variety of alternatives, or to analyze fewer possibilities in depth. He also provides a number of techniques that can be used at any level of the iterative design process.

A brief review of some of the cost models and their use in the query optimization is given below.

Even though working storage requirements [23, 70, 116] or CPU costs [122] are considered to be important factors by some researchers, the number of secondary storage accesses or I/O operations is considered to be the most important one in designing a cost model for analyzing any physical access plan. The number of secondary storage accesses is mainly dependent on the size of the operands used in the query, the type of access structures used, and the size of the temporary buffers available in the main memory.

In the initial stages of the evaluation, the sizes of most of the operands are known,

at least approximately. In addition, any available index methods for the attribute values involved are also known. But in the later stages of the evaluation, most intermediate relations are the results of preceding operations, and consequently, the cost model must estimate their size by using information about the original data structures and any other information available from the DBMS catalogue, such as the selectivity of the operations already performed on them. Unfortunately, there are no generally accepted well defined formulae for estimating the size of intermediate results. Some of the on-going research works, including the ideas proposed in this thesis work, consider various approaches to approximately estimate the intermediate result sizes.

It is obvious that one can construct a simpler model, using only a few parameters by imposing restrictions on the assumptions about the data. The result size estimate analysis discussed in [122] uses only information already available in the database catalogue. Obviously this requires many assumptions about data and queries to complete the cost model [2]. On the other hand, the result size estimate analysis proposed by Yao assumes that detailed selectivity data is known [142], without stating how this detailed information can be obtained.

In the late 1980s and early 1990s, researchers began to carefully describe and critically analyze all underlying assumptions about database characteristics. The objective was to formally generate valid parameters for the cost model that enable the optimizer to

1. estimate a result size for any relational operation, and

2. estimate parameter values for intermediate results based on the previous results and base relations.

Cost models based on such techniques relate the database state at run time as the result of a random process. This random process is assumed to generate relations from the Cartesian product of the attribute value domains, based on some probability distribution (usually assumed to be uniform) and by invoking certain other general principles such as the functional dependencies of the underlying database schema

[43, 113]. These assumptions enable one to derive parameters in order to compute the size of any intermediate result of more complex relational operations.

These assumptions have been critically analyzed by Christodoulakis [19, 20] and Montgomery *et al.* [96]. They have shown that these assumptions often lead to a bias against direct-access structures in selection plans. However, to date, there has been no practical and simple cost model or formula with more general assumptions, which does not require detailed catalogue information.

Selection of Access Plans

As we previously mentioned, selection of access plans can be based on either heuristic rules or cost estimates. Some approaches, proposed in the literature, combine heuristic reduction of access plan choices with enumerative cost minimization techniques [141]. A number of experimental studies show that combinatorial analysis significantly improves the database performance [35].

Cost estimates can be used in the access plan selection process in two different ways. The first approach estimates the costs of each alternative access plan completely [11, 142]. The major advantage of this approach is that it is possible to make use of techniques like parallelism or feedback in such situations. But, obviously the overall optimization effort is high.

In the second method, a parallel incremental generation of the cost of strategies is proposed. The main advantage of this method is that it permits whole families of strategies with common parts to be evaluated concurrently. Hence this approach results in a significant reduction in the overall optimization effort. One good example using this approach is the heuristic method proposed by Rosenthal and Reiner [115], in which the "optimizer" keeps only the current-optimal way to generate each intermediate result, and discards any other sub-optimal methods.

A more popular method is the dynamic-programming query optimization strategy. This is actually an extension to the second method, in which, at each level, only the next operation to be performed is determined based on the optimal sub-plans. As in any dynamic programming technique, this method has actual information about all its operands, including intermediate result sizes that can be used to update the

estimates of the remaining steps. This dynamic approach has two obvious drawbacks. Firstly, it has a potential of getting stuck in local optima if no adequate look ahead is planned. Furthermore, in the more general setting, the method leads to prohibitively high optimization cost.

2.2 Query Optimization Using Join Re-Ordering

In this section we review the work done in query optimization using join re-ordering, i.e., by the selection of optimal join trees. Before we describe what has been done with regard to optimal join trees, we will first introduce a few definitions that will be used throughout this section.

2.2.1 Some Definitions

Query Graphs and Join Trees

Query graph is a commonly used pictorial representation of a relational query. In a query graph, nodes denote the relation names, and edges represent the respective predicates. If a predicate expression, p, references the attributes of two relations, say R_i and R_j, then an edge labeled p is said to exist between the nodes of these two relations. Denoting the edges of the query graph as $\{p_1, \ldots, p_n\}$ and nodes as $\{R_1, \ldots, R_m\}$, we define the result of a query graph $G = (V, E)$ as a Cartesian product followed by relational selection: $\sigma_{p_1 \wedge p_2 \ldots \wedge p_n}(R_1 \times \cdots \times R_m)$.

Instead of using the straight definition of product followed by selection of the query tree stated above, simple *Join trees* are used to evaluate queries. A join tree is defined as an operator tree whose inner nodes are labeled by the relational join operators and whose leaves are labeled by base relations. The computation of the result of a join tree is always carried out in a computed bottom-up fashion. The nodes of the join trees are usually annotated to reflect the type of join-algorithm used. These generally include algorithms such nested loops, hash, sort merge and so on. It should be noted that some of the binary trees on the relations of the query are not actually join trees since they may involve the use of Cartesian products.

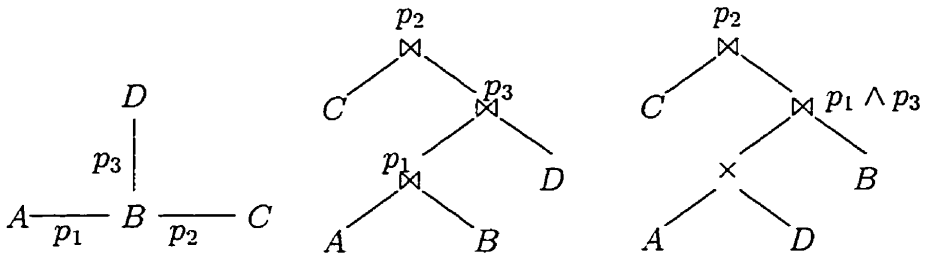


Figure 2.2: Query Graph and its Operator Trees.

The query graphs and join trees are two different forms of representing a given query. A query graph merely represents a collection of relations and their corresponding predicate expressions. It does not specify any evaluation order for the expressions involved. On the other hand, a join tree specifies clearly the inputs to each relational operator, and the order in which how they should be evaluated.

Figure 2.2 shows a query graph, and two possible operator trees that solve the query.

Join Tree Topologies

Join trees can be classified into two major categories, namely *linear* join trees and *bushy* join trees. For every join operator in a join tree, if at least one of the two input relations is a base relation, then the join tree is called *linear*, otherwise it is called *bushy*. Using this classification, *bushy search space* is defined as a set of join trees when there is no restriction on the topology of the trees. Similarly a *linear search space* is defined as a set of join trees where the topology of the trees is restricted to only linear join trees. It is obvious that the linear search space is actually a subset of the bushy search space. Since join trees do not distinguish between right and left subtrees, they are unordered. Since each of the $n-1$ internal nodes of an unordered tree of n leaves can have two choices, a total of 2^{n-1} ordered trees can be generated from such an unordered tree.

The number of alternative join trees exponentially increases when considering the various implementation algorithms available for each join operator in the tree. Some of the commonly used join algorithms are hash-join, nested-loop and merge-scan. For

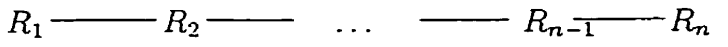


Figure 2.3: A string query.

example, for a query graph involving n relations where each join has x alternative implementation schemes available, then for each query tree which was previously unordered, one can generate a pool of x^{n-1} ordered trees.

Structured Query Graphs

When presented with an arbitrary query graph, finding the total number of valid join trees is a computationally intractable problem. On the other hand, when the topology of the query graph is "structured", it is relatively easy to find the exact number of join trees using standard combinatorial methods [81]. The number of unlabeled binary trees is shown in [75] to be equal to the Catalan number. Three well known query graph topologies, *string, completely connected* and *star* query graphs are discussed below.

String Query: Both end nodes of a string query have degree 1 and all other $n-2$ nodes have degree 2 as shown in Figure 2.3. String queries are frequently encountered in the verification of foreign key relationships and in object-oriented database environments.

Completely Connected Query: This is also known as a *clique*. These type of query graphs are not usually found in day to day operations of a DBMS. The main use of such a graph is as a test case for its large number of evaluation orders. It is clear that the query graph for a completely connected query of n relations, has n nodes of degree $n-1$. Since every node of this graph is connected to every other node, these graphs are also called *complete* or $n-1$ *regular* graphs. Figure 2.4 depicts a clique on four nodes.

Star Query: Online analytical processing (OLAP) applications often use star queries. In the query graph of a star query of n relations, there are $n-1$ nodes with degree 1 and one central node with degree $n-1$ as shown in Figure 2.5. It is interesting to note that for a star query, all valid join trees involve join operators that have at least one operator as a base relation. Hence in a star query, the number of

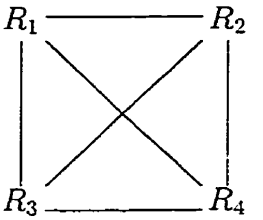


Figure 2.4: A completely connected query on 4 nodes.

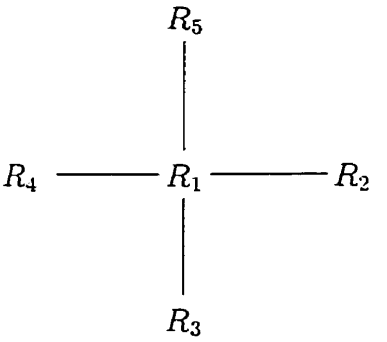


Figure 2.5: A star query on 5 nodes.

bushy search space is the same as the linear join tree search space.

For the three query graph topologies discussed above, namely *string, completely connected* and *star* [104] the exact number of unordered linear and bushy join trees is listed in Table 2.1.

Search Space	String Query	Completely Connected	Star
Linear join trees	2^{n-2}	$n!$	$(n-1)!$
Bushy join trees	$\frac{(2n-2)!}{n!(n-1)!}$	$\frac{(2n-2)!}{(n-1)!}$	$(n-1)!$

Table 2.1: Number of join trees for structured query graphs.

2.2.2 Join Tree Selection Methods

Optimal join tree selection is a search problem which depends on the three factors, namely the *join tree search space*, the *cost model* and the *search strategy* or *search algorithm*. The join tree search space contains all valid join trees, both bushy and

linear. The cost model is usually a simple mathematical model that annotates a cost to each operator in the tree. The search algorithm is the actual procedure that finds the optimal join tree in the search space using the cost model. Most of the modern-day query optimizers use three different search strategies, namely (a) exhaustive search (b) probabilistic techniques (c) non-exhaustive deterministic algorithms. We discuss each of them briefly below.

Exhaustive Search

Exhaustive search is only possible if the join tree search spaces are not too large. For most of the relational queries this should be less than ten relations. There are two different exhaustive search techniques commonly in use. These are namely, *dynamic programming* [122] and *transformation-based* search methods [92]. Dynamic programming and transformation-based search are also known as the *bottom-up* and *top-down* search methods.

1. **Dynamic Programming:** Early database systems such as System R [122] and Starburst [53, 54, 104] used dynamic programming to find optimal join trees or query evaluation plans. This strategy is very simple and easy to comprehend. The dynamic programming algorithm first considers the individual base relations in a bottom-up fashion and adds new relations until all pairs of relations are joined to generate the final result. A table for storing the intermediate result sizes is maintained so that the sequence of the next join operation can be determined based on the cheapest alternative currently known.

2. **Transformation-based:** The Volcano query optimizer [45, 92] and the Cascades [48] query optimizer use the transformation based exhaustive search approach. This strategy works by applying various transformation rules to the initial query tree recursively until no new join trees are generated. These optimizers use an efficient storage structure for storing the information about the partially explored search space, in order to avoid generating the same query trees.

Probabilistic Algorithms

Finding a globally optimal join tree in a very large search space, specially in a bushy search space involving more than 10 relations, is usually a difficult problem. One approach is to use probabilistic algorithms to eliminate the less promising portions of the search space. These algorithms improve the search performance by avoiding the worst join trees during the search process, instead of looking for the best join tree. A number of different probabilistic algorithms such as *Iterative Improvement* (II) [130], *Simulated Annealing* (SA) [66] and their variations *Toured Simulated Annealing* (TSA) [81] and *Two Phase Optimization* (2PO) [65] are currently in use for searching large query tree search space. All these algorithms are based on rewriting join trees based on the algebraic properties of the join operator (such as commutativity and associativity) in order to generate alternative query trees.

All of the above probabilistic algorithms look for an optimal join tree in the following manner. First, an arbitrary join tree is generated by the algorithm to represent the given query. Then, applying the logical transformation rules that were discussed earlier, new join trees are generated. For each new join tree, using a simple cost model, the algorithm computes an estimated cost. The new join tree is retained as the *current solution*, if the cost is found to be below a specified value. The transformation rules are applied again on the current solution to generate new join trees in a recursive manner. This recursive process is carried out until a stopping condition, such as a time limit or level of cost improvement, is met.

The algorithm TSA finds the optimal join tree by performing simulated annealing repeatedly. Each time the algorithm picks a new query tree as its starting point. The 2PO algorithm combines the features of the II algorithm and the SA algorithm. It first performs II for a given period of time, then using the output from the II phase as its starting query tree, it performs SA until the final globally optimal query tree is obtained.

Non-exhaustive Deterministic Algorithm

An efficient non-exhaustive deterministic algorithm using a block-wise nested-loop join algorithm was first proposed in [61]. This algorithm generates optimal linear join trees, especially for acyclic queries. A variant of this algorithm was described in [17], in which the search strategy employs an arbitrary join algorithm based on the available access paths instead of a nested-loop join algorithm.

2.3 Parallel and Distributed Query Optimization

Almost all of the modern-day high-performance computers support some form of parallelism in their architecture. Hence it is very important that the database systems designed to process large volumes of data should be able to take advantage of such parallel processing capabilities. There has been considerable developments in the parallel and distributed query evaluation methods in the last two decades, and therefore a review of them here is necessary.

Even though the fundamental concepts behind both the distributed and parallel databases are almost similar, research in distributed query optimization was done in the early 1980s, at a time at which communication over a network was prohibitively expensive and computer equipment, such as processors, memory, secondary storage, was very expensive to provide parallel processing capabilities.

In order to improve the query execution performance in distributed systems, new techniques, such as semi-joins were developed to reduce the communication bottlenecks. During the early days, techniques for exploiting parallelism were ignored. Since the potential query execution space is different for both distribute and parallel systems, the optimization problems are obviously different under both systems.

Any parallel execution strategy requires an overhead of starting and initializing processes. The initial processes may then transmit substantial amounts of data between various processors. Obviously these *startup* and *communication* overheads increase total work. Therefore it is important to note that the increase is overhead work can be significant enough that careless use of parallelism can result in severe

performance degradation rather than performance increase. The communication cost is usually a function of the size of data transmitted. While an individual operator may examine a relatively small portion of each tuple, all data that are used by any subsequent operator need to be communicated. This invariably results in a significant percentage of the total cost of executing the query.

In a *distributed* database system, each participating system (or node) is a complete database management system in itself, with access control, meta-data (catalogue), query processing, etc. This means each node in a distributed database system can function autonomously on its own, regardless of the status of the other nodes. Since each node performs its own access control, cooperation of each node in a distributed transaction is not mandatory. Some of the popular distributed systems of academic interest are R* [50], distributed Ingres [35], and SDD-1 [9]. In addition to these, there are now a number of commercial distributed database management systems. Some of the works described by Ozsu and Valduriez [109, 110] detail distributed database systems and query evaluation plans.

Unlike the distributed systems, in a parallel database system, the overall control is at a single location. This means that there is only one database management system that divides user queries into sub-tasks and executes the sub-tasks in parallel. Access control to data is more difficult to implement than a distributed system. Also unlike a distributed system, the query optimizer needs to assume that all nodes in the system are available to participate in coordinated execution of queries. The participation of nodes or processors in a given query execution is usually controlled by a global resource manager, as opposed to the voluntary cooperation found in distributed systems. Research work on parallel database systems has been going on for a number of years now and consequently a number of research prototypes can be found in the literature. Some of the popular ones include, Gamma [31, 32], Bubba [14, 15], Grace [41, 72], Volcano [46], Tandem's NonStop SQL [34], Teradata's DBC/1012 [101], and Informix [28].

Distributed database systems can be either homogeneous or heterogeneous. In a homogeneous system all participating nodes are of the same type, whereas in a heterogeneous system, the participating nodes can be of different types. In addition, it is

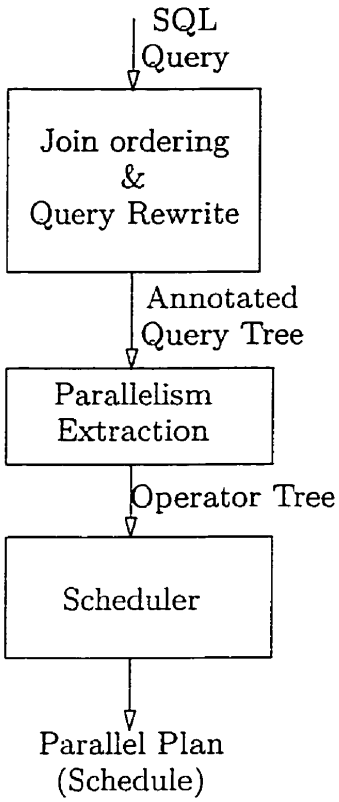


Figure 2.6: Different Phases of Parallel Query Optimization.

possible for a distributed system to make use of parallelism at any of its nodes. It can also employ parallelism by pipelining data between nodes. Parallel database systems can be based on a shared-memory, shared-disk, distributed-memory, or hierarchical computer architectures consisting of multiple nodes, each with multiple CPUs and disks and a large shared memory. Some researchers [35] have proposed that considering various scenario, a system based on distributed memory architecture is possibly the most promising one in terms of efficiency and cost.

2.3.1 Forms of Parallelism

There are many different forms of parallelism that can be used for query evaluation. In *interquery* parallelism, multiple queries from many users can be evaluated concurrently within a single node. Since most DBMSs can process requests from multiple

users simultaneously, interquery parallelism can be exploited by the query optimize.r The major disadvantage of the interquery parallelism is that there may be resource contention among competing processes, mainly for accessing memory and disks.

Another commonly employed parallel processing technique known as the *inter-operator* parallelism involves pipelining successive operators of a given query.

There are two main forms of inter-operator parallelism, namely *vertical inter-operator* parallelism and *horizontal inter-operator* parallelism. Vertical inter-operator parallelism involves executing successive operators and relations in a pipeline, whereas horizontal inter-operator parallelism involves executing independent subtrees of a bushy-query tree concurrently.

Another form of parallelism commonly employed in practice is known as the *intra-operator* parallelism, which involves the execution of a single operator concurrently on multiple processors. This is useful when the underlying data sets can be easily partitioned among multiple processors.

Modern-day database query optimizers use both vertical inter-operator parallelism and intra-operator parallelism to achieve increased performance. Even though these techniques offer speedup and scale-up from a theoretical point of view, they both present significant implementation difficulties. For example, load balancing is extremely difficult to achieve with pipelining. This is because the amount of data going to each processor in a pipeline cannot be predicted or chosen by the query optimizer easily. Similarly the intra-operator, partitioning-based parallelism cannot work efficiently unless the data partitions are all of equal size. This is often not true as the real world data distributions are not always uniform.

2.3.2 Load Balancing and Skew

Assignment of data to individual processors and disks equally is one of the important factors in achieving optimal speedup and scale-up in a parallel query evaluation environment. In inter-operator parallelism, one or more processors may become the bottleneck unless the query operators are properly grouped together. The main difficulty in achieving proper load balancing is that intermediate result sizes are not easy

to predict. This is why most of the query optimizers do not base their architectures solely on inter-operator parallelism. The data from the secondary storage must be partitioned in intra-operator parallelism in such a way that the distribution of load is equal for each processor.

A number of techniques have been proposed in the literature to minimize the effect of skew or to completely avoid it in parallel query processing [4, 30, 60, 71, 79, 80, 123]. Each of these techniques have associated drawbacks and thus achieving a theoretically ideal load balancing is almost impossible.

2.3.3 Parallel Algorithms

Parallel algorithms for query optimization in database systems mainly based on the concept of partitioning an input data set using range or hash partitioning. These partitioning techniques can be combined with sort and hash-based query processing algorithms to evaluate the final results.

A main issue when building a parallel DBMS is deciding whether to parallelize a slower sequential algorithm with better speedup behavior or a fast sequential algorithm with inferior speedup behavior. This should be properly planed since the overall efficiency depends on the design goals and the desired degree of parallelism. For example, the goal in a single user database system would to minimize response time. Hence in this case, parallelizing a slow algorithm with linear speedup may be the right option. But in a multi-user system which is encountered in day-to-day life, the goal typically is to minimize the overall resource consumption in order to maximize throughput. Hence the right approach would be to choose a fast sequential algorithm and then to parallelize it. It has been shown by Boral and Dewitt [16] that effective and efficient indices cannot be substituted with any amount of parallelism. Hence proper choice of the underlying sequential algorithm is always essential before any attempts to parallelize the query processing.

Parallel Selections

One of the main issues in query processing is the performance bottle-neck due to the disk I/Os. Hence it is a natural candidate for parallelization. This is usually accomplished by using an asynchronous I/O or assigning one process per I/O device. When the partitioning attribute is the selection attribute, then usually fewer disks or partitions will contain the results of the selection operation. This has the advantage of using fewer processes and disks. Whenever local indexes are available, parallel selection can be usually implemented efficiently.

Parallel Updates

When partitioning attributes are updated, the records may have to be moved between disks. For example, a record may have to move whenever a clustering attribute value is updated. Hence in order to maintain consistency of the partitioning used, the updates of partitioning attributes need to perform data transfers from old to new positions of the updated records.

Parallel Sorting

Parallel sorting has been extensively dealt with in the last two decades due to its importance in modern-day DBMSs [5, 7, 10, 47, 67, 73] and thus we provide a brief overview here. Since sequential input or output usually restricts the throughput of parallel sorts, most of the parallel sorting strategies include multiple inputs and multiple outputs. Most parallel sorting algorithms consist of a local sort and a single data exchange step. Assuming that the data exchange step takes place first, then quantiles (partitions) must be known to guarantee proper load balancing during the local sort step within each node. Quantiles can be usually obtained from the DBMS catalogue. Whenever the local sort step is carried out first, the local merging step should transfer records directly to the subsequent data exchange step. As the sorted records emerge from the output stream, they can be merged together by the data exchange step.

Parallel Aggregation and Duplicate Removal

The parallel algorithms for aggregation and duplicate removal are usually carried out in two stages, i.e: locally and globally. In the local step, the duplicates are first eliminated, and then data is partitioned to detect and remove duplicates from different original sites. It should be noted that during a parallel aggregation operation, the local and global aggregate functions can differ.

Parallel Binary Operations, including Joins

Unlike the previous operations, parallel computation of the binary operations, namely, join, semi-join, outer join, intersection, union, and difference are considered differently as they operate on two operands. Computing a join for which two sub-plans create the two inputs independently from one another in parallel is known as bushy parallelism. This is usually achieved by symmetric hash join algorithms. Symmetric hash join uses two hash tables, one for each input. When an element arrives, the join algorithm first determines which input it came from and then carries out the join using the new element with the hash table built from the other input. The new element is also inserted into its hash table such that subsequent elements from the other input arriving can be joined correctly. XPRS, a popular shared memory high-performance extensible database system, uses symmetric hash join algorithm [57, 58, 126, 127]. Symmetric hash join algorithm is also used in Prisma/DB, a popular shared-nothing main-memory database system [139, 140]. The main advantage of symmetric hash join algorithms is that they are not dependent on how the elements/records arrive at their inputs. Their main disadvantage is that both their inputs need to fit into the main memory.

Symmetric partitioning and *fragment and replicate* partitioning are two well known techniques for parallelizing a single binary matching operation. Both techniques concatenate the local results to obtain the final result.

The symmetric partitioning technique requires that both inputs are partitioned on the attributes corresponding to the operation before the operation is performed at each site. Two popular database systems, namely, Gamma and Teradata, use this

method.

The fragment-and-replicate technique partitions one input and broadcasts the other one to all sites. Usually the larger input is partitioned and the smaller one is broadcast. The early distributed database systems such as R*, SDD-1 and distributed Ingres used fragment-and-replicate technique as their choice for parallel join operation. It is important to note that fragment-and-replicate methods do not work correctly for semi-join, outer join, difference, and union operations.

Semi-join technique is typically used for reducing network traffic during join evaluation in distributed database systems [9, 44]. This method can also be used in distributed memory parallel systems.

Another type of algorithm known as *fetch-as-needed*, is often used in distributed database systems. In this method, one site performs the join by explicitly fetching only those elements from the other input needed to perform the join [26]. Obviously if one input is relatively small, fetching only the required elements of the larger input is advantageous. The main difference between the semi-join strategy and fetch-as-needed method is that the semi-join reads the first input twice, once to obtain the join values and once to actually carry out the join operation, whereas fetch-as-needed works on each element only once.

2.3.4 Distributed Query Optimization

Query processing and optimization in a distributed database system pose additional challenges to the query optimizer due to their inherent complexity resulting from the introduction of communication networks as well as the presence of heterogeneous nodes. One of the main problems in distributed query optimization is that not all relations can be expected to reside at the site of the query optimizer. More often than not, the relations being joined need to be transmitted over slow networks before an optimal access plan can be chosen. Obviously this introduces additional parameters, such as communication delay and relation/site information, into the cost model, which, in turn, increases the computation cost. In addition to these factors, heterogeneity which is inherent to any distributed computing environment, introduces a

number of unpredictable issues. These include, type of databases at various sites, local optimization strategies, communication protocols, compatibility of query languages used[1], failure of nodes and local access methods.

Due to the enormity and complexity of this problem, we will not attempt to tackle it in this thesis work. We merely allude to some of the recent developments in the research on heterogeneous database interconnection [95] as one possible way to integrate query optimization in such a distributed database system. In [95], an interoperable database communication prototype was proposed by Merrett *et al.* for handling communication among the various databases on a heterogeneous database system for supporting GIS. Using a simple communication protocol based on UNIX sockets and an XML-like language, this prototype demonstrates that complex SQL-based communications can be easily and efficiently implemented between various heterogeneous database systems supporting different commercial products.

Hence one approach to query optimization in such a heterogeneous system would be to extend the language used for the communication protocol to support adequate syntax, so that the statistical information from the database systems can be properly transmitted between nodes. Assigning various priority codes to distinguish between the statistical data originating from the catalogue and the actual data, can also be implemented to facilitate query optimization. Armed with such an additional language support, we envision that the query result size estimation techniques proposed in this thesis for centralized database systems can be easily adopted in such a distributed database system.

2.4 Statistical Estimations in Database Systems

The quantitative properties or statistics that summarize the instances of a database are important for query optimization. In this section we review the major estimation techniques used to obtain the database statistics that have some commonalities with

[1]ANSI SQL-92 is the widely accepted standard for querying databases. Even though most of the major database vendors these days adhere to this standard, each of their products contains minor variations. Thus they present problems in a heterogeneous distributed database system.

this thesis work.

As we discussed in Chapter 1, query optimization is an NP-hard problem due to the exponential growth of the number of alternative QEPs with the number of relations in the query. Due to this reason, most database systems are forced to make a number assumptions in order to quickly select an optimal QEP. Christodoulakis investigated the relationships of a number of frequently used assumptions and the accuracy of the query optimizer [20, 96]. In specific, he considered the following issues in his study:

1. **Uniformity of attribute values:** Every attribute value in the corresponding value domain has the same frequency.

2. **Attribute independence:** The data distributions of various attributes in a relation are independent of each other. The value of two attributes (say A and B) are independent if the conditional probability of an A value given a B value is equal to the probability of obtaining the A value by itself.

3. **Uniformity of queries:** Queries reference all attribute values in a given attribute value domain with equal probabilities.

4. **Constant number of records per block:** The number of tuples in each file block is the same. Hence the probability of referencing any block is $1/B$, where B is the number of blocks.

5. **Random placement:** The probability of a record in a file to appear in a query is the same, regardless of its location among the different pages of a disk. That is, the probability of referencing any tuple is $1/N$, where N is the number of tuples.

One can easily observe that assumptions 1 and 2 have a direct relationship to the estimates of the sizes of plan operations, thus determine their accuracy. Similarly assumption 3 impacts the size estimate of queries that are associated with a given parameter, as well as the physical database design problems. Assumptions 4 and 5 affect the estimation of logical block accesses for a given attribute value.

Though these statistical assumptions simplify the factors involved in the estimation of the cost of a QEP, they also decrease accuracy of these estimates. In order to increase the estimation accuracy, some modern query optimizers incorporate a more detailed cost models for assumptions 1 and 2. Few optimizers incorporate assumptions 3, 4, and 5 in more detail, since these assumptions are more difficult to analyze.

Christodoulakis studied the effects of the above assumptions on a number of frequently encountered problems such as the estimation of the (1) number of block accesses for a given number of tuples, (2) number of block accesses for all queries on a given attribute, (3) number of block accesses for queries with multi-attributes, and (4) number of distinct attribute values resulting from a selection operation. Using these studies, he proved that the expected cost of a query estimated using these assumptions is an upper bound on the actual cost of the QEP. For example, he showed that the uniformity and independence assumptions lead to a worst-case estimate for the number of distinct attribute values. He further demonstrated that most commercial database systems using these assumptions are prone to use expensive query evaluation strategies. This indicates that non-uniformity, non-independence, and non random placement that usually exist in a real-world data distribution could be exploited in a query optimizer in order to reduce the system cost. He also argued that optimizers that use these often erroneous assumptions tend to ignore direct access structures because the these cost estimates often favor the simpler structures. His investigation in this area was a catalyst in motivating further research in this area for better result size estimation techniques. He also discussed a number of mathematical techniques such as *Schur concavity* [91] in database performance evaluation.

The size estimation of select and join queries on a heavily skewed data distribution were thoroughly investigated by Montgomery *et al* [96]. In this study, they estimated the result sizes of select and join operations using uniformity assumptions and compared them with the actual result sizes. They also constructed a few models of size estimation using formulae tnat closely describe the skewed data distributions under study. They reported that the size estimates for the select operations based on the widely used uniformity assumption actually overestimate the actual result size by

200-300%. The size estimates for the join operations were reported to be 200-300% lower than the actual join sizes.

The local and distributed cost models based on the uniformity assumption that is used for single table sorting and two table joins in R* were experimentally verified and validated by Mackert and Lohman [88, 87]. They also further reported that the CPU costs are usually a significant portion of the overall cost for sort operations. In addition, they also found that size estimation is a significant issue in the overall cost, including I/O and CPU costs. Considering the nested loop join method, they reported that most query optimizers based on uniformity assumption overstate the evaluation cost, when an access structure is used to retrieve the attribute values and when the inner table fits in main memory. This indicates that the cost of evaluating the nested loop join is usually sensitive to parameters such as join cardinality, the outer table's cardinality, and the memory used to maintain the inner table.

The relationship of join selectivity and the selection of the optimal nesting order involving four and five variable queries were studied by Kumar and Stonebraker [78]. A query processor that behaves similar to the System R optimizer [122] was built. This query processor considered the nested loop and merge scan join algorithms, using only two-way joins. It also considered the availability of any secondary indexes on the inner join table. The sensitivity of a query with respect to changes in the joint selectivity was derived. This was obtained as the ratio between the cost of the optimal evaluation plan and the plan used by the query processor. The found that most optimal evaluation plan under varying selectivities was either the one that minimizes the average cost ratio or the one that minimizes the maximum cost ratio. Using the sensitivity factor as a parameter, they further measured the sensitivity of four and five variable queries under a number of join selectivities. Their experimental results showed that, assuming the query evaluation plan was selected using their criteria, the best query evaluation plan is usually insensitive to changing join selectivities.

The relationship of the assumption of random placement of tuples to pages and the dependence of multiple attribute values were investigated by Vander Zander *et al.* [136]. A query of the form, "Select the tuples from relation R where $R.A = constant$" when R is clustered according to another attribute, say B, was used for this type of

experiments. They found that whenever there is a high correlation between the attributes A and B, the assumption of random placement of tuples to pages is always violated. This was evident from the skewed pattern of tuple distributions to the pages.

Recently Ioannidis and Christodoulakis [62] proved that the worst case errors incurred by the uniformity assumption propagate exponentially as the number of joins in the query increases. As a result, except for very small queries, errors may become extremely high, resulting in inaccurate estimates for result sizes and hence for the execution costs.

Despite the overwhelming evidence that most of the common assumptions either overestimate or underestimate the true statistics by a wide margin, the query optimization literature abounds with a variety of estimation techniques, most of them discussed in the extensive survey by Mannino, Chu, and Sager [90]. The various estimation techniques can be grouped into four broad classes as follows:

1. Probabilistic counting techniques
2. Sampling based techniques
3. Parametric techniques
4. Non-parametric techniques

We describe each of them briefly in the following sections.

2.4.1 Probabilistic Counting Techniques

The probabilistic counting techniques are useful for estimating the number of distinct attribute values in the result of projecting a relation over a given subset of attributes [40, 43, 124]. Flajolet and Martin [40] proposed a technique for estimating the number of distinct values in a multi-set, which generates an estimate during a single pass through the data. This technique was extended by Shukla *et al* for estimating the size of multidimensional projections (the cube operator) [124]. Their experiments have shown that these techniques based on multidimensional projections can usually

result in more accurate estimates than the sampling based techniques [124]. The applicability of these techniques to other operators has not yet been resolved.

2.4.2 Sampling based Techniques

Sampling based techniques are mainly used to compute result estimates during query optimization time. They compute their estimates by collecting and processing random samples of the data. The major advantage of these techniques is that since they do not rely on any precomputed information about the data, they are not usually affected by database changes. Another advantage is that they do not require any storage overheads. These techniques also provide a probabilistic guarantee on the accuracy of the estimates. The major disadvantages of these techniques include the disk I/Os, CPU overhead during query optimization and the extra overhead for recomputing the same information since they are not preserved across queries. The sampling based techniques are highly desirable whenever a parameter needs to be estimated once with high accuracy in a dynamic environment. One good example is in the context of a query profiler. Number of different methods on the sampling based techniques have been extensively discussed in the literature [1, 3, 42, 55, 56, 83, 103, 123]. In this section, we give an overview of the sampling-based size estimation methods.

Traditionally the sampling based size estimation techniques use two distinct models, namely, *point space model* [59] and *urn model* [84, 85, 86] for result size estimation. These size estimation techniques can be classified into *random sampling* and *systematic sampling*.

The idea behind the *random sampling* technique is that the tuples are chosen randomly from their set so that every tuple has an equal probability of being chosen. The random sampling techniques can be either random sampling with replacement or random sampling without replacement. In random sampling with replacement, a tuple that is drawn randomly is put back into the data set and available for subsequent selections. But, in random sampling without replacement, any selected tuple is not returned to the original data set.

In *systematic sampling* technique, all the tuples in a sampled block (or page) will

be used as sampled tuples. This obviously increases efficiency by reducing the number of I/O operations. But its accuracy is highly related to the uniformity of data across blocks. This means, whenever the tuples are clustered, sampled data could become highly biased resulting in increased estimation error [59].

Most of the sampling techniques used in modern-day database systems are based on random sampling with replacement. Some of the widely used techniques are discussed briefly in the following sections.

Adaptive Sampling

Adaptive sampling technique and its variants were proposed by Lipton, Naughton, and Schneider [84, 85, 86]. This technique expresses the stopping condition for sampling in terms of the sum of samples and the amount of sample taken. The provide an asymptotic efficiency of their algorithm and discuss the practicality of their method for estimating the result sizes of select and join operations. Their method provides a bound on the sample size necessary to obtain a desired level of estimation accuracy. They also provide sanity bounds to handle queries for which the underlying data distribution is skewed.

Double Sampling

In *Double sampling* technique [59], sampling is divided into two phases. In the first phase, x number of tuples are sampled. This sample is used to obtain the estimated mean and variance and to compute the number of samples y to be obtained in the second phase. The number of samples y is computed based on the desired level estimation accuracy, confidence level, and the estimated mean and variance obtained in the first phase. If $y > x$, then an additional $y - x$ samples are obtained during the second phase of sampling. Otherwise, the number of samples, x, obtained in the first phase is sufficient to provide the desired level of estimation accuracy and confidence level.

Sequential Sampling

Sequential sampling is a random sampling technique for estimating the sizes of query results [55]. This is a sequential process in which the sampling is stopped after a random number of steps based on a stopping criterion. Stopping criterion is usually determined based on the observations obtained from the random samples so far. It can be shown that for a sufficient number of samples, the estimation accuracy can be predicted to lie within a certain bound. The authors have shown that this method is asymptotically efficient and does not require any *ad hoc* pilot sampling or any assumptions about the underlying data distributions.

2.4.3 Parametric Techniques

Parametric techniques use a mathematical distribution to approximate the actual data distribution A few popular examples include the uniform distribution [122], multivariate normal distribution or Zipf distributions [20]. The parameters used to construct the mathematical distribution are usually obtained from the actual data distributions, and consequently the accuracy of this parametric approximation mainly depends on the similarity between the actual and parameterized distributions. Two major advantages of the parametric technique are (a) their relatively small overhead and (2) small run-time costs. But their downside is that the real-world data hardly conform to any simple mathematical formula. Consequently any such parametric approximation may result in estimation errors. Another disadvantage with this method is that since the parameters for the mathematical distribution are always precomputed, this technique always results in further errors whenever there is a change in the actual data distribution.

An interesting variants of this approach is the *algebraic* technique, where the actual data distribution is approximated by a polynomial function. Regression techniques are usually used to obtain the coefficients of this polynomial [129]. Another promising algebraic technique involves adaptively approximating the distribution by a six-degree polynomial, whose coefficients are obtained dynamically based on a query feedback mechanism [43]. The main disadvantages in using the algebraic techniques

include the difficulties in choosing the degree of the polynomial model and uniformly handling result-size estimates for operators other than simple selection predicates. Some of the positive results obtained in the work of Wei Sun *et al* [129] on algebraic techniques indicate that they have the potential of generating closely matching parametric distributions.

The uniform distribution is the simplest and may apply to all types of variables, from categorical to numerical and from discrete to continuous. For categorical or discrete variables, the uniform distribution provides equal probability for all distinct categories or values. In the absence of any knowledge of the probability distribution of a variable, the uniform distribution is a conservative, minimax assumption. Early query optimizers such as System R [122] relied on the uniformity (and independence) assumptions. This was primarily due to the small computational overhead and the ease of obtaining the parameters (maximum and minimum values).

The use of the uniform distribution assumption has been criticized, however, because many attributes have few occurrences with extreme values. For example, few companies have very large sales, and few employees are very old or very young. The Zipf distribution has been suggested by Fedorowicz [38, 39] and Samson and Bendell [117] for attributes with a skewed distribution such as the occurrence of words in a text.

Christodoulakis [21] demonstrated that many attributes have unimodal distributions that can be approximated by a family of distributions. He proposed a model based on a family of probability density functions, which includes the Pearson types 2 and 7, and the normal distributions. The parameters of the models (mean, standard variation, and other moments) can be estimated in one pass and can be dynamically updated. Christodoulakis demonstrated the superiority of this model over the uniform distribution approach using a set of queries against a population of Canadian engineers.

Faloutsos [36] showed that using multi-fractal distribution and 80-20 law, one can obtain better approximation of the real-world data than the uniform and Zipf distributions. His method can also be used to provide estimates for supersets of a relation, which the uniformity assumption based schemes cannot provide.

Vendor	Product	Histogram Type
IBM	DB2-6000 (Client-Server)	Compressed(V,F) Type
IBM	DB2-MVS	Equidepth, Subclass of End-Biased(F,F)
Oracle	Oracle7	Equidepth
Sybase	System 11	Equidepth
Tandem	NonStop SQL/MP	Equidepth
NCR	Teradata	Equidepth
Informix	Online Data Server	Equidepth
Microsoft	SQL Server	Equidepth

Table 2.2: Histograms used in commercial DBMSs.

2.4.4 Non-parametric or Histogram-based Techniques

In order to avoid the restrictions of particular parametric methods, many researchers have proposed non-parametric methods for estimating a distribution. The oldest and most common of these methods is the histogram. These techniques approximate the underlying data distribution using precomputed histograms. They are probably the most common techniques used in the commercial database systems such as DB2, Informix, Ingres, Microsoft SQL Server, Oracle, Sybase, and Teradata - see Table 2.2. The essence of the histogram is to divide the range of values of a variable into intervals, or *buckets* and, by exhaustive scanning or sampling, tabulate frequency counts of the number of observations falling into each bucket. The frequency counts and the bucket boundaries are stored as a distribution table. The distribution table can be used to obtain upper and lower selectivity estimates. Within those bounds, a more precise estimate is then computed by interpolation or other simple techniques. Since the histograms are precomputed, they may incur errors in estimation if the database is updated and hence require regular re-computation. It has been shown that significant effort is required to identify the correct form of histograms that incur small errors for all estimation problems [64].

Most of the database research on histograms is in the context of simple elementary relational operations such as selections. One of the first work on histograms was by Piatetsky-Shapiro and Connel where they analyzed the effect of histograms on

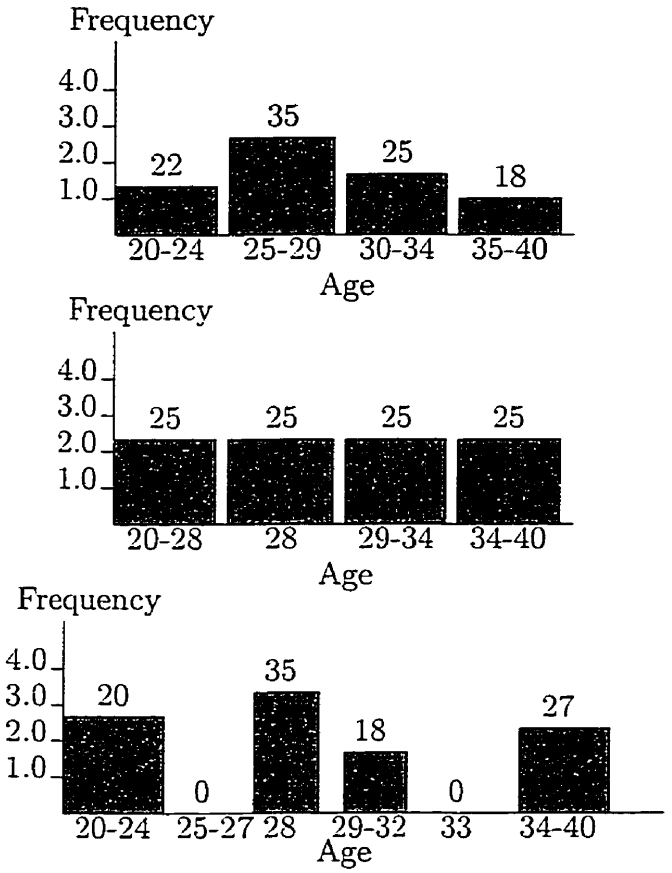


Figure 2.7: Equi-width, Equi-depth and Variable-width Histograms.

minimizing the error for selection queries [112]. These histograms involved two distinct classes, namely, *equi-width* histograms and *equi-depth* histograms [76]. One of their major results is that the worst-case and average case errors in both the equality and range selection queries are usually smaller in the equi-depth histograms than the equi-width histograms.

Extending the work of Kooi, Muralikrishna and DeWitt [98] investigated the use of *multi-dimensional equi-depth* histograms for result size estimations. Multidimensional attributes are very common in geographical, image and design databases. A typical query with a multidimensional attribute is to find all the objects that overlap a given grid area. By building histograms on multiple attributes together, their techniques were able to capture dependencies between those attributes. They proposed

an algorithm to construct equal-height histograms for multidimensional attributes, a storage structure, and two estimation techniques. Their estimation methods are simpler than the single-dimension version because they assume that multidimensional attributes will not have duplicates. Performance analysis by the authors was conducted to compare the estimation techniques and the use of random sampling to construct the histograms.

Many other related works have been done using *variable-width* histograms for estimating the result size of selection queries, where the buckets are chosen based on various criteria [69, 76, 99]. A detailed description of the use of histograms inside a query optimizer can be found in Kooi's thesis [76]. It also deals with the concept of variable-width histograms for query optimization. The survey by Mannino, Chu, and Sager [90] provides a list good references to work in the area of statistics on choosing the appropriate number of buckets in a histogram for sufficient error reduction. Again this work also mainly deals with the context of selection operations. The query result size estimations on simple join queries have not yet been throughly discussed in the literature. Most of the work in this area does not address the issue of optimal result size estimation [20, 76, 99].

Merrett and Otoo [94] showed how to model a distribution of tuples in a multidimensional space. They derived distributions for the relations that result from applying the relational algebraic operations. In one sense, this work, done two decades ago, sowed the seeds for future research in this area.

In their work, a relation was modeled as a multidimensional structure where the dimension was equal to the number of attributes in the relation. They divided each dimension, i, into c_i number of equal width sectors. The values for c_i, $1 \leq i \leq n$, are chosen such that the resulting structure can completely fit into the available memory, where n is the dimension of the multidimensional array. Scanning through the relation for each attribute, the cells of the multidimensional structure are filled with integer numbers corresponding to the number of tuples that occupy the respective value ranges of the attributes.

Denoting the value of the count for a cell (a, b) of relation $R(A, B)$ as $d_{ab}^{AB}(R)$, and assuming uniformity of tuples within each cell, they showed that the expected

number of tuples within a cell from a projection is,

$$d_a^A(R[A]) = n\left[1 - \left(1 - \frac{1}{n}\right)^{d_a^A(R)}\right].$$

Using the multidimensional representation, they show that the number of tuples resulting from joining a sector a_1 of R and a_2 of S is equal to,

$$d^{ABC}_{a_1 \cap a_2 bc}(Q) = \frac{n d^{AB}_{a_1 b}(R) d^{AC}_{a_2 c}(S)}{n_1 n_2},$$

where n_1 is the number of values in sector a_1, n_2 is the number of values in sector a_2, n is the overlap between a_1 and a_2, and b and c are the remaining attributes in the respective relations.

The above value for the join is valid only when the two cells from the relations have overlapping sectors on the joining attribute. The above join estimation depends on the number of overlapping attribute values, n. Since there is no way to compute the value of n, they resort to estimate it based on how the two sectors overlap along the joining attributes. They showed that there are 11 possible ways in which the sectors can overlap. By defining a rule to approximately guess the value for n for each of the 11 possible ways of overlap, they show how to estimate the result size of join from two cells.

Estimating the result sizes of selection operations using the equi-width histogram is very simple. We look for the histogram bucket where the attribute value corresponding to the search key lies. Then the result size of the selection operation is simply the average number of tuples in that bucket. But the main problem with this simple strategy is that the maximum estimation error is related to the height of the tallest bucket. Consequently a data distribution with widely varying frequency values will result in very high estimation error rate. Another problem associated with the equi-width histogram is that there are no proven statistical method to choose the boundaries of a bucket as this would require some prior knowledge about the underlying data distribution. This invariably has an effect on the estimation accuracy of the selection operations.

Fortunately there are some statistical techniques that can be used to select the bucket widths (or the number of buckets) based on the number of tuples in the attribute value domain being mapped. During early days of statistical analysis of data using histograms, a popular rule known as Sturge's rule [128] was used to obtain the number of buckets. The Sturge's rule, based on the binomial distribution, states that the number of buckets for n tuples is equal to $1 + \log_2 n$, which is a reasonably accurate value under most circumstances. The modern statistical techniques provide more optimal rules for any given attribute value domain and error rates [131]. It is important to note that the simple Sturges' rule usually yields estimation results which are closer to the modern statistical techniques for up to 500 tuples. The modern statistical literature also provides a number of parameters such as the mean error rate and mean-squared error rate, which are very useful in computing the size of the error within a multiple of a power of the given attribute value domain without any further knowledge of the data distribution. Although the equi-width approach is very simple to implement, its asymptotic accuracy is always lower than the estimation accuracy of the most modern techniques, including the kernel and nearest neighbor techniques [131]. It has been shown in [131] that the asymptotic mean-squared error obtained using the equi-width approach is $O(n^{-2/3})$, while that of the more recent methods are $O(n^{-4/5})$ or even better.

In the equi-width histogram, proper choice of the bucket width usually results in an optimal estimation error rate. Similarly, in the equal-height histogram method, the result size estimation error is directly related to the height of the histogram buckets. A strategy known as the *nearest neighbor* method in the field of density estimation, in which the bucket height is kept the same over the entire attribute value domain, can be used to control the estimation error in the equal-height method by varying the number of buckets. The main problem with this approach is that we need to have some prior knowledge about the data distribution in order to construct the sector widths so as to guarantee that same number of tuples falling in each bucket. One common approach is to arrange the attribute value in an ascending or descending order, and to then set the bucket boundaries at every k^{th} attribute value. A less expensive approach would be to employ some sampling techniques to approximately

set the bucket boundaries. It has been shown that the estimation accuracy and performance can be improved asymptotically by allowing buckets to have overlapping attribute value ranges.

The above technique was further improved by Moore and Yackel [97], by applying the large-n properties. They showed that by allowing some adjustment for the differences between two methods, using the large-n theorem for one method, it could be translated into a large-n theorem for another estimation method without any additional proof.

A detailed analysis and discussion about various density estimation techniques is given by Wegman [137] and Wertz [138].

Another type of histogram method, often known as *variable kernel* technique in the density estimation field, uses a partitioning strategy based on criteria other than the equal frequency. Mathematical analysis of the variable kernel technique has been found to be difficult. Again one can find references regarding this technique in Wegman [137], Wertz [138], Devroye [29] and Breiman *et al* [18].

Lecoutre showed how to obtain the value of the cell width which minimizes the integrated mean squared error of the histogram estimate of a multivariate density distribution [82]. By considering a density estimator based on k statistically equivalent blocks, he further proved the L^2 consistency of this estimator, and described the asymptotic limiting behavior of the integrated mean square error. He also gave a functional form for the optimum k expressed in terms of the sample size and the underlying data distribution.

A similar technique was proposed by Scott for choosing the bin-width based on the underlying data distribution [121]. But his procedure assumes a Gaussian reference standard and requires only the sample size and an estimate of the standard deviation. He also investigated the sensitivity of this procedure using several probability models that violate the Gaussian assumption [2].

Using a maximal difference criterion, Christodoulakis [19] proposed a variable-width histogram. This utilized a uniformity measure for selecting the bucket ranges.

[2]The expressions for the estimation errors resulting from the select and join operations under the traditional histograms can be found in the appropriate sections of Chapters 3 and 4.

This maximal difference criterion brings together attribute values that have a similar ratio of tuples. In other words, the criterion states the maximum and minimum proportions of the attribute values in the bucket must be less than some constant, which is usually based on the individual attribute. He further showed that this criterion reduces the estimation error on all attribute values. This strategy seems most applicable where queries are not uniformly spread over the attribute value domain.

Another method proposed by Kamel and King [69] is based on pattern recognition to partition the buckets of variable-width distribution tables. In pattern recognition, a frequently encountered problem is to compress the storage space of a given image without distorting its actual appearance. This problem can be restated in the context of the selectivity estimation problem to partition the data space into nonuniform buckets that reduce the estimation error using an upper bound on storage space. The idea is to partition the attribute domain into equal-width buckets and compute a homogeneity measure for each bucket. The homogeneity is defined as the measure of the non-uniformity or dispersion around the average number of occurrences per value in the cell. The value of homogeneity for a given bucket can be computed by a given function or by using sampling techniques.

A number of multivariate analogs of the equal-width and equal-height methods can be found in the density estimation literature. Considering the equi-width case, a two or three dimensional cells of equal area or volume can be partitioned, in the joint attribute value domain. The number of tuples with combined values in each cell can be obtained. Considering the equi-depth case, variable width buckets, each with approximately, say x, nearest neighbors, can be derived. The difficulty in this multiattribute situation is that the shape of the cell must also be selected. A square or a cube can be a natural choice, when dealing with the equi-width analog. When the range or variation of one attribute's values is smaller than another's, a rectangular shaped cell is an obvious choice.

Ioannidis and Christodoulakis proposed optimal histograms for limiting worst-case error propagation in the size of join results [63]. They identified two classes of histograms, namely, *serial* and *end-biased* histograms that are associated with optimal worst-case error for join operations. End-biased histograms are frequently

encountered in modern-day database systems. By considering simple *t-clique* queries, where queries involve only equi-join operations without any function symbols, they showed that the optimal histograms for reducing the worst-case error is always serial. They also further showed that for t-clique queries with a large number of joins, *high-biased* histograms, which are a subclass of *end-biased*, are always optimal. The main disadvantage of the serial histograms are that the attribute values need to be sorted with respect to their respective frequencies before being assigned to buckets.

Ioannidis *et al.* [64] discussed the design issues of various classes of histograms and balancing practicality and optimality of them in the use of query optimization. They also investigated various classes of histograms using different constraints (V-Optimal, MaxDiff, Compressed, and Spline-based) and sort and source parameters (Frequency, Spread, and Area). They further provided various sampling techniques for constructing the above histograms and concluded that the V-optimal histogram is the most optimal one for estimating the result sizes of equality-joins and for selection predicates.

Chapter 3

Rectangular Attribute Cardinality Map[1]

In Chapter 1, we discussed the importance of estimating the sizes of results resulting from the various relational operations encountered during a query optimization strategy. We also reviewed, in Chapter 2, some of the popular query result-size estimation techniques found in the literature. In this chapter we introduce the first contribution of this thesis, which is a new catalogue-based non-parametric statistical model called the *Rectangular Attribute Cardinality Map* (R-ACM) that can be used to obtain more accurate estimation results than the currently known estimation techniques. In the next chapter, we propose the second fundamental contribution of this thesis, which is also a catalogue based non-parametric statistical model called the *Trapezoidal Attribute Cardinality Map* (T-ACM) which generalizes the contributions of this chapter from a "step" representation to a "linear" representation. In this and the next chapters we shall introduce these attribute cardinality maps as fundamental tools for query result-size estimation and provide the mathematical foundation for their use. We also, in this and the next chapters, investigate how they can be used in the result-size estimation of various relational operations. Unlike the traditional histograms which are usually created at run-time during query optimization, we propose

[1]Some preliminary results on the theoretical aspects of the work done in this chapter will appear in the *Proceedings of the International Database Engineering and Application Symposium (IDEAS'99)* to be held in Montreal, Canada in August, 1999 [106]. They can also be found as a Carleton University Technical Report, TR-99-01 [107].

to store and maintain these ACMs in the DBMS catalogues.

The traditional histograms, namely equi-width and equi-depth histograms, which we reviewed in Chapter 2, have two major drawbacks. First of all, their fundamental design objective does not address the problem of extreme frequency values within a given bucket. For example, in the equi-width case, all the bucket widths are the same regardless of the frequency distribution of the values. This means large variations between one frequency value to the other is allowed in the same bucket, resulting in higher estimation errors. This problem is somewhat reduced in the equi-depth case. But to avoid the extreme frequency values in a given bucket, a single attribute value may need to be allocated to several consecutive locations in the same bucket. This not only increases the storage requirement but also reduces the efficiency of estimation. The second drawback is that the traditional histograms are usually built at run-time during the query optimization phase using sampling methods. This obviously requires an I/O overhead during the optimization phase and tends to increase the overall optimization time. Moreover, unless sufficiently many samples are taken, the histograms built from the sampled data may not closely reflect the underlying data distribution.

The proposal of ACM as a new tool for query result-size estimation stems from our motivation to address the above drawbacks in the traditional histograms. Specifically, as we shall see, the R-ACM reduces the estimation errors by disallowing large frequency variations within a single bucket. The T-ACM, as we shall see in the next chapter, reduces the estimation errors by using the more accurate trapezoidal-rule of numerical integration. Moreover, as both types of ACMs are catalogue based, they are precomputed, and thus will not incur I/O overhead during run-time.

3.1 Definition

The Rectangular ACM (R-ACM) of a given attribute, in its simplest form, is a one-dimensional integer array that stores the count of the tuples of a relation corresponding to that attribute, and for some subdivisions for the range of values assumed by that attribute. The R-ACM is, in fact, a modified form of histogram. But unlike

Symbol	Explanation
x_i	Number of tuples in attribute X for the i^{th} value of X.
$E(X_i)$	Expected number of tuples in attribute X for the i^{th} value of X.
n_j	No of tuples in the j^{th} sector of an ACM.
l_j	No of distinct values in the j^{th} sector. (Also known as sector width).
s	Number of sectors in the ACM.
τ	Allowable tolerance for an R-ACM
ξ	Size of a relation.
N	Number of tuples in the relation.

Table 3.1: Notations Used in the Thesis

the two major forms of histograms, namely, the equi-width histogram, where all the sector widths are equal, and the equi-depth histogram, where the number of tuples in each histogram bucket is equal, the R-ACM has a variable sector width, and varying number of tuples in each sector. The sector widths or subdivisions of the R-ACM are generated according to a rule that aims at minimizing the estimation error within each subdivision.

Before we proceed, we shall present the notations that will be used throughout this thesis in Table 3.1.

The R-ACM can be either a one-dimensional or a multi-dimensional depending on the number of attributes being mapped. To introduce the concepts formally, we shall deal with the one-dimensional case first.

Definition 3.1 A One dimensional Rectangular ACM: *Let* $\mathcal{V} = \{v_i : 1 \leq i \leq |\mathcal{V}|\}$, *where* $v_i < v_j$ *when* $i < j$, *be the set of values*[2] *of an attribute* X *in relation* R. *Let the value set* $\mathcal{V}$ *be subdivided into* s *number of sector widths according to the range partitioning rule described below. Then the Rectangular Attribute Cardinality Map of*

[2] *In this work, only ordinal numerical values are considered for attributes. It is possible to convert non-numeric attributes (symbolic, scalar-typed, fuzzy etc.) into floating point numbers using a mapping function. This includes most type of attribute values. For non-ordinal data (for example, "real valued" data), we have proposed another type of structure called the Trapezoidal Attribute Cardinality Map (T-ACM), which is discussed in the next chapter.*

attribute X is an integer array in which the j^{th} index maps the number of tuples in the j^{th} value range of the set $\mathcal{V}$ for all j, $1 < j \leq s$.

Rule 3.1 Range Partitioning Rule: *Given a desired tolerance value τ for the R-ACM, the sector widths, $l_j, 1 \leq j \leq s$, of the R-ACM should be chosen such that for any attribute value X_i, its frequency x_i does not differ from the running mean of the frequency by more than the tolerance value τ, where running mean is the mean of the frequency values examined so far in the current sector.*

For example, consider the frequency set $\{8, 6, 9, 7, 19, 21, 40\}$ corresponding to the values $\{X_0, X_1, X_2, X_3, X_4, X_5, X_6\}$ of an attribute X. Using a tolerance value $\tau = 2$, the attribute value range will be partitioned into the three sectors, $\{8, 6, 9, 7\}$, $\{19, 21\}$, $\{40\}$ with sector widths of 4, 2, and 1 respectively.

3.1.1 Generating the Rectangular ACM

Using the *range partitioning rule*, Algorithm `Generate_R-ACM` partitions the value range of the attribute X into s variable width sectors of the R-ACM.

The input to the algorithm are the tolerance value τ for the ACM and the actual frequency distribution of the attribute X. The frequency distribution is assumed to be available in an integer array A, which has a total of L entries for each of the L distinct values of X. For simplicity reasons, we assume that the attribute values are ordered integers from 0 to $L - 1$. The output of the algorithm is the R-ACM for the given attribute value set.

It is obvious that the algorithm, `Generate_R-ACM` generates the R-ACM corresponding to the given frequency value set. Assuming that the frequency distribution of X is already available in array A, the running time of the algorithm `Generate_R-ACM` is $O(L)$ where L is the number of distinct attribute values.

The reader will observe that we have assumed that the tolerance value, τ, is an input to the above algorithm. The question of how to determine an "optimal" tolerance value for an R-ACM is addressed in Chapter 6, using adaptive techniques.

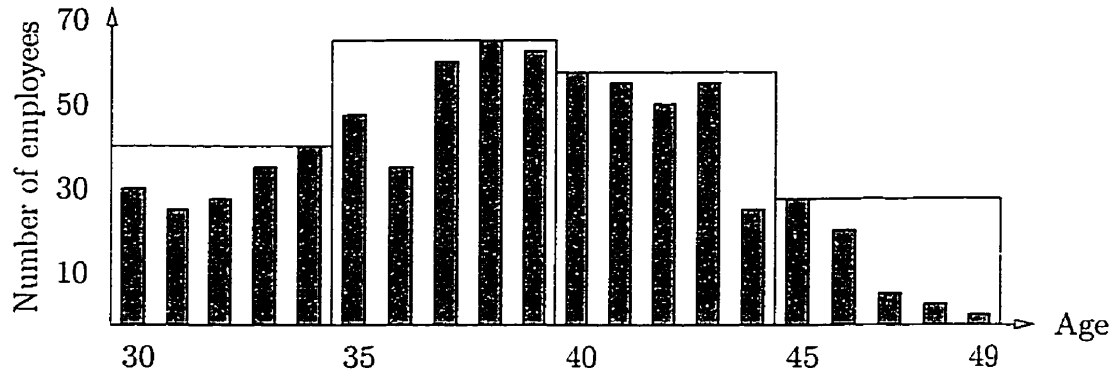
Example 3.1 *Figure 3.1 shows the histogram and the R-ACM of the* `Age` *attribute of a relation* `Emp(SIN, Age, Salary)` *between* `Age = 30` *and* `Age = 49`. *Note that the*

actual frequency for every age value is shown in the histogram as shaded rectangles. As can be noticed, whenever the frequency of the current age differs from the frequency of the running mean value for age in the sector by more than the allowed tolerance value $\tau = 8$, a new partition is formed. From the ACM, the number of tuples in this relation with ages in the range of $34 \leq Age \leq 36$ is 130 and the estimate for the number of employees having age = 48 is 15/3 = 5.

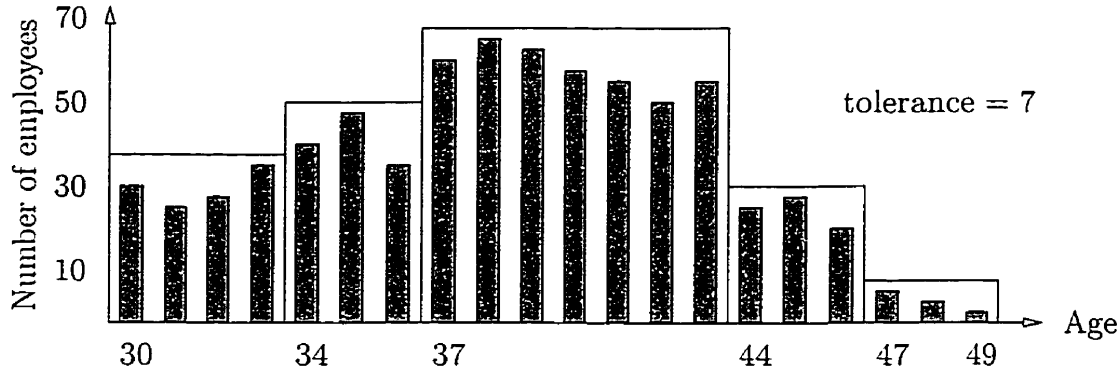
Algorithm 3.2 Generate_R-ACM

```
Input: tolerance τ, frequency distrib. of X as A[0...L-1]
Output: R-ACM
begin
   Initialize_ACM;      /* set all entries in ACM to zero */
   current_mean := A[1]; j := 0;
   ACM[j] := A[1];
   for i:=1 to L-1 do      /* for every attribute value */
      if abs(A[i]- current_mean) < τ
         ACM[j] := ACM[j] + A[i];
            /* compute the running mean */
         current_mean := (current_mean*i + A[i])/(i+1);
      else begin
         l_j := i - 1;      /* set the sector width */
         j++;      /* move to next sector */
         current_mean := A[i];
         ACM[j] := A[i];
      end;
end;
EndAlgorithm Generate_R-ACM;
```

Since the ACM only stores the count of the tuples and not the actual data, it does not incur the usually high I/O cost of having to access the base relations from secondary storages. Secondly, unlike the histogram-based or other parametric and probabilistic counting estimation methods we reviewed in Chapter 2, ACM does not use sampling techniques to approximate the data distribution. Each cell of the ACM maintains the *actual* number of tuples that fall between the boundary values of that



(a) Equi-width Histogram for the Employe Age Distribution



(b) Corresponding Rectangular ACM

Figure 3.1: An Example of the Rectangular Attribute Cardinality Map

cell, and thus, although this leads to an approximation of the density function, there is no approximation of the number of tuples in the data distribution.

The one-dimensional R-ACM as defined above can be easily extended to a multi-dimensional one to map an entire multi-attribute relation. A multi-dimensional ACM can also be used to store the multi-dimensional attributes that commonly occur in geographical, image, and design databases. Multi-dimensional ACM is a promising area for future research work.

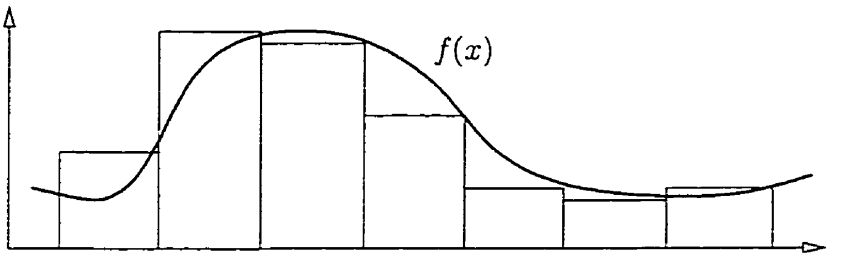
Before we derive any of the analytical properties of the R-ACM, it would be fitting to compare the three methodologies from a common conceptual perspective.

3.2 Rationale for the Rectangular ACM

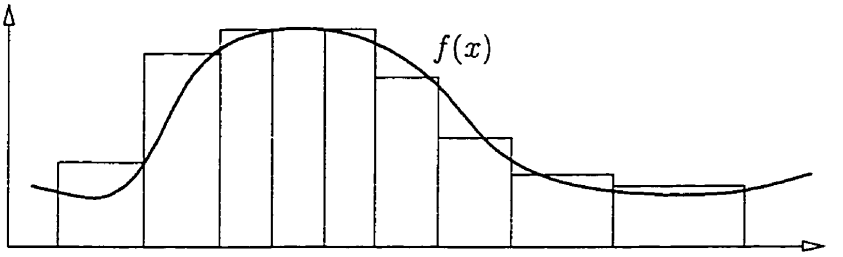
Without loss of generality, let us consider an arbitrary continuous frequency function $f(x)$. Figure 3.2 shows the histogram partitioning of $f(x)$ under the traditional equi-width, equi-depth methods and the R-ACM method. We note that in the equi-width case, regardless of how steep the frequency changes are in a given sector, the sector widths remain the same across the attribute value range. This means the widely different frequency values of all the different attribute values are assumed to be equal to that of the average sector frequency. Thus there is an obvious loss of accuracy with this method. On the other hand, in the equi-depth case, the area of each histogram sector is the same. This method still has a number of sectors with widely different frequency values and thus suffers from the same problem as the equi-width case. In the R-ACM method, we note that whenever there is a steep frequency changes, the corresponding sector widths proportionally decrease (or the number of sectors proportionally increases). Hence the actual frequencies of all the attribute values within a sector are kept closer to the average frequency of that sector. This partitioning strategy obviously increases the estimation accuracy. Figure 3.3 shows a comparison of probability estimation errors obtained on all three estimation methods on synthetic data.

The rationale for partitioning the attribute value range using a tolerance value is to minimize the variance of values in each ACM sector, and thus to minimize the estimation errors. Since variance of an arbitrary attribute value X_k is given as $Var(X_k) = E[(x_k - \frac{n_j}{l_j})^2]$, forcing the difference between the frequency of a given value and the mean frequency to be less than the tolerance τ, i.e: $x_k - \frac{n_j}{l_J} \leq \tau$, will ensure that the variance of the values falls within the acceptable range. To get a flavor for the ultimate goal of our endeavor, we allude to the expression for the ACM variance which we shall derive in a subsequent lemma (Lemma 3.5). It will later become clear that minimizing the variance of the individual sectors will result in a lower value for the variance of the ACM.

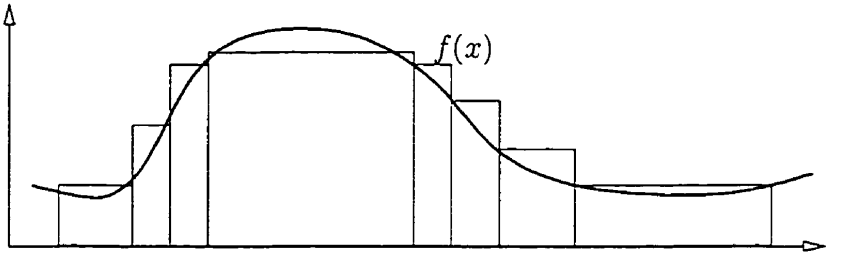
To demonstrate the relationship between the selection (random match) estimation error and the variance of the ACM, we conducted an experiment in which we



(a) Equi-width Histogram



(b) Equi-depth Histogram



(c) Rectangular ACM

Figure 3.2: Comparison of R-ACM and Traditional Histograms. Note in (a), the sector widths are equal, in (b) the areas of the sectors are equal, in (c) the sector widths become smaller whenever there is a significant change in the frequencies of the attribute values.

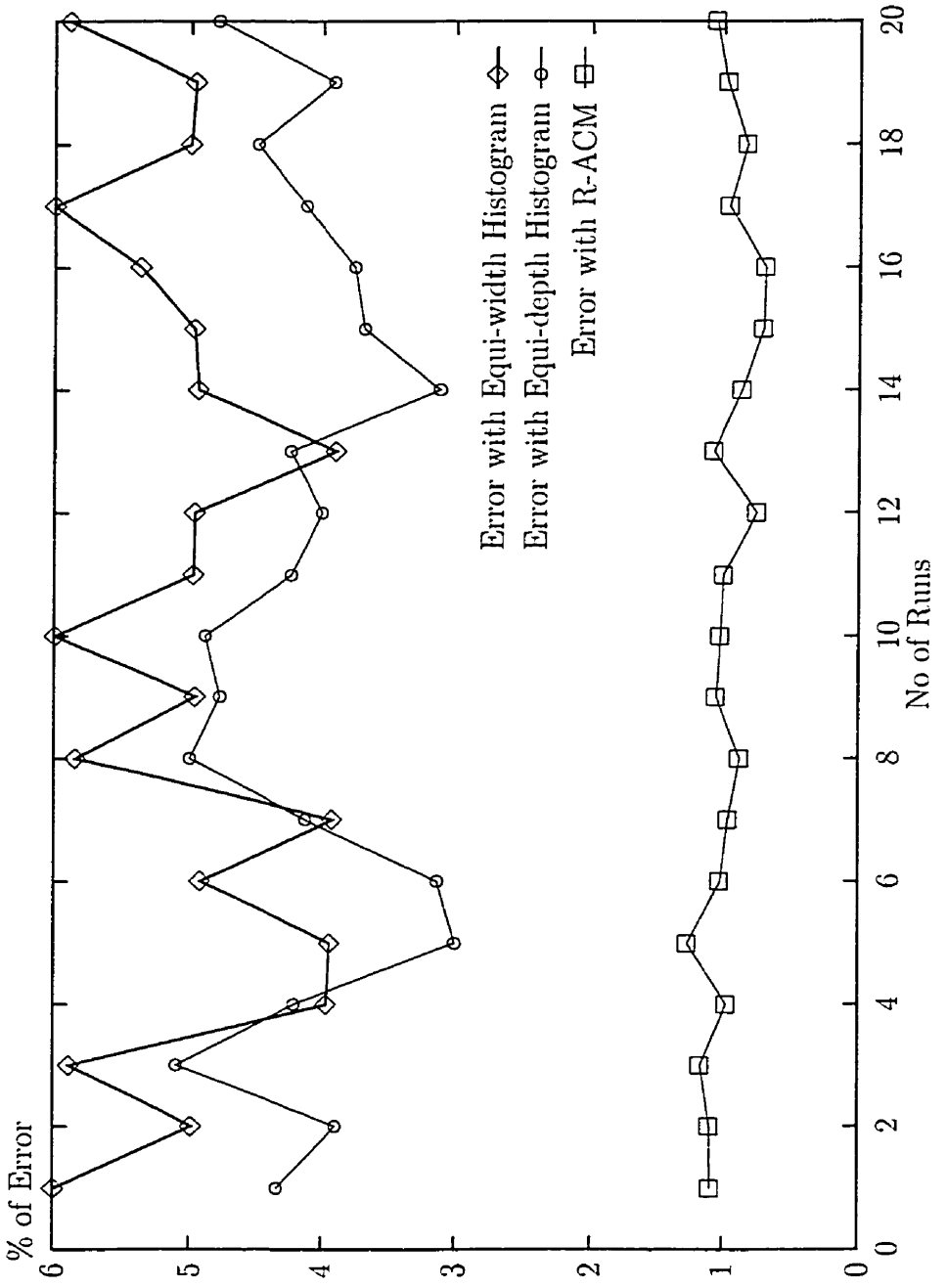


Figure 3.3: Comparison of Equi-width, Equi-depth Histograms and the R-ACM for Frequency (Probability) Estimation: Each experiment was run 500,000 times to get the average percentage errors. Estimation errors are given for exact match on a random distribution with 100,000 tuples and 1000 distinct values. For the R-ACM, tolerance value was $\tau = 3$.

changed the underlying data distribution and computed the variance of the ACM (i.e: $Var(ACM)$) for three different tolerance values. The errors between the estimated and actual size of random matches are plotted against the computed variance of the ACM, and is shown in Figure 3.4. The advantages of using the R-ACM are obvious. Indeed, more detailed expressions and experimental results will later strengthen this *initial* claim.

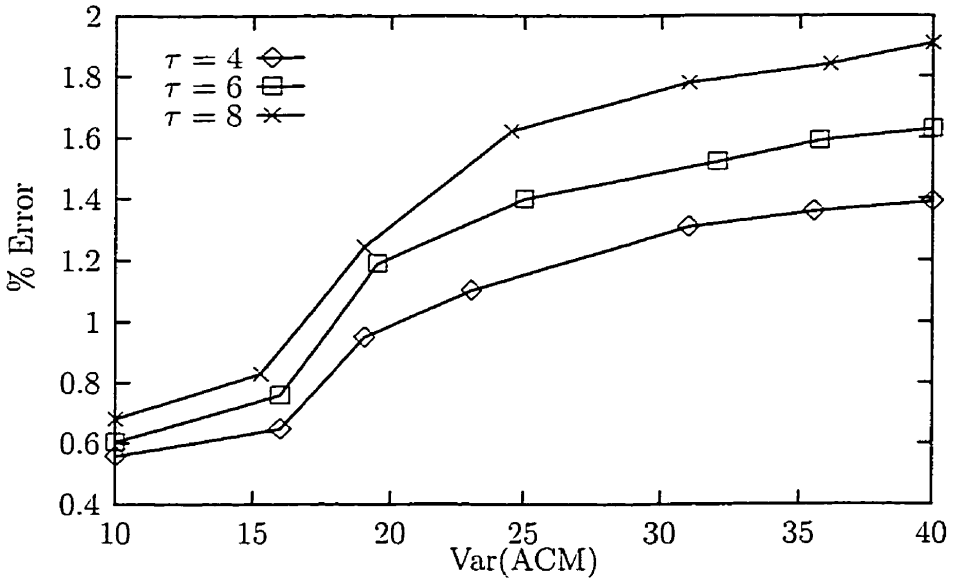


Figure 3.4: Frequency Estimation Error vs ACM Variance: The ACM sectors were partitioned using three different tolerance values and the resulting ACM variances were computed for various data distributions. Using random selection queries (matches), the errors between the actual and the expected frequencies were obtained.

3.3 Density Estimation Using Rectangular ACM

We shall now study the properties of the R-ACM with regard to density approximation. To do this we consider a one-dimensional Rectangular ACM sector of width l with n tuples. Also to render the analysis feasible, we make the following assumption.

Assumption 1 *The attribute values within a Rectangular ACM sector are uniformly distributed.*

Rationale: Since the sector widths of the R-ACM are chosen so that the frequency of the values within the sectors do not differ by more than the allowed tolerance, τ, we can see that these frequencies are guaranteed to be close to each other. The original System R research work [122] relied on the (often erroneous) assumption that the frequencies of an attribute value are uniformly distributed across the *entire attribute value domain.* With the adoption of the equi-width and equi-depth histograms in the modern-day database systems (see Table 2.2), this was improved by making the uniformity assumptions only within the histogram buckets. Our uniformity assumption within an R-ACM sector is a much weaker assumption than that used in any other previous works on histograms, due to its partitioning strategy. Indeed, as we shall show from the experimental results later, this assumption is satisfied with almost no additional approximation.

Using the above uniformity assumption, we shall now derive the probability mass distribution for the R-ACM.

Lemma 3.1 *The probability mass distribution for the frequencies of the attribute values in an R-ACM is a Binomial distribution with parameters* $(n, \frac{1}{l})$.

Proof: Since there are l distinct values in the sector, the probability of any of these l values, say X_i, occurring in a random assignment of that value to the sector is equal to $\frac{1}{l}$. (See Figure 3.5).

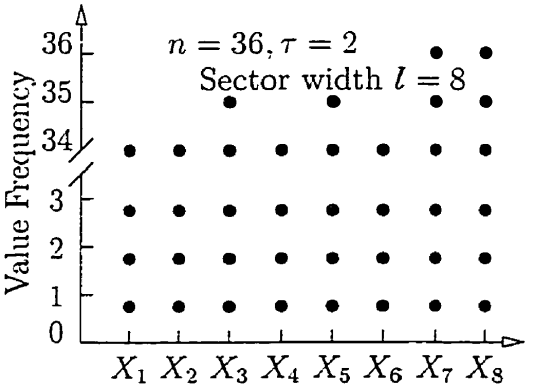


Figure 3.5: Distribution of Values in an ACM Sector

Consider an arbitrary permutation (or arrangement) of the n tuples in the sector. Suppose the value X_i occurs exactly x_i times. This means all the other $(l-1)$ values must occur a combined total of $(n-x_i)$ times. Since the probability of X_i occurring once is $\frac{1}{l}$, the probability of it not occurring is $(1-\frac{1}{l})$. Hence the probability of an arbitrary permutation of the n tuples, where the value X_i occurs exactly x_i times and the other values collectively occur $n-x_i$ times is,

$$\left(\frac{1}{l}\right)^{x_i}\left(\frac{l-1}{l}\right)^{n-x_i}. \tag{3.1}$$

Clearly there are $\binom{n}{x_i}$ different permutations of the n tuples in the sector where the above condition is satisfied. Hence we find that the total probability that an arbitrary value X_i occurs exactly x_i times is,

$$p_{X_i}(x_i) = \binom{n}{x_i}\left(\frac{1}{l}\right)^{x_i}\left(\frac{l-1}{l}\right)^{n-x_i} \tag{3.2}$$

which is exactly the Binomial distribution with parameters $(n, \frac{1}{l})$. This proves the lemma. □

3.4 Maximum Likelihood Estimate Analysis for the Rectangular ACM

In the previous section we showed that the frequency distribution for a given attribute value in the R-ACM obeys a Binomial distribution. With this as a background, we shall now derive a maximum likelihood estimate for the frequency of an arbitrary attribute value in an R-ACM sector. In classical statistical estimation theory, we are usually interested in estimating the parameters (such as the mean or other unknown characterizing parameters) of the distribution of one or more random variables. In our problem, we are interested in estimating the value of the occurrence of the random variable (the frequency x_i) which we assume is "inaccessible". We do this however in terms of an observation of one or more accessible random variables (the total number

of tuples, n and the width of the R-ACM sector, l). To do this we shall derive the maximum likelihood estimate, which maximizes the **corresponding** likelihood function. Indeed the result which we get is both intuitively appealing and quite easy to comprehend.

Theorem 3.1 *For a one-dimensional rectangular ACM, the maximum likelihood estimate of the number of tuples for a given value X_i of attribute X is given by,*

$$\hat{x}_{ML} = \frac{n}{l}$$

where n is the number of tuples in the sector containing the value X_i and l is the width of that sector.

Proof: We know from Lemma 3.1 that the frequency distribution of a given attribute value in an R-ACM sector is a Binomial distribution. So the probability mass function of the frequency distribution of an attribute value $X = X_\alpha$ in an R-ACM sector can be written as,

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

where x is the number of occurrences of X_α. Let

$$\mathcal{L}(x) = f(x) = \binom{n}{x} p^x (1-p)^{n-x}.$$

$\mathcal{L}(x)$ is the traditional *likelihood function* of the random variable X on the parameter x which we intend to maximize. We are interested in finding out the maximum likelihood estimate for this parameter x. Taking natural logarithm on both sides of the likelihood function, we have,

$$\begin{aligned} \ln \mathcal{L}(x) &= \ln n! - \ln x! - \ln(n-x)! + x \ln p + (n-x)\ln(1-p) \\ &= \ln \Gamma(n+1) - \ln \Gamma(x+1) - \ln \Gamma(n-x+1) + \\ &\quad x \ln p + (n-x)\ln(1-p) \end{aligned} \tag{3.3}$$

where $\Gamma(x)$ is the Gamma function given by, $\Gamma(x) = \int_0^\infty e^{-t}t^{x-1}dt$. Now using the well known identity,

$$\Gamma(\alpha) = \frac{\Gamma(\alpha+k+1)}{\alpha(\alpha+1)\ldots(\alpha+k)}$$

we find that,

$$\Gamma(n-x+1) = \frac{\Gamma(n+1)}{(n-x+1)(n-x+2)\cdots n} \text{ and } \Gamma(x+1) = \frac{\Gamma(n+1)}{(x+1)(x+2)\cdots n}.$$

Thus substituting the above expressions for $\Gamma(n-x+1)$ and $\Gamma(x+1)$ in Equation 3.3, we find,

$$\begin{aligned}\ln\mathcal{L}(x) &= -\ln\Gamma(n+1) + x\ln p + (n-x)\ln(1-p) + \\ &\quad \ln(x+1) + \ln(x+2) + \ldots + \ln n + \\ &\quad \ln(n-x+1) + \ln(n-x+2) + \ldots + \ln n\end{aligned}$$

Now differentiating $\ln\mathcal{L}(x)$ with respect to x, we obtain,

$$\frac{d}{dx}\ln\mathcal{L}(x) = \ln p - \ln(1-p) + \sum_{r=x+1}^{n-x}\frac{1}{r}.$$

Setting $\frac{d\{\mathcal{L}(x)\}}{dx} = 0$, and noting that $\sum_{r=x+1}^{n-x}\frac{1}{r} \leq \ln\left(\frac{n-x}{x}\right)$, $\hat{x}_{ML}$ of x is obtained as,

$$\frac{p(n-x)}{(1-p)x} \geq 1.$$

This inequality is solved for $x \leq np$. But, by virtue of the underlying distribution, since we know that the likelihood function monotonically increases till its maximum, we conclude that,

$$\hat{x}_{ML} = np.$$

But we have already seen earlier that, due to the uniformity assumption within an R-ACM sector, $p = \frac{1}{l}$. So we have,

$$\hat{x}_{ML} = \frac{n}{l}.$$

Hence the theorem. □

The maximum likelihood estimate, $\hat{x}_{ML} = np$, which we derived using the Gamma function above is, most of the time, not an integer. In fact, the maximum likelihood estimate reaches its upper limit of np at integer values only in very special cases. If we are interested in the integer maximum likelihood value which is related to the above maximum likelihood estimate, we have to discretize the space. Thus, considering the analogous discrete case, we have the following theorem.

Theorem 3.2 *For a one-dimensional rectangular ACM, the maximum likelihood estimate of the number of tuples for a given value X_i of attribute X falls within the range of,*

$$\frac{(n+1)}{l} - 1 \leq \hat{x}_{ML} \leq \frac{(n+1)}{l},$$

where n is the number of tuples in the sector containing the value X_i and l is the width of that sector.

Proof: The probability mass function $f(x) = \binom{n}{x} p^x (1-p)^{n-x}$ is a steadily increasing function until it reaches the maximum likelihood value, $x = \hat{x}_{ML}$. For any $x > \hat{x}_{ML}$, $f(x)$ is a steadily decreasing function. Hence we can obtain an integer value for the maximum likelihood estimate by solving the following two discrete inequalities simultaneously.

$$f(x) - f(x+1) > 0 \tag{3.4}$$

$$f(x) - f(x-1) > 0 \tag{3.5}$$

From Equation (3.4), we have,

$$\begin{aligned} f(x) &- f(x+1) > 0 \\ \binom{n}{x} p^x (1-p)^{n-x} &- \binom{n}{x+1} p^{x+1} (1-p)^{n-x-1} > 0 \\ \frac{n!}{x!(n-x)!}(1-p) &- \frac{n!}{(x+1)!(n-x-1)!} p > 0 \\ \frac{1-p}{n-x} &- \frac{p}{x+1} > 0 \text{ or} \\ x &> p(n+1) - 1. \end{aligned}$$

Similarly considering Equation (3.5), by using similar algebraic manipulation, we get,

$$\begin{aligned} f(x) &- f(x-1) > 0 \\ \binom{n}{x} p^x (1-p)^{n-x} &- \binom{n}{x-1} p^{x-1} (1-p)^{n-x-1} > 0 \text{ or} \\ x &< p(n+1). \end{aligned}$$

Since $p = \frac{1}{l}$, the theorem follows. □

3.5 Expected Value Analysis for the R-ACM and Self-Join Error Estimate

The maximum likelihood estimate of the frequency of a given attribute value tells us that the attribute value would have a frequency of $\hat{x}_{ML}$ with maximum degree of certainty when compared to the other possible frequency values. But even though the attribute value occurs with the maximum likelihood frequency with high probability, it can also occur with other frequencies with smaller probabilities. Hence when we need to find the worst-case and average-case errors for our result size estimations, we need to obtain another estimate which includes all these possible frequency values. One such estimate is the expected value of the frequency of a given attribute value. We use our Binomial model to find the expected value of the frequency of an attribute

value as given in the following lemma and develop a sequence of results regarding the corresponding estimates.

Lemma 3.2 *For a one-dimensional rectangular ACM, the expected number of tuples for a given value X_i of attribute X is $E(X_i) = n/l$, where n is the number of tuples in the sector containing the value X_i and l is the width of that sector.*

Proof: From Equation (3.2), the probability that the value X_i occurs exactly k times is,

$$p_{X_i}(k) = \binom{n}{k} \left(\frac{1}{l}\right)^k \left(\frac{l-1}{l}\right)^{n-k}$$

which is a Binomial distribution with parameters $(n, \frac{1}{l})$. The result follows directly from the fact that the mean of the binomial distribution, *Binomial* (n, p), is np, where p is the probability of success. □

The above result is very useful in estimating the results of selection and join operations.

3.5.1 Estimation Error with Rectangular ACM

It has been shown that even a small error in the estimation results, when propagated through several intermediate relational operations, can become exponential and be devastating to the performance of a DBMS [62]. In this section we provide some definitions for estimating the errors based on the variance, and provide a technique to measure the estimation errors obtained from the R-ACM.

The variance of a random variable X measures the spread or dispersion that the values of X can assume and is defined by $Var(X) = E\{[X - E(X)]^2\}$. It is well known that $Var(X) = E(X^2) - [E(X)]^2$. Thus the variance of the frequency of the k^{th} value of the attribute X in the j^{th} sector is given as,

$$Var(X_k) = E\left[\left(x_k - \frac{n_j}{l_j}\right)^2\right]$$

Expanding the right hand side, we obtain,

$$Var(X_k) = \sum_{i=0}^{n_j} x_k^2 \left(\frac{1}{l_j}\right)^i \left(1 - \frac{1}{l_j}\right)^{n_j - i} - \left(\frac{n_j}{l_j}\right)^2 \tag{3.6}$$

Lemma 3.3 *The variance of the frequency of an attribute value X in sector j of an R-ACM is,*

$$Var(X) = \frac{n_j(l_j - 1)}{l_j^2} \tag{3.7}$$

Proof: It is well known that the variance of a Binomial distribution with parameters (n, p) is $np(1-p)$. Using the property of the Binomial distribution, the expression for the variance given in Equation (3.6) can be reduced to the one given in the lemma. □

Lemma 3.4 *The sector variance of the j^{th} rectangular ACM sector is,*

$$Var_j = \frac{n_j(l_j - 1)}{l_j} \tag{3.8}$$

Proof: We note that $Var(X_k)$ is same for all $k, 1 \leq k \leq l_j$, in a given sector. Since the random variables are independent[3], summing up the variances of all the values in the sector will give us an upper bound for the estimation error or variance of the sector. The result follows. □

Similarly, summing up the variances of all the sectors, we obtain an expression for the variance of the entire ACM, which is given in the following lemma.

Lemma 3.5 *The variance of an R-ACM is given by,*

$$Var(ACM) = \sum_{i=1}^{s} Var_i \tag{3.9}$$

[3]We really don't need independence for this. Uncorrelatedness is sufficient.

where s is the number of sectors in the ACM.

Proof: The result follows directly from the fact that the variances in each sector are independent, and thus summing up the sector variances yields the variance of the entire ACM. □

3.5.2 Error Estimates and Self-Joins

It is interesting to study the join estimation when a relation is joined with itself. These self-joins frequently occur with 2-way join queries. It is well known [91] that the self-join is a case where the query result size is maximized because the highest occurrences (frequencies) in the joining attributes correspond to the same attribute values. Assuming that the duplicate tuples after the join are not eliminated, we have the following lemma.

Lemma 3.6 *The error, ϵ, resulting from a self-join[4] of relation R on attribute X using a rectangular ACM is given by,*

$$\epsilon = Var(ACM) + \sum_{j=1}^{s} \left\{ \sum_{k=1}^{l_j} x_k^2 - \frac{n_j^2 + n_j l_j - n_j}{l_j} \right\}.$$

Proof: Since there are $L = \sum_{i=1}^{s} l_j$ values for attribute X, the actual value, ξ and expected value κ of the join size can be estimated as follows.

$$\xi = \sum_{i=1}^{L} x_i^2 = \sum_{j=1}^{s} \sum_{k=1}^{l_j} x_k^2.$$

The frequency of an arbitrary attribute value is computed from the R-ACM as the expected value $E(x_i)$, which is the average frequency of the R-ACM sector. Hence

[4]In one his recent works [64], considering a V-optimal histogram (defined in his work), Ioannidis has claimed that the error resulting from a self-join is equal to the histogram variance. Indeed, although his result is basically true from an order-notation point of view, the more exact expression is given in this lemma.

the result of self-joining this attribute value would be $[E(x_i)]^2$. Hence the size of the join computed by the ACM, κ, is,

$$\begin{aligned}\kappa &= \sum_{i=1}^{L}[E(x_i)]^2 \\ &= \sum_{j=1}^{s}\sum_{i=1}^{l_j}\left(\frac{n_j}{l_j}\right)^2 = \sum_{j=1}^{s} l_j \left(\frac{n_j}{l_j}\right)^2.\end{aligned}$$

Hence the error in estimation of the self-join is,

$$\begin{aligned}\xi - \kappa &= \sum_{j=1}^{s}\sum_{k=1}^{l_j} x_k^2 - \sum_{j=1}^{s} l_j \left(\frac{n_j}{l_j}\right)^2 \\ &= \sum_{j=1}^{s} l_j \left\{ \frac{1}{l_j}\sum_{k=1}^{l_j} x_k^2 - \left(\frac{n_j}{l_j}\right)^2 \right\}.\end{aligned}$$

But since $Var(ACM) = \sum_{j=1}^{s} l_j Var_j$, adding and subtracting $Var(ACM)$ from RHS of the above expression results in,

$$\begin{aligned}\xi - \kappa &= \sum_{j=1}^{s} l_j Var_j + \sum_{j=1}^{s} l_j \left\{ \frac{1}{l_j}\sum_{k=1}^{l_j} x_k^2 - \left(\frac{n_j}{l_j}\right)^2 \right\} - \sum_{j=1}^{s} l_j \frac{n_j(l_j-1)}{l_j^2} \\ &= Var(ACM) + \sum_{j=1}^{s} \left\{ \sum_{k=1}^{l_j} x_k^2 - \frac{n_j^2 + n_j l_j - n_j}{l_j} \right\}. \qquad (3.10)\end{aligned}$$

and the result is proved. □

Observation 3.1 *The error in self-join estimation of a relation from the R-ACM is* $O(Var(ACM))$.

Our extensive experiments with R-ACM indicate that the error in the self-join estimation is approximately equal to the variance of the ACM since the contribution due to the second term in Equation (3.10) seems to be less dominant. This agrees well with the results claimed by Ioannidis[64].

Theorem 3.3 *The variance of a rectangular ACM corresponding to attribute X is,*

$$Var(ACM) = N - \sum_{j=1}^{s} \frac{n_j}{l_j}. \tag{3.11}$$

Proof: From Lemma 3.5, the variance of an R-ACM is given by $Var(ACM) = \sum_{j=1}^{s} Var_j$, where Var_j is the variance of the j^{th} sector. But from Lemma 3.4, $Var_j = n_j(l_j - 1)/l_j$. Hence,

$$\begin{aligned} Var(ACM) &= \sum_{j=1}^{s} \frac{n_j(l_j - 1)}{l_j} \\ &= \sum_{j=1}^{s} n_j - \sum_{j=1}^{s} \frac{n_j}{l_j} \\ &= N - \sum_{j=1}^{s} \frac{n_j}{l_j}. \end{aligned}$$

Hence the theorem. □

3.6 Worst-Case Error Analysis for the R-ACM

As mentioned earlier, forcing a frequency value to be within a given tolerance τ to the current mean ensures that the frequency distribution within an R-ACM sector is very close to uniform. We note that whenever every frequency value is always consistently smaller (or always consistently greater) than the current mean by the tolerance value τ, the resulting sectors will be far from uniform. So we have the following definition.

Definition 3.2 *A distribution is said to be "least uniform" if for every attribute value of X_i, the frequency x_i attains the value $x_i = \mu_{i-1} - \tau$, if x_i is decreasing or $x_i = \mu_{i-1} + \tau$ if x_i is increasing, where μ_{i-1} is the mean of the first $(i-1)$ frequencies. A sector of the former type is called a monotonically decreasing R-ACM sector. Similarly a sector of the latter type is called a monotonically increasing R-ACM sector.*

The motivation for the above definition comes from the following observation. Assume that during the process of constructing an R-ACM sector, the next value x_i is smaller than the current mean μ_{i-1}. We note that if $x_i < \mu_{i-1} - \tau$ then, we will have generated a new sector. Hence the smallest value that x_i can assume is $x_i = \mu_{i-1} - \tau$. The resulting distribution is shown in Figure 3.6(a). This is formally given by the following lemma. This lemma is given for the case when the sector is a decreasing R-ACM sector, or in other words, when every frequency value is always smaller than the previous mean. The case for the increasing R-ACM sector is proved in an analogous way.

Lemma 3.7 *A decreasing R-ACM sector is "least uniform", if and only if*

$$x_k = a - \sum_{i=1}^{k-1} \frac{\tau}{i} \qquad \text{for } 1 \leq k \leq l_j.$$

Proof: Note that in this case, least uniformity occurs when the quantity $E_S(X) - x_i$ assumes its largest possible value (or when x_i assumes its minimum value), where $E_S(X)$ is the expected value of X in a sector. We assume that the frequency of the first value is $x_1 = a$.

Basis: $x_1 = a$: Since the current mean is $E_S(X) = a$, and the sector is a monotonically decreasing R-ACM sector, the minimum possible value for x_2 is obtained from $E_S(X) - x_2 = \tau$. This is achieved when $x_2 = a - \tau$.

Inductive hypothesis: Assume the statement is true for $n = k$. So the sector is the least uniform for the first k values and the frequencies take the following values:

$$\begin{aligned} x_1 &= a \\ x_2 &= a - \tau \\ x_3 &= a - \frac{3\tau}{2} \\ &\vdots \quad \vdots \quad \vdots \\ x_k &= a - \sum_{i=1}^{k-1} \frac{\tau}{i} \end{aligned}$$

For $n = k + 1$: The minimum possible value for x_{k+1} without creating a new sector is obtained when $E_S(X) - x_{k+1} = \tau$. This is achieved when $x_{k+1} = E_S(X) - \tau$. So,

$$\begin{aligned} x_{k+1} &= \frac{x_1 + x_2 + \ldots + x_k}{k} - \tau \\ &= a - \frac{\tau + \frac{3\tau}{2} + \ldots + \sum_{i=1}^{k-1} \frac{\tau}{i}}{k} - \tau \\ &= a - \frac{(k-1)\tau + (k-2)\frac{\tau}{2} + (k-3)\frac{\tau}{3} + \ldots + \frac{\tau}{k-1}}{k} - \tau \\ &= a - \frac{k\tau(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k-1}) - (k-1)\tau}{k} - \tau = a - \sum_{i=1}^{k} \frac{\tau}{i} \end{aligned}$$

This proves the lemma. □

Lemma 3.8 *An increasing R-ACM sector is "least uniform", if and only if*

$$x_k = a + \sum_{i=1}^{k-1} \frac{\tau}{i} \qquad \text{for } 1 \leq k \leq l_j.$$

Proof: The proof is analogous to that of Lemma 3.7 and omitted in the interest of brevity. □

We shall now present a tight bound for the frequency x_i of an arbitrary attribute value X_i in the following theorem.

Theorem 3.4 *If the value X_i falls in the j^{th} sector of an R-ACM, then the number of occurrences of X_i is,*

$$\frac{n_j}{l_j} - \left|\tau\left[\ln\left(\frac{l_j}{i-1}\right) - 1\right]\right| \leq x_i \leq \frac{n_j}{l_j} + \left|\tau\left[\ln\left(\frac{l_j}{i-1}\right) - 1\right]\right|$$

where n_j and l_j are the number of tuples and the sector width of the j^{th} sector.

Proof: Consider a sector from an R-ACM. Let the frequency of the first value x_1 be a. We note that the R-ACM sector will become a "skewed" one, if the subsequent

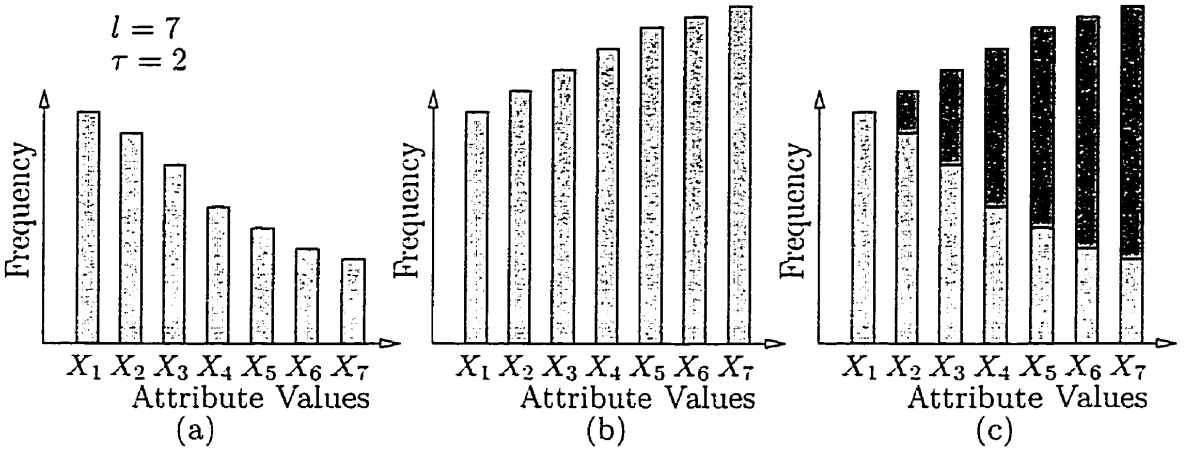


Figure 3.6: (a) A decreasing R-ACM sector; (b) An increasing R-ACM sector; (c) Superimposition of the decreasing and increasing frequency distributions.

values are **all** smaller or **all** greater than the previous mean by τ. From Lemma 3.7 and Lemma 3.8, it is obvious that such sectors are the "least uniform" ones in an R-ACM, and consequently the largest estimation error occurs in such a "skewed" sector. Assume that the frequency values from x_1 to x_{l_j} decrease monotonically in this manner. In other words, if the mean of the first k values is μ_k, then the next value will take its lowest allowable frequency, $\mu_k - \tau$. The resulting distribution is shown in Figure 3.6(a). From Lemma 3.7, the frequency of an arbitrary attribute value X_i is given by,

$$x_i = a - \sum_{k=1}^{i-1} \frac{\tau}{k}$$

The expected value $E(X_i)$ is the mean frequency for the entire sector. So,

$$\begin{aligned} E(X_i) &= \frac{\sum_{k=1}^{l_j} x_k}{l_j} \\ &= a - \tau\left(1 + \frac{1}{2} + \ldots + \frac{1}{l_j - 1} - \frac{l_j - 1}{l_j}\right) \end{aligned}$$

But the frequency of an arbitrary value X_i is,

$$x_i = a - \tau\left(1 + \frac{1}{2} + \ldots + \frac{1}{i-1}\right)$$

So the estimation error is,

$$\begin{aligned} x_i - E(X_i) &= \tau\left(\sum_{k=i}^{l_j}\frac{1}{k} - 1\right) \\ &\leq \left|\tau\left[\ln\left(\frac{l_j}{i-1}\right) - 1\right]\right| \end{aligned}$$

Hence, $$x_i \geq \frac{n_j}{l_j} - \left|\tau\left[\ln\left(\frac{l_j}{i-1}\right) - 1\right]\right|$$

Similarly using symmetry, for an R-ACM sector with monotonically increasing frequency value, we can show that,

$$x_i \leq \frac{n_j}{l_j} + \left|\tau\left[\ln\left(\frac{l_j}{i-1}\right) - 1\right]\right|.$$

The theorem follows. □

The reader should note that the composite effect of the monotonically decreasing frequency sequence and the monotonically increasing sequence restricts the set of frequency values which the attributes can take. Figure 3.6(c) represents the range of likely frequencies when the frequencies in the sector are generated from the frequency of the first attribute value. We should note that when the frequencies are generated based on the running mean, the worst case error will be at its maximum at the beginning of the sector and then gradually decrease as we move towards the end of the sector. This is the result of the fact that the sample mean converges to the true mean and the variance tends to zero by the law of large numbers. What is interesting, however, is that by virtue of the fact the next sample deviates from the current mean by at most τ, the error in the deviation is bounded from the mean in a logarithmic

manner. If this is not taken into consideration, the above figure can be interpreted erroneously. The following example illustrates the implications of the above theorem.

Example 3.2 *Consider an R-ACM sector of width 10 containing 124 tuples, where the R-ACM is partitioned using a tolerance value $\tau = 3$. Let us attempt to find the estimated frequency ranges for the attribute values (a) X_3 and (b) X_6.*

(a). The frequency range of X_3 is,

$$\begin{aligned} 12.4 - |3(\ln 5 - 1)| \leq \quad & x_3 \quad \leq 12.4 + |3(\ln 5 - 1)| \\ 10.57 \leq \quad & x_3 \quad \leq 14.23 \end{aligned}$$

(b). The frequency range of X_6 is,

$$\begin{aligned} 12.4 - |3(\ln 2 - 1)| \leq \quad & x_6 \quad \leq 12.4 + |3(\ln 2 - 1)| \\ 11.48 \leq \quad & x_6 \quad \leq 13.32 \end{aligned}$$

Notice that in both the above cases, the possible frequency values from an equi-width or equi-depth histograms are $0 \leq x \leq 124$, where $x = x_3$ or x_6. The power of the R-ACM in the estimation is obvious!

3.6.1 Worst-Case Error in Estimating the Frequency of an Arbitrary Value X_i

The previous theorem gives an estimate of the number of occurrences of an arbitrary attribute value by considering the worst-case R-ACM sector. So the worst-case error in the above estimate is given by the frequency range that the attribute value can assume. The following theorem provides a worst-case error for this frequency estimation.

Theorem 3.5 *The worst-case error, ϵ, in estimating the frequency of an arbitrary attribute value X_i in a rectangular ACM is given by,*

$$\epsilon = \left| \tau \left[\ln \left(\frac{l}{i-1} \right) - 1 \right] \right|$$

where τ is the tolerance value used in generating the R-ACM.

Proof: It was proved in Theorem 3.4 that the actual frequency of an attribute value can at most differ from the estimated frequency value by, $\left| \tau \left[\ln \left(\frac{l}{i-1} \right) - 1 \right] \right|$. Hence the theorem. □

3.6.2 Worst-Case Error in Estimating the Sum of Frequencies in an Attribute Value Range

Unlike the previous case, when estimating the sum of frequencies in an attribute value range, we have to consider three distinct cases. These are namely the cases when,

1. The attribute value range spans across one R-ACM sector.
2. The attribute value range falls completely within one R-ACM sector.
3. The attribute value range spans across more than one R-ACM sector.

In the first case, estimation using the R-ACM gives the accurate result (n_j) and there is no estimation error. The estimation error in the second case is shown in the example Figure 3.7 (a) and is given by the theorem below. The estimation error in the third case can be obtained by noting that it is in fact the combination of the first and second cases.

Theorem 3.6 *An estimate for the worst-case error, ϵ, in estimating the sum of frequencies of X_i in the attribute value range, $X_\alpha \leq X_i \leq X_\beta$, when both X_α and X_β fall completely within an R-ACM sector is given by,*

$$\epsilon = \left| \ln \left\{ l_j^{(\beta-\alpha+1)\tau} \frac{(\beta-1)!}{(\alpha-2)!} \right\} \right| - (\beta - \alpha + 1)\tau$$

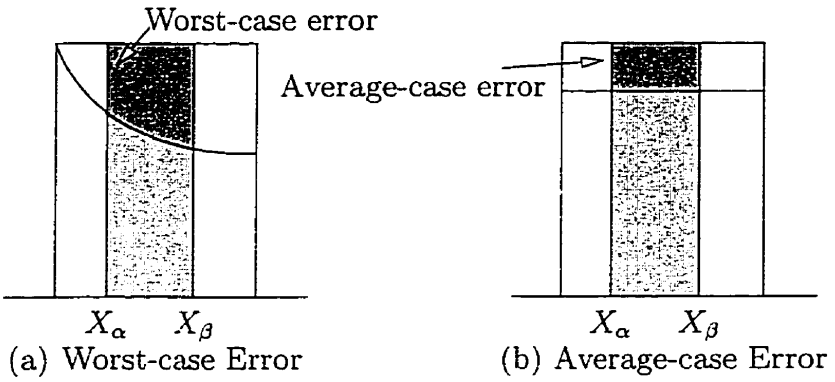


Figure 3.7: Estimation of a Range Completely within an R-ACM Sector

where $\beta > \alpha$.

Proof: Using Theorem 3.4, we can compute the error resulting from this result size estimation by summing up the worst-case error for each of the attribute values in the given value range. Thus we have the following cumulative error.

$$\begin{aligned}
\epsilon &= \sum_{i=\alpha}^{\beta} \left\{ \tau \left| \ln\left(\frac{l_j}{i-1}\right) - 1 \right| \right\} \\
&= \tau \sum_{i=\alpha}^{\beta} \ln\left(\frac{l_j}{i-1}\right) - (\beta - \alpha + 1)\tau \\
&= \left| \ln\left\{ l_j^{(\beta-\alpha+1)\tau} \frac{(\beta-1)!}{(\alpha-2)!} \right\} \right| - (\beta - \alpha + 1)\tau.
\end{aligned}$$

Hence the theorem follows. □

3.7 Average Case Error with Rectangular ACM

The previous theorem gives the bounds for the worst-case estimation error by considering the "least uniform" sector. But what about the estimation error on the average?

3.7.1 Average Case Error in Estimating the Frequency of an Arbitrary Value X_i

Average case error occurs in a truly random sector. In a random sector, frequency values do not monotonically increase or decrease as in the least uniform case we considered before. Instead, they take on a random value around the mean bounded by the tolerance value. In other words, if the current mean is μ, then the next frequency value is a random variable between $\mu - \tau$ and $\mu + \tau$. Whenever the next frequency value falls outside the range of $[\mu-\tau, \mu+\tau]$, then a new sector is generated.

Theorem 3.7 *The average case error with a rectangular ACM is bounded by* 2τ.

Proof: To compute the average case error, we are required to compute the error at every attribute value and then take its expected value by weighting it with the probability of the corresponding error. However, since the frequency of an attribute value can vary only in the range of $[\mu-\tau, \mu+\tau]$, the maximum variation the frequency can have is 2τ. It follows that the maximum error is always bounded by 2τ, and the result follows. □

3.7.2 Average Case Error in Estimating the Sum of Frequencies in an Attribute Value Range

As in the case of the worst-case error estimation, we have to consider three different cases depending on where the attribute value range falls. We only consider the case when the attribute value range completely falls within an R-ACM sector as the other two cases are trivial. This is given by the following theorem.

Theorem 3.8 *The average case error with a rectangular ACM in estimating the sum of frequencies in the value range,* $X_\alpha \leq X \leq X_\beta$, *when both* X_α *and* X_β *fall completely within the* k^{th} *sector of the R-ACM is given by,*

$$\epsilon = 2(\beta - \alpha + 1)\tau$$

where $\beta > \alpha$ *and* τ *is the tolerance value used to build the R-ACM.*

Proof: From Theorem 3.7, the average-case error bound in estimating the frequency of an arbitrary attribute value X_i is 2τ. Hence considering the errors for the attribute values in the range of $X_\alpha \leq X \leq X_\beta$, we find the cumulative error is,

$$\epsilon = \sum_{i=\alpha}^{\beta} 2\tau = 2(\beta - \alpha + 1)\tau$$

and the theorem follows. □

3.7.3 Tight Error Bound with R-ACM: A Special Case

In a random R-ACM sector, almost half of the attribute values tend to have frequency values below the sector mean and the other half above the frequency mean. So it is possible to *re-arrange* the attribute values in such a random sector so that the frequency values alternatively increase and decrease about the current mean. We consider such a random R-ACM sector where the frequency values alternatively increase and decrease by the tolerance τ about the current mean. For example, the current frequency value x_k is derived from the current mean μ_{k-1} by the expression,

$$x_k = \mu_{k-1} + (-1)^{k+1}\tau.$$

The current frequency mean can be given by the following recurrence expression.

$$\mu_k = \mu_{k-1} + \frac{(-1)^{k+1}\tau}{k}.$$

A random sector with frequency values which alternatively increase and decrease about the current mean is depicted in Figure 3.8 and the corresponding frequency and mean values are given in Table 3.2. Considering the case of such a random sector leads us to the following lemma.

Lemma 3.9 *The estimation error in a rectangular ACM with frequency values which*

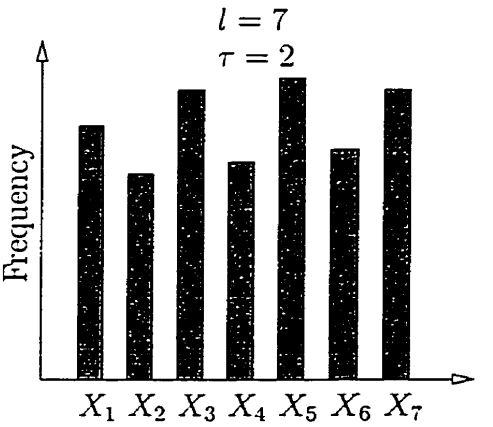


Figure 3.8: A Random R-ACM Sector

k	x_k	μ_k
1	a	a
2	$a-\tau$	$a-\tau/2$
3	$a+\tau/2$	$a-\tau/6$
4	$a-7\tau/6$	$a-5\tau/12$
5	$a+7\tau/12$	$a-13\tau/60$

Table 3.2: First few values of x_k and μ_k

alternatively increase and decrease about the current mean is given by,

$$x_i - E(X_i) = \tau \left| \left(\sum_{k=2}^{i} \frac{(-1)^{k+1}}{k} + \ln 2 - 1 \right) \right|.$$

Proof: The expected value $E(X_i)$ is the mean frequency for the entire sector. So,

$$\begin{aligned} E(X_i) &= \frac{\sum_{k=1}^{l_j} x_k}{l_j} = \mu_{l_j} \\ &= a + \tau \left\{ -\frac{1}{2} + \frac{1}{3} - \ldots + \frac{(-1)^{l_j+1}}{l_j} \right\}. \end{aligned}$$

For sufficiently large l_j, the above becomes,

$$E(X_i) \approx a - (1 - \ln 2)\tau.$$

But the frequency of an arbitrary value X_i is,

$$\begin{aligned} x_k &= \mu_{k-1} + (-1)^{k+1}\tau \\ &= a + \tau\left\{-\frac{1}{2} + \frac{1}{3} - \ldots + \frac{(-1)^{i+1}}{i}\right\} \\ &= a + \tau\sum_{k=3}^{i+1}\frac{(-1)^k}{k-1} \end{aligned}$$

Hence the estimation error is,

$$x_i - E(X_i) = \tau\left|\left(\sum_{k=2}^{i}\frac{(-1)^{k+1}}{k} + \ln 2 - 1\right)\right|.$$

The lemma follows. □

Note that the above estimation error is always smaller than 0.8069τ and thus is better than the average case error.

3.8 Size Estimation of Selection Operations Using the R-ACM

In the previous sections of this chapter, we introduced the Rectangular Attribute Cardinality Map and argued that it could be used as the method for query result size estimations in database systems. We also provided some theoretical analysis to show that their result size estimations are more accurate than the traditional equi-width and equi-depth histograms. Our analysis in the previous sections was mainly in the context of frequency (probability) density estimations of data distributions where we were able to get estimates for the frequencies of attribute values. In this and the

next sections, we discuss the actual application of the R-ACM for query result size estimation in a relational database system. In particular, we discuss how R-ACM can be used for estimating the result sizes of *select* and *join* query operations which are the most predominant operations done in query processing.

3.8.1 Introduction to Selection Operations

The selection operation is one of the most frequently encountered operations in relational queries. In general, the select operation is denoted by $\sigma_p(R)$, where p is a Boolean expression or the selection predicate and R is a relation containing the attributes specified in p. The selection predicate p is specified on the relation attributes and is made up of one or more clauses of the form,

$$< Attribute > \ < comparison\ operator > \ < Constant >,$$
$$\text{or } < Attribute1 > \ < comparison\ operator > \ < Attribute2 > .$$

The comparison operator can be any operator in the set, $\{=, <, \leq, >, \geq, \neq\}$. The selection predicates can be either range predicates or equality predicates depending upon the comparison operators. Furthermore, the selection clauses can be arbitrarily connected by the Boolean operators, AND, OR, and NOT to form a general selection condition. The comparison operators in the set $\{=, <, \leq, >, \geq, \neq\}$ apply to attributes whose domains are ordered values, such as numeric or date domains. Domains of strings of characters are considered ordered based on the collating sequences of the characters used in the system under consideration. If the domain of an attribute is a set of unordered values, only the comparison operators in the set $\{=, \neq\}$ can be applied to that attribute.

In general, the result of a select operation can be determined as follows. The selection predicate, p, is applied independently to each tuple t in the relation R. This is done by substituting each occurrence of an attribute A_i in the selection predicate with its value in the tuple $t[A_i]$. If the condition evaluates to be true, then the tuple t is selected. All the selected tuples appear in the result of the select operation. The

relation resulting from the select operation has the same attributes as the relation specified in the selection operation.

In this work, we assume the selection predicates contain only a single clause. First of all, this is done for ease of analysis. Although this may appear to be of just academic value, extending our work to any arbitrary number of clauses is straight forward. Finding the estimation for the result sizes of various selection predicates requires finding the estimates, $E(\sigma_{X=X_\alpha}(R))$, $E(\sigma_{X<X_\alpha}(R))$, $E(\sigma_{X>X_\alpha}(R))$, $E(\sigma_{X\leq X_\alpha}(R))$, $E(\sigma_{X\geq X_\alpha}(R))$ and $E(\sigma_{X\neq X_\alpha}(R))$. Using the theoretical results we obtained earlier in this chapter, we can find estimation bounds for each of these six conditions. But instead of computing each of these six estimates independently, we shall find only the estimation bounds for the estimates $E(\sigma_{X\leq X_\alpha}(R))$ and $E(\sigma_{X=X_\alpha}(R))$. The estimation bounds for the other four selection conditions can be obtained from the following relationships between the various selection conditions.

3.8.2 Relationships between Selection Conditions

We briefly discuss the various relationships between the selection condition below. Understanding these relationships would enable us to obtain the estimation results for every selection condition from the two estimates, $E(\sigma_{X\leq X_\alpha}(R))$ and $E(\sigma_{X=X_\alpha}(R))$. These relationships are given in the form of lemmas here. Their proofs are omitted in the interest of brevity as they are easy to derive.

Lemma 3.10 *If N is the total number of tuples for the attribute, then for every X the following conditions hold:*

$$E(\sigma_{X<X_\alpha}(R)) + E(\sigma_{X=X_\alpha}(R)) + E(\sigma_{X>X_\alpha}(R)) = N$$
$$E(\sigma_{X\leq X_\alpha}(R)) = E(\sigma_{X<X_\alpha}(R)) + E(\sigma_{X=X_\alpha}(R))$$
$$E(\sigma_{X\geq X_\alpha}(R)) = E(\sigma_{X>X_\alpha}(R)) + E(\sigma_{X=X_\alpha}(R))$$

□

Lemma 3.11 *If X_{min} and X_{max} are the minimum and maximum values for the attribute respectively, then,*

$$\begin{aligned} E(\sigma_{X>X_{max}}(R)) &= 0, \\ E(\sigma_{X<X_{min}}(R)) &= 0. \end{aligned}$$

Lemma 3.12 *For every X_α and X_β such that $X_\alpha < X_\beta$,*

$$E(\sigma_{X<X_\alpha}(R)) \leq E(\sigma_{X<X_\beta}(R)).$$

Lemma 3.13 *For every X_α,*

$$E(\sigma_{X=X_\alpha}(R)) \geq 0.$$

The following corollary gives some further properties, which can be inferred from the above lemmas.

Corollary 3.1 *For any arbitrary attribute value X and a given attribute value X_α, we have,*

$$\begin{aligned} E(\sigma_{X\leq X_{max}}(R)) &= N \\ E(\sigma_{X\geq X_{min}}(R)) &= N \\ E(\sigma_{X<X_\alpha}(R)) &\leq E(\sigma_{X\leq X_\alpha}(R)). \end{aligned}$$

Hence from the above lemmas and corollary, using the estimates $E(\sigma_{X=X_\alpha}(R))$, and $E(\sigma_{X\leq X_\alpha}(R))$, we can obtain the estimates for the other four selection predicates as

follows:

$$\begin{aligned} E(\sigma_{X>X_\alpha}(R)) &= N - E(\sigma_{X\leq X_\alpha}(R)) \\ E(\sigma_{X\geq X_\alpha}(R)) &= N - E(\sigma_{X\leq X_\alpha}(R)) + E(\sigma_{X=X_\alpha}(R)) \\ E(\sigma_{X<X_\alpha}(R)) &= E(\sigma_{X\leq X_\alpha}(R)) - E(\sigma_{X=X_\alpha}(R)) \\ E(\sigma_{X\neq X_\alpha}(R)) &= N - E(\sigma_{X=X_\alpha}(R)). \end{aligned}$$

Earlier in this chapter, we discussed the error estimates for equality match and range match in the context of probability distributions. In the following sections we shall derive the estimates for the sizes of both equality select ($E(\sigma_{X=X_\alpha}(R))$) and range select ($E(\sigma_{X\leq X_\alpha}(R))$) operations using the R-ACM and the T-ACM.

3.8.3 Result Estimation of Equality Select Using the R-ACM

Let us consider a relational query with an equality select predicate, $X = X_\alpha$, whereX_α is a constant value in the domain of attribute X. We want to find the estimate of the result size of the query, $\sigma_{X=X_\alpha}(R)$ where the attribute value X_α is in position α of the k^{th} R-ACM sector. This is given by the following lemma.

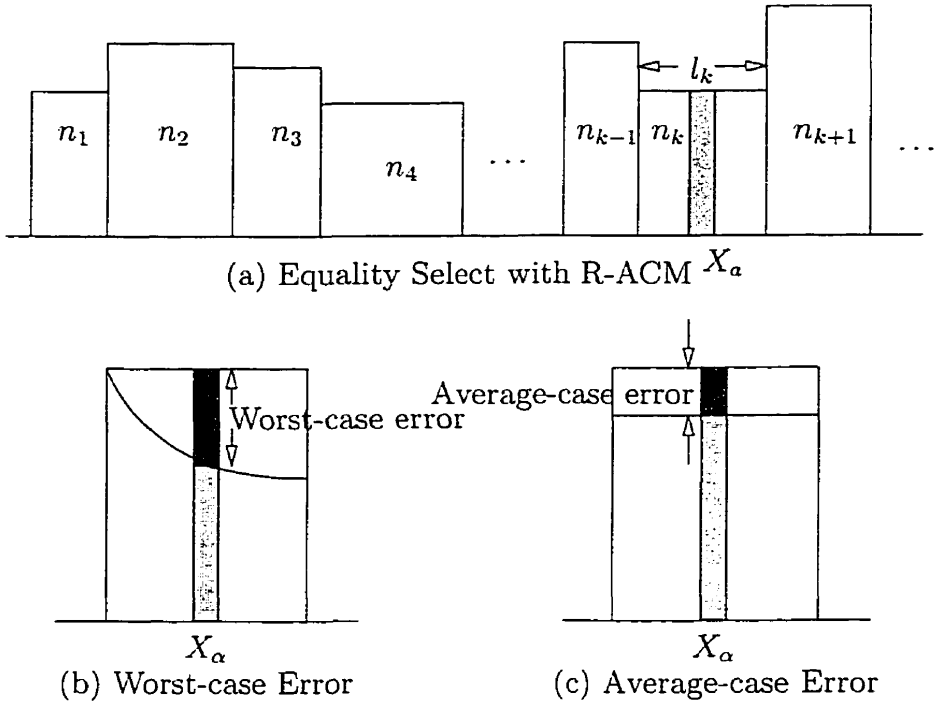
Lemma 3.14 *An estimate obtained by using the expected value analysis for the result of the equality select query, $\sigma_{X=X_\alpha}(R)$, using an R-ACM is,*

$$E(|\sigma_{X=X_\alpha}(R)|) = \frac{n_k}{l_k}$$

where n_k is the number of tuples in the k^{th} R-ACM sector and l_k is the number of distinct attribute values (or width) of the k^{th} sector respectively.

Proof: This result is a direct consequence of Theorem 3.2. □

In future, because of the fact that the estimate using the expected value and the maximum likelihood estimate are essentially identical, we shall refer to the estimate as the maximum likelihood estimate. When estimating the result size of the above



(a) Equality Select with R-ACM

(b) Worst-case Error

(c) Average-case Error

Figure 3.9: Estimation of Equality Select Using the R-ACM

query using the R-ACM, obviously we are also interested in the associated worst-case and average-case errors. These errors are shown in Figure 3.9 (b) and (c).

Lemma 3.15 *If the value X_α falls in the k^{th} sector of an R-ACM, then the worst-case error in estimating the number of tuples of X_α by using its maximum likelihood estimate is,*

$$\epsilon = \tau \left| \ln \left(\frac{l_k}{\alpha - 1} \right) - 1 \right|$$

where n_k and l_k are the number of tuples and the sector width of the k^{th} sector.

Proof: This is a direct result from Theorem 3.4. □

Lemma 3.16 *The average case error with a rectangular ACM in estimating the equality select query, $\sigma_{X=X_\alpha}(R)$ by using its expected value is 2τ, where τ is the tolerance value used to build the R-ACM.*

Proof: This is a direct result of Theorem 3.7. □

3.8.4 Result Estimation of Range Select Using the R-ACM

As mentioned earlier, among the various selection predicates, it is sufficient to find the estimate for the range select query $\sigma_{X \leq X_\alpha}(R)$. With this estimate and the estimate we found above for $\sigma_{X=X_\alpha}(R)$, we can find the estimates for all other range predicates. To obtain this we present the following theorem.

Theorem 3.9 *The maximum likelihood estimate for the range select query $\sigma_{X \leq X_\alpha}(R)$ using the R-ACM is given by,*

$$S_{ML}(|\sigma_{X \leq X_\alpha}(R)|) = \sum_{j=1}^{k-1} n_j + \frac{z_\alpha n_k}{l_k}$$

where the attribute value X_α is in the k^{th} sector of the R-ACM and is the z_α^{th} attribute value in the sector.

Proof: The attribute values which satisfy the query $\sigma_{X \leq X_\alpha}(R)$ occupy the first $k-1$ R-ACM sectors and upto and including the z_α^{th} location of the k^{th} R-ACM sector. Hence the query result size is found by summing up the number of tuples in the first $k-1$ sectors of the R-ACM and the maximum likelihood estimate of the tuples for the first z_α attribute values in the k^{th} sector of the R-ACM. First of all, we stress that there is no estimation error in the first $k-1$ sectors. We find the maximum likelihood estimate of the tuples for the first z_α attribute values in the k^{th} sector as below.

Let us assume that the frequencies of the first z_α attribute values in the k^{th} R-ACM sector are $x_1, x_2, \ldots, x_{z_\alpha}$. We can obtain a likelihood function on these frequencies by considering their join probability mass functions. But instead, without loss of generality, we consider the join probability mass function for the first two attribute

values. Due to their uncorrelatedness, the likelihood function can be written as,

$$\begin{aligned}\mathcal{L}(x_1, x_2) &= \binom{n_k}{x_1} p^{x_1}(1-p)^{n_k-x_1} \binom{n_k}{x_2} p^{x_2}(1-p)^{n_k-x_2} \\ &= \binom{n_k}{x_1}\binom{n_k}{x_2} p^{x_1+x_2}(1-p)^{2n_k-(x_1+x_2)}.\end{aligned}$$

Taking natural logarithm on both sides of the likelihood function, we have,

$$\begin{aligned}\ln \mathcal{L}(x_1, x_2) &= 2\ln n_k! - \ln x_1! - \ln(n_k - x_1)! - \ln x_2! - \ln(n_k - x_2)! + \\ &\quad (x_1 + x_2)\ln p + [2n_k - (x_1 + x_2)]\ln(1-p) \\ &= 2\ln\Gamma(n_k + 1) - \ln\Gamma(x_1 + 1) - \ln\Gamma(n_k - x_1 + 1) + -\ln\Gamma(x_2 + 1) \\ &\quad - \ln\Gamma(n_k - x_2 + 1) + (x_1 + x_2)\ln p + [2n_k - (x_1 + x_2)]\ln(1-p)\end{aligned}$$

Now using the well known identity,

$$\Gamma(\alpha) = \frac{\Gamma(\alpha + k + 1)}{\alpha(\alpha+1)\ldots(\alpha+k)}$$

and simplifying the above log likelihood function we find,

$$\begin{aligned}\ln \mathcal{L}(x_1, x_2) &= -2\ln\Gamma(n_k + 1) + (x_1 + x_2)\ln p + [2n_k - (x_1 + x_2)]\ln(1-p) + \\ &\quad \ln(x_1 + 1) + \ln(x_1 + 2) + \ldots + \ln n + \\ &\quad \ln(n_k - x_1 + 1) + \ln(n_k - x_1 + 2) + \ldots + \ln n \\ &\quad \ln(x_2 + 1) + \ln(x_2 + 2) + \ldots + \ln n + \\ &\quad \ln(n_k - x_2 + 1) + \ln(n_k - x_2 + 2) + \ldots + \ln n\end{aligned}$$

Now taking the partial derivative of $\ln \mathcal{L}(x_1, x_2)$ with respect to x_1, we obtain,

$$\frac{\partial\{\ln \mathcal{L}(x_1, x_2)\}}{\partial x_1} = \ln p - \ln(1-p) + \sum_{r=x_1+1}^{n_k - x_1} \frac{1}{r}.$$

Setting $\frac{\partial\{\ln \mathcal{L}(x_1,x_2)\}}{\partial x_1} = 0$, and noting that $\sum_{r=x_1+1}^{n_k-x_1} \frac{1}{r} \leq \ln\left(\frac{n_k-x_1}{x_1}\right)$, $\hat{x}_{ML}$ of x_1 is obtained as,

$$\frac{p(n_k - x_1)}{(1-p)x_1} \geq 1,$$

or arguing as in Theorem 3.1, we get,

$$\hat{x}_{ML} = n_k p = \frac{n_k}{l_k}.$$

Similarly, finding $\frac{\partial\{\ln \mathcal{L}(x_1,x_2)\}}{\partial x_1}$ and setting it to zero, we find the maximum likelihood estimate for x_2 as $\hat{x}_{ML} = \frac{n_k}{l_k}$. Hence considering the join probability mass function of all other attribute values in this sector, we can show that each of their maximum likelihood estimate is equal to $\hat{x}_{ML} = \frac{n_k}{l_k}$. Thus the maximum likelihood estimate of the number of tuples for the first z_α attribute values is equal to,

$$\sum_{i=1}^{z_\alpha} \frac{n_k}{l_k} = \frac{z_\alpha n_k}{l_k}.$$

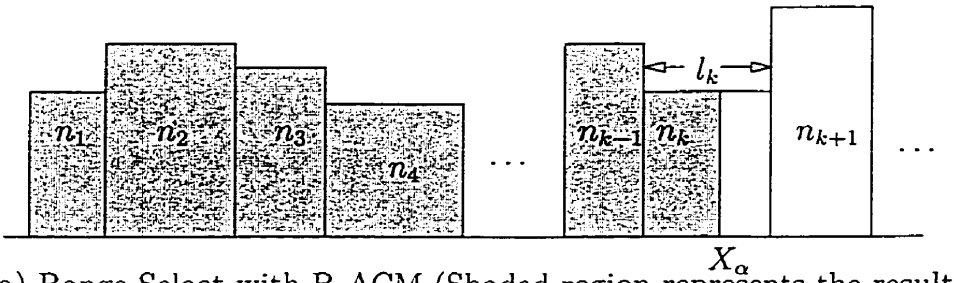
Hence the maximum likelihood estimate for the range select query $\sigma_{X \leq X_\alpha}(R)$ is given by,

$$\mathcal{S}_{ML}(|\sigma_{X \leq X_\alpha}(R)|) = \sum_{j=1}^{k-1} n_j + \frac{z_\alpha n_k}{l_k}.$$

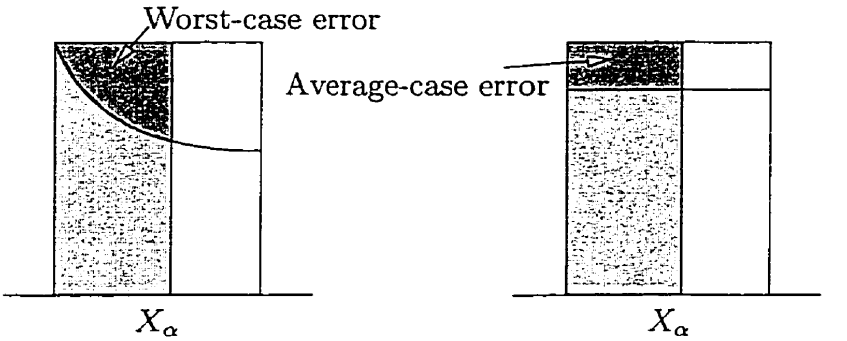
Hence the theorem. □

As mentioned earlier (see Section 3.6.2), we note that when estimating the range select queries, we have to consider three distinct cases. As we argued in that section, it is sufficient to consider the second case only. Let us consider the selection query $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$, where the attribute values X_α and X_β fall within the k^{th} R-ACM sector and $\alpha < \beta$. So we have the following theorem.

Theorem 3.10 *Using the R-ACM, the maximum likelihood estimate of the range*



(a) Range Select with R-ACM (Shaded region represents the result)



(b) Worst-case Error (c) Average-case Error

Figure 3.10: Estimation of Range Select Using the R-ACM

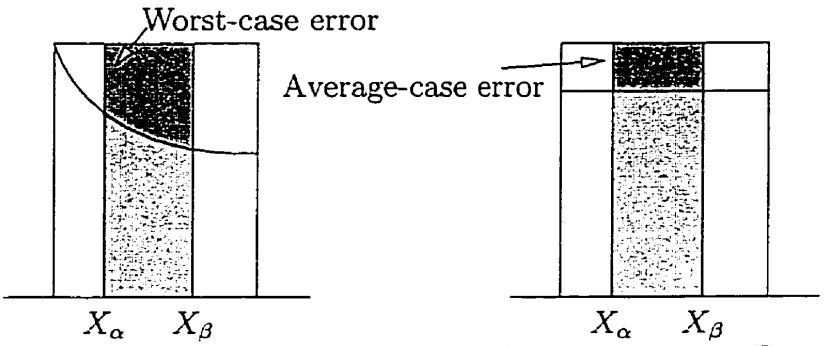
selection query $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$*, where the attribute values* X_α *and* X_β *fall completely within the* k^{th} *R-ACM sector is given by,*

$$S_{ML}(|\sigma_{X_\alpha \leq X \leq X_\beta}(R)|) = \frac{(\beta - \alpha + 1)n_k}{l_k}$$

where $\beta > \alpha$.

Proof: Using Theorem 3.9, we obtain,

$$\begin{aligned} E_{ML}(|\sigma_{X_\alpha \leq X \leq X_\beta}(R)|) &= E_{ML}(\sigma_{|X \leq X_\beta}(R)|) - E_{ML}(|\sigma_{X < X_\alpha}(R)|) \\ &= \sum_{j=1}^{k-1} n_j + \frac{\beta n_k}{l_k} - \sum_{j=1}^{k-1} n_j - \frac{(\alpha - 1)n_k}{l_k} \\ &= \frac{(\beta - \alpha + 1)n_k}{l_k}. \end{aligned}$$



(a) Worst-case Error (b) Average-case Error

Figure 3.11: Estimation of a Range Completely within an R-ACM Sector

The theorem follows. □

In Theorem 3.15, considering a monotonically decreasing R-ACM sector, we found the worst-case error bound for an arbitrary attribute value as,

$$\epsilon = \left| \tau \left[\ln \left(\frac{l_k}{\alpha - 1} \right) - 1 \right] \right|.$$

The following lemma gives a bound for the worst-case error (shown in Figure 3.11 (a)) in the above select operation.

Lemma 3.17 *The worst-case error, ϵ, in estimating the query $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$ by using the maximum likelihood estimate, when both X_α and X_β fall completely within the k^{th} sector of the R-ACM is given by,*

$$\epsilon = \left| \ln \left\{ l_j^{(\beta-\alpha+1)\tau} \frac{(\beta-1)!}{(\alpha-2)!} \right\} \right| - (\beta - \alpha + 1)\tau$$

where $\beta > \alpha$.

Proof: This follows directly from Theorem 3.6. □

The following lemma gives a bound for the average-case error (shown in Figure 3.11 (b)) in the above select operation.

Lemma 3.18 *The average case error with a rectangular ACM in estimating the range select query, $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$ by using its maximum likelihood estimate, when both X_α and X_β fall completely within the k^{th} sector of the R-ACM is given by,*

$$\epsilon = 2(\beta - \alpha + 1)\tau$$

where $\beta > \alpha$ and τ is the tolerance value used to build the R-ACM.

Proof: This is a direct result of Theorem 3.8. □

3.9 Size Estimation of the Join Operation Using the R-ACM

The join operation, denoted by the symbol ⋈, is used to combine related tuples from two relations into single tuples. This operation is very important for any relational database system with more than a single relation, because it allows us to process relationships among relations. Also joins are among the most expensive operators in relational DBMSs. In fact, a large part of query optimization consists of determining the optimal ordering of join operators. Since estimating the result size of a join is very essential to finding a good QEP, we shall discuss here the result size estimation of joins using the R-ACM.

3.9.1 Introduction to Processing Joins

The general form of a join operation with an arbitrary predicate on two relations $R(A_1, \ldots, A_n)$ and $S(B_1, \ldots, B_n)$ is,

$$R \bowtie_{predicate} S$$

where *predicate* is the join condition. A join condition is of the form,

$$< condition > \text{ AND } < condition > \text{ AND } \ldots \text{ AND } < condition >$$

where each condition is of the form $A_i \theta B_j$. A_i and B_j have the same domain and θ is one of the comparison operators $\{=, <, \leq, >, \geq, \neq\}$. A join operation with such a general join condition is called a *theta join*. The most common join involves join conditions with equality comparisons only. Such a join, where the only comparison operator used is $=$, is called an *equijoin*. We consider only the equijoin in this work, and propose to consider the other theta join operations in future research endeavors.

As mentioned earlier in this chapter, the approximate data distribution corresponding to an ACM can be used in the place of any actual distribution to estimate a required quantity. Specifically, the approximate data distributions can be derived using the techniques developed earlier for all joining relations. When the value domain is approximated in a discrete fashion (uniform spread assumption or storing the entire set of values) the result size can be estimated by joining these data distributions using, say, a merge-join algorithm. Essentially, for each distinct value in the approximate value domains of the ACMs, its frequencies in all the ACMs are multiplied and the products are added to give the join result size. It is clear that, in addition to approximating the frequency domain with high accuracy, an accurate ACM for this case should also approximate the value domain with high accuracy in order to correctly identify the joining values[5].

3.9.2 Assumption of Attribute Value Independence

Several queries in practice contain multiple attributes from the same relation. The result sizes of such queries depend on the joint data distributions of those attributes. Due to the large number of such attribute value combinations, determining the probabilistic attribute dependencies is very difficult. In fact, most commercial DBMSs adopt the *attribute value independence assumption*[6] [122]. Under this assumption,

[5]The case when the join operation involves multiple attribute predicates requires the multi-dimensional version of the R-ACM, which is a promising area for future research.

[6]Unfortunately, real-life data rarely satisfies the attribute value independence assumption. For instance, functional dependencies represent the exact opposite of the assumption. For example, it is natural for the `salary` attribute of the `Employee` relation to be 'strongly' dependent on the age attribute (i.e: higher/lower salaries mostly related to older/younger people). Making the attribute value independence assumption in these cases may result in very inaccurate approximations of joint

the data distributions of individual attributes in a relation are statistically[7] *independent* of each other.

In this section we assume that the join attributes are independent of each other. Later, we will consider attribute value dependencies between the join attributes and obtain a more realistic result.

The following lemma provides an estimation of the result for joining two sectors of the R-ACMs of domain compatible attributes X and Y.

Lemma 3.19 *We consider the equality join of two domain compatible attributes X and Y. If the i^{th} sector from the R-ACM of attribute X has α matching attribute values with the j^{th} sector from the R-ACM of attribute Y, the expected number of tuples resulting from joining the* **two sectors** *is,*

$$\hat{\xi} = \frac{\alpha n_{i_X} n_{j_Y}}{l_{i_X} l_{j_Y}}$$

where n_{i_X} and n_{i_Y} are the number of tuples in the i^{th} sector of attribute X and j^{th} sector of attribute Y respectively.

Proof: From Lemma 3.2 we know, that the expected frequency of a given attribute value in the i^{th} R-ACM sector is given by,

$$E(x) = \frac{n_{i_X}}{l_i}$$

where n_i is the number of tuples in the i^{th} R-ACM sector and l_i is the sector width of the i^{th} R-ACM sector. Hence we find that, if α matching values (which need not be contiguous) are in the sectors, each of them will yield the same expected value. Thus the total join size is,

$$\hat{\xi} = E\left(\sum_{k=1}^{\alpha} x_k \times y_k\right)$$

data distributions and therefore inaccurate query result size estimations [22].

[7]The reader must differentiate here between "independence" as used in the normal form definition of the relations and the concept of statistical independence.

where x_k and y_k are the frequencies of the attribute values X_k and Y_k. We assume that the attributes X and Y are independent of each other[8] and using the expressions for $E(x_k)$ and $E(y_k)$, the above becomes,

$$\hat{\xi} = \sum_{k=1}^{\alpha} \{E(x_k) \times E(y_k)\}$$

whence,

$$\begin{aligned} \hat{\xi} &= \sum_{k=1}^{\alpha} \left(\frac{n_{i_X}}{l_{i_X}}\right)\left(\frac{n_{j_Y}}{l_{j_Y}}\right) \\ &= \frac{\alpha n_{i_X} n_{j_Y}}{l_{i_X} l_{j_Y}}. \end{aligned}$$

Since the term inside the summation above is the same for **all** the α attribute values, the lemma follows. □

Corollary 3.2 *For the special case, when the sectors of the join attributes have equal sector widths, $l_{i_X} = l_{j_Y} = l = \alpha$, then the estimated size of the sector join is,*

$$\hat{\xi} = \frac{n_{i_X} n_{j_Y}}{l}$$

3.9.3 Estimated Result Sizes

Consider an equijoin $R \bowtie_{X=Y} S$ where attributes X and Y are from relations R and S respectively. Assume that the R-ACMs for both attributes X and Y are available. We consider a more general case, where a sector from the R-ACM for attribute X has matching values with more than one sector from the R-ACM for attribute Y. We want to estimate the size of the join $R \bowtie_{X=Y} S$.

We note that the attribute values in the R-ACMs are in an ascending order. Let us consider the i^{th} sector from the R-ACM for the attribute X. Suppose it has matching values in the R-ACM for attribute Y with all the sectors in the range from sector j to sector k. Assume that there are α matching values from the j^{th} sector and β

[8] We don't need independence; uncorrelatedness is sufficient.

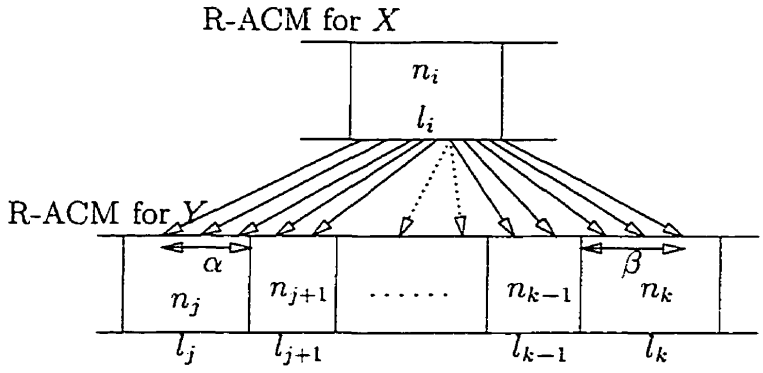


Figure 3.12: Estimating Result Size of $R \bowtie_{X=Y} S$

matching values from the k^{th} sector. We also assume that all the values in sectors $j+1$ to $k-1$ of Y have matching values from sector i of X. See Figure 3.12.

Hence the estimated join size resulting from joining sector i of X to the matching sectors of Y,

$$\begin{aligned}\hat{\xi}_i &= \frac{n_i}{l_i}\frac{\alpha n_j}{l_j} + \frac{n_i}{l_i}n_{j+1} + \ldots + \frac{n_i}{l_i}n_{k-1} + \frac{n_i}{l_i}\frac{\beta n_k}{l_k} \\ &= \frac{n_i}{l_i}\left(\frac{\alpha n_j}{l_j} + n_{j+1} + \ldots + n_{k-1} + \frac{\beta n_k}{l_k}\right)\end{aligned}$$

Hence the estimated join size of $R \bowtie_{X=Y} R$ is the summation of all such $\hat{\xi}_i$, i.e: $\hat{\xi}_i = \sum_{i=1}^{s_x} \hat{\xi}$ where s_x is the number of sectors in attribute X.

Similarly, in order to estimate the size of the entire join of relations R and S, we make use of Lemma 3.19 for every intersecting sector of the R-ACM of attribute X and the R-ACM of attribute Y. This is given by the following theorem.

Theorem 3.11 *Considering an equijoin $R \bowtie_{X=Y} S$ of two relations R and S, with domain compatible attributes X and Y, the maximum likelihood estimate for the size of the join using an R-ACM is,*

$$\hat{\xi} = \sum_{i=1}^{s_X}\sum_{j=1}^{\gamma_i} \alpha_{ij}\frac{n_{i_X} n_{j_Y}}{l_{i_X} l_{j_Y}}$$

where s_X is the number of sectors in the R-ACM of attribute X and α_{ij} is the number

of intersecting values of the i^{th} sector of the R-ACM of X and the j^{th} sector of the R-ACM of Y. γ_i is the number of sectors from the R-ACM of Y which intersect the i^{th} R-ACM sector of X.

Proof: The proof follows directly from Lemma 3.19. □

We note that the estimation of join sizes using an ACM is similar to the merge phase of the sort-merge join algorithm. Since the attribute values in the ACM are already in a sequentially ascending order, unlike the sort-merge join algorithm, the sorting step is not required for an ACM. When estimating the join size using two ACMs, if the number of distinct values in the smaller ACM is L, then the join estimation only requires L number of integer multiplications and an equal number of additions.

3.9.4 Estimation of Join Error

The estimation error resulting from an equality join of two attributes is usually much higher than the estimation errors resulting from the equality select and range select operations.

Lemma 3.20 *Considering the equality join of two domain compatible attributes X and Y with $X_i = Y_j$, if the expected result size of the equality selection query, $\sigma_{X=X_i}$, using an ACM is $\hat{x}_i$ and that of $\sigma_{Y=Y_j}$ is $\hat{y}_j$, then the maximum error resulting from joining the attributes X and Y on the values X_i and Y_j is given by,*

$$\epsilon = |(\hat{x}_i\epsilon_y + \hat{y}_j\epsilon_x + \epsilon_x\epsilon_y)|$$

where ϵ_x and ϵ_y are the estimated errors resulting from the equality selection queries $\sigma_{X=X_i}$ and $\sigma_{X=Y_j}$ respectively.

Proof: Assume that the actual frequency values of X_i and Y_j are x_i and y_j respectively. Hence the actual size of the join contribution from these values is,

$$\xi = x_i y_j.$$

But the expected size of the join contribution from the above values is,

$$\hat{\xi} = \hat{x}_i \hat{y}_j.$$

Thus the maximum error resulting from joining the values $X = X_i$ and $Y = Y_j$ is,

$$\begin{aligned} \epsilon &= |\xi - \hat{\xi}| \\ &= |x_i y_j - \hat{x}_i \hat{y}_j| \end{aligned}$$

The possible values for x_i can be either $(\hat{x}_i - \epsilon_x)$ or $(\hat{x}_i + \epsilon_x)$. Similarly the possible values for y_j can be either $(\hat{y}_j - \epsilon_y)$ or $(\hat{y}_j + \epsilon_y)$. We note that out of the 4 possible value combinations of these expected values, only $(\hat{x}_i + \epsilon_x)(\hat{y}_j + \epsilon_y)$ gives the largest error. Hence the maximum error becomes,

$$\begin{aligned} \epsilon &= |\hat{x}_i \hat{y}_j - (\hat{x}_i + \epsilon_x)(\hat{y}_j + \epsilon_y)| \\ &= |\hat{x}_i \epsilon_y + \hat{y}_j \epsilon_x + \epsilon_x \epsilon_y|. \end{aligned}$$

The lemma follows. □

Considering all the values of attributes X and Y, it is possible to find the cumulative error in the estimation of a join. Hence using the results on estimation errors we obtained earlier, we can find the join errors for both the worst-case and average-case situations in R-ACM and T-ACM.

Corollary 3.3 *The error resulting from an equality join of two domain compatible attributes X and Y, is given by,*

$$\epsilon = \sum_{j=1}^{s_X} \sum_{i=1}^{l_j} (\hat{x}_i \epsilon_{y_k} + \hat{y}_k \epsilon_{x_i} + \epsilon_{x_i} \epsilon_{y_k})$$

where k is an index on the R-ACM of Y such that $X_i = Y_k$ and $\epsilon_{x_i}, \epsilon_{y_k}$ are the errors resulting from the equality selection queries $\sigma_{X=X_i}$ and $\sigma_{Y=Y_k}$ respectively.

Proof: The proof follows from the previous lemma. □

Lemma 3.21 *Considering the equality join of two domain compatible attributes X and Y with $X_i = Y_j$, if the expected result size of the equality selection query, $\sigma_{X=X_i}$, using an R-ACM is $\hat{x}_i$ and that of $\sigma_{Y=Y_j}$ is $\hat{y}_j$, then the worst-case error resulting from joining the attributes X and Y on the values X_i and Y_j is given by,*

$$\epsilon = \ln\left(\left[\frac{l_q}{e(j-1)}\right]^{\tau_y \hat{x}_i}\left[\frac{l_p}{e(i-1)}\right]^{\tau_x \hat{y}_j}\right) + \ln\left[\frac{l_q}{e(j-1)}\right]^{\tau_y} \ln\left[\frac{l_p}{e(i-1)}\right]^{\tau_x}$$

where τ_x and τ_y are the tolerance values used to generate the R-ACMs of attributes X and Y respectively and l_p and l_q are the sector widths of the R-ACMs corresponding to the attribute values X_i and Y_j respectively.

Proof: It follows from Lemma 3.20 that the maximum join error when joining the values X_i and Y_j is equal to,

$$\epsilon = |\hat{x}_i \epsilon_y + \hat{y}_j \epsilon_x + \epsilon_x \epsilon_y|.$$

But from Theorem 3.5, the worst-case error in estimating the selection query, $\sigma_{X=X_i}$ is equal to,

$$\epsilon_x = \left|\tau_x\left[\ln\left(\frac{l_p}{i-1}\right) - 1\right]\right|, \text{ for } i > 1.$$

Similarly, the worst-case error in estimating the selection query, $\sigma_{Y=Y_j}$ is equal to,

$$\epsilon_y = \left|\tau_y\left[\ln\left(\frac{l_q}{j-1}\right) - 1\right]\right|, \text{ for } j > 1.$$

Hence we find that the worst-case error in joining the attribute values X_i and Y_j can be written as,

$$\begin{aligned}\epsilon &= |\hat{x}_i \epsilon_y + \hat{y}_j \epsilon_x + \epsilon_x \epsilon_y| \\ &= \ln\left(\left[\frac{l_q}{e(j-1)}\right]^{\tau_y \hat{x}_i}\left[\frac{l_p}{e(i-1)}\right]^{\tau_x \hat{y}_j}\right) + \ln\left[\frac{l_q}{e(j-1)}\right]^{\tau_y} \ln\left[\frac{l_p}{e(i-1)}\right]^{\tau_x}\end{aligned}$$

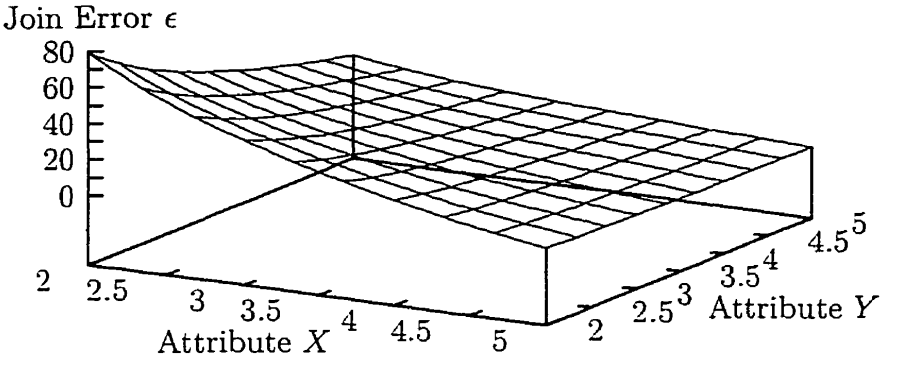


Figure 3.13: Join Estimation Error and the Positions of Attribute Values

and the lemma follows. □

Since the worst-case error in estimating the selection queries $\sigma_{X=X_i}$ and $\sigma_{Y=Y_j}$ is dependent on the positions of the attribute values X_i and Y_j within the corresponding R-ACM sectors, we note that the worst-case error in the above join is also dependent on the positions of the attribute values being joined. Figure 3.13 shows the relationship of the worst-case join estimation error and the positions i, j of the attribute values X_i and Y_j within the R-ACM sectors. Note that the join estimation has the lowest worst-case error when both X_i and Y_j are the last attribute values in their corresponding sectors.

Having formulated the properties of the R-ACM, we are now in a position to test the analytical results using simulations. To achieve this we shall briefly describe the data distributions that we will be using in our experiments throughout this work.

3.10 Data Distributions for the Experiments

The synthetic data from a random number generator invariably produces a uniform distribution. Since real-world data is hardly uniformly distributed, any simulation

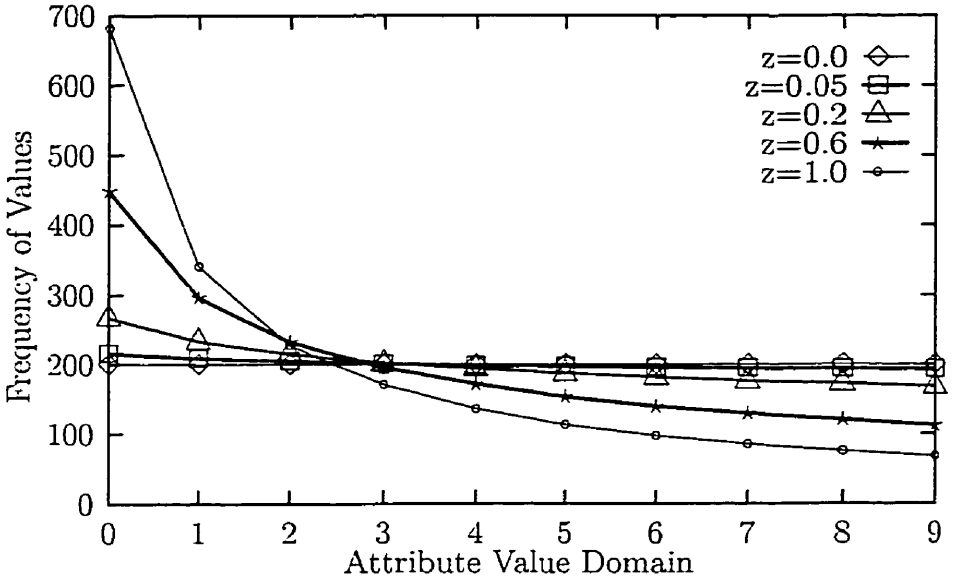


Figure 3.14: Zipf Distributions for Various z Parameters

results from using such synthetic data will, obviously, be of limited use. Hence we have resorted to the use of two powerful mathematical distributions, namely the *Zipf distribution* and the *multi-fractal distribution*. Since these distributions can generate frequencies with wide range of skews, they provide us with more realistic simulation results. In the interest of continuity we provide below a brief overview of these two distributions.

3.10.1 Overview of Zipf Distributions

Nature is full of phenomena which seem to obey a few laws. Some, such as falling apples, can be explained satisfactorily based on certain laws of physics or mechanics. On the other hand, there are some events, such as those occurring in chaotic systems, which do not exhibit any apparent regularity. There is another type of phenomena which exhibit empirically observable regularities, but do not directly yield to explanation using simple laws of nature. These empirical phenomena have been observed in domains as diverse as population distributions, frequency distributions of words, income distributions and biological genera and species.

G.K. Zipf first proposed a law, called the Zipf's law, which he observed to be approximately obeyed in many of these domains [143]. Zipf's law is essentially an algebraically decaying function describing the probability distribution of the empirical regularity. Zipf's law can be mathematically described in the context of our problem as follows.

For an attribute value X of size N with L distinct values, the frequencies generated by the Zipf distribution are given by,

$$f_i = N.\frac{1/i^z}{\sum_{i=1}^{L} 1/i^z}, \text{ for } 1 \leq i \leq L.$$

The skew of the Zipf distribution is a monotonically increasing function of the z parameter, starting from $z = 0$, which is the uniform distribution. We have plotted the frequency sets of several Zipf distributions with different z values in Figure 3.14. These frequency distributions were all generated for $N = 2000$ and $L = 10$.

One of the common claims in database literature is that many attributes in real-world databases contain a few attribute values with high frequencies and the rest with low frequencies [22], and hence can be modeled satisfactorily by Zipf distributions. Statistical literature abounds with information on modeling real-life data by Zipf distributions. Due to this reason, in Chapters 5 and 6 we shall resort to using Zipf distribution to generate frequencies for the value domains in the U.S. CENSUS, NBA performance statistics and TPC-D benchmark databases to validate the results presented in this chapter and the following.

3.10.2 Overview of Multi-fractal Distributions

The relationship of multi-fractals with the "80-20 law" is very close, and seem to appear often. Several real-world distributions follow a rule reminiscent of the 80-20 rule in databases. For example, photon distributions in physics, or commodities (such as gold, water, etc) distributions on earth etc., follow a rule like "*the first half of the region contains a fraction p of the gold, and so on, recursively, for each sub-region.*" Similarly, financial data and people's income distributions follow similar patters.

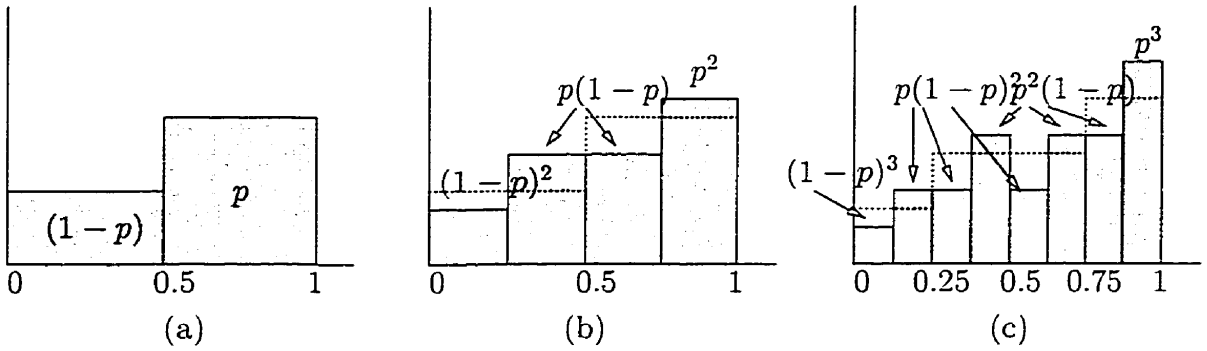


Figure 3.15: Generation of a Multi-fractal Distribution - First three steps

With the above rule, we assume that the attribute value domain is recursively decomposed at k levels; each decomposition halves the input interval into two. Thus, eventually we have 2^k sub-intervals of length 2^{-k}.

We consider the following distribution of probabilities, as illustrated in Figure 3.15. At the first level, the left half is chosen with probability $(1-p)$, while the right half is with p; the process continues recursively for k levels. Thus, the left half of the sectors will host $(1-p)$ of the probability mass, the left-most quarter will host $(1-p)^2$ etc.

For our experiments we will use a *binomial multi-fractal* distribution with N tuples and parameters p and k, with 2^k possible attribute values. Note that when $p = 0.5$, we have the uniform distribution. For a binomial multi-fractal, we have

Count	Relative Frequency
C_0^k	p^k
C_1^k	$p^{(k-1)}(1-p)^1$
...	...
C_a^k	$p^{(k-a)}(1-p)^a$
...	...

In other words, out of the 2^k distinct attribute values, there are C_a^k distinct attribute values, each of which will occur $N * p^{(k-a)}(1-p)^a$ times. Thus, for example, out of the 2^k distinct attribute values, there is $C_0^k = 1$ attribute value that occurs p^k times.

3.11 Experiments Using Synthetic Data

Having derived the analytic properties of the Rectangular ACM in the previous sections, we shall now present a number of experimental results on the R-ACM using *synthetic data*. A more extensive set of benchmarking experiments on *real-world data* is described in Chapter 5. In order to demonstrate the validity of our theoretical results on the R-ACM, we conducted three distinct set of experiments as follows.

In the first set of experiments, we found the actual estimation errors of both selection and join operations using the R-ACM on various synthetic data distributions such as

1. uniform data distribution,
2. Zipf data distribution, and
3. multi-fractal data distribution.

Using these experiments, we verified the performance of the R-ACM, in terms of the result size estimation accuracy, under various frequency skews.

In the second set of experiments, we studied the performance of the R-ACM under various tolerance values, τ. Using these experiments, we also investigated the relationship between the variance of the R-ACM and the query result size estimation errors.

Finally, in the third set of experiments, we compared the performance of the R-ACM to that of the traditional equi-width and equi-depth histograms under various synthetic data distributions for both select and join operations.

It should be noted that even though our experiments are conducted under various frequency distributions, the value domains of all the attributes involved are constructed using a uniform spread[9].

[9] In a uniform spread, the difference between the consecutive attribute values is equal. This does not imply that the distribution in the attribute *domain* is uniform.

3.11.1 Queries Used in the Experiments

The *select* and *join* operations are the two most frequently used relational operations in database systems. Thus for our experiments we will use queries that use these two operations.

For estimating the result sizes of select operations, we will actually use two types of select operations, namely the *exact-match select* and the *range select*. The exact-match select operation, denoted $\sigma_{X=X_i}(R)$, retrieves all the tuples from the relation R, for which the attribute X has the value X_i. The range select operation retrieves all the tuples falling within an attribute value range. For example, the query $\sigma_{X \leq X_i}(R)$, retrieves all the tuples from the relation R, for which the attribute value X has values less than X_i. For the join operation, we will use the most frequently encountered equi-join operation. The equi-join operation, denoted $R \bowtie_{X=Y} S$, combines all the tuples in the relations R and S whenever the value of attribute X from relation R is equal to the value of attribute Y from relation S.

3.11.2 Estimation Accuracy of the R-ACM under Various Synthetic Data Distributions

As we discussed in Section 3.2, whenever there is a steep frequency change in a data distribution, there will be a proportional increase in the number of sectors in the corresponding portions of the R-ACM. In other words, generally speaking, $\tau \propto \frac{1}{\mathcal{S}}$, where $\mathcal{S}$ is the storage requirement for the given tolerance value, τ. Hence for an R-ACM to map a data distribution with a large number of frequency skews, the scheme would require proportionally large storage requirements to maintain the desired estimation accuracy. Thus, a fair comparison study of the performance of the R-ACM under various data distribution should be based on the storage requirements for the R-ACM as opposed to the tolerance values. Consequently, we have used a fixed size for R-ACMs in this group of experiments. This is done by increasing or decreasing the tolerance values as required for each of the data distributions under study.

Operation	Actual Size	Estimated Size	Percentage Error
Equi-select	96	99.2	3.33%
Range-select	3048	3090.8	1.40%
Equi-join	253124	277727.6	9.72%

Table 3.3: Estimation Accuracy of the R-ACM under Uniform Distribution

Operation	Actual Size	Estimated Size	Percentage Error
Equi-select	326	354.4	8.70%
Range-select	1451	1535.2	5.81%
Equi-join	263688	320460	21.53%

Table 3.4: Estimation Accuracy of the R-ACM under Zipf Distribution

In this set of experiments, we computed the relative estimation errors for the selection and join operations under the above mentioned data distributions. The relative estimation error is obtained as a ratio by subtracting the estimated size from the actual result size and dividing it by the actual result size. Obviously, the cases where the actual result sizes were zero were not considered for error estimation. The results were obtained by averaging the estimation errors over a number of experiments and are shown in Tables 3.3, 3.4, and 3.5 for the three different frequency distributions. A uniform attribute value domain was consistently used for this group of experiments. Although the superiority of the R-ACM is clear, this will be further discussed in the

Operation	Actual Size	Estimated Size	Percentage Error
Equi-select	131	138.7	5.91%
Range-select	2058	2134.6	3.72%
Equi-join	600632	689525.5	14.80%

Table 3.5: Estimation Accuracy of the R-ACM under Multifractal Distribution

analysis of the results in Section 3.11.5.

3.11.3 Estimation Accuracy of the R-ACM and τ

In this set of experiments, we compared the result size estimates from the R-ACM for three different tolerance values. Again the experiments were conducted with the uniform, Zipf and multifractal frequency distributions. As in the first group of experiments, the comparisons were done for simple (a) equality-select queries (b) range-select queries and (c) equi-join queries. The percentage estimation error corresponding to each tolerance value was computed as an average over a number of experiments and are shown in Tables 3.6, 3.7 and 3.8.

Operation	Actual Size	Estimated Result			Percentage Error		
		$\tau = 4$	$\tau = 6$	$\tau = 8$	$\tau = 4$	$\tau = 6$	$\tau = 8$
Equi-select	83	84.8	86.84	91.58	2.1%	4.63%	10.34%
Range-select	1378	1390.7	1421.6	1480.1	0.92%	3.17%	7.41%
Equi-join	68938	72453.8	78520	85179.6	5.1%	13.9%	23.56%

Table 3.6: Result Estimation Using R-ACM: Uniform Frequency Distribution

To present the results related to the variances, we also computed these variances of the R-ACM corresponding to each tolerance value under the uniform frequency distribution for equality-select operations in the above set of experiments. The percentage estimation errors and the corresponding variance of the R-ACM are given in Table 3.9. Observe that the percentage estimation error corresponding to the variance $\mathcal{V} = 1328$ is only 2.1%, but the percentage estimation error corresponding to the

Operation	Actual Size	Estimated Result			Percentage Error		
		$\tau = 4$	$\tau = 6$	$\tau = 8$	$\tau = 4$	$\tau = 6$	$\tau = 8$
Equi-select	396	427.36	469.97	504.27	7.92%	18.68%	27.34%
Range-select	2219	2278.9	2343.7	2449.1	2.70%	5.62%	10.37%
Equi-join	276328	304237	335020	369616	10.01%	21.24%	33.76%

Table 3.7: Result Estimation Using R-ACM: Zipf Frequency Distribution

Operation	Actual Size	Estimated Result			Percentage Error		
		$\tau = 4$	$\tau = 6$	$\tau = 8$	$\tau = 4$	$\tau = 6$	$\tau = 8$
Equi-select	218	223.19	229.75	243.46	2.38%	5.39%	11.68%
Range-select	3195	3233.0	3322.2	3450.9	1.19%	3.98%	8.01%
Equi-join	517566	548775	595097	661087	6.03%	14.98%	27.73%

Table 3.8: Result Estimation Using R-ACM: Multifractal Frequency Distribution

variance $\mathcal{V} = 1537$ is more than 10%.

Actual Size	Estimated Result			Percentage Error		
	$\mathcal{V} = 1328$	$\mathcal{V} = 1461$	$\mathcal{V} = 1537$	$\mathcal{V} = 1328$	$\mathcal{V} = 1461$	$\mathcal{V} = 1537$
83	84.8	86.84	91.58	2.1%	4.63%	10.34%

Table 3.9: Variance of the R-ACM and the Estimation Errors

3.11.4 R-ACM and the Traditional Histograms

This final set of experiments were conducted on both the traditional equi-width and equi-depth histograms and the R-ACM. In order to provide a fair comparison, we used a fixed amount of storage for all three techniques, thus varying the build parameters for the structures as required. The build parameters for the equi-width and the equi-depth histograms are the sector-width and the number of tuples per sector

Operation	Actual Size	Equi-width		Equi-depth		R-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	1632	2067.7	26.7%	2022.0	23.9%	1684.4	3.21%
Range-select	28567	30832.4	7.93%	29741.1	4.11%	28924.1	1.25%
Equi-join	698436	862568	23.5%	811583	16.2%	758571	8.61%

Table 3.10: Comparison of Equi-width, Equi-depth and R-ACM: Uniform Frequency Distribution

Operation	Actual Size	Equi-width		Equi-depth		R-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	368	593.9	61.4%	563.8	53.2%	401.9	9.23%
Range-select	1982	2538.9	28.1%	2414.1	21.8%	2093.9	5.65%
Equi-join	185368	314198	69.5%	302149	63.0%	223164	20.39%

Table 3.11: Comparison of Equi-width, Equi-depth and R-ACM: Zipf Frequency Distribution

Operation	Actual Size	Equi-width		Equi-depth		R-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	618	808.3	30.8%	781.1	26.4%	644.14	4.23%
Range-select	29076	31721.9	9.1%	30995.0	6.6%	29811.6	2.53%
Equi-join	691444	979084	41.6%	921695	33.3%	763078	10.36%

Table 3.12: Comparison of Equi-width, Equi-depth and R-ACM: Multifractal Frequency Distribution

respectively. The build parameter for the R-ACM is the tolerance value, τ. As before we computed the percentage estimation errors for the three type of queries, namely, (a) equality-select (b) range-select and (c) equi-join. The experiments were again conducted for the uniform, Zipf and multifractal frequency distributions. An analysis of the results follows.

3.11.5 Analysis of the Results

The results from the first set of experiments show that the estimation error with the uniform data distribution is the smallest and that with the Zipf data distribution is the largest. This is consequent to the fact that a much smaller tolerance value was used for the uniform data distribution than that for the Zipf data distribution, thus resulting in higher estimation accuracy.

The results from the second set of experiments show that the estimation accuracy is inversely proportional to the tolerance value, τ. Even with a highly skewed Zipf

distribution, a tolerance value of $\tau = 4$ provides a relatively small percentage estimation error of 7.92% for equality-select operations. As in the first group of experiments, it is evident that the estimation errors are the smallest with the uniform frequency distribution and are the largest with the highly skewed Zipf frequency distribution. In addition, the experimental results, summarized in Table 3.9, demonstrate that the variance of the R-ACM is inversely related to the estimation error. For example, the percentage estimation error corresponding to the variance $\mathcal{V} = 1328$ is only 2.1%, but the percentage estimation error corresponding to the variance $\mathcal{V} = 1537$ is more than 10%.

The third group of experiments demonstrate the superiority of the R-ACM over the traditional equi-width and the equi-depth histograms for query result size estimation. The result from this set of experiments show that the estimation error resulting from the R-ACM is a fraction of the estimation error from the equi-width and the equi-depth histograms. For example, from Table 3.10, the percentage estimation error with the R-ACM for equi-select operation on uniform frequency distribution is only 3.21%, whereas for the same operation, the equi-width and equi-depth histograms result in 26.7% and 23.9% estimation errors respectively - which is an order of magnitude larger! This disparity is more evident and striking for the highly skewed Zipf distribution. As can be seen from Table 3.11, the percentage estimation error for the equi-select operation using the R-ACM is only 9.23%, whereas for the same operation, the equi-width and equi-depth histograms result in an unacceptably high 61.4% and 53.2% respectively. The power of the R-ACM is obvious!

3.12 Summary of Work Done

In this chapter we have introduced a new histogram-like approximation strategy, called the Rectangular Attribute Cardinality Map, for query result size estimation. Since this technique is based on a user-specified tolerance value for partitioning the sectors, its worst-case and average-case errors are assured to be within a desired bound, and thus it is more accurate than the traditional histograms. By proving a Binomial distribution to represent frequency variations within sectors, we have

provided theoretical results to compare their accuracy to that of the traditional histograms, both in the average-case and worst-case. Our argument that the R-ACM can be used as a fundamental tool for query result-size estimation is fully supported by a formal maximum likelihood analysis, an expected-case analysis of the variance, and the resulting worst-case and average-case errors.

In addition, we have also conducted a number of experiments using synthetic data to support the validity of our theoretical results. Since we resorted to computer simulations to validate our results, we also presented a brief overview of the two most popular distributions, namely, Zipf and multi-fractal distributions, which were used for generating frequencies in the above experiments.

The theoretical results from this chapter will appear in the *Proceedings of the International Database Engineering and Application Symposium (IDEAS'99)* to be held in Montreal, Canada in August, 1999 [106]. Some of the results from a set of prototype validating experiments conducted on real-world data have been published in the *Proceedings of the BIS'99 Conference* held in Poznan, Poland in April, 1999 [132].

We anticipate that due to its high accuracy and low construction costs, the R-ACM (along with the T-ACM to be introduced in the next chapter) could prove to be a standard tool for query result size estimation in future database systems.

Chapter 4

Trapezoidal Attribute Cardinality Map[1]

In Chapter 3, we introduced a non-parametric statistical model called the Rectangular Attribute Cardinality Map that can be used to obtain more accurate estimation results than the traditional equi-width and equi-depth histograms. In this chapter we introduce the second major contribution of this thesis, namely the Trapezoidal Attribute Cardinality Map (T-ACM), which is another non-parametric histogram-like estimation technique based on the trapezoidal-rule of numerical integration. The T-ACM generalizes the R-ACM from a "step" representation to a "linear" representation. As in the case of the R-ACM, we introduce the T-ACM in this chapter as a fundamental tool for query result-size estimation and provide a mathematical foundation for its use. We also investigate the use of the T-ACM in the result-size estimation of various relational operations. Also, as in the case of the R-ACM, we propose to store and maintain the T-ACM in the DBMS catalogue.

4.1 Definition

A trapezoidal ACM is a modified form of the equi-width histogram where each histogram partition is a trapezoid instead of a rectangle. In fact, the trapezoidal ACM

[1]The theoretical aspects of the work done in this chapter are currently being reviewed for publication, and can also be found as a Carleton University Technical Report, TR-99-04 [108].

is obtained by replacing each of the rectangular sectors of the equi-width histogram by a trapezoid. The beginning and ending frequency values of each trapezoid sector is chosen so that the area of the resulting trapezoid will be equal to the area of the "rectangle" of the histogram it is replacing.

Definition 4.1 A One dimensional Trapezoidal ACM: *Let* $\mathcal{V} = \{v_i : 1 \leq i \leq |\mathcal{V}|\}$, *where* $v_i < v_j$ *when* $i < j$, *be the set of values of an attribute* X *in relation* R. *Let the value set* $\mathcal{V}$ *be subdivided into* s *equi-width sectors, each having sector width,* l. *We approximate each equi-width sector by a trapezoid in which the* j^{th} *trapezoid is obtained by connecting the starting value,* a_j, *to the terminal value,* b_j, *where the quantities* a_j *and* b_j *satisfy:*

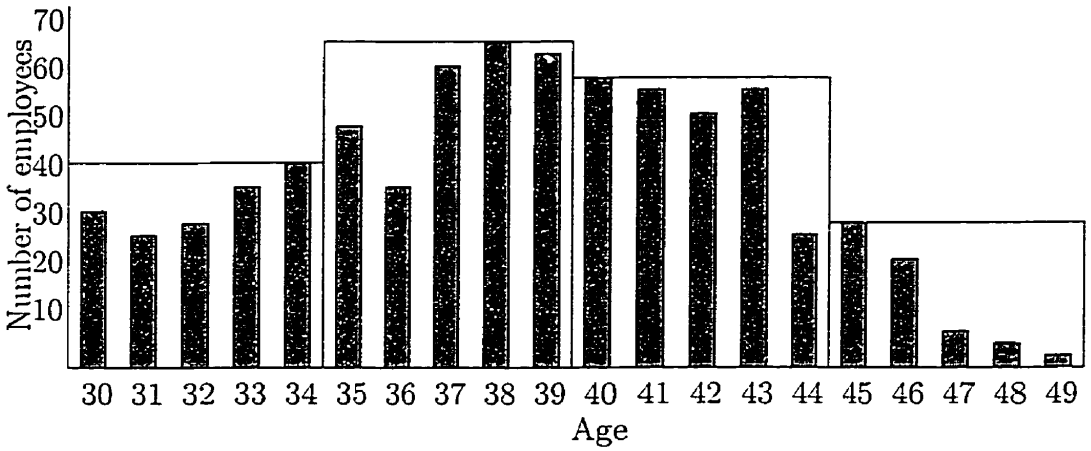
(a) The starting value a_1 *is a user-defined parameter.*

(b) For all $j > 1$, *the starting value of the* j^{th} *trapezoid,* a_j, *is the terminal value of the* $(j-1)^{st}$ *trapezoid,* b_{j-1}.

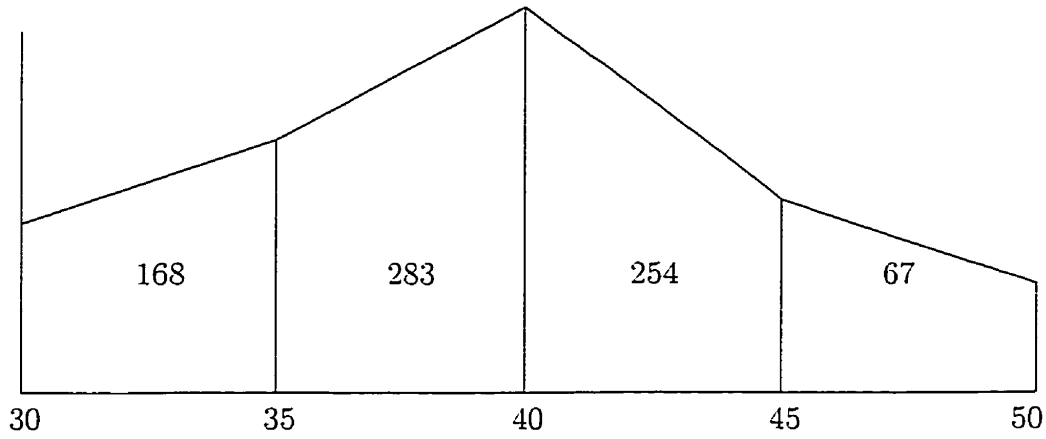
(c) The area of the j^{th} *trapezoid exactly equals the area of the* j^{th} *equi-width sector from which the exact computation of the quantity,* b_j, *is possible.*

Then the Trapezoidal Attribute Cardinality Map of attribute X *with initial attribute value* X_1 *and width* l *is the set* $\{(a_i, b_i) | 1 \leq i \leq s\}$.

Example 4.1 *Figure 4.1 shows the equi-width histogram and the trapezoidal ACM of the* `Age` *attribute of a relation* `Emp(SIN, Age, Salary)` *between* `Age = 30` *and* `Age = 49`. *Note that the actual frequency for every age value is shown in the histogram as shaded rectangles. As can be noticed, the starting and ending frequencies of each trapezoidal sector is chosen so that the area under the trapezoid is equivalent to the area of the corresponding rectangular sector of the histogram. From the trapezoidal ACM, the number of tuples in the relation with ages in the range of* `35 ≤ Age < 40` *is 283 and the estimate for the number of employees having* `Age = 48` *is 6. Compare this with the rectangular ACM in Example 3.1.*



(a) Equi-width Histogram for the Employe Age Distribution



(b) Corresponding Trapezoidal ACM

Figure 4.1: An Example for Constructing the Trapezoidal Attribute Cardinality Map

Our motivation for proposing the trapezoidal ACM for density approximation (and the query result size estimation) originates from considering the various techniques used in numerical integration. Finding the result size of a selection query on a range-predicate can be considered as a discrete case of finding the area under a curve. Thus any numerical integration technique used to find the area under a curve will fit our purpose well. Though more accurate and sophisticated methods such as Simpson's Rule exist, since the trapezoidal method is relatively easy to use in a DBMS setting and is much superior to the traditional equi-width and equi-depth histograms currently in use, we have opted to use the trapezoidal method. In addition to providing more accurate result estimation on selection queries on range predicates, it also gives better results on equality-match predicates.

4.1.1 Generating Trapezoidal ACM

Unlike the R-ACM, where the sector widths are variable, the sector widths of a T-ACM are all equal. Each sector or cell of a T-ACM stores the *modeled* frequency values of the first and last attribute values in that sector, which naturally leads to the number of tuples in the sector. Algorithm `Generate_T-ACM` partitions the value range of the attribute X into s equal width sectors of the T-ACM. The input to the algorithm is the number of partitions, s. The frequency distribution is assumed to be available in an integer array A, which has a total of L entries for each of the L distinct values of X. Note that the input can also be an equi-width histogram. For simplicity, we assume that the attribute values are ordered integers from 0 to $L-1$. The output of the algorithm is the T-ACM for the given attribute value set. Since choosing the starting frequency value of the first trapezoidal sector is important for obtaining the subsequent a_j's and b_j's, we briefly discuss it below.

Determining the First Frequency Value, a_1

As we shall see later from Lemmas 4.1 and 4.2, if the frequency of the first attribute value of the first sector of a T-ACM is known, the subsequent a_j's and b_j's can be easily obtained. The problem of obtaining an optimal starting frequency for building

a T-ACM is still open and is currently being investigated. Below we have listed some of the methods that can be used to obtain this quantity:

(1) a_1 is a user-defined frequency value.

(2) a_1 is obtained using the average of all the frequencies in the given attribute value domain.

(3) Use the frequency value from (2) above as the starting frequency of the first sector and compute all the a_j's and b_j's in a *left-to-right* manner. Again use the frequency value from (2) above as the terminal frequency of the last sector and compute all the a_j's and b_j's in a *right-to-left* manner. One possibility is to assign a_1 to be the average of the first frequency values resulting from these two methods.

Before presenting the `Generate_T-ACM` algorithm that generates a T-ACM, we shall first present two lemmas that are used in this algorithm.

Lemma 4.1 *For each sector in the T-ACM, the number of tuples, n_j, is equal to,*

$$n_j = \left(\frac{a+b}{2}\right).l,$$

where a, b are the frequencies of the first and last attribute value in the sector and l is the number of distinct values in the sector.

Proof: This can be easily shown using the geometry of the trapezoidal sector. □

This lemma is important because ensuring that n_j is close to $(a+b)l/2$ would provide us the desired accuracy using trapezoidal approximation.

Let a_j be the frequency of the first attribute value in the j^{th} sector. The first frequency value of the first sector, a_1 can be chosen to be either the actual frequency of the attribute value (i.e: $a_1 = x_1$) or the average frequency of the entire attribute value range (i.e: $a_1 = \frac{N}{sl}$). The subsequent values for $a_j, 2 \leq j \leq s$, do not need to be stored explicitly and can be obtained from Lemma 4.2.

Algorithm 4.3 Generate_T-ACM

```
Input:  No of sectors, s, frequency distrib.of X as A[0...L-1]
        Alternatively, input can also be an equi-width histogram.
Output: T-ACM
begin
   Initialize_ACM;      /* set all entries in ACM to zero */
   ACM[1].a := (sum_{i=0}^{L-1} A[i]) / L ; /* set a_1 to average frequency */
   for j := 1 to s do /* for every sector */
      for i := 1 to l do /* for every attrib.value */
         ACM[j].n := ACM[j].n + A[(j-1)*l + i];
      end; { for }
      if (j > 1) then ACM[j].a := ACM[j-1].b;
      ACM[j].b := 2 * ACM[j].n/l - ACM[j].a;
   end; { for }
end
```

EndAlgorithm Generate_T-ACM;

Lemma 4.2 *If the frequency of the first attribute value of the first sector of a T-ACM is a_1, then the frequency of the first attribute value of the j^{th} T-ACM sector, $a_j, 2 \leq j \leq s$, is given by,*

$$a_j = (-1)^{j-1}\frac{2}{l}\left\{a_1 + \sum_{k=1}^{j-1}(-1)^k n_k\right\}$$

where n_k is the number of tuples in the k^{th} sector.

Proof: Given the frequency of the first attribute value in a T-ACM sector, the frequency of the last attribute value in that sector can be obtained by using Lemma 4.1. Hence we have the following first and last frequency values a_j's and b_j's for a T-ACM.

$$\begin{array}{ll} a_1 = a & b_1 = \frac{2n_1}{l} - a \\ a_2 = b_1 = \frac{2n_1}{l} - a & b_2 = \frac{2n_2}{l} - a_2 \\ \vdots & \vdots \\ a_j = b_{j-1} = (-1)^{j-1}\frac{2}{l}\left\{a_1 + \sum_{k=1}^{j-1}(-1)^k n_k\right\} & b_j = \frac{2n_j}{l} - a_j \end{array}$$

Hence the lemma. □

In practice, we can obtain a_j easily from a_{j-1} as shown in the Algorithm 4.3. We also obtain a_1 by averaging the frequency values of the entire attribute range. Note that each entry of the ACM array is a record with three fields, namely n, a, b, which store the number of tuples, the frequency of the first value and the frequency of the last value in the sector respectively.

It is obvious that the algorithm, `Generate_T-ACM` generates the T-ACM corresponding to the given frequency value set. Assuming the frequency distribution of X is already available in array A, the running time of the algorithm `Generate_T-ACM` is $O(L)$ where L is the number of distinct attribute values.

4.2 Density Estimation Using Trapezoidal ACM

Consider a trapezoidal ACM sector of sector width l with n_j tuples. We assume that the tuples in this sector occur according to a trapezoidal probability distribution. In other words, the number of occurrences of the attribute values is not uniform, but it increases (decreases) from the left most value a to the right most value b in the sector in a linear fashion as shown in Figure 4.2 (a).

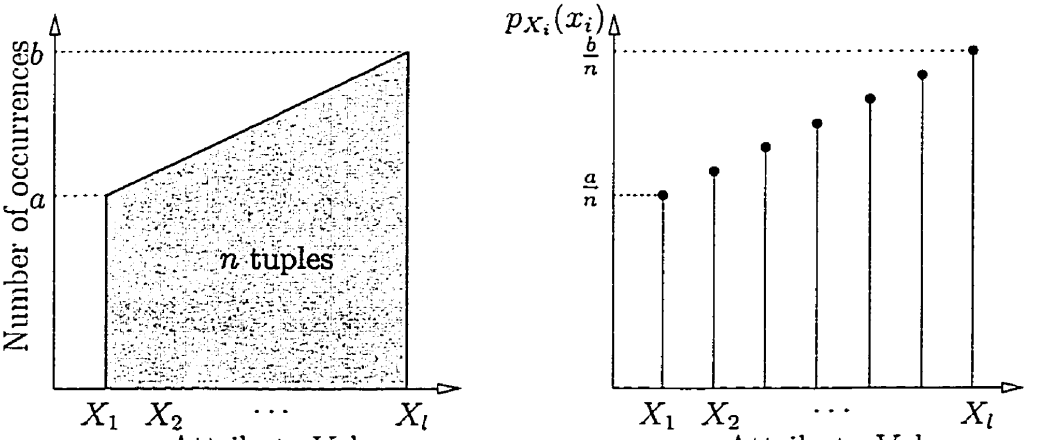


Figure 4.2: Trapezoidal ACM sector and its corresponding probability mass function

Since the probability of a given attribute value X_i occurring is the number of

occurrences of X_i divided by the total number of tuples n_j, the probability mass function (pmf) $p_{X_i}(x_i)$ can be sketched as shown in Figure 4.2 (b).

Lemma 4.3 *The probability of a given value X_i occurring in the trapezoidal sector is,*

$$p_{X_i}(x_i) = \frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}.i \qquad 1 \leq i \leq l-1 \tag{4.1}$$

where a_j is the frequency for the first attribute value in the j^{th} sector.

Proof: From the geometry of Figure 4.2 (b), we know that

$$\begin{aligned} p_{X_i}(x_i) &= \frac{a_j}{n_j} + \frac{b_j - a_j}{n_j(l-1)}.i \qquad 1 \leq i \leq l-1 \\ \text{But, } b_j &= \left(\frac{2n_j}{l} - a_j\right) \text{ from Lemma 4.1.} \\ \text{So, } p_{X_i}(x_i) &= \frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}.i \qquad 1 \leq i \leq l-1. \end{aligned}$$

This proves the lemma. □

Lemma 4.4 *The probability mass distribution for the frequencies of the attribute values in a T-ACM is a Binomial distribution with parameters $(n, p_{X_i}(x_i))$.*

Proof: Consider an arbitrary permutation (or arrangement) of the n tuples in the sector. Suppose the value X_i occurs exactly x_i number of times, then all other $(l-1)$ values must occur a combined total of $(n - x_i)$ times. Since the probability of X_i occurring once is $p_{X_i}(x_i)$, where $p_{X_i}(x_i)$ is given by Lemma 4.3, the probability of this value not occurring is $(1 - p_{X_i}(x_i))$. From hereafter, let us denote $p_{X_i}(x_i)$ simply as p_i for convenience. Hence the probability of an arbitrary permutation of the n tuples, where the value X_i occurs exactly x_i number of times is,

$$p_i^{x_i}(1 - p_i)^{n - x_i}. \tag{4.2}$$

There are $\binom{n}{x_i}$ different permutations of the n tuples in the sector where X_i occurs exactly x_i number of times and all other values occur a combined total of $(n - x_i)$

times. Hence we find that the probability that an arbitrary value X_i occurs exactly x_i number of times is,

$$\binom{n}{x_i} p_i^{x_i} (1 - p_i)^{n - x_i}. \tag{4.3}$$

In other words, we note that each of the attribute values $X_1, X_2, \ldots, X_l$ forms a binomial distribution *Binomial* (n, p_i) with a parameter determined by its location in the trapezoidal sector. □

4.3 Maximum Likelihood Estimate Analysis for the Trapezoidal ACM

In the previous section we showed that the frequency distribution of an attribute value in a T-ACM sector is a Binomial distribution whose parameters change with the location of the attribute value. With this knowledge, we shall now derive a maximum likelihood estimate for the frequency of an arbitrary attribute value in a T-ACM sector. Again, as in Section 3.4, contrary to the classical estimation theory, where we are interested in estimating the parameters, such as the mean, of a distribution of one or more random variables, in our problem, we are interested in estimating the value of the occurrence of the random variable (the frequency x_i) which we assume is "inaccessible". As in the case of the R-ACM, we do this in terms of an observation of one or more accessible random variables (i.e: the total number of tuples, n, and the slope and width of the T-ACM sector). In order to do this, we shall derive the maximum likelihood estimate, which maximizes the corresponding likelihood function. The analysis closely follows the R-ACM proofs and so the details will be omitted to avoid repetition.

Theorem 4.1 *For a one-dimensional trapezoidal ACM, the maximum likelihood estimate of the number of tuples for a given value X_α of attribute X in the k^{th} T-ACM*

sector is given by,

$$\hat{x}_{ML} = a_k + \frac{2(n_k - a_k l)}{l(l-1)}.z_\alpha$$

where n_k is the number of tuples in the k^{th} T-ACM sector, a_k is the frequency of the first attribute value in the k^{th} sector, l is the number of distinct attribute values (or width) of the T-ACM sectors and X_α is the z_α^{th} value in the T-ACM sector.

Proof: The proof of the theorem follows closely the analysis given for Theorem 3.1. The difference however is that the parameters of the Binomial distribution vary with the location of the attribute values. However this is, technically speaking, a minor irritant, since the quantity p would be replaced by p_i, and the rest of the proof would follow directly. The details are omitted in the interest of brevity.

Thus using the arguments given in the proof for Theorem 3.1, $\hat{x}_{ML}$ of x is obtained as,

$$\hat{x}_{ML} = n_k p.$$

But we have already seen in Lemma 4.3 that, the probability of the z_α^{th} attribute value, X_α, occurring in a T-ACM sector is given by,

$$p_{X_\alpha}(x_\alpha) = \frac{a_k}{n_k} + \frac{2(n_k - a_k l)}{n_k l(l-1)}.z_\alpha$$

where a_k is the frequency of the first attribute value in the k^{th} sector. So we have,

$$\hat{x}_{ML} = a_k + \frac{2(n_k - a_k l)}{l(l-1)}.z_\alpha.$$

Hence the theorem. □

In most of the cases, the maximum likelihood estimate, $\hat{x}_{ML} = np$, which we derived using the Gamma function above is not an integer. In fact, as we discussed in the case of the R-ACM, the maximum likelihood estimate reaches its upper limit

of np at integer values only in very special cases. If we are interested in the integer maximum likelihood value which is related to the above maximum likelihood estimate, we have to discretize the space. Thus, considering the analogous discrete case, we have the following theorem.

Theorem 4.2 *For a one-dimensional trapezoidal ACM, the maximum likelihood estimate of the number of tuples for a given value X_i of attribute X falls within the range of,*

$$\frac{a_k}{n_k} + \frac{2(n_k - a_k l)}{n_k l(l-1)}.z_\alpha(n_k+1) - 1 \leq \hat{x}_{ML} \leq \frac{a_k}{n_k} + \frac{2(n_k - a_k l)}{n_k l(l-1)}.z_\alpha(n_k+1).$$

where a_k is the frequency of the first attribute value in the k^{th} sector, n_k is the number of tuples in the k^{th} sector containing the value X_i and l is the width of that sector.

Proof: Again the proof of this theorem follows closely that of Theorem 3.2, so the details are omitted in the interest of brevity. Using the arguments from the proof of Theorem 3.2, we find that

$$p(n+1) - 1 < x < p(n+1).$$

Since $p = \frac{a_k}{n_k} + \frac{2(n_k - a_k l)}{n_k l(l-1)}.z_\alpha$, the theorem follows. □

4.4 Expected and Worst-Case Error Analysis for the T-ACM

The maximum likelihood estimation of the frequency of an attribute value tells us that the attribute value would have a frequency of $\hat{x}_{ML}$ with high degree of certainty when compared to the other possible frequency values. But even though the attribute value occurs with the maximum likelihood frequency with high probability, it can also occur with other frequencies with smaller probabilities. Hence, as we did in the case of the R-ACM, when we need to find the worst-case and average-case errors for the

result size estimations, we need to obtain the expected value of the frequency of a given attribute value. We use our Binomial model for the T-ACM sector to find the expected value of the frequency of an attribute value as given in the following lemma and develop a series of results regarding the corresponding query result-size estimates.

Lemma 4.5 *Using a trapezoidal approximation, the expected number of tuples for a given value X_i of attribute X is,*

$$E(X_i) = a_j + \frac{2(n_j - a_j l)}{l(l-1)}.i,$$

where n_j is the number of tuples in the sector which contains value X_i and l is the sector width. The quantity a_j is the number of occurrences of the first attribute value in the j^{th} sector.

Proof: The frequency distribution of attribute values in a T-ACM sector is a binomial distribution with parameters (n, p_i) where p_i is given by Lemma 4.3. Hence the expected value $E(X_i)$ is its mean,

$$E(X_i) = n_j p_i = a_j + \frac{2(n_j - a_j l)}{l(l-1)}.i$$

and the lemma follows. □

4.4.1 Estimation Error with the Trapezoidal ACM

In this section, we discuss the techniques for obtaining the estimation errors using the T-ACM. These results are similar to the ones we obtained for the R-ACM in Section 3.5.1.

Lemma 4.6 *The variance of the frequency of an attribute value X_i in sector j of a trapezoidal ACM is,*

$$Var(X_i) = n_j p_i (1 - p_i), \text{ where } p_i = \frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}.i$$

Proof: The frequency distribution in a T-ACM sector is a binomial distribution with parameters (n_j, p_i), where p_i is given by Lemma 4.3. Hence the variance is $n_j p_i(1-p_i)$. □

Lemma 4.7 *The sector variance of the j^{th} trapezoidal ACM sector is,*

$$Var_j = n_j - \frac{a_j(l+1)(a_j l - 2n_j)}{3n_j(l-1)} - \frac{2n_j(2l-1)}{3l(l-1)}$$

where a_j is the frequency of the first attribute value in the sector, n_j is the number of tuples in the sector and l is the sector width.

Proof: Since the frequency values in the sector are assumed independent, summing up the variances of all frequency values in the sector will give us the expression for the variance of the entire sector. So we have,

$$\begin{aligned} Var_j &= \sum_{i=0}^{l-1} n_j p_i(1-p_i) \\ &= \sum_{i=0}^{l-1} n_j \left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}.i\right)\left(1 - \frac{a_j}{n_j} - \frac{2(n_j - a_j l)}{n_j l(l-1)}.i\right). \end{aligned}$$

Simplifying the above expression gives us,

$$Var_j = n_j - \frac{a_j(l+1)(a_j l - 2n_j)}{3n_j(l-1)} - \frac{2n_j(2l-1)}{3l(l-1)}$$

and the lemma follows. □

Lemma 4.8 *The variance of a T-ACM is given by,*

$$Var(ACM) = \sum_{j=1}^{s} Var_j$$

where s is the number of sectors in the T-ACM, and Var_j is the sector variance given in Lemma 4.7.

Proof: The lemma follows directly from the fact that the frequency values in each sector are independent of each other and thus summing up the variances of all the sectors will give the overall variance which is also an estimate for the estimation error. □

4.4.2 Self-join Error with the Trapezoidal ACM

Since query result sizes are maximized for self-joins, as in the case of the R-ACM, in this section we consider self-join and error estimation with the T-ACM. Assuming that the duplicate tuples after the join are not eliminated, we have the following lemma.

Lemma 4.9 *The error, ϵ, resulting from a self-join of relation R on attribute X using a trapezoidal ACM is given by,*

$$\epsilon = \sum_{j=1}^{s} \left(\sum_{k=1}^{l} x_k^2 - n_j^2 + n_j Var_j \right)$$

where s is the number of sectors in the T-ACM, and n_j is the number of tuples in the j^{th} sector and Var_j is the variance of the j^{th} sector given in Lemma 4.7.

Proof: Since there are $L = sl$ distinct values for attribute X, the actual value, ξ and expected value κ of the join size can be estimated as follows.

$$\xi = \sum_{i=1}^{L} x_i^2 = \sum_{j=1}^{s} \sum_{k=1}^{l} x_k^2.$$

The frequency of an arbitrary attribute value is computed from the T-ACM as the expected value $E(x_i)$, which is the average frequency of the T-ACM sector. Hence the result of self-joining this attribute value would be $[E(x_i)]^2$. Hence the size of the

join computed by the T-ACM, κ, is,

$$\begin{aligned}\kappa &= \sum_{i=1}^{L}[E(x_i)]^2 = \sum_{j=1}^{s}\sum_{i=0}^{l-1}[E(x_i)]^2 \\ &= \sum_{j=1}^{s} n_j^2 \sum_{i=0}^{l-1} p_i^2. \end{aligned} \tag{4.4}$$

But the variance of the j^{th} sector Var_j is,

$$\begin{aligned} Var_j &= \sum_{i=0}^{l-1} n_j p_i(1-p_i) = n_j - n_j \sum_{i=0}^{l-1} p_i^2 \\ \text{So, } \sum_{i=0}^{l-1} p_i^2 &= 1 - \frac{Var_j}{n_j}. \end{aligned}$$

Substituting the above expression in Equation (4.4), we obtain,

$$\kappa = \sum_{j=1}^{s}(n_j^2 - n_j Var_j).$$

Hence the error in estimation of self-join is,

$$\begin{aligned}\xi - \kappa &= \sum_{j=1}^{s}\sum_{k=1}^{l} x_k^2 - \sum_{j=1}^{s}(n_j^2 - n_j Var_j) \\ &= \sum_{j=1}^{s}\left(\sum_{k=1}^{l} x_k^2 - n_j^2 + n_j Var_j\right) \end{aligned}$$

and the lemma is proved. □

Corollary 4.4 *The error, ϵ, resulting from a self-join of relation R on attribute X using a trapezoidal ACM is given by,*

$$\epsilon = \sum_{j=1}^{s}\left(\sum_{k=1}^{l} x_k^2 - \frac{a_j(l+1)(a_j l - 2n_j)}{3(l-1)} - \frac{2n_j^2(2l-1)}{3l(l-1)}\right)$$

where a_j is the frequency of the first attribute value in the j^{th} sector.

Proof: The proof follows from substituting Var_j in the previous lemma with the expression obtained in Lemma 4.7. □

4.4.3 Worst Case Error with the Trapezoidal ACM

We have already seen that Trapezoidal ACMs give smaller errors than the histograms. In this section we shall find estimates for these errors in the worst-case for both an equality-match and a range-select operations.

Theorem 4.3 *The worst-case error, ϵ, in estimating the frequency of an arbitrary attribute value X_i in a trapezoidal ACM is,*

$$\epsilon = \begin{cases} a_j + \frac{2(n_j - a_j l)}{l(l-1)} i & \text{if } i < \frac{l(l-1)(n_j - 2a_j)}{4(n_j - a_j l)}, \\ n_j - a_j - \frac{2(n_j - a_j l)}{l(l-1)} i & \text{if } i \geq \frac{l(l-1)(n_j - 2a_j)}{4(n_j - a_j l)}. \end{cases}$$

Proof: The expected frequency of the attribute value X_i reported as a result of the T-ACM is,

$$\hat{\xi} = n_j \left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)} i \right)$$

where $\left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)} i \right)$ is the probability that attribute value X_i occurs in the T-ACM sector. But the actual frequency ξ of attribute value X_i can be anywhere in the range of,

$$0 \leq \xi \leq n_j.$$

Hence the maximum worst case error is,

$$\epsilon = \max(\hat{\xi}, n_j - \hat{\xi}).$$

It is easy to observe that whenever $\hat{\xi} < \frac{n_j}{2}$, the maximum worst case error occurs when the actual frequency ξ is equal to n_j. The maximum worst case error in this case is $n_j - \hat{\xi}$. Whereas whenever $\hat{\xi} \geq \frac{n_j}{2}$, the maximum worst case error occurs when the actual frequency $\xi = 0$. The maximum worst case error in this case is of course $\hat{\xi}$. We note that whether the expected frequency value $\hat{\xi}$ is smaller or larger than $\frac{n_j}{2}$ depends on the location of the attribute value X_i within the T-ACM sector. The location of the attribute value when the expected frequency $\hat{\xi}$ is equal to $\frac{n_j}{2}$ can be obtained by solving,

$$a_j + \frac{2(n_j - a_j l)}{l(l-1)} i = \frac{n}{2}$$

and is equal to,

$$i = \frac{l(l-1)(n_j - 2a_j)}{4(n_j - a_j l)}.$$

The theorem follows from the above. □

As we discussed in Chapter 3, when estimating the sum of frequencies in an attribute value range, we have to consider three distinct cases. These are namely the cases when,

1. The attribute value range spans across one T-ACM sector.
2. The attribute value range falls completely within one T-ACM sector.
3. The attribute value range spans across more than one T-ACM sector.

In the first case, estimation using the T-ACM gives the accurate result (n_j) and there is no estimation error. The estimation error in the second case is given by the theorem below. The estimation error in the third case can be obtained by noting that it is in fact the combination of the first and second cases.

Theorem 4.4 *The worst-case error, ϵ, in estimating the sum of frequencies between*

the attribute values of $X = X_\alpha$ and $X = X_\beta$, when these attribute values fall completely within a T-ACM sector, is given by,

$$\epsilon = \begin{cases} n_j - \mathcal{A} & \text{if } \mathcal{A} < \frac{n_j}{2}, \\ \mathcal{A} & \text{if } \mathcal{A} \geq \frac{n_j}{2} \end{cases}$$

where $\mathcal{A}$ is the sum of the expected frequencies between the attribute values X_α and X_β and is equal to,

$$\mathcal{A} = a_j(\beta - \alpha + 1) + \frac{(n_j - a_j l)(\beta - \alpha + 1)(\beta - \alpha + 2)}{l(l-1)}.$$

Proof: The sum of the expected frequencies between the attribute values X_α and X_β within a T-ACM sector is,

$$\begin{aligned} \mathcal{A} &= \sum_{i=\alpha}^{\beta} \hat{\xi} = \sum_{i=\alpha}^{\beta} \left(a_j + \frac{2(n_j - a_j l)}{l(l-1)} i \right) \\ &= a_j(\beta - \alpha + 1) + \frac{(n_j - a_j l)(\beta - \alpha + 1)(\beta - \alpha + 2)}{l(l-1)}. \end{aligned}$$

But the actual sum of frequencies ξ between the attribute values X_α and X_β can be anywhere in the range of,

$$0 \leq \xi \leq n_j.$$

Hence the maximum worst case error is,

$$\epsilon = \max(\mathcal{A}, n_j - \mathcal{A}).$$

It is easy to observe that whenever $\mathcal{A} < \frac{n_j}{2}$, the maximum worst case error occurs when the actual sum of frequencies, ξ, is equal to n_j. The maximum worst case error in this case is $n_j - \mathcal{A}$. Whereas whenever $\mathcal{A} \geq \frac{n_j}{2}$, the maximum worst case error occurs when the actual sum of frequencies, $\xi = 0$. The maximum worst case error in

this case is of course $\mathcal{A}$. Hence the theorem. □

4.4.4 Average Case Error with Trapezoidal ACM

In this section we give an estimate for the average-case error with a trapezoidal ACM. As we shall see, the average case error is much smaller than the worst-case error that we derived in the previous section. The average-case is synonymous with a truly random sector in which all attribute values have the same (or approximately same) frequency value equal to the mean sector frequency, $\frac{n_j}{l}$. The average case error in estimating the frequency of an arbitrary value X_i can be obtained by two different methods, depending on how the frequencies of the attribute values are averaged. In the first case, the expected frequency of all attribute values in the sector is obtained by averaging over the entire sector. In the second case, we obtain the average frequency of *an attribute* value by averaging all the possible frequencies that this particular attribute value can assume. The average case errors in these two situations are given below.

Theorem 4.5 *Assuming that the T-ACM sector has been obtained by processing a histogram bucket of size l with n_j tuples, the average error in estimating the frequency of an arbitrary attribute value X_i, obtained by averaging over* all *attribute values in this sector of the trapezoidal ACM is exactly zero.*

Proof: The expected frequency of the attribute values in the sector computed by the T-ACM can be obtained by averaging **over the entire sector** as,

$$\begin{aligned} E(\hat{\xi}_i) &= \frac{1}{l}\sum_{i=0}^{l-1}\hat{\xi}_i \\ &= \frac{1}{l}\sum_{i=0}^{l-1} n_j\left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}i\right) = \frac{n_j}{l} \end{aligned}$$

where $\left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}i\right)$ is the probability that attribute value X_i occurs in the T-ACM sector. But, if we assume that the T-ACM sector has been obtained by processing an equivalent histogram bucket of size l with n_j tuples, then the actual frequency ξ_i

of attribute value X_i in the average case is equal to, $\xi_i = \frac{n_j}{l}$. Hence the average case error obtained by averaging **over the entire range** is equal to,

$$\epsilon = \hat{\xi}_i - \xi_i = \frac{n_j}{l} - \frac{n_j}{l} = 0.$$

The theorem follows. □

Note that in the case of an R-ACM, the actual frequencies of the attribute values are controlled by the tolerance, τ. That is why in the R-ACM (unlike the T-ACM), the average case error is not equal to zero. In a T-ACM, due to the geometry of the T-ACM sector, each of the negative estimation errors in the right half of the sector cancels out with the corresponding positive estimation error on the left half of the sector, thus resulting in an overall zero average case error **when the expectation operation is carried out by averaging over the entire sector.** Note that this will not be the case if we perform the expectation at any one particular attribute value, and this is discussed in the theorem below.

Theorem 4.6 *The upper bound of the average-case error, ϵ, in estimating the frequency of an arbitrary attribute value X_i in a trapezoidal ACM is,*

$$\epsilon = a_j + \frac{2(n_j - a_j l)}{l(l-1)}.i - \frac{n_j}{l}.$$

Proof: The expected frequency of the attribute value computed by the T-ACM is,

$$\hat{\xi} = n_j \left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)} i \right)$$

where $\left(\frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)} i \right)$ is the probability that attribute value X_i occurs in the T-ACM sector. But assuming that the T-ACM sector has been obtained from an equivalent histogram bucket of size l with n_j tuples, we note that, due to the uniformity assumption within the histogram bucket, the frequency ξ_i of attribute value X_i in this

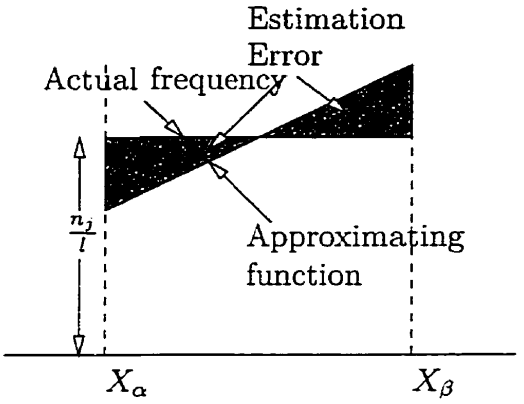


Figure 4.3: Average Case Error in T-ACM

histogram bucket is equal to, $\xi_i = \frac{n_j}{l}$. Hence the average-case error is equal to,

$$\epsilon = \hat{\xi} - \xi = a_j + \frac{2(n_j - a_j l)}{l(l-1)}.i - \frac{n_j}{l}.$$

The theorem follows. □

As before in the worst-case error analysis, when estimating the sum of frequencies in an attribute value range, we have three distinct cases. The theorem given below deals with the case of the average-case error when attribute value range falls completely within one T-ACM sector. The case when the attribute value range spans across one entire T-ACM sector is trivial and does not result in any estimation error. The case when the attribute value range spans across more than one T-ACM sector can be solved by decomposing it into the first two cases.

Theorem 4.7 *The average-case error, ϵ, in estimating the sum of frequencies between the attribute values of $X = X_\alpha$ and $X = X_\beta$, when these values fall completely within a T-ACM sector, is given by,*

$$\epsilon = \frac{(\beta - \alpha)(\alpha + \beta - 3)(n_j - a_j l)}{l - 1}.$$

Proof: In a random T-ACM sector, all the frequency values are equal (or close) to the mean frequency value. This is shown in Figure 4.3 along with the T-ACM frequency

distribution that is used to approximate the actual frequency distribution. We note that the shaded area between the actual and approximate frequency distribution represents the cumulative estimation error. Also we see that both lines intersect at the center of the sector or at $i = \frac{l-1}{2}$. Hence an estimate for the estimation error is,

$$\begin{aligned}
\epsilon &= \frac{n_j}{l}\left(\frac{l-1}{2} - \alpha + 1\right) - \sum_{i=\alpha}^{\frac{l-1}{2}} \left(a_j + \frac{2(n_j - a_j l)}{l(l-1)}\right) i \\
&\quad + \sum_{i=\frac{l-1}{2}}^{\beta} \left(a_j + \frac{2(n_j - a_j l)}{l(l-1)}.i\right) - \frac{n_j}{l}\left(\frac{l-1}{2} - \beta + 1\right) \\
&= \frac{(\beta - \alpha)(\alpha + \beta - 3)(n_j - a_j l)}{l-1}
\end{aligned}$$

and the theorem follows. □

4.5 Comparison of Trapezoidal ACM and Equi-Width Histogram

The rationale for the trapezoidal ACM is that the trapezoidal rule for numerical integration provides a more accurate estimation of the area under a curve than the left-end or right-end histogram (or rectangular rule) based methods. This is formally given by the following lemma.

Lemma 4.10 *If the estimation error for computing the sum of frequencies of all the attribute values falling between $X = a$ and $X = b$, using the trapezoidal ACM is $Error_T$ and that of using the histogram method is $Error_H$, then $Error_T < Error_H$.*

Proof: Without loss of generality, let us consider a continuous function $f(x)$. (See Figure 4.4.) Let A^* and A^{**} be the approximations to the area under the function $f(x)$ between $x = a$ and $x = b$ by these two methods respectively. Also let ϵ_1 and ϵ_2 be the errors introduced by the trapezoidal ACM and histogram methods in the

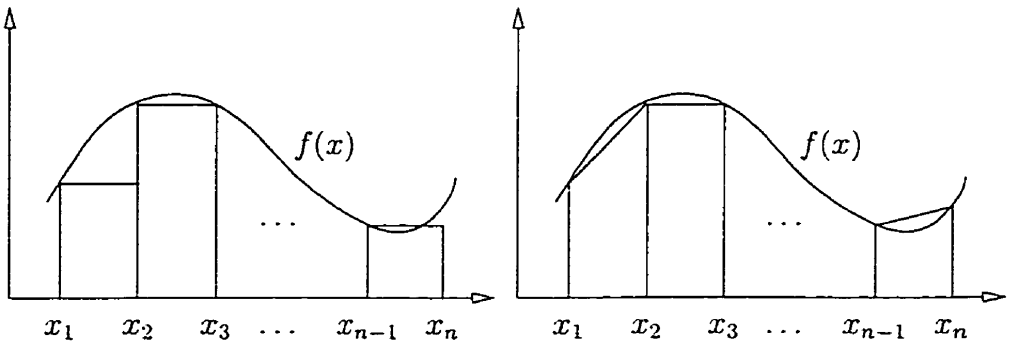


Figure 4.4: Comparison of Histogram and Trapezoidal ACM

estimated areas A^* and A^{**} respectively. Hence,

$$\epsilon_1 = A^* - \int_a^b f(x)dx$$
$$\text{and } \epsilon_2 = A^{**} - \int_a^b f(x)dx.$$

The histogram method, also known as the rectangular rule, and the trapezoidal ACM, also known as the trapezoidal rule are two well known numerical integration techniques to approximate the area under a curve. It can be shown [77] that using the trapezoidal rule, the estimation error, ϵ_1, has the following bounds.

$$\frac{(b-a)^3}{12n^2}M_2^* \leq \epsilon_1 \leq \frac{(b-a)^3}{12n^2}M_2.$$

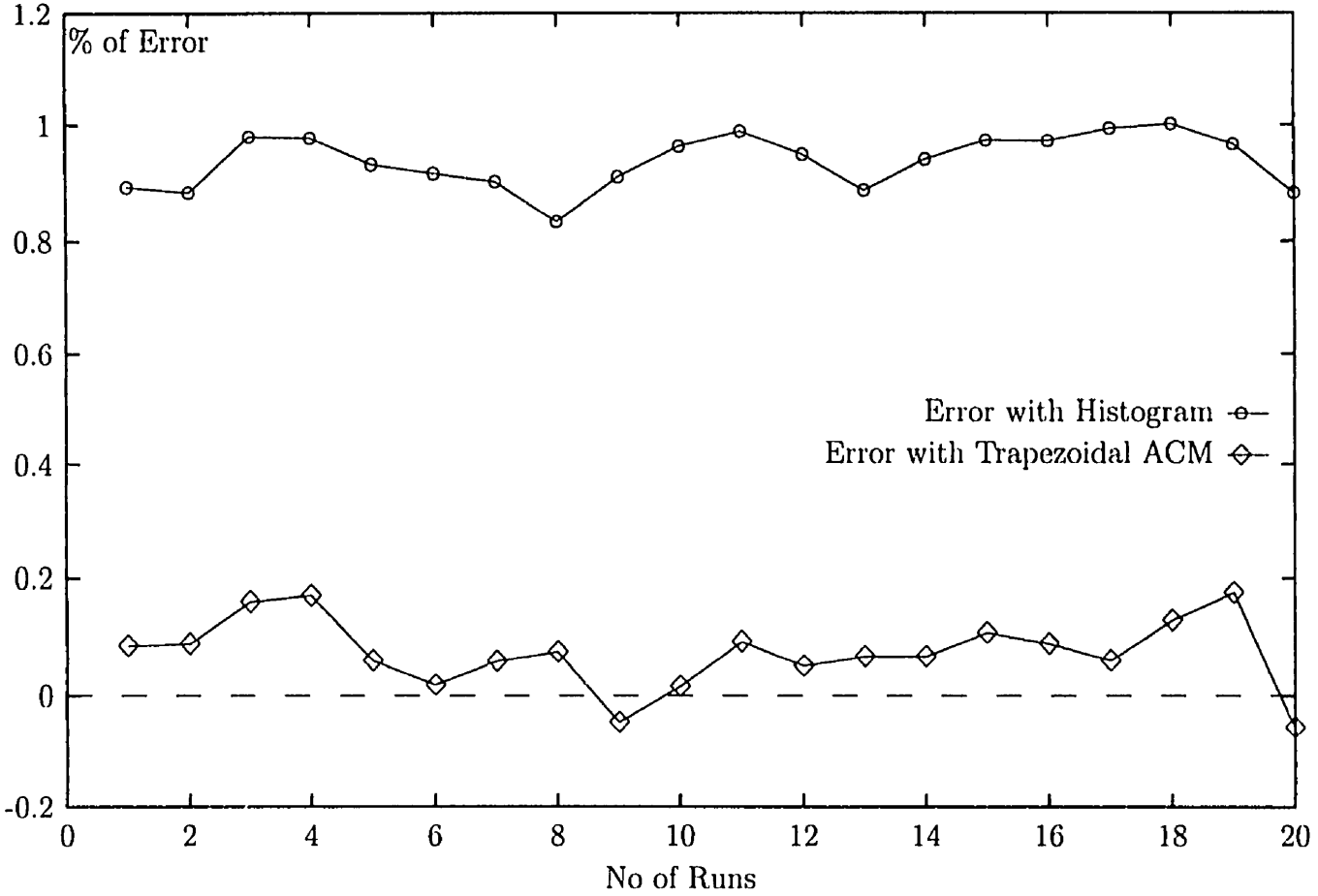
Similarly it can be shown that using the rectangular rule, the estimation error, ϵ_2, has the following bounds.

$$\frac{(b-a)^3}{6n^2}M_2^* + \frac{b^2}{2n}M_1^* \leq \epsilon_2 \leq \frac{(b-a)^3}{6n^2}M_2 + \frac{b^2}{2n}M_1.$$

In both bounds, the quantities M_1^*, M_1 are the smallest and largest values for the first derivative of f and M_2^*, M_2 are the smallest and largest values for the second derivative of f between $x = a$ and $x = b$. Hence it follows that $Error_T < Error_H$. □

Claim 4.1 *If the frequency estimation error for an arbitrary attribute value using*

Figure 4.5: Comparison of Histogram and the T-ACM for Probability Estimation: Each experiment was run 100,000 times to get the average percentage of errors in the estimated occurrence of the attribute values. Estimation errors are given for exact match on a random distribution with 100,000 tuples and 1000 distinct values. Both histogram and T-ACM were of equi-width type with a sector width of 5 and no of sectors equal to 200.



the trapezoidal ACM is $Error_T$ and that of using the equi-width histogram with the same number of sectors is $Error_H$, then $Error_T < Error_H$.

Rationale: Assume the actual frequency of an arbitrary attribute value X_i is ξ. Let the frequencies computed by an equi-width histogram and a T-ACM with the same number of sectors be $\hat{\xi}_H$ and $\hat{\xi}_T$ respectively. We use $E[(\xi - \hat{\xi})^2]$ as the measure for comparing the errors resulting from the histogram and the T-ACM. So we have,

$$\begin{aligned} E_H[(\xi - \hat{\xi}_H)^2] &= E_H\left[\left(\xi - \frac{n}{l}\right)^2\right] \\ &= E(\xi^2) - \left(\frac{n}{l}\right)^2 \\ &= \frac{\sum_{k=0}^{l-1}\sum_{i=0}^{n}\binom{n}{x_i}x_i^2 p_H^{x_i}(1-p_H)^{n-x_i}}{l} - \left(\frac{n}{l}\right)^2 \end{aligned}$$

where p_H is the probability of selecting one of the l attribute values in the histogram sector and is equal to $\frac{1}{l}$. Similarly,

$$\begin{aligned} E_T[(\xi - \hat{\xi}_T)^2] &= E(\xi^2) - [E(\hat{\xi}_T)]^2 \\ &= \frac{\sum_{k=0}^{l-1}\sum_{i=0}^{n}\binom{n}{x_i}x_i^2 p_{T_k}^{x_i}(1-p_{T_k})^{n-x_i}}{l} - \left(\frac{\sum_{k=0}^{l-1} a + \frac{2(n-al)}{l(l-1)}.k}{l}\right)^2 \\ &= \frac{\sum_{k=0}^{l-1}\sum_{i=0}^{n}\binom{n}{x_i}x_i^2 p_{T_k}^{x_i}(1-p_{T_k})^{n-x_i}}{l} - \left(\frac{n}{l}\right)^2 \end{aligned}$$

where p_{T_k} is the probability of selecting the k^{th} attribute value in a trapezoidal sector and is equal to $\frac{a}{n} + \frac{2(n-al)}{nl(l-1)}.k$.

Analytically proving that

$$\sum_{k=0}^{l-1}\sum_{i=0}^{n}\binom{n}{x_i}x_i^2 p_H^{x_i}(1-p_H)^{n-x_i} > \sum_{k=0}^{l-1}\sum_{i=0}^{n}\binom{n}{x_i}x_i^2 p_{T_k}^{x_i}(1-p_{T_k})^{n-x_i}$$

is difficult. But extensive simulations demonstrate that this is true, and we hope that one can use a symbolic mathematical package such as Mathematica or Maple to show this to be true. However we believe it is true due to the well acclaimed superiority of the trapezoidal rule over the rectangular rule in numerical integration, indicating

that $E_H[(\xi - \hat{\xi}_H)^2] > E_T[(\xi - \hat{\xi}_T)^2]$. □

4.6 Size Estimation of Selection Operations Using the T-ACM

In this chapter, we introduced a new catalogue based non-parametric histogram-like technique called Trapezoidal Attribute Cardinality Map. We also provided some theoretical analysis to show that their result size estimations are more accurate than the traditional histograms. Our analysis in the previous sections was mainly in the context of probability density estimations of data distributions. In this and the following sections, we discuss the actual application of the T-ACM for query result size estimation in a relational database system. In particular, we discuss how the T-ACM can be used for estimating the result sizes of *select* and *join* operations which are the most predominant operations in query processing.

4.6.1 Result Estimation of Equality Select Using the T-ACM

Let us consider a relational query with an equality select predicate, $X = X_\alpha$, where X_α is a constant value in the domain of attribute X. We are interested in finding the estimate of the result size of the query, $\sigma_{X=X_\alpha}(R)$, where the attribute value X_α is in position α of the k^{th} T-ACM sector. The following lemma gives us this estimate.

Theorem 4.8 *An estimate obtained by using the expected value analysis for the size of the equality select query, $\sigma_{X=X_\alpha}(R)$, using a T-ACM is,*

$$E(|\sigma_{X=X_\alpha}(R)|) = a_k + \frac{2(n_k - a_k l)}{l(l-1)}.z_\alpha$$

where n_k is the number of tuples in the k^{th} T-ACM sector, a_k is the frequency of the first attribute value in the k^{th} sector, l is the number of distinct attribute values (or width) of the T-ACM sectors and X_α is the z_α^{th} value in the T-ACM sector.

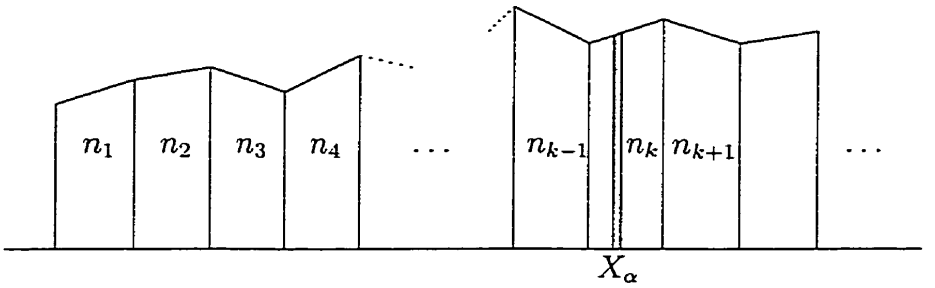


Figure 4.6: Equality Select Using the T-ACM

Proof: This is a direct result from Theorem 4.1. □

Since the result size estimates using the expected value and the maximum likelihood estimate are essentially identical, in the subsequent derivations, we shall refer to the estimate as the maximum likelihood estimate. As before, we are interested in both the worst-case and average case errors in the above equality select estimation. We can now make use of the results we have already obtained earlier in this chapter for the probability density estimation in the T-ACM sector. The following lemma gives the worst-case error in an equality select operation.

Lemma 4.11 *The worst-case error, ϵ, in estimating the equality select operation, $\sigma_{X=X_\alpha}(R)$ in a T-ACM using the maximum likelihood estimate is given by,*

$$\epsilon = \begin{cases} a_k + \frac{2(n_k - a_k l)}{l(l-1)} z_\alpha & \text{if } z_\alpha < \frac{l(l-1)(n_k - 2a_k)}{4(n_k - a_k l)}, \\ n_k - a_k - \frac{2(n_k - a_k l)}{l(l-1)} z_\alpha & \text{if } z_\alpha \geq \frac{l(l-1)(n_k - 2a_k)}{4(n_k - a_k l)}. \end{cases}$$

where the attribute value X_α is in the z_α^{th} position within the k^{th} T-ACM sector.

Proof: This is a direct result from Theorem 4.3. ■

As discussed in earlier in the chapter in the context of probability density estimation using the T-ACM, the average-case error in estimating the result size of the equality selection query, $\sigma_{X=X_\alpha}(R)$ can be obtained by two different methods, depending on how the frequencies of the attribute values are averaged. In the first case,

the expected frequency of all attribute values in the sector is obtained by averaging over the entire sector. In the second case, we obtain the average frequency of an attribute value by averaging all the possible frequencies that this particular attribute value can assume. These are given in the following lemmas.

Lemma 4.12 *Assuming that the T-ACM sector has been obtained by processing a histogram bucket of size l with n_j tuples, the average error in estimating the result size of the equality selection query, $\sigma_{X=X_\alpha}(R)$, obtained by averaging over all attribute values in this sector of the T-ACM is exactly zero.*

Proof: This is a direct result from Theorem 4.5. □

Lemma 4.13 *The upper bound of the average-case error, ϵ, in the maximum likelihood estimate of the equality selection query, $\sigma_{X=X_\alpha}(R)$ in a trapezoidal ACM is given by,*

$$\epsilon = a_k + \frac{2(n_k - a_k l)}{l(l-1)}.z_\alpha - \frac{n_k}{l}$$

where a_k is the frequency of the first attribute value in the k^{th} sector and X_α is in the z_α^{th} position of the T-ACM sector.

Proof: This is a direct consequence of Theorem 4.6. □

4.6.2 Result Estimation of Range Select Using the T-ACM

As we derived earlier for the case of R-ACM, here we shall find the estimate for the range query $S_{ML}(\sigma_{X\leq X_\alpha}(R))$. With this result estimate and the estimate we found above for $S_{ML}(\sigma_{X=X_\alpha}(R))$, we can find the estimates for all other range predicates. The following theorem finds the required estimate.

Theorem 4.9 *The maximum likelihood estimate for the range select query $\sigma_{X\leq X_\alpha}(R)$ using a T-ACM is given by,*

$$S_{ML}(\sigma_{X\leq X_\alpha}(R)) = \sum_{j=1}^{k-1} n_j + \alpha a_k + \frac{\alpha(\alpha+1)(n_k - a_k l)}{l(l-1)}$$

where a_k is the frequency of the first attribute value in the k^{th} sector.

Proof: The attribute values which satisfy the query $\sigma_{X \leq X_\alpha}(R)$ occupy the first $k-1$ T-ACM sectors and up to and including the α^{th} location of the k^{th} T-ACM sector. Hence the query result size is found by summing up the number of tuples in the first $k-1$ sectors of the T-ACM and the estimate of the number of tuples for the first α attribute values in the k^{th} sector of the T-ACM. We find the maximum likelihood estimate of the tuples for the first z_α attribute values in the k^{th} sector as below.

Let us assume that the frequencies of the first z_α attribute values in the k^{th} R-ACM sector are $x_1, x_2, \ldots, x_{z_\alpha}$. We can obtain a likelihood function on these frequencies by considering their join probability mass functions. But instead, without loss of generality, we consider the join probability mass function for the first two attribute values. So our likelihood function can be written as,

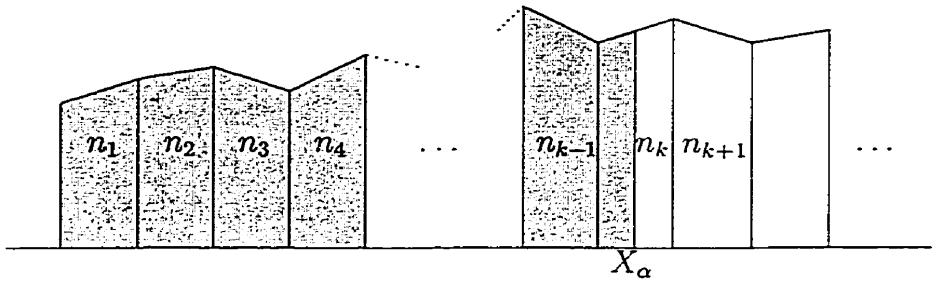
$$\begin{aligned} \mathcal{L}(x_1, x_2) &= \binom{n_k}{x_1} p^{x_1} (1-p)^{n_k - x_1} \binom{n_k}{x_2} p^{x_2} (1-p)^{n_k - x_2} \\ &= \binom{n_k}{x_1} \binom{n_k}{x_2} p^{x_1 + x_2} (1-p)^{2n_k - (x_1 + x_2)}. \end{aligned}$$

Taking natural logarithm on both sides of the likelihood function and then taking the partial derivative of $\ln \mathcal{L}(x_1, x_2)$ with respect to x_1, we obtain (after omitting the tedious algebraic details),

$$\frac{\partial\{\ln \mathcal{L}(x_1, x_2)\}}{\partial x_1} = \ln p - \ln(1-p) + \sum_{r=x_1+1}^{n_k - x_1} \frac{1}{r}.$$

Setting $\frac{\partial\{\ln \mathcal{L}(x_1, x_2)\}}{\partial x_1} = 0$, and noting that $\sum_{r=x_1+1}^{n_k - x_1} \frac{1}{r} \leq \ln\left(\frac{n_k - x_1}{x_1}\right)$, $\hat{x}_{ML}$ of x_1 is obtained as,

$$\begin{aligned} \frac{p(n_k - x_1)}{(1-p)x_1} &\geq 1 \quad \text{or} \\ \hat{x}_{ML} &= n_k p = \frac{n_k}{l_k}. \end{aligned}$$



Range Select with T-ACM (Shaded region is the result of select)

Figure 4.7: Result Estimation of Range Select Using the T-ACM

Similarly, finding $\frac{\partial\{\ln \mathcal{L}(x_1,x_2)\}}{\partial x_1}$ and setting it to zero, we find the maximum likelihood estimate for x_2 as $\hat{x}_{ML} = \frac{n_k}{l_k}$. Hence considering the join probability mass function of all other attribute values in this sector, we can show that each of their maximum likelihood estimate is equal to $\hat{x}_{ML} = \frac{n_k}{l_k}$. Thus the maximum likelihood estimate of the number of tuples for the first z_α attribute values is equal to,

$$\sum_{i=1}^{z_\alpha} \frac{n_k}{l_k} = \frac{z_\alpha n_k}{l_k}.$$

Hence the maximum likelihood estimate for the range select query $\sigma_{X \leq X_\alpha}(R)$ is given by,

$$\mathcal{S}_{ML}(|\sigma_{X \leq X_\alpha}(R)|) = \sum_{j=1}^{k-1} n_j + \alpha a_k + \frac{\alpha(\alpha+1)(n_k - a_k l)}{l(l-1)}$$

Hence the theorem. □

Using Theorem 4.4, we can find that the above estimate for the range select operation has the following worst-case estimation error.

$$\epsilon = \begin{cases} n_k - \mathcal{A} & \text{if } \mathcal{A} < \frac{n_k}{2}, \\ \mathcal{A} & \text{if } \mathcal{A} \geq \frac{n_k}{2} \end{cases}$$

where $\mathcal{A}$ is the estimated number of tuples that are less than the attribute value, X_α, and is equal to,

$$\mathcal{A} = \alpha a_k + \frac{\alpha(\alpha+1)(n_k - a_k l)}{l(l-1)}.$$

where a_k is the frequency of the first attribute value in the k^{th} sector.

Similarly, using Theorem 4.7, we obtain the following estimate for the average-case error.

$$\epsilon = \frac{(\alpha-1)(\alpha-2)(n_k - a_k l)}{l-1}$$

where a_k is the frequency of the first attribute value in the k^{th} sector.

In the next theorem, we consider the selection query, $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$, where the attribute values X_α and X_β fall within the k^{th} R-ACM sector and $\alpha < \beta$.

Theorem 4.10 *The maximum likelihood estimate of the result size for the range selection query $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$, where the attribute values X_α and X_β fall completely within the k^{th} T-ACM sector is given by,*

$$\mathcal{S}_{ML}(\sigma_{X_\alpha \leq X \leq X_\beta}(R)) = (\beta - \alpha + 1)\left\{a_k + \frac{(\beta+\alpha)(n_k - a_k l)}{l(l-1)}\right\}$$

where a_k is the frequency of the first attribute value in the k^{th} sector and $\beta > \alpha$.

Proof: The maximum likelihood value of the number of tuples in the i^{th} attribute value X_i of the k^{th} T-ACM sector is given by,

$$\hat{x}_{ML_i} = a_k + \frac{2(n_k - a_k l)}{l(l-1)}.i.$$

Hence the maximum likelihood number of tuples falling in the range of $X_\alpha \leq X \leq X_\beta$ is,

$$\begin{aligned} S_{ML}(\sigma_{X_\alpha \leq X \leq X_\beta}(R)) &= \sum_{i=\alpha}^{\beta} \left(a_k + \frac{2(n_k - a_k l)}{l(l-1)} . i \right) \\ &= (\beta - \alpha + 1) \left\{ a_k + \frac{(\beta + \alpha)(n_k - a_k l)}{l(l-1)} \right\} \end{aligned}$$

and the theorem follows. □

The following lemmas give estimates for the worst-case and average-case errors in the above select operation.

Lemma 4.14 *The worst-case error in estimating the result size of the selection query, $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$, where the attribute values X_α and X_β fall completely within the k^{th} T-ACM sector is given by,*

$$\epsilon = \begin{cases} n_k - \mathcal{A} & \text{if } \mathcal{A} < \frac{n_k}{2}, \\ \mathcal{A} & \text{if } \mathcal{A} \geq \frac{n_k}{2} \end{cases}$$

where $\mathcal{A}$ is the expected number of tuples between the attribute values X_α and X_β and is equal to,

$$\mathcal{A} = a_k(\beta - \alpha + 1) + \frac{(n_k - a_k l)(\beta - \alpha + 1)(\beta - \alpha + 2)}{l(l-1)}.$$

Proof: This is a direct consequence of Theorem 4.4. □

Lemma 4.15 *The average-case error in estimating the result size of the selection query, $\sigma_{X_\alpha \leq X \leq X_\beta}(R)$, where the attribute values X_α and X_β fall completely within the k^{th} T-ACM sector is given by,*

$$\epsilon = \frac{(\beta - \alpha)(\alpha + \beta - 3)(n_k - a_k l)}{l - 1}$$

where $\beta > \alpha$.

Proof: This is a direct consequence of Theorem 4.7. □

4.7 Size Estimation of the Join Operation Using the T-ACM

In Chapter 3, we discussed how the R-ACM can be used to estimate the result size of the join of two relations. In this section we shall see how the T-ACM can be used for the same purpose. Specifically, we shall use the maximum likelihood estimate obtained for the equality select operation in the T-ACM, to obtain an estimate for the join result size. Unlike the R-ACM, where the expected frequencies for the values within a sector are all equal, the expected frequency for every value in a T-ACM is different depending on the location of the attribute value, thus adding more complexity to the size estimation.

Let L_X and L_Y be the number of distinct attribute values in attributes X and Y respectively. Evaluating the equi-join operation, $R \bowtie_{X=Y} S$ involves finding the matching attribute value Y_q of relation S for every attribute value X_p, $1 \leq p \leq L_X$, of relation R. Since the attribute value domain in the T-ACM is already in an ascending order, this turns out to be a linear operation.

Algorithm 4.4 Locate_Matching_Tuple(X_i, j)

```
   Input:  Attribute value, X_i, from the first T-ACM and location j
      of the current pointer into the second T-ACM.
   Output:  Location, j, of the matching attribute value, Y_j,
      from the second T-ACM.
   begin
   while (X_i < V[j]) and (j ≤ L_Y) do
      j++;
   if X_i = V[j] then
      return(j);            /* location of matching tuple */
      else return(-1);      /* there is no matching tuple */
   end;
EndAlgorithm Locate_Matching_Tuple;
```

The algorithm `Locate_Matching_Tuple`(X_i), given below, returns the location of the matching attribute value in the joining T-ACM. The array $V[0 \ldots L_Y]$ maintains the value domain of the joining attribute Y. Note that the search for the matching tuple does not have to start from the first location of the T-ACM. Since the current attribute value $X_i, i > 0$, is always larger than the previous attribute value X_{i-1}, it is sufficient to begin the search for the matching tuple from where the previous search ended. Indeed, the philosophy of the traversal is analogous to *merge-sort*, where the "sortedness" of the individual values is taken advantage of.

We can now estimate the size of $R \bowtie_{X=Y} S$, where X and Y are domain compatible attributes, using the following algorithm, `Estimate_Join_Using_T-ACM`. This algorithm works in the following manner. For every attribute value, X_p, of attribute X in relation R, the matching attribute value, Y_q, of attribute Y is found using the algorithm, `Locate_Matching_Tuple`. The corresponding sector index k and the location β of the attribute value Y_q within this sector are computed, and then the estimated result size of the selection queries $E(\sigma_{X=X_p})$ and $E(\sigma_{Y=Y_q})$ are computed. The correctness of the algorithm is omitted as it is straightforward.

Algorithm 4.5 Estimate_Join_Using_T-ACM

`Input: T-ACMs for attributes` X `and` Y.
`Output: Estimated join size of the query` $R \bowtie_{X=Y} S$.
begin
$\hat{\xi}_{XY} = 0$;
for $p = 1$ **to** L_X **do**
  $j = p$ **div** l_X; `/* index of the sector where` X_p `falls */`
  $\alpha = p$ **mod** l_X; `/* location of` X_p `within the sector */`
  $\hat{\xi}_X = a_j + \frac{2(n_j - a_j l_X)}{l_X(l_X - 1)}.\alpha$;
  q =`Locate_Matching_Tuple`$(X_p, current\_ptr)$;
  $k = q$ **div** l_Y; `/* index of the sector where` Y_q `falls */`
  $\beta = q$ **mod** l_Y; `/* location of` Y_q `within the sector */`
  $\hat{\xi}_Y = a_k + \frac{2(n_k - a_k l_Y)}{l_Y(l_Y - 1)}.\beta$;
  $\hat{\xi}_{XY} = \hat{\xi}_{XY} + \hat{\xi}_X \times \hat{\xi}_Y$;
endfor;
end;
EndAlgorithm Estimate_Join_Using_T-ACM;

4.7.1 Estimation of Join Error

As in the case with the R-ACM, the estimation error resulting from the size estimation using a T-ACM of an equality join of two attributes are usually much higher than the estimation errors resulting from the equality select and range select operations. We earlier derived in Lemma 3.20 that this estimation error ϵ is equal to,

$$\epsilon = |(\hat{x}_i\epsilon_y + \hat{y}_j\epsilon_x + \epsilon_x\epsilon_y)|$$

where ϵ_x and ϵ_y are the estimated errors resulting from the equality selection queries $\sigma_{X=X_i}$ and $\sigma_{X=Y_j}$ respectively. Substituting the average-case and worst-case errors of equality selection queries that we obtained earlier for ϵ_x and ϵ_y, we can obtain the average-case and worst-case errors of equi-join estimations.

4.8 Case for ACM as a Better Estimation Tool

As discussed earlier, the current state of the art estimation techniques use either parametric techniques or sampling based techniques, including probabilistic and non-parametric techniques. In all brevity we shall now argue why the ACM is a better estimation tool for query optimization in database systems compared to the current state of the art schemes. The reader should notice that the arguments presented here are to advocate the use of *both* the R-ACM and the T-ACM over traditional methods.

4.8.1 Comparison of ACM and the Current State of the Art

Parametric vs ACM

Estimations based on ACM is superior to parametric techniques, because ACM does not make any assumptions about the underlying data distribution. Parametric techniques, on the other hand, approximate the actual data distribution by a parameterized mathematical distribution, such as the uniform distribution, multivariate distribution or Zipf distribution. Obviously the accuracy of the approximation depends heavily on the similarity between the actual and parameterized distribution.

Since real data often does not resemble any simple mathematical distribution, such approximations are prone to cause higher inaccuracies.

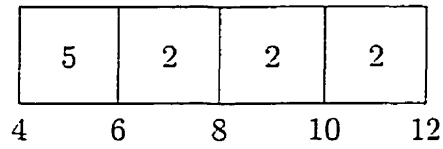
For example, System R query optimizer assumes that the data distribution is uniform over the entire value domain $\mathcal{D}$. Example 4.2 compares the selection[2] results based on the System R's uniform parametric model and that of rectangular ACM for a small relation. One can note that the estimation using the rectangular ACM is closer to the actual result, 3, since the uniformity assumption is made only within a sector.

With the Rectangular ACM, in addition to the assumption that the data distribution is uniform only within each sector, the frequency values within each sector are assured to be within a given tolerance value to the running mean. This eliminates the problem of widely different frequency values within a sector. With the trapezoidal ACM, data distribution is assumed to take a trapezoidal probability distribution, following the actual distribution very closely within each sector. In addition to our analytical results discussed in this chapter, our extensive experiments also confirm that both the rectangular and trapezoidal ACMs provide a much more accurate result size estimations both in selection and join queries, as we will see in the next chapter. Thus by appropriately choosing a suitable tolerance value in the case of R-ACM and a suitable number of sectors in the case of T-ACM, one can obtain a very close approximation to the underlying data distribution and the desired accuracy in the result estimation.

Example 4.2 *Consider a small relation $R(A, B)$ with total number of tuples, $N_R = 11$, selection cardinality of attribute A, $\varphi(A, R) = 1.38$, and number of distinct values of attribute A, $\delta(A, R) = 8$. If we estimate the number of tuples with $A = 5$, using both $\varphi(A, R)$ and its rectangular ACM, we get results as given below. Note that in this example the Rectangular ACM happens to be an equi-width rectangular ACM.*

[2]This study will be done in greater detail in the forthcoming chapter for both synthetic and real-world data.

A	B
5	31
5	39
5	42
6	37
6	39
7	12
8	13
9	12
10	39
11	53
12	59



Using $\varphi(A, R)$, the number of values when $A = 5$ is 1.38

Using the ACM, there are 5 tuples for the values of $A = 5$ and 6. Hence the expected value of $A = 5$ is $5/2 = 2.5$

Sampling vs ACM:

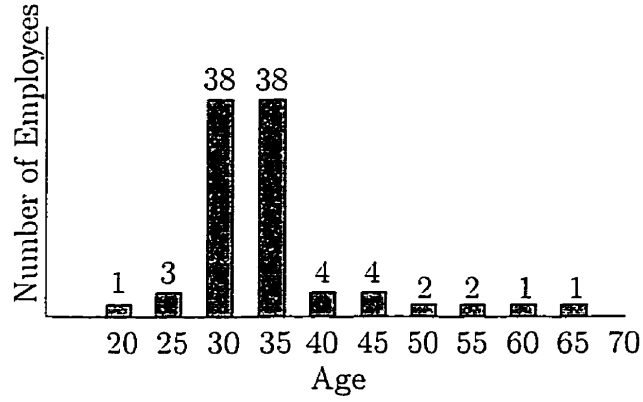
Estimations based on ACM is superior to sampling techniques. Sampling based methods are mainly designed for run time. Because of this they incur disk I/Os and CPU overheads during query optimization. Moreover the information gathered is not preserved across queries and hence they may incur the costs repetitively. The accuracy of the results mainly depends on the sampling method used and the size of the sampled population. ACM, both rectangular and trapezoidal, on the other hand, maps the entire tuples over the value range of the attribute(s) hence is much more accurate than any sampling method. Since the ACMs of the base relations are not built during run-time, they do not incur the disk I/Os and CPU overhead at the time of query optimization.

Histogram vs ACM:

As argued in the entire body of this chapter, estimation based on ACM is superior to histogram based techniques. Both R-ACM and T-ACM are improved histogram-like strategies. By using a tolerance value, the frequency values in each R-ACM sector are guaranteed to be close to the running mean frequency. Similarly, in the T-ACM, the trapezoidal sectors very closely follow the actual data distribution, so the estimation errors are minimized. In addition, histograms are usually constructed

based on sampling the data distribution, and their accuracy heavily depends on the type of sampling methods and the size of sampling population used. But ACMs are constructed to map the entire data distribution, providing further accuracy. The following example illustrates why the R-ACM is better than the equi-width histogram.

Example 4.3 *Let us consider a hypothetical distribution of attribute* age *in a relation with $N = 100$ tuples. Assume that* age *ranges from 20 to 70. Let us construct an equi-width histogram for this relation with $m_x = 10$ equi-width sectors as shown in the figure below.*



From the figure, we can observe that the selectivity $\xi_{age<37}$ (i.e: the percentage of the employees who are younger than 37) is,

$$0.42 \leq \xi_{age<37} \leq 0.80.$$

The actual fraction of tuples with age < 37 *can be anywhere between 0.42 to 0.80. If the estimate of $\xi_{age<37}$ is assumed to be the mid-point of this range (i.e: 0.61), then it is obvious that the estimate can be wrong by nearly 0.19.*

In general, the maximum error in estimating the result size of $\xi_{X<A}$, where A is a constant, is half the value of the sector in which X falls. For an unlucky distribution of attribute values (where the cell with the largest value contains almost 100% of the tuples), a selectivity estimate from an equi-width histogram can be wrong by almost 0.5. It should be noted that such a situation is very common with real world databases.

Hence from this example, we see that the way to control the maximum estimation

Histogram Type	Worst-case Error	Average-case Error
Equi-width	$\max\left(n_j - \frac{n_j}{l}, \frac{n_j}{l}\right)$	$\max\left(n_j - \frac{n_j}{l}, \frac{n_j}{l}\right)$
Equi-depth	$\frac{2n_j}{3l_j}$	$\frac{n_j}{2l_j}$
R-ACM	$\tau\left\lvert\ln\left(\frac{l_j}{i-1}\right) - 1\right\rvert$	2τ
T-ACM	$\max\left(a_j + \frac{2(n_j - a_j l)}{l(l-1)}i, n_j - a_j - \frac{2(n_j - a_j l)}{l(l-1)}i\right)$	$a_j + \frac{2(n_j - a_j l)}{l(l-1)}i - \frac{n_j}{l}$

Table 4.1: Comparison of Histogram Errors

error is to control the number of tuples (or height) in each sector so as to maintain the frequencies of every value close to a certain height. This can be easily achieved by choosing a suitable tolerance value and forcing the values in every sector to not exceed it, as is done in the rectangular ACM. □

We compare the worst-case and average-case estimation errors of the traditional equi-width, equi-depth histograms and the new histogram-like techniques proposed here in Table 4.1.

4.9 Maintaining ACMs in the DBMS Catalogue

A major drawback of traditional histograms is that they are mostly generated at run-time during query optimization incurring a high I/O cost. We believe that one of the practical advantages of ACMs is that both the types of ACMs defined in this chapter can be easily created and maintained in the DBMS catalogue with only a minimal amount storage requirement. Looking from an implementation point of view, storing an integer array with a few hundred or thousand entries require only a few kilobytes of disk storage. Unlike the late 1970s and early 80s when most of the database technology was developed when the disk storage was considered to be a premium, the 1990s have seen technology yielding unprecedentedly huge storage capacities along with a dramatic drop in the cost of disk storage.

Since a relation with several million tuples can be mapped to an ACM with a few hundred or few thousand entries, and considering that our current technology has made very large capacity and low cost disks possible, ACMs for an entire DBMS can be easily materialized in the catalogue. Even for a complex database, with 100 relations and say 500-1000 distinct attributes, the cost of storing the ACMs would be less than one megabyte, which is less than 0.01% of the size of a typically large database with ten gigabytes of data.

In most commercial database systems, the DBMS catalogue is designed as part of the query optimizer module. Thus storing ACMs in the DBMS catalogue is a natural choice as it would reduce the communication costs. For the intermediate relations resulting from relational operations, the query optimizer can construct the ACMs in the main memory for further optimization.

4.10 Experiments Using Synthetic Data

In this section, we provide a number of experimental results on the T-ACM using *synthetic data*. These experimental results demonstrate the validity of the theoretical results that were presented throughout this chapter, and also prove the superiority of the T-ACM over the traditional equi-width and equi-depth histograms. We shall present a more extensive set of benchmarking experiments on *real-world data*, including a set of experiment using the TPC-D queries and database, in the next chapter.

We conducted three distinct set of experiments on the T-ACM as follows. In the first set of experiments, we computed the actual estimation errors of both the selection and join operations using the T-ACM on various synthetic data distributions such as:

1. uniform data distribution,
2. Zipf data distribution, and
3. multi-fractal distribution.

In the second set of experiments, we computed the variance of the T-ACM and the corresponding percentage estimation errors for various equi-select, range-select and equi-join operations.

The scope of the third set of experiments included a performance comparison study of the T-ACM and the traditional equi-width and equi-depth histograms. As in the case of the experiments with the R-ACM, we conducted this set of experiments with various synthetic data distributions for both select and join operations.

4.10.1 Queries Used in the Experiments

As in the case of the experiments with the R-ACM, the following experiments on the T-ACM, included queries that use both the select and join operations.

For estimating the result sizes of select operations, we actually used two types of select operations, namely the *exact-match select* and the *range select.* The exact-match select operation, denoted $\sigma_{X=X_i}(R)$, retrieves all the tuples from the relation R, for which the attribute X has the value X_i. The range select operation retrieves all the tuples falling within an attribute value range. For example, the query $\sigma_{X \leq X_i}(R)$, retrieves all the tuples from the relation R, for which the attribute value X has values less than X_i. For the join operation, we used the most frequently encountered equi-join operation. The equi-join operation, denoted $R \bowtie_{X=Y} S$, combines all the tuples in the relations R and S whenever the value of attribute X from relation R is equal to the value of attribute Y from relation S.

4.10.2 Estimation Accuracy of the T-ACM under Various Synthetic Data Distributions

In this set of experiments, we computed the relative estimation errors for the selection and join operations under the above mentioned data distributions. The relative estimation error was obtained as a ratio by subtracting the estimated size from the actual result size and dividing it by the actual result size. Obviously, the cases where the actual result sizes were zero were not considered for error estimation. The results were obtained by averaging the estimation errors over a number of experiments and are shown in Tables 4.2, 4.3, and 4.4 for the three different frequency distributions. A uniform attribute value domain was consistently used for this group of experiments.

Operation	Actual Size	Estimated Size	Percentage Error
Equi-select	107	112.15	4.81%
Range-select	4028	3948.64	1.97%
Equi-join	395602	435795.16	10.16%

Table 4.2: Estimation Accuracy of the T-ACM under Uniform Distribution

Operation	Actual Size	Estimated Size	Percentage Error
Equi-select	413	450.99	9.20%
Range-select	1607	1693.62	5.39%
Equi-join	298710	368667.88	23.42%

Table 4.3: Estimation Accuracy of the T-ACM under Zipf Distribution

4.10.3 Estimation Accuracy and Variance of the T-ACM

We computed the variance of the T-ACM under the uniform frequency distribution for equality-select, range-select and equi-join operations and the corresponding percentage estimation errors in this set of experiments. The percentage estimation errors and the corresponding variance of the T-ACM are given in Table 4.5. Note that the row numbers I, II, and III correspond to equality-select, range-select and equi-join operations respectively.

Operation	Actual Size	Estimated Size	Percentage Error
Equi-select	98	104.24	6.37%
Range-select	1813	1893.68	4.45%
Equi-join	480662	563960.72	17.33%

Table 4.4: Estimation Accuracy of the T-ACM under Multi-fractal Distribution

No	Size	Estimated Result			Percentage Error		
		$\mathcal{V} = 1007$	$\mathcal{V} = 1196$	$\mathcal{V} = 1493$	$\mathcal{V} = 1007$	$\mathcal{V} = 1196$	$\mathcal{V} = 1493$
I	72	74.36	75.60	78.66	3.28%	4.99%	9.25%
II	318	297.55	343.79	360.90	6.43%	8.11%	13.49%
III	163	171.67	175.06	323.72	5.32%	7.40%	9.86%

Table 4.5: Variance of the T-ACM and the Estimation Errors, where I, II, and III denote the equi-select, range-select and equi-join operations respectively.

Operation	Actual Size	Equi-width		Equi-depth		T-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	491	608.00	23.83%	599.91	22.18%	469.74	4.33%
Range-select	20392	22251	9.12%	21487	5.37%	21054	3.25%
Equi-join	489074	627482	28.3%	591779	21.0%	534362	9.26%

Table 4.6: Comparison of Equi-width, Equi-depth and T-ACM: Uniform Frequency Distribution

4.10.4 T-ACM and the Traditional Histograms

This group of experiments were conducted on both the traditional equi-width and equi-depth histograms and the T-ACM. In order to provide a fair comparison, we used a fixed amount of storage for all three techniques, thus varying the build parameters for the structures as required. The build parameter for both the equi-width histogram and the T-ACM is the sector width, whereas the build parameter for the equi-depth histogram is the number of tuples within the sector. As before we computed the percentage estimation errors for the three type of queries, namely, (a) equality-select (b) range-select and (c) equi-join. The experiments were again conducted for the uniform, Zipf and multi-fractal frequency distributions. An analysis of the results follows.

Operation	Actual Size	Equi-width		Equi-depth		T-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	191	305.22	59.8%	287,26	50.4%	218.35	14.32%
Range-select	14560	18389.3	26.3%	17457.4	19.9%	15619.9	7.28%
Equi-join	125606	210264	67.4%	201221	60.2%	153779	22.43%

Table 4.7: Comparison of Equi-width, Equi-depth and T-ACM: Zipf Frequency Distribution

Operation	Actual Size	Equi-width		Equi-depth		T-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	266	340.75	28.1%	330.64	24.3%	284.08	6.80%
Range-select	24091	26668	10.7%	26066	8.2%	25091	4.15%
Equi-join	398440	574550	44.2%	545863	37.0%	452508	13.57%

Table 4.8: Comparison of Equi-width, Equi-depth and T-ACM: Multi-fractal Frequency Distribution

4.10.5 Analysis of the Results

As in the case of the R-ACM, the results from the first group of experiments show that the estimation errors with the uniform data distribution are smaller than the corresponding errors with the Zipf and multi-fractal distributions. For example, the percentage estimation error for the equi-select operation using a uniform distribution is 4.81% (Table 4.2), whereas the percentage estimation errors for the same operation under the Zipf and multi-fractal distributions are 9.20% (Table 4.3) and 6.37% (Table 4.4) respectively. This is obviously due to the fact that extreme frequency variations within the sectors of uniform data distribution are much lower than that with the Zipf and multi-fractal distributions. The frequency variations are the largest with the Zipf distribution, and consequently the estimation errors from the T-ACM are proportionately larger than that with other distributions.

The results from the second set of experiments show that the estimation accuracy is inversely proportional to the variance of the T-ACM for all three types of query

operations considered. For example, considering the range-select query in row II of Table 4.5, we see that the percentage estimation error from the T-ACM with the variance of $\mathcal{V} = 1007$ is only 6.43%, whereas the percentage estimation error from the T-ACM with the variance of $\mathcal{V} = 1493$ is equal to 13.49%, which is more than a 100% increase!

The result from the third group of experiments show that the estimation error resulting from the T-ACM is **consistently** much smaller than the estimation errors from the equi-width and the equi-depth histograms. For example, from Table 4.6, the percentage estimation error with the T-ACM for equi-select operation on uniform frequency distribution is only 4.33%, whereas for the same operation, the equi-width and equi-depth histograms result in 23.83% and 22.18% estimation errors. This demonstrates that the estimation errors from the T-ACM are indeed a fraction of the estimation errors from the equi-width and equi-depth histograms and proves the superiority of the T-ACM over the traditional histograms. This behavior can be observed with both the Zipf and multi-fractal distributions as well, and can be seen from the Tables 4.7 and 4.8. This is consequent to the fact that by rendering a trapezoidal approximation valid, the slope of the trapezoid would lead to a structure superior to the traditional equi-width and equi-depth histograms where the slope has no bearing. Furthermore the experimental results show that the degree of approximation is much better with the T-ACM than the other techniques, since the average error (error averaged over the entire sector) can be arbitrarily small. The superiority of the T-ACM over the traditional histograms is obvious!

4.11 Summary of Work Done

In this chapter we have introduced a new histogram-like approximation strategy, called the Trapezoidal Attribute Cardinality Map, for query result size estimation. Since this technique is based on the philosophy of numerical integration, it is much more accurate than the traditional histograms. As in the case with the R-ACM which we discussed in Chapter 3, by proving a Binomial distribution (whose parameters vary with the location of the attribute value) to represent frequency variations within

sectors, we have provided theoretical results to compare the accuracy of the T-ACM to that of the traditional histograms, both in the average-case and worst-case. As in the case of the R-ACM, we have provided a rigorous maximum likelihood analysis, an expected-case analysis of the variance, and the resulting worst-case and average-case errors to support our argument that the T-ACM can indeed be used as a fundamental tool for query result size estimation.

In addition, we have also conducted extensive experiments using synthetic data to support the validity of our theoretical results. Although we discussed a few simple methods to obtain the starting frequency for building the T-ACM, the problem of finding an optimal starting frequency for a given data distribution remains open.

The theoretical results from this chapter are currently being reviewed for publication. Some of the results from a set of prototype validating experiments conducted on real-world data have been published in the *Proceedings of the ICEIS'99 Conference* held in Setubal, Portugal in March, 1999 [133].

In summary, due to its superiority over the traditional histograms in query result size estimation and its relatively low construction costs, we hope that the T-ACM (along with the R-ACM which we discussed in Chapter 3) could prove to be a standard tool for query optimization in future database systems.

Chapter 5

Prototype Validation of the Attribute Cardinality Maps[1]

5.1 Introduction

Modern-day database management systems (DBMS) provide competitive advantage to businesses by allowing quick determination of answers to business questions. Intensifying competition in the business marketplace continues to increase the sizes of databases as well as the level of sophistication of queries against them. This has resulted in a greater focus (both academically and in the industrial world) to develop systems with superior DBMS functionalities that would, in turn, minimize the response times for business and other queries. Current business database systems utilize histograms to approximate frequency distributions of attribute values of relations. These are used to efficiently estimate query result sizes and access plan costs and thus minimize the query response time for business (and non-commercial) database systems. In Chapter 3 and Chapter 4, we proposed two new forms of histogram-like techniques called the Rectangular and Trapezoidal Attribute Cardinality Maps respectively that give much smaller estimation errors than the traditional equi-width

[1]Some preliminary results about the work on the prototype validation of the R-ACM from this chapter have been published in the *Proceedings of the International Conference on Business Information Systems (BIS'99)*, Poznan, Poland, April, 1999 [132]. Similarly, some preliminary results about the work on the prototype validation of the T-ACM from this chapter have been published in the *Proceedings of the International Conference on Enterprise Information Systems (ICEIS'99)*, Setubal, Portugal, March, 1999 [133].

and equi-depth histograms currently being used by many commercial database systems. We also provided a fairly extensive mathematical analysis for the average and worst case errors of the R-ACM and T-ACM for their frequency estimates which was verified for synthetic data.

The goal of this chapter is to demonstrate the power of these ACMs for query result size estimation by extensive experiments on real-world data (described presently) using *synthetic* queries. The aim of this exercise is to justify the claim of our analytical results that the ACMs are superior to the current state-of-the-art techniques used in the commercial database systems. Specifically, we provide prototype validations of the Attribute Cardinality Maps for query optimization in two popular real-world databases, namely, the U.S. CENSUS database and the NBA Player Statistics database.

First of all we briefly describe the real-world data distributions and the queries used in our experiments. We then provide a number of prototype validation results that have been obtained using the R-ACM and the T-ACM with the above set of databases and some synthetic queries. This is followed by an analysis of the experimental results.

In the above experiments we also make use of the well-known Zipf distributions and multi-fractal distributions[2] to generate frequencies for the attribute values from real-world databases. As mentioned earlier, these mathematical distributions are so powerful that when used in conjunction with real-world attribute value domains, they provide "infinite" possibilities of "simulated" real-world data distributions for our experiments.

5.2 Databases Used in the Experiments

Because synthetic data is usually generated as random numbers or on the basis of some mathematical distributions, it is impossible to simulate real-world data distributions using synthetic data. Consequently, we have resorted to conduct our experiments

[2]We refer the reader to Sections §3.10.1 and §3.10.2 where we earlier gave a brief overview of both the Zipf and multi-fractal distributions.

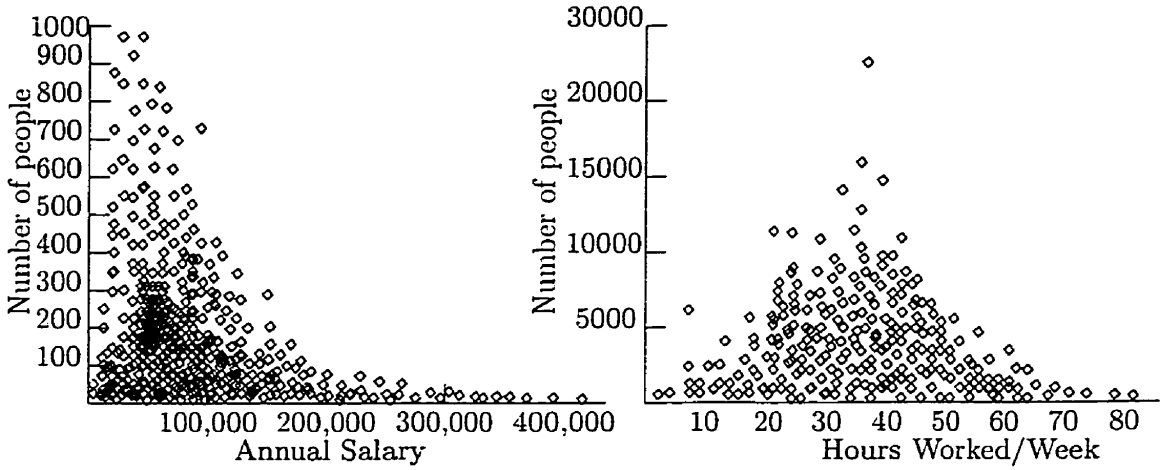


Figure 5.1: Frequency Distributions of Selected Attributes from the U.S. CENSUS.

on two real-world databases, namely, the United States CENSUS database and the database on the performance statistics of NBA players.

5.2.1 U.S. CENSUS Database

The CENSUS database contains information about households and persons in the United States for the years 1993 to 1995 [135]. Most of the relations in this database contains hundreds of attributes, both scalar-typed (such as a person's sex and type of job) and numerical (such as salary and age). The Data Extraction System (DES) at the U.S. Census Bureau allows extracting records and fields from very large public information archives such as governmental surveys and census records. The DES produces custom extracts in selectable data file formats that can be later analyzed by statistical packages.

Tables 5.1 and 5.2 describe the set of relations and attributes chosen from this database for our experiments. The data distributions of some of the selected attributes from CENSUS are plotted in Figure 5.1. The queries for our experiments consisted of either (a) equality join (b) equality selection or (c) range selection operators.

Relation Name	No of Tuples	Description
cpsm93p	155197	Population survey for 1993 - Person
cpsm94p_1	83455	Population survey for 1994 (Set 1) - Person
cpsm94p_2	150943	Population survey for 1994 (Set 2) - Person
cpsm95f	63756	Population survey for 1995 - Family
cpsm95h	72152	Population survey for 1995 - Household
pums905h	828932	Decennial Census Microdata Samples

Table 5.1: Relations in the CENSUS Database

Relation	Attribute	No of Distinct Values	Description
cpsm93p	hours	95	No of hours worked/week
	industry	48	Industry code
	wages	31321	Person's wages
cpsm94p	income	8143	Total income
cpsm94p_2	age	91	age
	hours	100	Hours worked/week
cpsm95f	income	32026	Annual income
	persons	15	No of persons/family
cpsm95h	state	51	State code
	wages	10496	Total wages
pums905h	wages	34668	Total wages

Table 5.2: Attributes in the CENSUS Database

5.2.2 NBA Performance Statistics Database

Due to the unusually high interest exhibited by the American basketball fans, there is a proportionally large amount of statistical analysis done on the basket ball players and games, especially in the news media. These statistics have been compiled by various people for various reasons, such as prediction of player behaviors etc. The database on the NBA players that we use for our experiments is a performance statistics of NBA players for the year 1991-92 [100]. This database contains a relation with 50 attributes and 458 tuples. We have given the distributions of a few attributes from this relation in Figure 5.2.

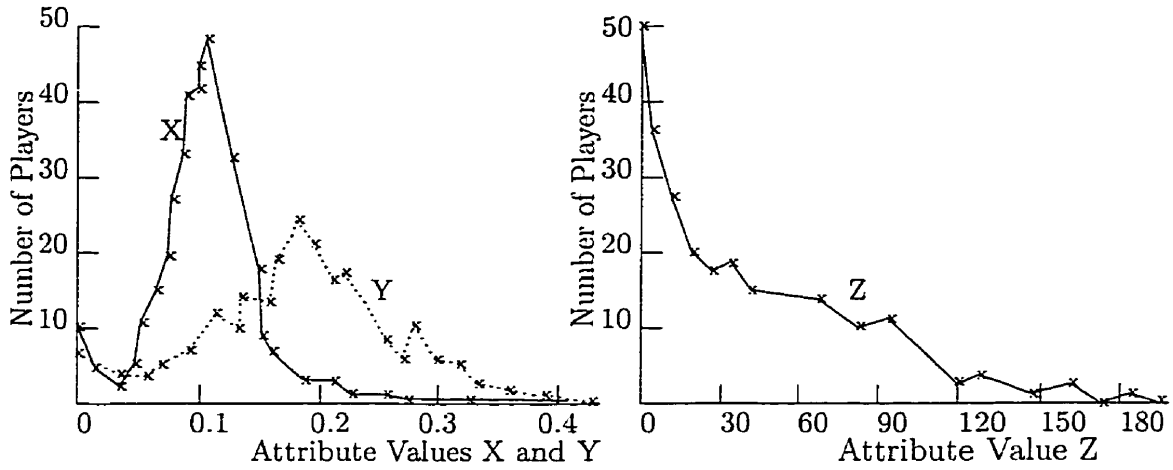


Figure 5.2: Frequency Distributions of Selected Attributes from the NBA Statistics.

5.3 Queries Used in the Experiments

Since the *select* and *join* operations are the two most frequently used relational operations in database systems, we used only those queries consisting of these two operations for our experiments.

For estimating the result sizes of select operations, we essentially used two types of select operations, namely the *exact-match select* and the *range select*. The exact-match select operation, denoted $\sigma_{X=X_i}(R)$, retrieves all the tuples from the relation R, for which the attribute X has the value X_i. The range select operation retrieves all the tuples falling within an attribute value range. For example, the query $\sigma_{X\leq X_i}(R)$, retrieves all the tuples from the relation R, for which the attribute value X has values less than X_i. For the join operation, we used the most frequently encountered equi-join operation. The equi-join operation, denoted $R \bowtie_{X=Y} S$, combines all the tuples in the relations R and S whenever the value of attribute X from relation R is equal to the value of attribute Y from relation S.

All the queries used in these set of experiments were synthetic queries, involving only one of the above elementary relational operations at a time. The attribute values for the matching conditions in the queries were generated for the specific attribute value domain using a random number generator.

Having described the database sets and the query patterns to be used in our

experiments, we shall now discuss the actual prototype validation on the R-ACM and T-ACM in the following sections.

5.4 Prototype Validation of the R-ACM

In this section, we describe in detail the prototype validating experiments that we conducted on the R-ACM using the real-world databases described earlier. Before proceeding, we emphasize that even though the following experiments were carried out on typical real-world databases, the query sets used were synthetic ones involving the elementary relational operations described earlier.

5.4.1 Experiments on U.S. CENSUS Database

The prototype validation of the R-ACM on the U.S. CENSUS database involved three distinct sets of experiments detailed below. The rationale for these experiments is to provide a performance comparison of the R-ACM and the traditional histograms, as well as to study its behavior under various distribution criteria and tolerance values.

The first group of experiments were conducted on equi-width, equi-depth histograms and the R-ACM. In each of our experimental runs, we chose different build-parameters for the histograms and the R-ACM. The build-parameters for the equi-width and equi-depth histograms are the sector width and the number of tuples within a sector respectively. The build-parameter for the R-ACM is the tolerance value, τ.

We obtained the relative estimation error as a ratio by subtracting the estimated size from the actual result size and dividing it by the actual result size. Obviously, the cases where the actual result sizes were zero were not considered for error estimation. We implemented a simple query processor to compute the actual result size. The results were obtained by averaging the estimation errors over a number of experiments and are shown in Table 5.3.

In the second group of experiments, we generated frequencies using the Zipf and multi-fractal distributions and applied them randomly to the value domains of the relations from the CENSUS database. Again the results were obtained by averaging

Operation	Actual Size	Equi-width		Equi-depth		R-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	1796	1279.1	28.8%	1365.6	23.4%	1702.3	5.23%
Range-select	32109	30008.3	6.5%	31214.9	2.8%	32319.2	0.65%
Equi-join	720988	543908	24.6%	610482	15.3%	660183	8.43%

Table 5.3: Comparison of Equi-width, Equi-depth Histograms and R-ACM: U.S. CENSUS Database

Operation	Actual Size	Equi-width		Equi-depth		R-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	932	1182.5	26.9%	1127.1	20.9%	989.4	6.12%
Range-select	27180	29028.2	6.8%	28049.8	3.2%	27533.3	1.30%
Equi-join	589066	743990	26.3%	688029	16.8%	640904	8.80%

Table 5.4: Comparison of Equi-width, Equi-depth Histograms and R-ACM: U.S. CENSUS Database, using frequencies from the Zipf and Multifractal distributions.

the estimation errors over a number of experiments and are shown in Table 5.4.

In the third group of experiments, we compared the result estimates from the R-ACM for three different tolerance values. Again we considered (a) exact match select queries (b) range select queries and (c) equi-join queries and obtained the average estimation errors by comparing the estimates to the actual result sizes. The results are given in Table 5.5. We also repeated these experiments by applying the frequency values generated by the Zipf and multifractal distributions to the attribute value domains of the CENSUS database. The results of these experiments are given in Table 5.6.

Analysis of the Results

The results from the first two sets of experiments show that the estimation error resulting from the R-ACM is **consistently** much lower than the estimation error from the equi-width and equi-depth histograms. This is consequent to the fact the

Operation	Actual Size	Estimated Result			Percentage Error		
		$\tau = 4$	$\tau = 6$	$\tau = 8$	$\tau = 4$	$\tau = 6$	$\tau = 8$
Equi-select	1435	1338	1292	1143	6.75%	9.97%	20.34%
Range-select	26780	26219	24918	22098	2.09%	6.95%	17.48%
Equi-join	563912	610180	680953	719320	8.2%	20.8%	27.56%

Table 5.5: Result Estimation Using R-ACM: Data - U.S. CENSUS Database

frequency distribution of an attribute value within an R-ACM is guaranteed to be close to the sector mean since the partitioning of the sectors is based on a user specified tolerance value and the deviation from the running sector mean. Thus we see that in the equi-select operation in Table 5.3, the percentage estimation error from the R-ACM is only 5.23%, but that of the equi-width and equi-depth histograms are 28.8% and 23.4% respectively, demonstrating an order of magnitude of superior performance. Such results are typical with both synthetic data and real-world data. The power of the R-ACM is obvious!

The Third set of experiments illustrate that the accuracy of the R-ACM is inversely proportional to the tolerance value. Since smaller tolerance value results in a proportionally larger number of sectors in the R-ACM, there is obviously a trade-off of estimation accuracy and the storage requirements of the R-ACM. From Table 5.5, we see that for the tolerance value $\tau = 4$, the percentage error for the range-select query is only 2.09% whereas when the the tolerance value is increased to $\tau = 8$, the percentage error for the same operation becomes 17.48%. Such results are again typical.

Our results from these three sets of experiments confirm our theoretical results, summarized in Table 4.1, and clearly demonstrate that the estimation accuracy of the R-ACM is superior to that of the traditional equi-width and equi-depth histograms.

Estimation Errors and Variance of the R-ACM

The variance of the R-ACM was shown in Chapter 3 as $Var(R\text{-}ACM) = N - \sum_{j=1}^{s} \frac{n_j}{l_j}$, where N is the total number of tuples mapped by the R-ACM, n_j is the number of

Operation	Actual Size	Estimated Result			Percentage Error		
		$\tau = 4$	$\tau = 6$	$\tau = 8$	$\tau = 4$	$\tau = 6$	$\tau = 8$
Equi-select	2018	2121	2229	2443	5.12%	10.46%	21.08%
Range-select	39076	39849	41569	45054	1.98%	6.38%	15.30%
Equi-join	790528	866418	943495	997804	9.6%	19.35%	26.22%

Table 5.6: Result Estimation Using R-ACM: Data - U.S. CENSUS Database, using frequencies from the Zipf and Multifractal distributions for different tolerance values.

tuples within the j^{th} R-ACM sector and l_j is the sector width of the j^{th} R-ACM sector. One of our major results in Section 3.5.2 is that the R-ACM with smaller variance produces better query result size estimations. To demonstrate this results, we conducted a number of experiments on the U.S. CENSUS database for various attribute values and computed the variances of the R-ACM and the estimation errors. The errors between the estimated and actual size of random equality select queries are plotted against the computed variance of the ACM, and shown in Figure 5.3. These results, in addition to confirming the relationship between the variance of an R-ACM and the estimation errors, also confirm that the R-ACMs with lower tolerance values produce lower estimation errors.

5.4.2 Experiments on NBA Performance Statistics Database

Our prototype validation on the NBA Performance Statistics database involved two different sets of experiments. The first set of experiments were conducted to compare the performance of the R-ACM to that of the traditional histograms. As opposed to this, the second set of experiments were conducted to study the behavior of the R-ACM under various tolerance values. These experiments are discussed in detail below.

In the first group of experiments, we compared the result estimations from equi-width, equi-depth, and the R-ACM. We considered (a) exact match select queries (b) range select queries and (c) equi-join queries. For the join queries, since the NBA database consists of a single relation, we generated various frequency values with

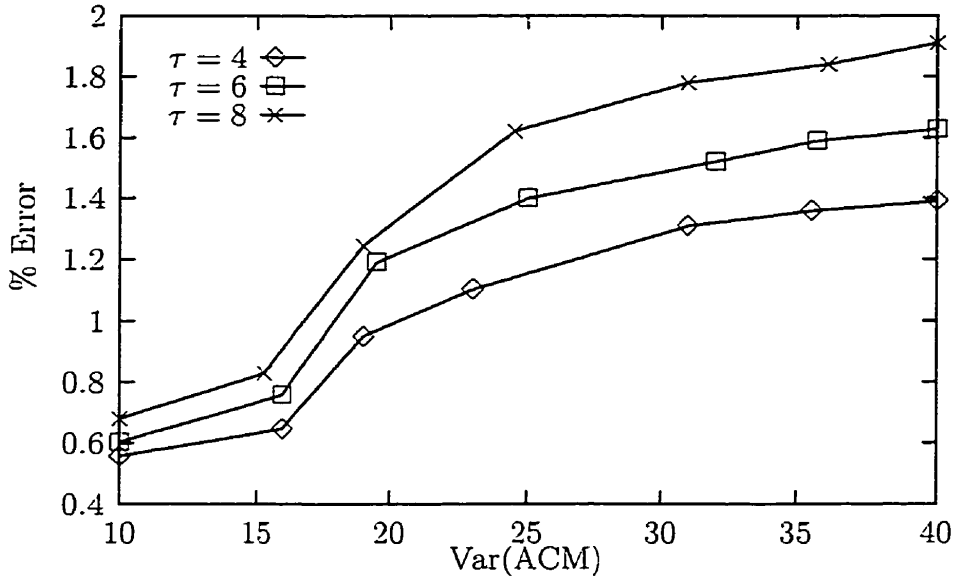


Figure 5.3: Estimation Error Vs Variance of the R-ACM: U.S. CENSUS Database

Operation	Result Size	Equi-width		Equi-depth		R-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	38	23.3	38.68%	26.5	30.26%	36.2	4.7%
Range-select	119	111.0	6.7%	113.7	4.4%	116.4	2.18%
Equi-join	242	192.8	20.33%	204.1	15.7%	249.5	3.09%

Table 5.7: Comparison of Equi-width, Equi-depth and R-ACM: NBA Statistics 1991/92

the Zipf and multifractal distributions using the same value domain of the joining attribute from the NBA relation as the second relation and joined it with the corresponding attribute from the NBA relation. As in the case with the experiment on the CENSUS database, for each of our experimental runs, we chose different build-parameters for the histograms and the R-ACM.

As before we obtained the relative estimation error as a ratio by subtracting the estimated size from the actual result size and dividing it by the actual result size, ignoring the cases where the actual result sizes were zero. Actual query result sizes were computed using a simple query processor. The results were obtained by

Operation	Result Size	Estimated Result			Percentage Error		
		$\tau = 4$	$\tau = 6$	$\tau = 8$	$\tau = 4$	$\tau = 6$	$\tau = 8$
Equi-select	42.3	39.8	37.6	35.0	5.9%	11.11%	17.26%
Range-select	132.0	129.8	126.6	124.7	1.67%	4.09%	5.53%
Equi-join	318.2	301.8	285.6	264.1	5.15%	10.24%	17.00%

Table 5.8: Result Estimation Using R-ACM: Data - NBA Statistics 1991-92

averaging the estimation errors over a number of experiments and are shown in Table 5.7.

In the second group of experiments, we compared the result estimates from the R-ACM for three different tolerance values. As in the case with the U.S. CENSUS database, we considered (a) exact match select queries (b) range select queries and (c) equi-join queries and obtained the average estimation errors by comparing the estimates to the actual result sizes. The results are given in Table 5.8.

Analysis of the Results

As in the case of the U.S. CENSUS database, the results from the first set of experiments show that the estimation error resulting from the R-ACM is a fraction of the estimation error from the equi-width and equi-depth histograms, again demonstrating the superiority of the R-ACM over the traditional histograms for query result size estimation. As we can see the percentage estimation error with the R-ACM for equi-select operation is only 4.7%, whereas for the same operation, the equi-width and equi-depth histograms result in 38.68% and 30.26% estimation errors - which is an order of magnitude larger!

As before, we see that our second set of experiments show that the accuracy of the R-ACM is inversely proportional to the tolerance value. The choice of the tolerance value is usually left to the database administrator as he/she would have the knowledge of how much trade-off the database system can afford in terms of the disk storage to achieve the desired response time for queries. In our experiment with the NBA database, we see that the R-ACM with the lowest tolerance value ($\tau = 4$)

gives the smallest estimation error, 5.9% for the equi-select operation. When the tolerance value is increased to $\tau = 6$, we see that the estimation error for the equi-select operation is much higher at 11.11% which increases to 17% when $\tau = 8$. Such results are typical.

5.5 Prototype Validation of the T-ACM

In the last section, we discussed the prototype validation of the R-ACM. In this section, we discuss the prototype validating experiments that we conducted on the T-ACM. As in the case of the experiments with the R-ACM, these prototype validating experiments also make use of the two real-world databases that were described earlier. We also reiterate that even though the frequency distributions used were from actual real-world databases, the query sets used for the experiments were all synthetic queries involving elementary relational operations. Before proceeding, we first present an implementation scheme for the T-ACM that dramatically improves the estimation accuracy using a small computational overhead for preprocessing, but not involving any additional *storage* overhead. As we shall see, this performance increase is achieved by computing the frequencies of the middle attribute values in each sector using the actual number of tuples contained in the sectors.

5.5.1 Implementing T-ACMs for the Experiments

Even though the algorithm, `Generate_T-ACM` (See Section 4.1.1), explains how to generate a T-ACM corresponding to the given frequency domain, there are a few implementation "tricks" which can be used to dramatically improve the computation accuracy in the actual experiments. The following algorithm `Implement_T-ACM` describes this implementation process.

Let us assume that T_A is the T-ACM derived from the equi-width histogram, H_A. Also assume that the frequencies of the starting attribute value of *every second* histogram sector are available. Observe that this can be computed in $O(s)$ time (as opposed to $O(L)$), where s is the number of sectors. Then we can generate a T-ACM

which has a sector width that is *half* of the sector width of H_A and is much more superior than the initial T-ACM, T_A. The strategy to achieve this is given in the algorithm, Implement_T-ACM.

Algorithm 5.6 Implement_T-ACM

Input: (i) Equi-width histogram H_A with sector width l.
(ii) Starting frequencies of every 2^{nd} sector.

Output: T-ACM with sector width $l/2$.

begin

Merge every two adjacent sectors of H_A to get H_B;
Generate T-ACM, T_A from H_B.
Estimate frequencies of T_A's middle attribute values.
Generate T_B from T_A using frequencies obtained from last step.
Estimate frequencies of T_B's middle attribute values.
Generate T_C from T_B using frequencies obtained from last step.

end;

EndAlgorithm Implement_T-ACM

Since the trapezoidal rule of numerical integration is more accurate than the left-end or right-end rectangular rule of numerical integration, it is obvious that by virtue of the construction of the T-ACM, T_A is more accurate than the equi-width histogram H_B. We note that the area of a sector in T_A may not be exactly equal to the actual number of tuples falling in that sector. Hence, using the actual number of tuples within the sectors and computing the frequency of the middle attribute value in the sector, we can partition the sectors to obtain the T-ACM, T_B, where the sector areas represent the *actual* number of tuples more accurately. Using the same arguments, we obtain the T-ACM, T_C, from T_B. The T-ACM, T_C, can be expected to be more accurate than the T-ACM, T_B, as its boundary frequencies are more accurate estimates based on the *actual number* of tuples falling within the sectors.

Thus, by invoking a small preprocessing step, we have generated a T-ACM that is much more accurate than the original T-ACM derived directly from the corresponding histogram. An example highlighting the steps taken by the above algorithm is shown

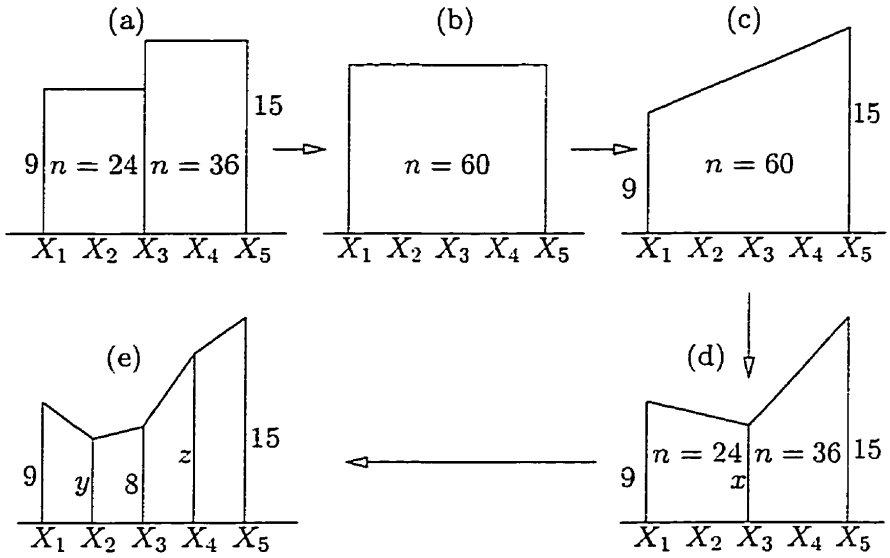


Figure 5.4: Construction of a T-ACM

in Figure 5.4.

Example 5.1 *In this example, the input to the algorithm is an equi-width histogram with two sectors as shown in Figure 5.4(a). The number of tuples in the sectors are* $n = 24$ *and* $n = 36$ *respectively. Also the starting frequency value of the first sector is 9 and the terminal frequency of the second sector is 15. The first step of the algorithm merges these two sectors to create a single histogram sector shown in Figure 5.4(b). The next step generates a trapezoidal sector equivalent to this larger histogram sector. Since the trapezoidal sector is only an approximation of the number of tuples represented by the rectangular sector, its area may not reflect the actual number of tuples* ($n = 60$) *falling within that sector. Hence in the next step of the algorithm, we* estimate *the middle frequency (i.e:* $x_3 = 8$*) by considering the total number of tuples in that sector. The resulting T-ACM sectors are shown in Figure 5.4(d). Since the actual number of tuples contained in these two T-ACM sectors are already known (they are* $n = 24$ *and* $n = 36$*), the next step of the algorithm, using the number of tuples within the sectors, estimate the middle frequencies of these two sectors. The result is the T-ACM shown in Figure 5.4(e).*

Operation	Actual Size	Equi-width		Equi-depth		T-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	1538	1138	25.98%	1198	22.06%	1476	3.98%
Range-select	29198	27178	6.92%	27931	4.34%	28673	1.80%
Equi-join	687962	532765	22.56%	562967	18.17%	612891	10.91%

Table 5.9: Comparison of Equi-width, Equi-depth and T-ACM: U.S. CENSUS Database.

5.5.2 Experiments on the U.S. CENSUS Database

Our experiments on the T-ACM using the U.S. CENSUS database consisted of three different groups. The first group of experiments were conducted as a performance comparison study of the traditional histograms and the T-ACM, using the row attribute values from the U.S. CENSUS database. The second group of experiments involved the use of Zipf and multifractal distribution with the U.S. CENSUS database and were conducted to study the behavior of the histograms and the T-ACM under various data distributions. The third group of experiments were conducted to verify the variance of the T-ACM and the estimation accuracy. We discuss each of these experiments in detail below.

The first group of experiments were conducted on equi-width, equi-depth histograms and the T-ACM. We chose different build-parameters for the histograms and the T-ACM in each of our experimental runs. The build-parameter for the equi-width histogram and the T-ACM is the sector width. The build-parameter for the equi-depth histogram is the number of tuples within a sector.

As in the case of the R-ACM, we obtained the relative estimation error as a ratio by subtracting the estimated size from the actual result size and dividing it by the actual result size. Obviously, the cases where the actual result sizes were zero were not considered for error estimation. We implemented a simple query processor to compute the actual result size. The results were obtained by averaging the estimation errors over a number of experiments and are summarized in Table 5.9.

The second group of experiments involved using the frequencies generated from the

Operation	Actual Size	Equi-width		Equi-depth		T-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	364	478	31.46%	466	28.12%	391	7.49%
Range-select	18712	20079	7.31%	19799	5.81%	19144	2.31%
Equi-join	528960	735202	38.99%	722401	36.57%	588521	11.26%

Table 5.10: Comparison of Equi-width, Equi-depth and T-ACM: U.S. CENSUS Database, using frequencies from the Zipf and Multifractal distributions.

No	Size	Estimated Result			Percentage Error		
		$\mathcal{V} = 1007$	$\mathcal{V} = 1196$	$\mathcal{V} = 1493$	$\mathcal{V} = 1007$	$\mathcal{V} = 1196$	$\mathcal{V} = 1493$
I	72	74.36	75.60	78.66	3.28%	4.99%	9.25%
II	318	297.55	343.79	360.90	6.43%	8.11%	13.49%
III	163	171.67	175.06	323.72	5.32%	7.40%	9.86%

Table 5.11: Variance of the T-ACM and the Estimation Errors: U.S. CENSUS Database

Zipf and the multifractal distributions. These frequency values were applied randomly to the value domains of the relations from the CENSUS database to generate a wide range of different data distributions. As before we considered (a) exact match select queries (b) range select queries and (c) equi-join queries. Again the results were obtained by averaging the estimation errors over a number of experiments and are shown in Table 5.10.

In the third group of experiments, we computed the variance of the T-ACM under different build-parameters for equality-select, range-select and equi-join operations and the corresponding percentage estimation errors in this set of experiments. The average percentage estimation errors and the corresponding variance of the T-ACM are given in Table 5.11. Note that the row numbers I, II, and III correspond to equality-select, range-select and equi-join operations respectively.

Analysis of the Results

The prototype validating experiment results using the U.S. CENSUS database show that the estimation errors resulting from the T-ACM is **consistently** much lower than the estimation errors from the equi-width and equi-depth histograms. As alluded to earlier, this is consequent to the fact that the trapezoidal rule of the numerical integration technique is more accurate than the right-end or left-end histogram approximation techniques. Thus, for example, we see in the equi-select operation in Table 5.9, the percentage estimation error from the T-ACM is only 3.98%, but that of the equi-width and equi-depth histograms are 25.98% and 22.06% respectively, demonstrating an order of magnitude of superior performance. A similar pattern can be observed in Table 5.10 as well. Such results are typical with both synthetic data and real-world data. The power of the T-ACM is obvious!

One of our major results in Chapter 4 is that the T-ACM with smaller variance produces better query result size estimations. The results from the third group of experiments confirm this result. For example, as can be seen from Table 5.11, for the equi-select operation (Query I), the percentage estimation error corresponding to the T-ACM variance of 1007 is only 3.28%. Whereas for the same operation, the percentage estimation corresponding to the T-ACM variance of 1493 is 9.25%, which is almost a three fold increase.

5.5.3 Experiments on NBA Performance Statistics Database

In the experiments on the NBA database, as in the case of the U.S. CENSUS database, we compared the result estimations from equi-width, equi-depth, and the T-ACM. We considered (a) exact match select queries (b) range select queries and (c) equi-join queries. For the join queries, since the NBA database consists of a single relation, we generated various frequency values with the Zipf and multi-fractal distributions using the same value domain of the joining attribute from the original NBA relation. The resulting relation was, in turn, joined with the corresponding attribute from the NBA relation. As in the case with the experiments on the CENSUS database, for each of our experimental runs, we chose different build-parameters for the histograms

Operation	Result Size	Equi-width		Equi-depth		T-ACM	
		Size	Error	Size	Error	Size	Error
Equi-select	41	25.1	38.78%	29.2	28.78%	26.3	4.61%
Range-select	136	124.0	8.82%	126.2	7.21%	129.6	4.71%
Equi-join	314	236.3	24.74%	251.2	20.0%	269.5	14.17%

Table 5.12: Comparison of Equi-width, Equi-depth and T-ACM: NBA Statistics 1991/92.

and the T-ACM.

As before we obtained the relative estimation error as a ratio by subtracting the estimated size from the actual result size and dividing it by the actual result size. The cases where the actual result sizes were zero were not considered for error estimation. Actual result sizes were computed using a simple query processor. Again the results were obtained by averaging the estimation errors over a number of experiments and are summarized in Table 5.12.

Analysis of the Results

These prototype validating experiments conducted using the NBA statistics database confirm our theoretical claim that the estimation error resulting from the T-ACM is a fraction of the estimation error from the equi-width and equi-depth histograms. As in the case of our experimental results with the U.S. CENSUS database, these results again demonstrate the superiority of the T-ACM over the traditional histograms. For example, as we can see, the percentage estimation error with the T-ACM for equi-select operation is only 4.61%, whereas for the same operation, the equi-width and equi-depth histograms result in 38.78% and 28.78% estimation errors - which is again an order of magnitude larger!

Our results from these experiments on both databases confirm our theoretical results, summarized in Table 4.1, and clearly demonstrate that the estimation accuracy of the T-ACM is superior to that of the traditional schemes.

5.6 Summary of Work Done

In this chapter, we have conducted a number of prototype validating experiments using the Rectangular and Trapezoidal Attribute Cardinality Maps. These experiments involved the use of two popular real-world databases, namely the U.S. CENSUS database and the NBA Player Performance Statistics database. Using the raw data from these databases and combining the value domains of these databases with the Zipf and multifractal distributions, we carried out a number of experiments involving the traditional equi-width, equi-depth histograms and our new methods. The results from all of our experiments **consistently** demonstrate that the R-ACM and the T-ACM are superior to the traditional histograms in terms of estimation accuracy, given the same storage requirements.

Some of the prototype validating experiments conducted on the R-ACM and the T-ACM have been published in the *Proceedings of the International Conference on Business Information Systems*, Poznan, Poland in April, 1999 and in the *Proceedings of the International Conference on Enterprise Information Systems*, Setubal, Portugal in March, 1999 respectively.

In summary, the prototype validating and testing experiments discussed in this chapter have justified the analytical results presented in Chapter 3 and Chapter 4, and have demonstrated the superiority of the R-ACM and the T-ACM over the traditional histograms which are currently used in many commercial database system.

Chapter 6

Benchmarking the Attribute Cardinality Maps[1]

6.1 Introduction

In the last chapter, we presented a set of prototype validating experiments using two popular real-world databases. Although these experiments used typical real-world databases, they utilized only synthetic queries involving simple elementary relational operations. In this chapter, we present the *benchmarking* (as opposed to prototype validation and testing) results from some of the experiments that were conducted using the industry-popular TPC-D benchmark specification. The TPC-D benchmark is a popular industry-standard decision support specification which includes a "simulated" scalable real-world database, and a set of complex "simulated" real-world query types, representing a business enterprise. Using the results from these rigorous benchmarking experiments, we demonstrate that the ACM structures introduced in Chapter 3 and Chapter 4 are well suited for real-world database application environment with general query patterns, and could replace the traditionally used equi-width and equi-depth histograms due to their superior estimation accuracy.

Benchmarking is an important step in the developmental cycle of any new software

[1]Some of the work on the TPC-D benchmarking of both the R-ACM and T-ACM is currently being reviewed for publication and can be found as a Carleton University Technical Report, TR-99-07 [105].

strategy as it validates the functional properties of the new software in a typical real-world environment. It also enables the user to compare the performance of various similar methods against a single standard. But benchmarking is often a challenging and a very complex problem, because it is difficult (if not impossible) to test, validate and verify the results of the various schemes in completely different settings. This is even more true in the case of database systems because the benchmarking also depends on the *types* of queries presented to the databases used in the benchmarking experiments. In this chapter, we study the benchmarking of the two new query optimization strategies, namely the R-ACM and the T-ACM, that we introduced in Chapter 3 and Chapter 4.

First of all we present an overview of the relevant parts of TPC-D benchmark specification that are used in our benchmarking experiments. These include the TPC-D benchmark database and the query types. We then list a number of experimental results obtained using the TPC-D database and query sets, and provide an analysis of the experimental results.

6.1.1 Purpose of Benchmarking

The purpose of the TPC-D benchmarking is to provide to industry users relevant objective performance data about any new system. The TPC-D benchmarking scheme is elegant because it has two distinct components. First of all, it provides the "testing platform" with a database which is both scalable and yet easily modeled. Secondly, it provides the platform with an easy and convenient query generator, from which queries of "arbitrary" complexity can be generated and tested against the database. Thus, the platform works with a set of real-world like business queries designed to run on a carefully selected complex business database.

Due to its closeness to the real-world model, and its industry-wide acceptance as a standard database system benchmark, TPC-D becomes an ideal test-bed to validate our analytical results of the R-ACM and the T-ACM. In addition, unlike the stand-alone elementary relational operations that were used in the prototype validating experiments, the queries used in the TPC-D benchmarking are highly complex.

The queries are also designed in such a way so that they examine a large percentage of available data, but are implemented with sufficient degree of ease. These query patterns, which are catalogued in Table 6.2, consist of a wide range of relational operations and represent the type of queries that are usually posed by DBMS users in a business environment. This enables us to rigorously test the ACMs with various types of queries and confirm their superiority over the traditional histogram based estimation techniques. More over, the DBGEN program supplied by the TPC-D allows us to generate scalable databases reflective of a true business enterprise. This again permits us to conduct a set of robust experiments so that an industry acceptable fair comparison of our new structures (or for that matter, any new strategy) can be presented against the traditional histograms currently in use. Since the TPC-D queries and databases can be accurately generated according to the TPC-D specification document [134], the results presented in this paper can be easily duplicated and verified by interested researchers.

6.1.2 Benchmarking Versus Prototype Validation

Any new system or strategy that is proposed has to be validated and thoroughly tested before it can be effectively used in the industry. This, of course, involves three distinct phases: the validation of the prototype, the testing of the prototype, and more importantly, the benchmarking of the strategy against industry-acclaimed benchmark standards. Although these issues are significant for any software solution, this is particularly important when a new database solution (for example, a query processor) is in question. This, indeed, is because it not only involves the database itself. It also involves the queries that are put against the database. Consequently, although it is possible to test a new strategy against a "real-world" database, the question of testing against families of query patterns is far from trivial, since the query patterns are usually obtained from the users themselves.

In this regard, we would like to mention that the prototype validation and testing phases of the two schemes which we proposed (the R-ACM and the T-ACM) were earlier studied for the real-world databases, namely the U.S. CENSUS and NBA player

statistics databases. However, the issue of testing the schemes against families of query patterns for a scalable database remained open. *This is exactly the contribution of this chapter.*

To achieve this fundamental goal of this task, we were posed with two issues. First of all we have to develop a database which was scalable and which represented a real-life scenario. Secondly, and more importantly, we were to be able to generate "real-life" queries without actually relying on a human interface. It is here that the TPC-D benchmarking platform has provided us with invaluable support.

As mentioned earlier, our previous experimental works related to the R-ACM and the T-ACM involved *prototype validation and testing* using two real-world databases, namely, the U.S. CENSUS database and the NBA player statistics database [132, 133]. Even though these databases were representative of the real-world data distributions, the scope of their use was rather limited to the specific attribute value domains (or data environments) under consideration. More over, the query types used for these specific databases were relatively primitive and included only one of the elementary relational operations (such as equi-join, equi-select and range-select), at a time, in a given query. Consequently, the query types used in these previous experiments conducted in Sections 5.4 and 5.5 cannot be argued to represent any typical real-world user queries, and thus, we do not feel that it is fair for us to extrapolate our results from the Sections 5.4 and 5.5 to the real-world query types.

In contrast to the above, the TPC-D benchmarking experiments presented in this chapter consist of two distinct components. First of all, the databases used for the experiments are scalable simulated real-world data that are typically found in a business enterprise. Secondly, the query types used are representative of a large collection of business oriented queries with broad industry-wide relevance. The query types, as can be seen from Table 6.2, are usually complex and involve multiple relational operations. Hence we expect the experimental results with these TPC-D benchmark database/query types to closely represent the behavior of the ACMs in a real-world database environment. More detailed discussion on the TPC-D queries and databases are found in Section 6.2.

6.2 TPC-D Benchmark Specification

The TPC Benchmark D (TPC-D) has been proposed by the Transaction Processing Performance Council (TPC) as a decision support benchmark. TPC-D models a decision support environment in which complex *ad hoc* business-oriented queries are submitted against a large database. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. We briefly describe the TPC-D benchmark queries and its database model below. A detail description about the complete benchmarking requirements, including auditing can be found in the TPC-D specification document [134].

6.2.1 TPC-D Benchmark Queries

The TPC-D benchmark queries typically involve, multi-table joins, extensive sorting, grouping and aggregation and sequential scans. The purpose of TPC-D is to assess cost/performance of a particular commercial system which supports the above queries. Since our objective is to study the performance of the ACMs in terms of their estimation accuracy, unlike a commercial DBMS, we do not require a fully functional DBMS for our experiments. Most of the TPC-D queries contain relational operators other than selection and equality join, and thus are not directly suitable for running on ACMs. Consequently, for our testing purposes, we shall use a slightly modified form of TPC-D queries by systematically eliminating the inapplicable operations such as grouping and aggregations etc., which are not supported by our current research work. Eliminating queries containing *views*, and modifying the remaining queries, we obtained a total of 11 query types from the original 17 query types specified in the TPC-D specification. These query types are given in Table 6.2.

For each of the relevant query types listed in Table 6.2, we construct an operator tree and estimate the result size of these operator trees in a bottom-up fashion. This bottom-up query execution strategy works as follows. First, we estimate the result sizes and distributions of leaf-level operators using the ACMs on the base relations. Then we construct the ACMs on the result distributions and use these ACMs for

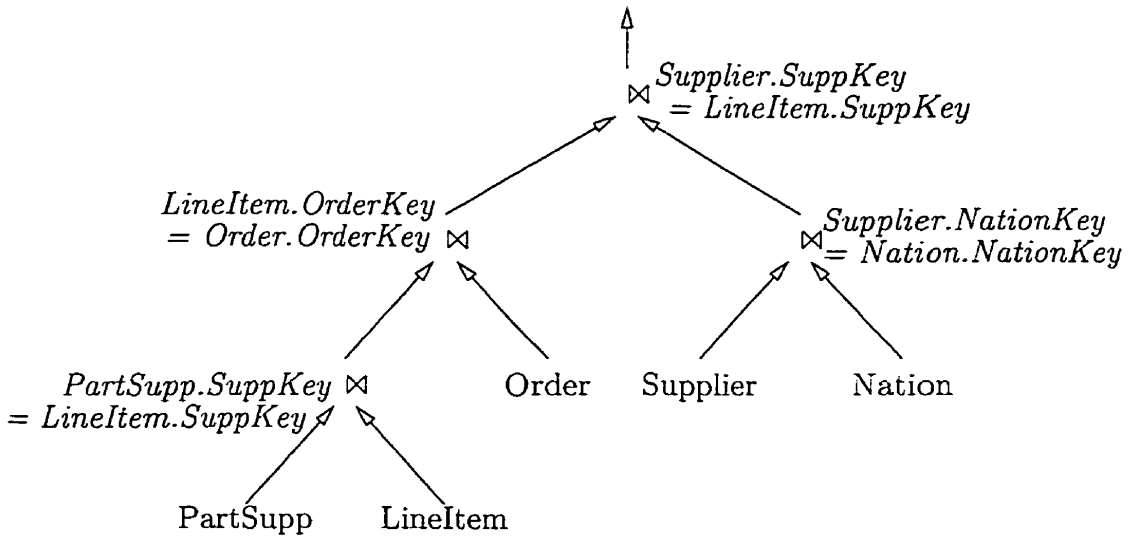


Figure 6.1: Operator Tree for Query Q9: Product Type Profit Measure Query

estimating the result size and distributions of the operators in the next level of the tree. This process is repeated until the result size of the root node of the operator tree is estimated. We also compute the exact result size of the query by actually executing the simplified queries on the same data using a simple query processor which we implemented. Using these we compute the error from the result size estimation using the ACMs. For example consider the query Q9 from Table 6.2. This query determines how much profit is made on a given line of parts.

```
select *
from supplier, partsupp, lineitem, order, nation
where partsupps.suppkey == lineitem.suppkey and
   order.orderkey == lineitem.orderkey and
   supplier.suppkey == lineitem.suppkey and
   supplier.nationkey == nation.nationkey
```

Observe that the query itself involves joining the relations `supplier`, `partsupp`, `lineitem`, `order` and `nation` based on the equality predicates:

(i) `partsupp.suppkey = lineitem.suppkey`,

(ii) `order.orderkey = lineitem.orderkey`,

(iii) `supplier.suppkey = lineitem.suppkey` and

(iv) `supplier.nationkey = nation.nationkey`.

Each of them, in turn, is joined by an and clause which is finally used to select all the tuples from the resulting relation. Note that this join is actually a multi-table equi-join with a multi-clause predicate. An operator tree corresponding to the query, `Q9`, is given in Figure 6.1.

All of the TPC-D benchmark queries involve business oriented questions using standard SQL syntax, conforming to SQL-92 specification. Table 6.1 describes each query in plain English.

6.2.2 TPC-D Benchmark Database

The TPC-D database schema consists of eight tables detailed in the TPC-D specification. This schema models a database for a worldwide distributor that purchases parts from suppliers and sells them to customers. The two largest tables are the master-detail pair of `Order` and `LineItem`, which together constitute about 85% of the database. The other tables describe the business's parts, suppliers and customers. All tables except the two small `Nation` and `Region` tables scale linearly with the size of the database. The TPC-D database can be scaled upto a size of 1000GB in order to support high-end systems. For our experiments, we use a prototype version of this, namely, a test database with a size of approximately 100MB. The TPC-D database itself is generated using the data generation utility, `DBGEN`, supplied by the TPC. The `DBGEN` program generates mainly uniformly distributed data. Since, we are interested in modeling skewed data distribution as well, we have opted to populate the database in three different ways listed below.

In the first method we simply use the data from the `DBGEN` program. This typically results in uniform data.

In the second method, we generate frequencies from a Zipf distribution with an appropriately chosen skew parameter, z, while retaining the original value domain

Query No	Name	Description
Q2	Minimum Cost Supplier Query	Find which supplier should be selected to place an order for a part in a given region
Q3	Shipping Priority Query	Retrieve the 10 unshiped orders with the highest value
Q5	Local Supplier Volume Query	List the revenue volume done through local suppliers
Q6	Forcasting Revenue Change Query	Quantify the amount of revenue increase in a given year
Q7	Volume Shipping Query	Determine the value of goods shipped between certain nations
Q9	Product Type Profit Measure Query	Determine how much profit is made on a given line of parts
Q10	Return Item Reporting Query	Identify customers who might be having problems with parts shipped to them
Q11	Identification of Stock Query	Find the most important subset of suppliers' stock in a given nation
Q12	Shipping Mode Query	Determine whether selecting less expensive modes of shipping affect priority orders
Q14	Promotion Effect Query	Monitor market response to a promotion such as a TV advertisement
Q17	Small-Quantity Order Query	Determine revenue lost if orders where not filled for small quantities of certain parts

Table 6.1: Description of the TPC-D Queries

and relation sizes. We combine the attribute values in various attributes in each relation randomly to generate the tuples for that relation, according to the frequencies obtained from the Zipf distribution.

It is claimed that the multi-fractals distributions occur more frequently in the real world data [36]. So in our third method, we generate frequencies using multi-fractal distributions with various bias values for the original value domain from the TPC-D database. The frequency values resulting from the multi-fractal distributions are combined with the values from the TPC-D database to generate the tuples for the new relations.

Query Number	TPC-D Query
Q2	select s_acctbal, s_name, n_name from part, supplier, partsupp, nation, region where r_regionkey == 3 and n_regionkey == r_regionkey and s_nationkey == n_nationkey and p_size == 40 and ps_partkey == p.partkey and ps_suppkey == s_suppkey
Q3	select l_orderkey, o_shippriority from lineitem, order, customer where l_orderkey == o_orderkey and c_custkey == o_custkey
Q5	select n_name from order, lineitem, customer, supplier, nation, region where o_orderkey == l_orderkey and l_suppkey == s_suppkey and c_nationkey == s_nationkey and s_nationkey == n_nationkey and n_regionkey == r_regionkey and r_regionkey == 1
Q6	select * from lineitem where l_quantity $\leq$ 25
Q7	select supp_nation, cust_nation from lineitem, order, customer, supplier, nation where s_suppkey == l_suppkey and l_orderkey == o_orderkey and c_custkey == o_custkey and n_nationkey == c_nationkey and n_nationkey == 3
Q9	select * from supplier, partsupp, lineitem, order, nation where ps_suppkey == l_suppkey and o_orderkey == l_orderkey and s_suppkey == l_suppkey and s_nationkey == n_nationkey
Q10	select c_custkey, c_name, c_acctbal, n_name from lineitem, order, customer, nation where c_custkey == o_custkey and l_orderkey == o_orderkey and c_nationkey == n_nationkey
Q11	select ps_partkey from partsupp, supplier, nation where ps_suppkey == s_suppkey and s_nationkey == n_nationkey and n_nationkey == 3
Q12	select * from order, lineitem where o_orderkey = l_orderkey
Q14	select * from part, lineitem where p_partkey == l_partkey
Q17	select * from lineitem, part where p_partkey = l_partkey and p_brand = 'Brand4'

Table 6.2: Simplified TPC-D Benchmark Queries

Since the second and third methods of populating the databases of our experiments involve the Zipf and multi-fractal distributions, the reader is referred to Section 3.10 for an overview of them.

6.3 TPC-D Benchmark Experiments

Unlike the prototype validating experiments which were discussed in Sections 5.4 and 5.5, the TPC-D benchmarking experiments involve arbitrarily large and complex query types. Consequently there are often thousands or more possible query evaluation plans (QEPs) for executing them. Since our objective is to compare the query result size estimation accuracy of various techniques as opposed to choosing the optimal QEP, we randomly selected an arbitrary QEP for a given query type and constructed the corresponding operator tree. The result size of this operator tree was then estimated in a bottom-up fashion from the leaves of the tree to the root of the tree. It is important to note that, unlike the case of a primitive relational operation, with these simulated "real-life" query types, it was necessary to construct the ACMs and histograms for all the resulting intermediate relations. What follows is the details of the results we obtained.

The experiments we conducted with the TPC-D benchmark database involved three distinct groups. In the first set of experiments, we used the uniform distribution data that was generated by the DBGEN program from TPC. We computed the estimation errors from the equi-width, equi-depth histograms, the R-ACM and the T-ACM for the set of TPC-D queries from Table 6.2. We conducted ten different set of experiments using the histograms and the ACMs with different build-parameters. The build-parameter for the equi-width histogram and the T-ACM is the sector width, the build-parameter for the equi-depth histogram is the number of tuples per sector and the build parameter for the R-ACM is the tolerance value, τ. For each attribute involved in the query, we generated the equi-width and equi-depth histograms with different sector-widths and different number of tuples/sector respectively. The corresponding T-ACMs were also generated with the same sector widths that were chosen for the equi-width histograms. The T-ACMs were generated using

TPC-D Query	Result Size	Error with Uniform TPC-D Database			
		Equi-width	Equi-depth	R-ACM	T-ACM
Q2	918	10.94%	9.60%	3.21%	3.72%
Q3	42190	13.62%	11.39%	4.37%	5.73%
Q5	209018	10.05%	9.68%	2.55%	3.61%
Q6	20706	12.33%	10.16%	4.38%	4.27%
Q7	2041	15.20%	10.75%	6.13%	5.99%
Q9	2.71e+06	13.83%	11.42%	5.48%	4.91%
Q10	44823	14.07%	12.93%	5.06%	6.15%
Q11	511	11.39%	10.40%	4.21%	5.03%
Q12	19	25.97%	22.73%	5.81%	5.62%
Q14	36724	14.66%	13.45%	4.77%	5.28%
Q17	523	20.35%	19.73%	5.81%	6.37%

Table 6.3: Estimation Error with Histograms, R-ACM and T-ACM on Uniform TPC-D Database

the `Implement_T-ACM` algorithm described in Section 5.5, which is a much improved version of the original `Generate_T-ACM` algorithm given in Section 4.1.1. Similarly for each attribute involved in the query, we generated the R-ACM with a different value for the tolerance, τ. In order to obtain a fair comparison, the build-parameters for all different techniques were chosen such that the resulting storage requirements were the same, regardless of the method. The result sizes and consequently the computed estimation errors were averaged over these set of experiments. The results are shown in Table 6.3.

In the second set of experiments, we used Zipf data distributions with three different skew values ($z = 2, z = 4, z = 6$) on the value domains of the various attribute values of the TPC-D database and computed the estimation errors from the equi-width, equi-depth histograms, the R-ACM and the T-ACM. Again we conducted ten different set of experiments using different build-parameters for the equi-width, equi-depth histograms, the R-ACM and the T-ACM. Due to the steep frequency variations of the Zipf distributions used, the R-ACM partitioning resulted in a very large number of sectors. In order to avoid this, we were forced to choose somewhat larger

TPC-D Query	Result Size	Error with Multi-fractal-TPC-D Database			
		Equi-width	Equi-depth	R-ACM	T-ACM
Q2	528	30.49%	28.26%	15.42%	17.22%
Q3	26210	29.83%	24.68%	14.58%	16.26%
Q5	132450	40.63%	36.27%	12.72%	10.67%
Q6	19204	30.20%	26.11%	14.32%	11.71%
Q7	3096	34.26%	27.42%	12.33%	15.24%
Q9	3.6e+06	28.39%	25.75%	9.97%	9.29%
Q10	40022	26.92%	24.18%	11.92%	13.33%
Q11	212	28.21%	22.73%	12.07%	13.15%
Q12	86	26.42%	23.12%	10.25%	12.63%
Q14	49270	26.25%	21.49%	13.92%	12.84%
Q17	976	23.46%	21.22%	14.67%	15.90%

Table 6.4: Estimation Error with Histograms, R-ACM and T-ACM on Zipf TPC-D Database

tolerance values for the R-ACM partitioning. As with the first set of experiments, in order to make a fair comparison of the various techniques, the build-parameters were chosen so that the resulting storage requirements were same for the histograms and the ACMs. The experiments were conducted for the set of TPC-D queries listed in Table 6.2. We averaged the results for the three Zipf distributions and computed the estimation errors. The results are given in Table 6.4.

The third set of experiments involve using the multi-fractal distributions. A single scan through the data generated by the `DBGEN` program returns the number of distinct attribute values for each of the attributes involved. We selected three different bias values ($p = 0.1, p = 0.3, p = 0.4$) that resulted in three completely different multi-fractal distributions. The frequencies from these multi-fractal distributions were applied randomly to the value domain of the attributes in the TPC-D database to generate a new set of relations. As before we conducted ten different set of experiments with different build-parameters for the equi-width, equi-depth histograms, the R-ACM and the T-ACM, using the same storage requirements. Again the experiments involved applying the TPC-D queries from Table 6.2. We averaged the results

TPC-D Query	Result Size	Error with Zipf-TPC-D Database			
		Equi-width	Equi-depth	R-ACM	T-ACM
Q2	161	27.46%	24.30%	9.62%	11.83%
Q3	26428	29.75%	26.42%	12.87%	12.90%
Q5	38620	19.07%	17.29%	9.46%	10.94%
Q6	16271	17.20%	14.65%	9.33%	6.85%
Q7	263	26.71%	25.80%	13.42%	15.25%
Q9	2.8e+06	19.81%	20.11%	10.66%	11.17%
Q10	22670	26.28%	23.04%	12.37%	8.42%
Q11	395	21.44%	17.59%	9.28%	9.91%
Q12	38	37.46%	31.72%	14.36%	15.11%
Q14	26538	23.81%	21.25%	15.30%	14.65%
Q17	1932	31.63%	29.28%	11.92%	10.87%

Table 6.5: Estimation Error with Histograms, R-ACM and T-ACM on Multi-fractal TPC-D Database

for the three multi-fractal distributions and computed the estimation errors. The results are given in Table 6.5.

6.3.1 Analysis of the Experimental Results

Our results from these three set of TPC-D benchmark experiments confirm our theoretical results, summarized in Table 4.1, and clearly demonstrate that the estimation accuracy of the R-ACM and the T-ACM structures is superior to that of the traditional equi-width and equi-depth histograms.

As we can observe, the estimation errors for the histograms and the ACMs are the lowest for the experiments with the original TPC-D database, which was generated by the TPC supplied `DBGEN` program. For example, for the TPC-D query $Q2$, the estimation errors for the original TPC-D database from Table 6.3 are 10.94%, 9.60%, 3.21% and 3.72% for the equi-width, equi-depth, R-ACM and T-ACM respectively. Whereas, as can be seen from Table 6.5, the estimation errors for query $Q2$ with the multi-fractal TPC-D database are 27.46%, 24.30%, 9.62%, and 11.83% in the same order. The estimation errors for the Zipf TPC-D database, given in Table 6.4, are

even higher. The reason for the lowest error with the original TPC-D database is, that it is uniformly distributed. Similarly the reason for the largest estimation errors with the Zipf TPC-D database is obviously due to the steep frequency changes by virtue of the underlying Zipf distribution. Another reason why the estimation errors for the R-ACM are larger than for the first set of experiments is our choice of comparatively larger tolerance values to avoid generating R-ACMs with large number of sectors. The estimation accuracy of the R-ACM and that of the T-ACM are almost comparable except for range-select queries, where the T-ACM out-performs the R-ACM. This is due to the fact that the trapezoidal rule of numerical integration is more accurate than the right-end or the left-end rectangular rule of numerical integration for computing the area under a curve.

The results from all three sets of experiments show that the estimation errors resulting from the R-ACM and the T-ACM are **consistently** much lower than the estimation errors from the equi-depth and equi-depth histograms. This is consequent to the fact the frequency distribution of an attribute value within an R-ACM is guaranteed to be close to the sector mean since the partitioning of the sectors is based on a user-specified tolerance value, and the deviation from the running mean. Similarly, the trapezoidal rule of the numerical integration technique is more accurate than the right-end or left-end histogram approximation techniques. Thus, these set of experiments with the TPC-D benchmark queries and databases demonstrate that the R-ACM and T-ACM strategies exhibit a distinctively superior performance over the traditional equi-width and equi-depth histograms. Such results are typical with both synthetic data and real-world data. The power of the ACM structures is obvious!

6.4 Summary of Work Done

In this chapter, we have conducted a set of benchmarking experiments using the Rectangular and Trapezoidal Attribute Cardinality Maps and shown how these experiments differ from those conducted in Chapter 5. Our benchmarking experiments involved the use of the industry-popular TPC-D benchmark database and query sets. Using a simulated real-world database and a set of complex simulated real-world query

sets, we conducted a performance comparison study of the ACM structures and the traditional equi-width and equi-depth histograms. The TPC-D database used in the experiments was a scalable database simulating a real-world business enterprise. We also made use of the Zipf and multifractal distributions to generate a wide range of frequency distributions for our experiments, retaining the same attribute value domains from the TPC-D database.

As we have seen from the results of these benchmarking experiments, the estimation accuracy of the R-ACM and T-ACM is **consistently** superior to the equi-width and equi-depth histograms regardless of the type of queries used, and again validate our theoretical results from Chapters 3 and 4. In addition, by using a scalable database, as well as query types from the industry-popular TPC-D benchmark specification, we have demonstrated the suitability of the ACM techniques in a real-world database system.

Some of the results from this chapter are currently being reviewed for publication. They can also be found as a Carleton University Technical Report, TR-99-07 [105].

We anticipate that due to their superior estimation accuracy and low implementation overheads, the attribute cardinality maps will replace the traditional equi-width and equi-depth histograms in the future database systems. A few commercial database vendors are currently investigating the possibility of using some of the concepts introduced in this work.

Chapter 7

(Near) Optimal Attribute Cardinality Maps[1]

The ability of a query optimizer to correctly identify an optimal QEP heavily depends on how accurately the query result sizes are computed. Obviously this, in turn, depends on how accurately the attribute cardinality maps, that are used to map the attribute values, reflect the underlying data distributions. Approximation to the underlying data distribution can be improved by changing the various build-parameters of the ACMs. These include, the tolerance value, τ in the case of an R-ACM and, the sector width and slope of the trapezoids in the case of a T-ACM.

In this chapter we present a few methods by which the ACMs can be made to perform *close* to optimal in a relatively static database environment[2]. This is achieved by choosing appropriate build-parameters for the ACMs as discussed in this chapter.

First we present a cost model of the problem. This cost model can serve as a parameter that can be minimized to generate a nearly optimal ACM. Then we present two different strategies by which a T-ACM structure can be made close to optimal. Finally in Chapter 8 we propose a hybrid attribute cardinality map that is generated by combining both the R-ACM and T-ACM. It will be argued that this can be used to further improve the estimation accuracy of the query result-sizes.

[1]Some of the work done in this chapter are currently being compiled for publication.

[2]Generating optimal ACMs for a highly dynamic database system, considering the changing frequency distributions and query patterns, requires the use of adaptive techniques such as the use of learning automata. This is an ambitious and promising area of research.

7.1 Cost Model

Estimation accuracy of a query result size using an ACM depends on many inter-related factors such as,

1. Estimation error,
2. Storage requirement for the ACM,
3. Data distribution being mapped, and
4. Type of queries being posed.

The first two factors, namely the *estimation error* and the *storage requirements* are inversely related to each other. For example, an ACM with larger number of sectors would obviously result in higher estimation accuracy than an ACM with fewer sectors. The third factor, *the type of data distribution* is an important one in the result estimation accuracy. For example, a uniform data distribution is invariably associated with small estimation errors, whereas a Zipf or a multi-fractal data distribution tends to produce large estimation errors. Since the information about the underlying data distribution is not known *a priori*, the ACM design process cannot make use of this knowledge. The last factor listed above is actually the interaction of the environment (in the form of DBMS users) and the DBMS itself, and is obviously difficult to model as human behavior is difficult to predict. Hence, for reasons of simplicity and ease of implementation, we will assume a fixed storage space and model the cost function, $\mathcal{C}$, as a function $f(\epsilon)$ of the estimation error as:

$$\mathcal{C} = kf(\epsilon),$$

where k is a constant.

7.2 (Near) Optimal Rectangular ACMs

Considering first an R-ACM, the function, $f(\epsilon)$, of the estimation error can be either the cumulative worst-case error of the ACM, $\mathcal{E}_{worst}$, or the variance of the ACM. The cumulative worst-case error of an R-ACM, $\mathcal{E}_{worst}$, is the theoretical maximum worst-case error that is possible for a given R-ACM. Since we know from Theorem 3.5 that the worst-case error, ϵ_i, in estimating the frequency of an arbitrary attribute value, X_i, is equal to,

$$\epsilon_i = \tau \left[\ln \left(\frac{l_j}{i-1} \right) - 1 \right],$$

the maximum possible worst-case error in the j^{th} sector is,

$$\mathcal{E}_j = \int_{x=2}^{l_j} \tau \left[\ln \left(\frac{l_j}{x-1} \right) - 1 \right] dx.$$

Hence the cumulative worst-case error for the entire R-ACM becomes,

$$\begin{aligned} \mathcal{E}_{worst} &= \sum_{j=1}^{s} \int_{x=2}^{l_j} \tau \left[\ln \left(\frac{l_j}{x-1} \right) - 1 \right] dx \\ &= \sum_{j=1}^{s} \tau \left[(l_j - 2)(1 + \ln l_j) - (l_j - 1)\ln(l_j - 1) \right]. \end{aligned}$$

Similarly, from Theorem 3.3, we know that the variance of the R-ACM is given by,

$$Var(ACM) = N - \sum_{j=1}^{s} \frac{n_j}{l_j}.$$

Hence for an R-ACM, we can model the cost function by one of the following expressions:

$$\text{Cost Function} = \begin{cases} \mathcal{C} = k\mathcal{E}_{worst} \\ \mathcal{C} = kVar(ACM). \end{cases}$$

Using the cumulative worst-case error, $\mathcal{E}_{worst}$, as $f(\epsilon)$, we see that the smallest possible tolerance value, τ, that satisfies the given storage requirement will result in the minimum value for the cost function, $\mathcal{C}$. Hence, generally ignoring other factors, we see that by arbitrarily reducing the tolerance value up to the ceiling imposed by the available storage space, we can correspondingly improve the estimation accuracy of an R-ACM.

This thus concludes the strategy by which we can increase the estimation accuracy of the R-ACM - we merely decrease the tolerance value, τ, till we get the desired accuracy.

7.3 (Near) Optimal Trapezoidal ACMs

We now consider the case of generating (near) optimal T-ACMs. In this case, the worst-case error in estimating an equality-select operation lies in the range of $0 \leq \epsilon \leq n_j$, where n_j is the number of tuples in the j^{th} T-ACM sector. Since this is not a narrow range (as in the case of an R-ACM), the cumulative worst-case error for a T-ACM will not help us to derive a meaningful cost function. Hence we shall define the cost function in this case as a function of the variance of the T-ACM itself.

We know from Lemma 4.6, that the variance of the frequency of an attribute value X_i in sector j of a T-ACM is,

$$Var(X_i) = n_j p_i (1 - p_i)$$

where p_i is given by,

$$p_i = \frac{a_j}{n_j} + \frac{2(n_j - a_j l)}{n_j l(l-1)}.i.$$

Let θ_j be the slope of the j^{th} trapezoid. From the geometry of the trapezoid we see that,

$$\tan\theta_j = \frac{2(n_j - a_j l)}{n_j l(l-1)}.$$

Hence the probability p_i can be written in terms of θ_j as,

$$p_i = \frac{a_j}{n_j} + i \tan\theta_j.$$

But the variance of the entire T-ACM is given by,

$$Var(ACM) = \sum_{j=1}^{s} \sum_{i=1}^{l} Var(X_i).$$

Hence we have,

$$Var(ACM) = \sum_{j=1}^{s} \sum_{i=1}^{l} \left(a_j - \frac{a_j^2}{n_j} + (n_j - 2a_j) i \tan\theta_j - n_j i^2 \tan^2\theta_j \right).$$

It is thus clear that the variance of a T-ACM is dependent on the slopes of the individual trapezoids of the T-ACM. Consequently the variance can be expressed as a function, $\varphi(\tan\theta)$ of $\tan\theta$. The cost function of a T-ACM can thus be expressed as,

$$\mathcal{C} = k\varphi(\tan\theta)$$

where k is a constant.

This shows that by choosing slopes of the individual trapezoidal sectors appropriately, we can minimize the value of the cost function and improve the estimation accuracy of the T-ACM. We shall now look at two different methods to find the optimal trapezoidal slopes, in the following sections.

7.3.1 (Near) Optimal T-ACMs: Average Estimation Errors

In this section, we discuss a method to optimize the T-ACM generated using the algorithm `Implement_T-ACM` presented in Chapter 5. As we shall see this optimization process has the effect of minimizing the cost function $\mathcal{C} = k\varphi(\tan\theta)$ where, as explained earlier, $\tan\theta$ is the slope of a trapezoid. In optimizing a trapezoidal sector,

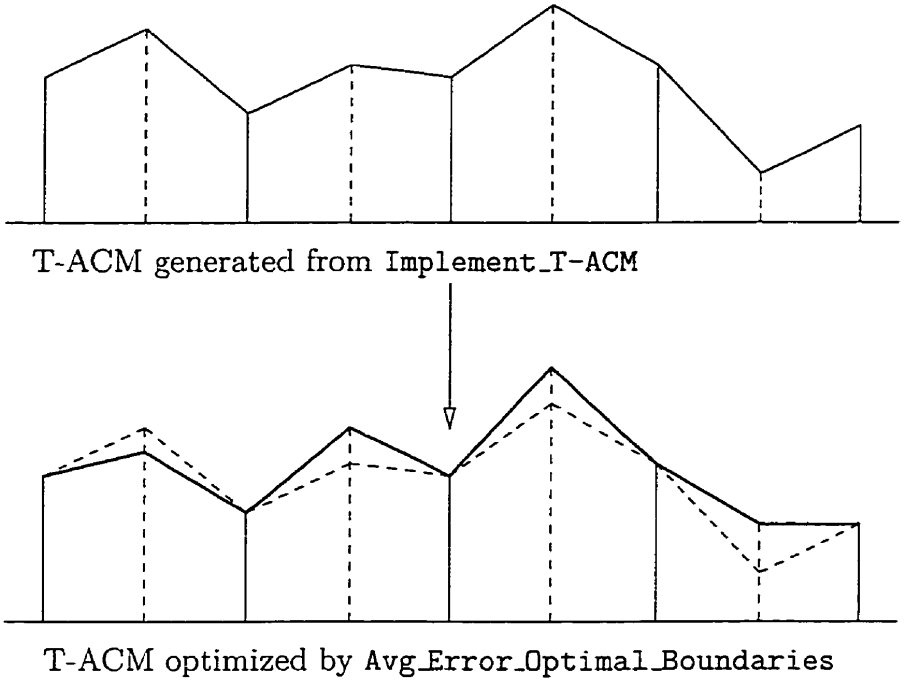


Figure 7.1: Generation of a (near) Optimal T-ACM

our aim is to find the slope of the trapezoid such that the estimation error is minimized for the equality select estimate of every single attribute value in that sector. In other words, in finding the best possible slope of a trapezoid, our aim is to minimize the estimation error for *every single* attribute value within that sector. Since the variance of the frequency of an attribute value is an indicator of how far away the actual frequency value deviates from the mean frequency, minimizing the estimation error for every single attribute value, in turn, has the effect of minimizing the cost function.

In the interest of simplicity, in this work we shall assume a stationary data distribution. We also assume that the T-ACM to be optimized has already been generated using the `Implement_T-ACM` algorithm from an equi-width histogram. The strategy works on every two adjacent T-ACM sectors as follows.

We assume that for every such pair of sectors, the frequencies of the first attribute value of the first sector and the terminal attribute value of the second sector are fixed (or rather known *a priori*). Our aim now is to find the optimal slopes for these

two sectors such that the estimation errors for the equality-select operation on each attribute value, within the sectors, are collectively (or rather simultaneously) minimal. To achieve this end, we will strive to estimate an optimal frequency for the attribute value that lies at the boundary of these two T-ACM sectors. The *criterion function* that we are interested in minimizing is the average error for the frequencies of the entire attribute values within the sector. In other words, we wish to find the straight line (that represents the roof of the trapezoid) that minimizes the overall average estimation errors. Our strategy is explained below.

Without loss of generality, we assume that the starting point of this straight line is at the origin $(0,0)$[3]. Hence our objective is to find the slope, m, of the straight line $x = ma$, where we assume that $A = \{a_1, a_2, \ldots, a_l\}$ is the set of attribute values in the sector and that the attribute value a_i occurs x_i times. The following lemma gives an optimal value for the slope m, where the criterion for optimization is the average error over the entire sector.

Lemma 7.1 *Let $A = \{a_1, a_2, \ldots, a_l\}$ be the set of all attribute values in the sector under consideration. Also let x_i be the frequency of the attribute value a_i. Given l points in the AX-plane*[4]*, the slope, m, of the straight line which goes through the first point $(a_1, x_1) = (0,0)$ and minimizes the* sum *of the vertical distances (frequency estimation errors) of the points to this straight line is given by,*

$$m = \frac{2\sum_{i=2}^{l} x_i}{(l^2 + l - 2)\Delta},$$

where the distances between the successive $a_i, 1 \leq i \leq l$, are the same, and equal to Δ.

Proof: Consider an arbitrary point $P_i \equiv (a_i, x_i)$ on the AX-plane. From Figure 7.2,

[3]If this is not the case, it can be obtained by a simple transformation. Thus, in the argument we set x_1 to 0.

[4]The AX-plane or the Attribute-Frequency plane is a 2-dimensional plane, where the A-axis represents the attribute value domain and X-axis represents the frequency domain.

the vertical distance of $P_i \equiv (a_i, x_i)$ from the straight line $x = ma$ is,

$$\epsilon_i = (x_i - ma_i).$$

This vertical distance ϵ_i represents the estimation error for the frequency of the

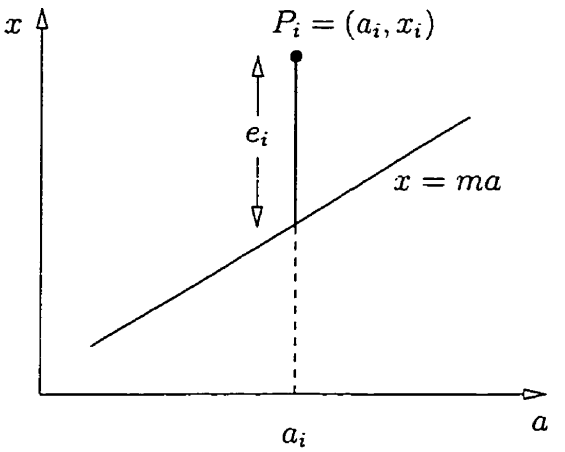


Figure 7.2: Minimizing the Average Estimation Error.

attribute value a_i. The sum of all such distances (estimation errors) in the sector (noting that there is no error in the estimation of the frequency of the first attribute value, a_1 - i.e: the straight line passes through this point), is:

$$\epsilon = \sum_{i=2}^{l} (x_i - ma_i).$$

The average frequency estimation error for an arbitrary attribute value can thus be given as the average of the above sum and is equal to,

$$\epsilon = \frac{1}{l} \sum_{i=2}^{l} (x_i - ma_i).$$

Ideally we would like to see that the above error equates to be zero. Setting this value

to be zero, we find that,

$$\frac{1}{l}\sum_{i=2}^{l}(x_i - ma_i) = 0$$

or

$$\begin{aligned} m &= \frac{\sum_{i=2}^{l} x_i}{\sum_{i=2}^{l} a_i} \\ &= \frac{\sum_{i=2}^{l} x_i}{(2+3+\ldots+l)\Delta} \\ &= \frac{2\sum_{i=2}^{l} x_i}{(l^2+l-2)\Delta}. \end{aligned}$$

The lemma follows. □

Now to proceed with our original task of finding the optimal boundary value for two adjacent sectors, let us consider two such sectors, T_1 and T_2. Assume that the optimal slopes for the sectors T_1 and T_2, using Lemma 7.1, are α and β respectively. Using the optimal slope, α, obtained for the sector, T_1, a boundary frequency value x_b' for the attribute value in the boundary of the two sectors can be obtained. Similarly, using the optimal slope, β, obtained for the sector, T_2, an optimal boundary frequency value, x_b'', can be obtained. But in order to obtain a pair of continuous trapezoidal lines for both sectors, we need to find a common (near) optimal boundary value x_b from the above optimal boundary values x_b' and x_b'' of the individual sectors T_1 and T_2 respectively. We shall now show that the average of x_b' and x_b'' is a suitable boundary value that minimizes the frequency estimation errors in both sectors simultaneously.

Lemma 7.2 *The average of the boundary values x_b' and x_b'', which are optimal for the respective individual sectors, minimizes the estimation errors for both T-ACM sectors simultaneously.*

Proof: In the absence of any better information, we assume that the straight lines $x = \alpha a$ and $x = \beta a$ (resulting in the boundary values x_b' and x_b'') obtained from Lemma 7.1 produce zero estimation errors in their respective T-ACM sectors.

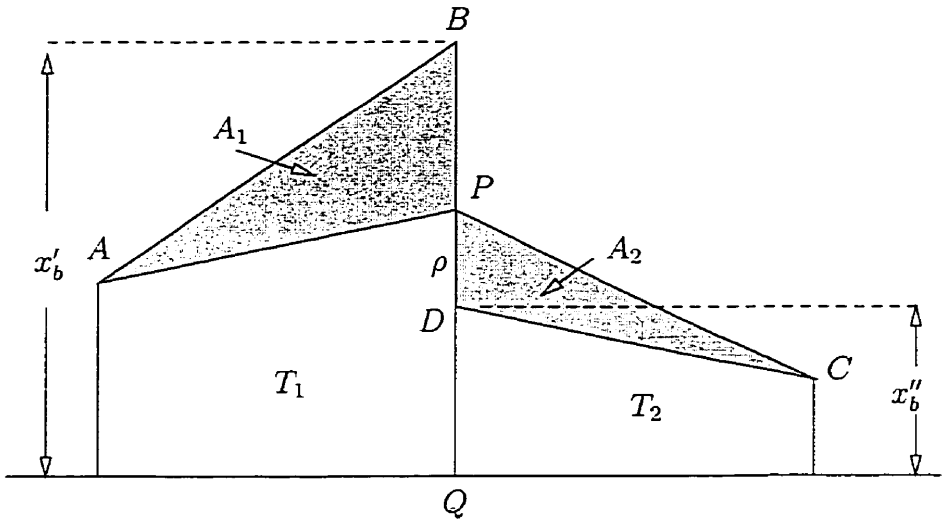


Figure 7.3: Finding Optimal T-ACM Sectors.

The lines AB and CD in Figure 7.3 denote the optimal straight lines obtained from Lemma 7.1. The lines BQ and DQ thus denote the optimal boundary frequencies for the sectors T_1 and T_2 respectively. We also note that $|BQ| = x'_b$ and $|DQ| = x''_b$. We are interested in finding an optimal point P on BD so that we could generate a pair of continuous trapezoidal sectors with straight line segments AP, and PC while minimizing the frequency estimation errors for the two sectors simultaneously. Assume that such an optimal point P lies on the line BD at distance ρ from point D as shown in Figure 7.3. Hence the shaded triangular areas ABP and CDP represent the estimation errors resulting from choosing the lines AP and CP as the trapezoidal roofs instead of the initial lines AB and CD which were optimal for their respective individual sectors.

Let the areas A_1 and A_2 denote the areas of the shaded triangles ABP and CDP respectively. We are interested in finding an optimal point P on line BD such that the total area $A = A_1 + A_2$, representing the overall estimation error, will be minimal. From the geometry of the triangles, we find that,

$$\begin{aligned} A_1 &= \frac{1}{2}(x'_b - x''_b - \rho)l, \\ A_2 &= \frac{1}{2}\rho l. \end{aligned}$$

or the total area is equal to,

$$A = A_1 + A_2 = \frac{1}{2}(x'_b - x''_b)l.$$

We thus find that the total estimation error resulting from choosing the lines AP and CP as the roofs of the respective trapezoids does not depend on the value ρ. Hence we are free to choose any arbitrary point P on the line BD. But we shall choose P as the mid-point of BD, for the simple reason that we do not want either of the new lines AP and CP to be too far away from the initial optimal lines, AB and CD respectively. Hence the lemma follows. □

The algorithm `Avg_Error_Optimal_Boundaries` computes a (near) optimal T-ACM *directly* from an equi-width histogram. The only additional information required is the frequency of the staring value of every second histogram sector. Since we do not require any additional information about other frequencies, we do not need an initial T-ACM and can thus by-pass the process of generating a T-ACM using the `Implement_T-ACM` presented in Chapter 5.

Algorithm 7.7 `Avg_Error_Optimal_Boundaries`

```
   Input:  An equi-width histogram.
   Output:  A (near) optimal T-ACM.
   begin
      for (j = 1; j < s; j + 2);      /* for every 2nd sector */
         α_j = 0;
         α_j = 2*(n_j − l*x_1) / (l^2+l−2)*Δ;  /* optimal slope */
         x'_b = a_j + α_j * l;      /* first estimate for boundary freq */
         β_j = 0;
         β_j = 2*(n_{j+1} − l*x_l) / l^2+l−2)*Δ;  /* optimal slope */
         x''_b = a_{j+1} + β_j * l;      /* second estimate for boundary freq */
      x_b = (x'_b + x''_b)/2;      /* average boundary frequencies */
      update_boundary_frequency(x_b, j);
      end;
   end;
EndAlgorithm Avg_Error_Optimal_Boundaries;
```

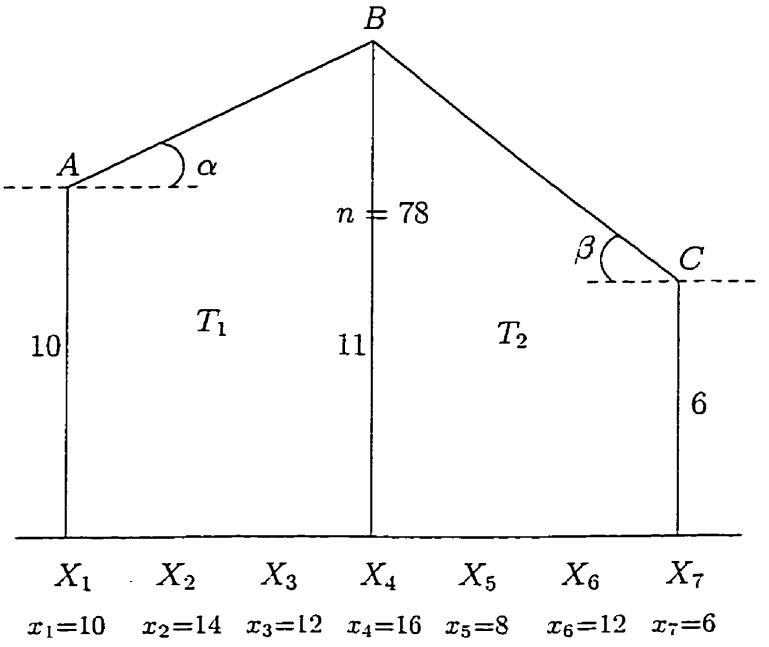


Figure 7.4: Optimizing the T-ACM sectors

Even though Lemma 7.1 uses actual frequency of every attribute value within the sector to compute the optimal slope, in practice we do not need this information. As can be seen from the algorithm `Avg_Error_Optimal_Boundaries`, we can easily compute the slope of the trapezoid only with the knowledge of the total number of tuples within the sector, and the frequency of the first attribute value of the sector.

The entire optimization process can be easily illustrated with an example given below.

Example 7.1 *Let us consider the following example with two T-ACM sectors, shown in Figure 7.4. The actual frequencies of each of the attribute values are denoted in the Figure. Also the total number of tuples falling within these two T-ACM sectors is* $n = 78$. *We assume that the starting frequency of the first sector and the terminal frequency of the second sector are fixed. Let us now look at how we can minimize the estimation error for each of the attribute values* $X_i, i = 1 \ldots 7$ *using the above T-ACM.*

Considering the sector T_1, *since the frequencies of the attribute value* X_1 *is fixed, the T-ACM will return the actual frequency value as the estimate for* X_1. *Thus there*

is no estimation error in this case.

Let α be the slope (tangent of the angle) which the trapezoidal roof sector T_1 makes with the positive *$X-$axis in order to provide minimal estimation error for the attribute values within the sector T_1. From Lemma 7.1, we can obtain the optimal slope as,*

$$\begin{aligned} \alpha &= \frac{2\sum_{i=2}^{l} x_i}{(l^2+l-2)\Delta} \\ &= \frac{2*12}{10} = \frac{12}{5}. \end{aligned}$$

Let x_4' be the frequency of X_4 such that the slope of $AB = \frac{12}{5}$. Hence,

$$x_4' = 10 + \alpha * l = 10 + \frac{12}{5} * 3 = 17\frac{1}{5}.$$

Similarly, let β be the slope (tangent of the angle) which the trapezoidal roof sector T_2 makes with the negative *$X-$axis in order to provide minimal estimation error for the attribute values within the sector. Again from Lemma 7.1, we can obtain the optimal slope as,*

$$\begin{aligned} \beta &= \frac{2\sum_{i=2}^{l} x_i}{l^2+l-2)\Delta} \\ &= \frac{2*18}{10} = \frac{18}{5}. \end{aligned}$$

Let x_4'' be the frequency of X_4 such that the slope of $CB = \frac{18}{5}$. So we have,

$$x_4'' = 6 + \beta * l = 6 + \frac{18}{5} * 3 = 17\frac{2}{5}.$$

Hence the optimal frequency of the boundary attribute value X_4 that would minimize the estimation errors **simultaneously** *for all the 7 attribute values in the sectors T_1 and T_2 is the average of x_4' and x_4'', and is,*

$$\begin{aligned} \overline{x}_4 &= \frac{17\frac{1}{5} + 17\frac{2}{5}}{2} \\ &= 17\frac{3}{10}. \end{aligned}$$

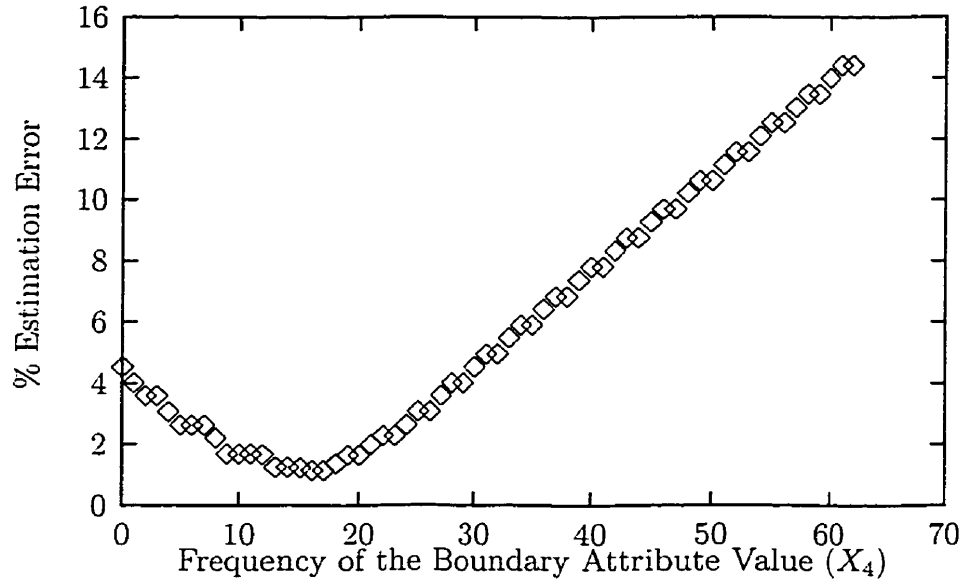


Figure 7.5: Percentage Estimation Error Vs Boundary Frequencies

A plot of percentage estimation errors for equality-select operations against the frequency of the boundary attribute value, X_4, in the above example is shown in Figure 7.5. Observe that the percentage estimation error is near 5% when the frequency of the boundary attribute value is near zero. As the frequency increases, the percentage estimation error gradually decreases and reaches its minimum at the frequency value of $17\frac{3}{10}$. The percentage estimation error increases as the frequency increases beyond this optimal value.

7.3.2 Experimental Results

We conducted a number of experiments to study the effect of optimizing a T-ACM using the method of optimal boundary values discussed in the previous section. As one would have expected the estimation accuracy of the results from the (near) optimal T-ACM is superior to that of a regular T-ACM generated using the original `Implement_T-ACM` algorithm. In this section we present a number of experimental results which highlight the performance increase due to the above optimization. Since

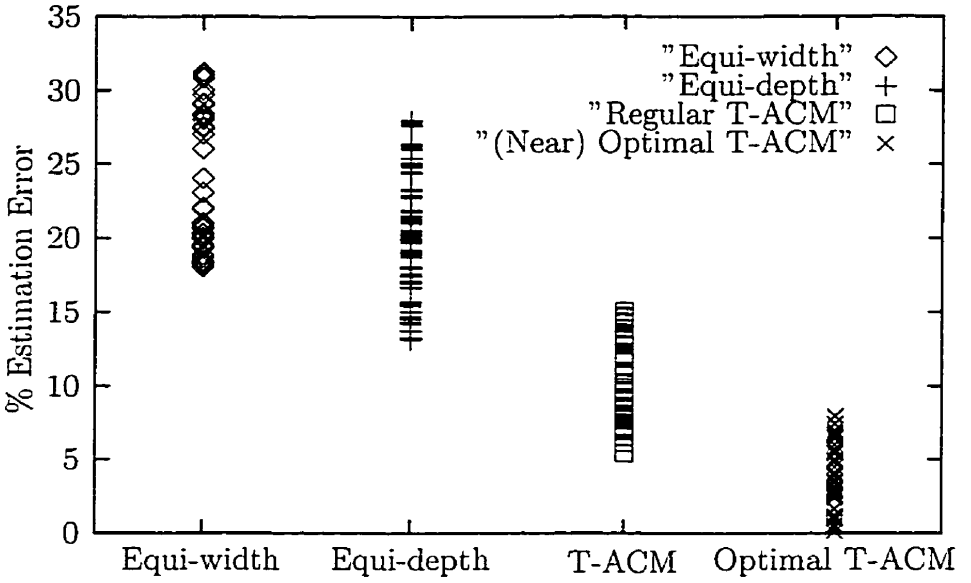


Figure 7.6: Comparison of the Range of Errors between Various Techniques

all of the experiments presented in this section are almost identical to the ones described in Chapters 5 and 6, we shall not delve into the details of their experimental setups, but merely summarize the results and provide appropriate analyses whenever required.

Experiments with Synthetic Data

In the first set of experiments, we compared the percentage estimation errors of the equi-width, equi-depth histograms, a T-ACM generated using the `Implement_T-ACM` algorithm presented in Chapter 5 and a T-ACM optimized by the above algorithm, `Avg_Error_Optimal_Boundaries`. The experiments were conducted using a number of synthetic data distributions, including (a) random (b) Zipf and (c) multi-fractal frequency distributions with simple equi-select operations. The results have been plotted in Figure 7.6, showing the range of errors for each of the above techniques. As can be seen from Figure 7.6, the range of errors for the original T-ACM is $4.9 \leq \epsilon \leq 17.0$, whereas the range of errors for the optimized T-ACM is only $0 \leq \epsilon \leq 8.3$. Needless to say that the range of errors for the equi-width and equi-depth histograms are much higher than the T-ACM techniques.

Operation	Equi-width	Equi-depth	T-ACM	Avg_Error_Optimal_T-ACM
Equi-select	25.30%	21.48%	7.16%	3.97%
Range-select	8.57%	7.11%	3.09%	2.18%
Equi-join	33.93%	29.54%	14.92%	9.01%

Table 7.1: Estimation Accuracy of the Avg_Error Optimal T-ACM: U.S. CENSUS

Operation	Equi-width	Equi-depth	T-ACM	Avg_Error_Optimal_T-ACM
Equi-select	32.07%	29.63%	9.59%	4.27%
Range-select	8.90%	7.26%	2.98%	2.33%
Equi-join	38.44%	35.40%	15.27%	9.85%

Table 7.2: Estimation Accuracy of the Avg_Error Optimal T-ACM: NBA Statistics

Prototype Validation Using Real-world Data

In the second set of experiments, we compared the percentage estimation errors of the equi-width, equi-depth histograms, a T-ACM generated using the original `Implement_T-ACM` algorithm presented in Chapter 5 and a T-ACM optimized by the `Avg_Error_Optimal_Boundaries` algorithm. The experiments were conducted on simple equi-select, range-select and equi-join operations using some selected attributes from the U.S. CENSUS database and the NBA Player Performance Statistics database. The results are shown in Tables 7.1 and 7.2. As can be seen from both Table 7.1 and Table 7.2, the percentage estimation errors from the (near) optimal T-ACM are much lower than the errors from the original T-ACM. For example, the percentage estimation error for the equality-select operation with the original T-ACM from Table 7.1 is more than 7%. But the corresponding quantity for the same operation with the optimized T-ACM is only 3.97%.

Benchmarking (Near) Optimal T-ACMs

In the third set of experiments, we compared the percentage estimation errors of the the equi-width, equi-depth histograms, a T-ACM generated using the original

TPC-D Query	Result Size	Error with Multi-fractal TPC-D Database			
		Equi-width	Equi-depth	T-ACM-1	T-ACM-2
Q2	1119	26.77%	22.38%	10.20%	6.07%
Q3	36729	24.35%	20.05%	9.13%	4.75%
Q5	314593	20.42%	17.03%	6.44%	3.87%
Q6	28186	25.31%	23.66%	9.81%	5.46%
Q7	7872	30.26%	28.43%	7.20%	4.11%
Q9	2.80e+06	24.76%	21.83%	9.47%	4.69%
Q10	72346	26.48%	26.30%	10.63%	4.91%
Q11	568	31.03%	28.90%	9.10%	3.96%
Q12	68	26.78%	23.00%	7.48%	4.39%
Q14	70908	22.61%	22.58%	9.20%	5.10%
Q17	648	28.40%	24.37%	8.29%	5.34%

Table 7.3: Estimation Error with Histograms, Original T-ACM and Avg_Error Optimal T-ACM on Multi-fractal TPC-D Database. The column T-ACM-1 denotes the results for original T-ACM and the column T-ACM-2 denotes the results for Avg_Error optimal T-ACM.

`Implement_T-ACM` algorithm presented in Chapter 5 and a T-ACM optimized by the `Avg_Error_Optimal_Boundaries` algorithm, using the TPC-D benchmark database and queries. This group of experiments were conducted to investigate the performance increase of the T-ACM in a simulated scalable real-world database using a set of complex simulated real-world query types. The results are shown in Table 7.3. Again the results indicate that the percentage estimation errors are much lower for the Avg_Error optimal T-ACM generated through the `Avg_Error_Optimal_Boundaries` algorithm. For example, the percentage estimation error for the TPC-D query, Q9, using the original T-ACM is more than 9%, but the corresponding value for the same query using the optimized T-ACM is only 4.69%, which is nearly a 50% improvement.

7.3.3 (Near) Optimal T-ACMs: Least Squares Errors

In this section, we discuss a second method to optimize the T-ACM generated using the algorithm `Implement_T-ACM` presented in Chapter 5. Again, as in the previous method, our goal is to minimize the cost function $\mathcal{C} = k\varphi(\tan\theta)$. To achieve this, we shall use the well-known *method of least squares approximation* to curve fitting.

As we mentioned earlier, our aim is to find the best trapezoidal approximation to the frequency values within a T-ACM sector. Since we are dealing with two variables (attribute value a_i and the corresponding frequency value x_i) where one is a random variable (frequency x_i) and the other is an independent variable, our problem is analogous to a *regression analysis* problem. In regression analysis one is interested in the dependence of the random variable (frequency, $Y = x_i$, also called the criterion variable) on the independent variable (attribute value, $X = a_i$).

Principle of Least Squares

The objective of regression [77] is to evaluate the coefficients of an equation relating the criterion variable to one or more other variables, which are called the independent variables. The most frequently used linear model relates a criterion variable Y to a single independent variable X by the equation,

$$\hat{Y} = mX + c \tag{7.1}$$

in which c = the intercept coefficient and m = the slope coefficient; c and m are called *regression coefficients* because they are obtained from a regression analysis. Because Equation (7.1) involves two variables, X and Y, it is sometimes referred to as the bivariate model. The intercept coefficient represents the value of Y when X equals zero. The slope coefficient represents the rate of change in Y with respect to change in X. Whereas c has the same dimensions as Y, the dimensions of m equal the ratio of the dimensions of Y to X.

The linear multivariate model relates a criterion variable to two or more independent variables:

$$\hat{Y} = m_1 X_1 + m_2 X_2 + \ldots + + m_p X_p + c \tag{7.2}$$

in which p =the number of independent variables, X_i = the i^{th} independent variable, m_i = the i^{th} slope coefficient, and c is the intercept coefficient, where $i = 1, 2, \ldots, p$.

The values of the slope and intercept coefficients of Equations (7.1) and (7.2) can be computed using the principle of least squares. The principle of least squares is a process of obtaining *best* estimates of the coefficients and is referred to as a regression method. Regression is the tendency for the expected value of one or two jointly correlated random variables to approach more closely the mean value of its set than any other. The principle of least squares is used to regress Y on either X or X_i values of Equations (7.1) and (7.2), respectively. To express the principle of least squares, it is important to define the error, ϵ, as the difference between the estimated and actual values of the criterion variable:

$$\epsilon = \hat{Y}_i - Y_i \tag{7.3}$$

in which $\hat{Y}_i$ = the i^{th} estimated value of the criterion variable, Y_i = the i^{th} actual value of Y, and ϵ_i = the i^{th} error. The objective function for the principle of least squares is to minimize the sum of the squares of the errors:

$$\mathcal{F} = \min \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 \tag{7.4}$$

in which n is the number of observation on the criterion variable.

After inserting the model of $\hat{Y}$, the objective function of Equation (7.4) can be minimized by taking the derivatives with respect to each unknown, setting the derivatives equal to zero, and then solving for the unknowns. The solution requires the model for predicting Y_i to be substituted into the objective function.

To illustrate the solution procedure, the model of Equation (7.1) is substituted

into the objective function of Equation (7.4), which yields

$$\mathcal{F} = \min \sum_{i=1}^{n} (mX_i + c - Y_i)^2.$$

The derivatives of the sum of squares of the errors with respect to the unknowns c and m are respectively,

$$\begin{aligned} \frac{\partial \sum_{i=1}^{n} (\hat{y}_i - y_i)}{\partial c} &= 2 \sum_{i=1}^{n} (c + mX_i - Y_i) = 0 \\ \frac{\partial \sum_{i=1}^{n} (\hat{y}_i - y_i)}{\partial m} &= 2 \sum_{i=1}^{n} (c + mX_i - Y_i) = 0 \end{aligned}$$

Dividing each equation by 2, separating the terms in the summations, and rearranging yield the set of normal equations:

$$\begin{aligned} nc + m \sum_{i=1}^{n} X_i &= \sum_{i=1}^{n} Y_i \\ c \sum_{i=1}^{n} X_i + m \sum_{i=1}^{n} X_i^2 &= \sum_{i=1}^{n} X_i Y_i \end{aligned}$$

The summations in the above equations are calculated over all values of the sample. The two unknowns c and m can be evaluated by solving the two simultaneous equations as follows:

$$\begin{aligned} m &= \frac{\sum_{i=1}^{n} X_i Y_i - \frac{1}{n} \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i}{\sum_{i=1}^{n} X_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} X_i \right)^2} \\ c &= \frac{\sum_{i=1}^{n} Y_i}{n} - \frac{m \sum_{i=1}^{n} X_i}{n}. \end{aligned}$$

(Near) Optimal T-ACMs Using the Principle of Least Squares

In order to apply the above method of least squares, our problem can be formulated as follows:

Given l points in the $AX-$plane (attribute-frequency plane), fit a straight line through the given points so that the sum of the squares of the distances of those points from the straight line is minimum, where the distance is measured in the vertical direction.

The following lemma gives an optimal value for the slope m of a straight line, where the criterion for optimization is the sum of squares of error over the entire sector.

Lemma 7.3 *Let $A = \{a_1, a_2, \ldots, a_l\}$ be the set of all attribute values in the sector under consideration. Also let x_i be the frequency of the attribute value a_i. Given l points in the AX-plane, the slope, m, of the straight line which goes through the first point $(a_1, x_1) = (0, 0)$ and minimizes the* sum *of the squares of the vertical distances (frequency estimation errors) of the points to this straight line is given by,*

$$m = \frac{6 \sum_{i=2}^{l} a_i x_i}{l(l-1)(2l-1)}.$$

Proof: Consider an arbitrary point $P_i \equiv (a_i, x_i)$ on the AX-plane. As in the first method, we assume without loss of generality that this straight line passes through the origin $(a_1, x_1) = (0, 0)$. From Figure 7.7, the vertical distance of $P_i \equiv (a_i, x_i)$ from the straight line $x = ma$ is,

$$\epsilon_i = (x_i - ma_i).$$

The sum of the squares of these distances is,

$$\mathcal{E} = \sum_{i=2}^{l} \epsilon_i^2 = \sum_{i=2}^{l} (x_i - ma_i)^2.$$

Using the method of least squares we choose m such that ϵ is minimum. Since the least square error ϵ depends on m, a necessary condition for ϵ to be minimum is,

$$\frac{\partial \mathcal{E}}{\partial m} = 0.$$

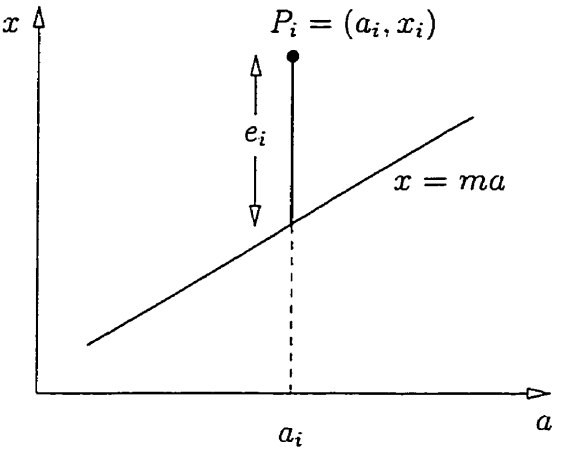


Figure 7.7: Least Squares Estimation Error.

This results in,

$$\frac{\partial \mathcal{E}}{\partial m} = -2\sum_{i=2}^{l} a_i(x_i - ma_i) = 0$$

Thus,

$$m\sum_{i=2}^{l} a_i^2 = \sum_{i=2}^{l} x_i a_i,$$

or

$$m = \frac{\sum_{j=i}^{l} x_i a_i}{\sum_{i=2}^{l} a_i^2} = \frac{6\sum_{j=i}^{l} x_i a_i}{l(l-1)(2l-1)}$$

and the lemma follows. □

Having found an optimal slope for the roof of a trapezoidal sector, as we did in the case of the previous method described in Section 7.3.1, we can proceed to compute an optimal boundary frequency for the attribute value in the boundary of two adjacent sectors. The process involves computing the optimal slopes α and β for a pair of adjacent trapezoidal sectors and obtaining the optimal boundary values, x_b' and x_b''

for the respective individual sectors from them. The question of whether the mean of x'_b and x''_b yields the overall (near) optimal solution for *both* the sectors remains open. But as this mean is easily computed we have tested it to verify that it is, indeed, an excellent solution to defining the overall (near) optimal solution. However, it seems intuitively clear that since the errors do not cancel, (as we consider the squared errors), this solution will be superior to the average error solution described earlier.

Algorithm 7.8 LSE_Optimal_Boundaries

```
Input:  A T-ACM generated from Implement_T-ACM algorithm.
Output:  A (near) optimal T-ACM.
begin
   for (j = 1; j < s; j + 2);        /* for every 2nd sector */
      α_j = 0;
      for (i = 2; i ≤ l; i++);       /* optimal slope for sector T_j */
         α_j = α_j + (i * Δ * x_ji) / (iΔ)^2;        /* optimal slope */
      x'_b = a_j + α_j * l;          /* first estimate for boundary freq */
      β_j = 0;
      for (i = 2; i ≤ l; i++);       /* optimal slope for sector T_{j+1} */
         β_j = β_j + (i * Δ * x_(j+1)i) / (iΔ)^2;    /* optimal slope */
      x''_b = a_{j+1} + β_j * l;     /* second estimate for boundary freq */
   x_b = (x'_b + x''_b) / 2;         /* average boundary frequencies */
   update_boundary_frequency(x_b, j);
   end;
end;
```

EndAlgorithm LSE_Optimal_Boundaries;

The algorithm `LSE_Optimal_Boundaries` computes a (near) optimal T-ACM *directly* from a T-ACM generated using the `Implement_T-ACM` presented in Chapter 5. It is important to note that, unlike the previous method described in Section 7.3.1, our algorithm requires the frequency value of every single attribute value in order to perform the optimization using the least squares method. The frequencies of the attribute values can be obtained in a single scan of the attribute value domain from

the DBMS catalogue periodically whenever such optimization is deemed necessary. In Algorithm 7.8, a frequency value is denoted by x_{ji}, where the first subscript j denotes the sector where the attribute value lies, and the second subscript i denotes the actual frequency value.

Now referring to the previous example, Example 7.1, an optimal boundary value for the two sectors can be obtained using the method of least squares as follows.

An optimal slope α for the first sector is,

$$\alpha = \frac{\sum_{i=2}^{l} a_i x_i}{\sum_{i=2}^{l} a_i^2} = \frac{13}{7}.$$

Hence an optimal boundary value x_b' for the two sectors is $x_b' = 10 + \frac{13}{7} * 3 = 15\frac{4}{7}$.

Similarly an optimal slope β for the second sector is,

$$\beta = \frac{\sum_{i=2}^{l} a_i x_i}{\sum_{i=2}^{l} a_i^2} = \frac{20}{7}.$$

Hence an optimal boundary value x_b'' for the two sectors is $x_b'' = 6 + \frac{20}{7} * 3 = 14\frac{4}{7}$.

Hence using an argument similar to the one provided in Lemma 7.1, we can *estimate* an optimal frequency of the boundary attribute value X_4 that would minimize the estimation errors **simultaneously** for all the 7 attribute values in the sectors T_1 and T_2. As mentioned earlier this value is the average of x_b' and x_b'', and is equal to,

$$\overline{x}_4 = \frac{15\frac{4}{7} + 14\frac{4}{7}}{2} = 15\frac{1}{14}.$$

Figure 7.8 is a plot of the percentage estimation errors for the above example, using equality-select operations against the frequency of the boundary attribute value, X_4. Comparing this with the plot given in 7.5, we see that the boundary value estimation using the method of least squares errors is marginally more accurate than using the method of average errors.

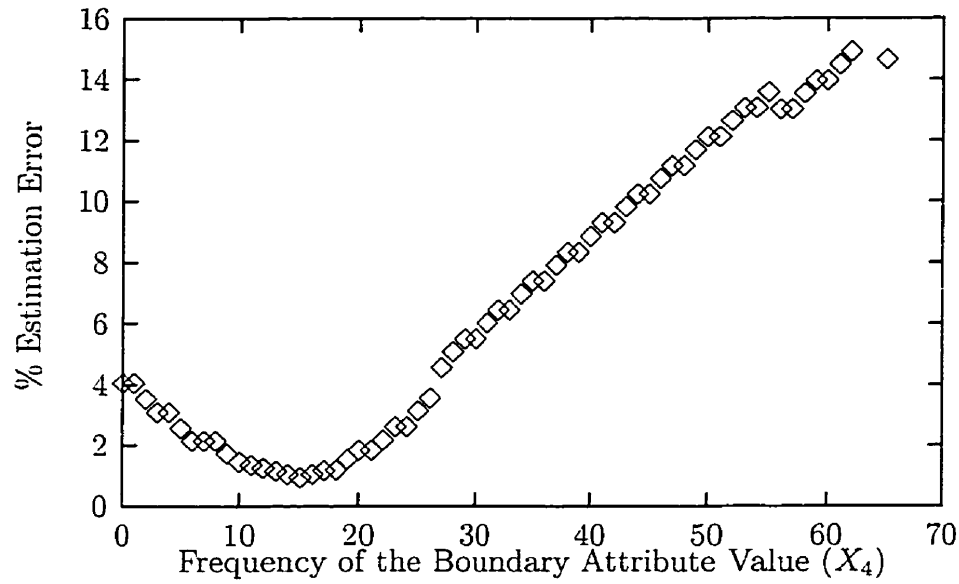


Figure 7.8: Optimal Boundary Frequencies: Least Square Error Method.

7.3.4 Experimental Results

This section presents a number of experimental results that compare the performance of the traditional histograms, the T-ACM generated using the `Implement_T-ACM` algorithm and the (near) optimal T-ACM generated using the method of least squares. These results clearly show that the (near) optimal T-ACMs generated using the least squares error method is superior to the regular T-ACM generated using the original `Implement_T-ACM` algorithm. In this section we present a number of experimental results which highlight the performance increase due to the above optimization. We omit the details about the experimental setups as they are very similar to the experiments conducted in Chapters 5 and 6.

Experiments with Synthetic Data

As in the case of the previous optimization method, in this group of experiments we compared the the percentage estimation errors of the equi-width, equi-depth histograms, a T-ACM generated using the `Implement_T-ACM` algorithm presented in Chapter 5 and a T-ACM optimized by the `LSE_Optimal_Boundaries` algorithm. The

Operation	Equi-width	Equi-depth	T-ACM	LSE_Optimal_T-ACM
Equi-select	21.49%	18.32%	7.20%	2.64%
Range-select	9.36%	5.18%	3.17%	1.48%
Equi-join	32.80%	27.55%	13.28%	7.33%

Table 7.4: Estimation Accuracy of the LSE-Optimal T-ACM: Synthetic Data

experiments were conducted using a number of synthetic data distributions, including (a) random (b) Zipf and (c) multi-fractal frequency distributions with equi-select, range-select and equi-join operations. The results are summarized in Table 7.4. These experimental results indicate that the estimation accuracy of the optimized T-ACM is much superior to the original T-ACM as well as the traditional histograms. For example, for the equi-select operation, the errors resulting from the equi-width, equi-depth, and the original T-ACM are 21.49%, 18.32%, and 7.20% respectively. But the estimation error resulting from the optimized T-ACM for the same operation is only 2.64%. We can also observe that this is a marginal improvement in accuracy over the average error optimization strategy presented in Section 7.3.1.

Prototype Validation Using Real-world Data

The second group of experiments were conducted using two real-world databases, namely the U.S. CENSUS and the NBA Player Performance Statistics databases. We compared the percentage estimation errors of the equi-width, equi-depth histograms, a T-ACM generated using the original `Implement_T-ACM` algorithm presented in Chapter 5 and a T-ACM optimized by the `LSE_Optimal_Boundaries` algorithm, in addition to comparing the estimation errors from the traditional histograms. The experiments were conducted on simple equi-select, range-select and equi-join operations using some selected attributes from the above mentioned databases. The results are summarized in Tables 7.5 and 7.6. The results from this set of experiments again confirm that the optimization using the least squares method indeed results in superior estimation accuracy. For example, as can be seen from the Table 7.6, the estimation error for the equi-select operation using the original T-ACM is 8.42%, but the estimation error

Operation	Equi-width	Equi-depth	T-ACM	LSE_Optimal_T-ACM
Equi-select	23.47%	20.32%	9.02%	2.91%
Range-select	9.58%	5.90%	3.42%	1.38%
Equi-join	31.20%	27.41%	11.07%	7.92%

Table 7.5: Estimation Accuracy of the LSE-Optimal T-ACM: U.S. CENSUS

Operation	Equi-width	Equi-depth	T-ACM	LSE_Optimal_T-ACM
Equi-select	31.81%	28.00%	8.42%	3.02%
Range-select	7.19%	6.15%	3.33%	1.65%
Equi-join	39.72%	32.01%	14.38%	8.11%

Table 7.6: Estimation Accuracy of the LSE-Optimal T-ACM: NBA Statistics

for the same operation using the LSE-Optimal T-ACM is only about 3%.

Benchmarking (Near) Optimal T-ACMs

The third group of experiments were conducted to study the performance increase of the (near) optimal T-ACM in a simulated scalable real-world database using a set of complex simulated real-world query types. Using the industry-popular TPC-D benchmark database and query sets, we compared the percentage estimation errors of the the equi-width, equi-depth histograms, a T-ACM generated using the original `Implement_T-ACM` algorithm presented in Chapter 5 and a T-ACM optimized by the `LSE_Optimal_Boundaries` algorithm. The results are summarized in Table 7.7. As can be seen from this table, the estimation accuracy of the (near) optimal T-ACM is obviously superior to the regular T-ACM and to the equi-width and equi-depth histograms.

Average Error Versus Least Squares Error

Having presented two strategies to optimize the T-ACMs in the previous sections, it is natural to provide a comparison between the two methods.

TPC-D Query	Result Size	Error with Multi-fractal TPC-D Database			
		Equi-width	Equi-depth	T-ACM-1	T-ACM-2
Q2	1032	25.32%	20.12%	9.44%	4.21%
Q3	32574	23.10%	21.42%	8.27%	3.98%
Q5	306278	19.98%	18.37%	4.30%	2.97%
Q6	32098	24.43%	23.74%	9.75%	5.30%
Q7	6910	29.11%	25.30%	6.53%	2.61%
Q9	3.47e+06	20.29%	20.78%	8.21%	3.28%
Q10	64184	27.03%	25.90%	11.39%	5.49%
Q11	327	30.15%	27.39%	8.24%	3.20%
Q12	42	25.63%	21.42%	6.31%	2.08%
Q14	57320	24.50%	23.94%	5.37%	2.85%
Q17	648	29.83%	26.66%	6.31%	3.42%

Table 7.7: Estimation Error with Histograms, Original T-ACM and LSE Optimal T-ACM on Multi-fractal TPC-D Database. The column T-ACM-1 denotes the results for original T-ACM and the column T-ACM-2 denotes the results for LSE optimal T-ACM.

The optimization strategy using the average error method, computes the optimum slope of the trapezoid by considering the sum of the overall estimation error within a T-ACM sector. Obviously, this strategy does not distinguish between negative estimation errors and positive estimation errors. Consequently, this results in some loss of accuracy due to the cancellation effect of the possible addition of some negative and positive estimation errors. On the other hand, this optimization strategy does not require the knowledge of the frequencies of every single attribute value in the relation. In addition to the total number of tuples in a sector, which is already available from an equi-width histogram, we only require the frequencies of the starting attribute value of every second T-ACM sector. This is indeed a tremendous saving on the computational and storage requirement.

As opposed to the first method, the optimization strategy using the least squares error method, computes the optimum slope of the trapezoid by considering the sum of the squares of the estimation error for every attribute value within a T-ACM sector. Since the estimation error at every attribute value is squared in this method, there

is no loss of accuracy due to the cancellation effect that was seen in the previous method. Comparison of the estimation errors from the previous method (see Tables 7.1, 7.2, and 7.3) and the method of least squares (see Tables 7.4, 7.5, 7.6, and 7.7), clearly shows that the method of least squares is marginally more accurate for both synthetic and real-world data distributions. For example, comparing the estimation results conducted on the U.S. CENSUS database which are summarized in Tables 7.1 and 7.5, we see that the estimation error for the equi-join operation using the average error optimization method is nearly 9.0%, but the estimation error for the same operation using the least squares error optimization method is nearly 8.0%, which is slightly superior. But the downside of the method of least squares is that we need the frequency of every single attribute value in the attribute value domain to compute the overall least squares error. This can be prohibitively expensive for a relatively large attribute value domain.

7.4 Summary of Work Done

In this chapter we discussed a few strategies to improve the performance of the rectangular and trapezoidal attribute cardinality maps. We initially presented a cost model and showed that the estimation accuracy of the R-ACM can be arbitrarily improved by decreasing the tolerance value, τ. As discussed, this strategy requires proportionally large amount of storage for the resulting R-ACM structure.

We then discussed two different strategies to improve the estimation accuracy of the T-ACM. These strategies are respectively based on minimizing the average error and least squares error when fitting a straight line through the points which represent the frequencies of the corresponding attribute values. The strategy based on the average error method is less accurate due to the fact that some of the positive estimation errors may be canceled out by some of the negative estimation errors. Nevertheless this estimation strategy is very useful in a practical settings as it does not require the knowledge of every single attribute value in the relation. The only additional information required for implementing this strategy is the frequency value of the starting attribute value of every *second* T-ACM sector.

The second optimization strategy is based on the popular least squares estimation method, which uses regression analysis to fit a straight line to a set of points so as to minimize the sum of squares of the errors.

Numerous experimental results validating the accuracy of these two optimization strategies have also been included.

Chapter 8

Conclusions

Estimation of the size of query results is an important functionality of the query optimizer in a DBMS. The steadily intensifying competition in the marketplace has been the cause of increasing the sizes of databases, as well as of escalating the level of sophistication of queries made against them. The reader can well understand that these have the effect of increasing the accuracy requirements of the estimation techniques. In spite of this, many commercial database systems still continue to use the traditional statistical techniques for estimation purposes even though their accuracy is hard to optimize.

In this thesis, we have focused on some novel histogram-like techniques for estimating the result sizes of the two most commonly used relational operations, namely, selections and equality-joins. Although histograms are used in most of the commercial database systems for the last two decades, very little attention has been paid (both academically and in the industrial world) to their accuracy and formal mathematical properties. Our main goal in this research work has been to develop more accurate classes of histogram-like techniques than the current state-of-the-art ones. Some of the results presented in this thesis are currently being considered by a few database vendors for possible inclusion in their products.

8.1 Contributions of the Thesis

Our main contributions of this research work are summarized below.

I. Proposal of Two Major Estimation Techniques

We have introduced two new histogram-like techniques called the Rectangular Attribute Cardinality Map (R-ACM) and the Trapezoidal Attribute Cardinality Map (T-ACM). They were formally introduced in Chapters 3 and 4 respectively. These techniques aim to approximate the density of the underlying attribute values using the philosophies of numerical integration. In the R-ACM, the density function within a given sector is approximated by a rectangular cell, where the height of the cell is obtained so as to guarantee that the actual probability density differs from the approximated one by a maximum of a user-specified tolerance τ. As opposed to this, in the T-ACM, the density function within a given sector is approximated by a trapezoidal cell, in which the slope of the trapezoid is obtained so as to guarantee that the actual probability mass within the cell equals the true probability mass.

II. Analytic Results Regarding the R-ACM and T-ACM

We have shown in Chapter 3 that the frequencies of attribute values within the sectors of the R-ACM are Binomially distributed. This has permitted us to derive a number of important analytical results which are listed below.

1. Maximum likelihood estimates for the frequency of any arbitrary attribute value.
2. Variance of the frequency of an attribute value, variance of the frequency within the sector, and the variance of the R-ACM.
3. Self-join estimation error.
4. Average-case and worst-case error bounds for both equality-select and range-select queries.

Similar (analogous but more involved) results have also been developed in Chapter 4 for the T-ACM.

III. Proofs of Superiority of the ACMs

Using theoretical analysis we showed in Chapters 3 and 4 that the R-ACM and the T-ACM are superior to the traditional equi-width and equi-depth histograms. Our experimental results, using synthetic data for all select and equi-join queries, also confirm our theoretical analysis and often yield an order of magnitude smaller estimation errors in the query result size estimates.

IV. Prototype Validation and Bench-marking Tests

We also conducted an extensive set of experiments (both prototype validations and bench-marking tests) on these new techniques. These experiments used two popular real-world databases and the TPC-D database and queries. We also augmented these real-world databases with frequencies generated from the Zipf and multi-fractal distributions. Even though we used real-world databases, namely the U.S. CENSUS and NBA Player Statistics databases, the prototype validating experiments presented in Chapter 5 only used simple synthetic queries. To augment this, the benchmarking experiments presented in Chapter 6 used a simulated real-world scalable database and many complex simulated real-world query types based on the industry-standard Transaction Processing Performance Council's TPC-D specifications.

V. Techniques for Obtaining Build-Parameters

Since the estimation accuracy of the R-ACM and the T-ACM mainly depends on their build-parameters, namely the tolerance value, τ and the slope of the trapezoidal sectors respectively, we have provided a few techniques to obtain these quantities. The estimation accuracy of the R-ACM can be arbitrarily increased by reducing the tolerance value, τ. But this will, of course, require proportionally larger storage space for the R-ACM. But in the case of a T-ACM, for a given storage space, the estimation accuracy can be improved by choosing suitable values for the trapezoidal slopes. We presented two different strategies for obtaining the trapezoidal slopes in Chapter 7. These strategies include minimizing the sum of the average estimation errors and minimizing the sum of the squares of the estimation errors, which is based on the well-known principle of least-squares method in curve fitting. Our experimental results

indicate that the build-parameters obtained using these techniques consistently yield an even superior estimation accuracy.

8.2 Novel Preliminary Solutions: Hybrid Attribute Cardinality Maps

Throughout this thesis we concentrated on the R-ACM and the T-ACM as two fundamental and distinct schemes for query result size estimation. There is no reason why we cannot incorporate these into a single scheme which has the advantages of the superior properties of both of them. In this section we intend to propose one such scheme. Even though we have some preliminary results on this new scheme, since it is a promising new area of research, we believe it is appropriate to describe it here.

The reader will observe the the R-ACM and the T-ACM were introduced as two fundamental tools for query result size estimation in database systems. In this section, we propose (and submit some preliminary ideas for) a potential new area of research, namely the development of an adaptive hybrid method, where a learning scheme (possibly an automaton or a neural network) combines both the R-ACM and the T-ACM to generate a hopefully superior hybrid structure.

Using this approach, the automaton can decide to partition one or more sectors of an R-ACM with smaller tolerance values generating a new structure with multiple tolerance values. This approach could be useful whenever there is a need for a higher degree of estimation accuracy for a particular attribute value range. This idea is easily explained with an example.

Consider an R-ACM (See Figure 8.1) which has been partitioned using a tolerance value $\tau = \tau_1$. Suppose that using this R-ACM in a practical-setting, we find that the attribute values in the range of 63 to 118 (belonging to sector 3) are retrieved more heavily than the other attribute values. We decide that a higher estimation accuracy for this attribute value range would improve the performance of the query optimizer. The obvious approach would be to partition the entire value domain using a smaller tolerance value $\tau_2 < \tau_1$. But this would, of course, require proportionately

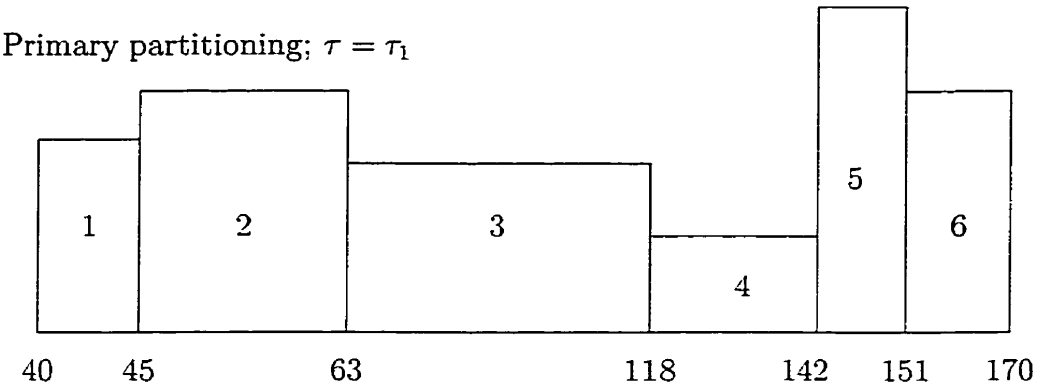


Figure 8.1: Primary Partitioning of an Attribute Value Domain

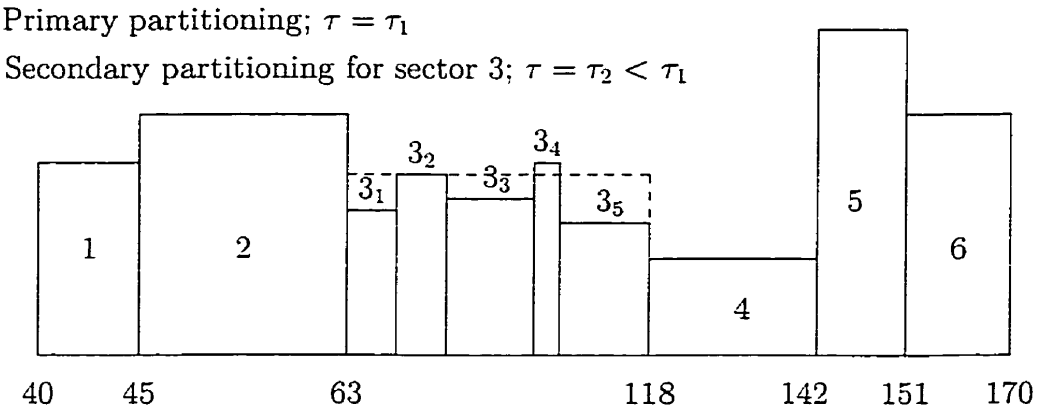


Figure 8.2: Secondary Partitioning of the Value Domain in Sector 3

larger storage space. Instead we partition only that individual sector (sector 3 in this example), using a tolerance value $\tau_2 < \tau_1$. This is shown in Figure 8.2. Here, using the secondary partitioning yields a finer granularization in five smaller sectors, (in this case $3_1, 3_2, 3_3, 3_4$ and 3_5 respectively), thus increasing the estimation accuracy for the queries hitting the attribute values in this domain.

This shows us that by using this hybrid approach instead of a single regular R-ACM sector, we will be able to generate a structure that gives higher estimation accuracy with modest storage requirements, if we appropriately choose a secondary partitioning schema (or tolerance values) for heavily used primary sectors.

We also propose that the automaton (or learning mechanism) can simultaneously

decide to partition some of the sectors using a trapezoidal partitioning method resulting in a new structure that combines the properties of the R-ACM and the T-ACM in a hierarchical manner. As discussed earlier, our goal is to increase the estimation accuracy for the attribute value range which is used more frequently than the others. Since an approximation based on the trapezoidal method would closely reflect the actual data distribution, this strategy would obviously provide a lower estimation error in *this* interval. Figure 8.3 shows an example where one of the heavily used sectors (sector 3) is partitioned using the trapezoidal method. The number of sectors for the secondary partitioning should be based on the available storage.

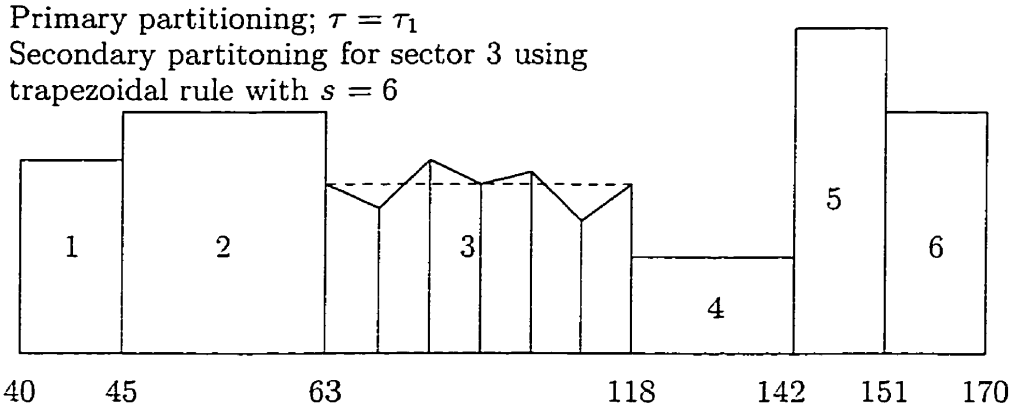


Figure 8.3: Secondary Partitioning of the Value Domain in Sector 3.

By extending these ideas and generalizing, we can also conceive of a way by which we can partition one or more sectors of an ACM based on *both* tolerance values and trapezoids. This would result in another structure where the secondary sectors can be either R-ACMs or T-ACMs. The decision to partition a sector based on a tolerance valued R-ACM or a trapezoid could be based on the learning mechanism's learning process. In other words, during the process of building the ACM, the learning mechanism can choose either an R-ACM or a T-ACM for partitioning the next sector, based on its current knowledge about the data distribution and the estimation accuracy.

8.3 Avenues for Future Work

I. Optimal Build-Parameters in a Static Database System

In Chapter 1 we stated that the selection of an optimal query evaluation plan (QEP) heavily depends on the query optimizer's ability to obtain result size estimates of the relational operations as accurately as possible. Obviously this, in turn, depends on how accurately the attribute cardinality maps, that are used to represent the attribute values, reflect the underlying data distributions. Approximations to the underlying data distribution can be improved by changing various parameters that make up the ACMs. For example, the approximation accuracy can be increased by reducing the sector widths of the ACMs. But obviously this will require proportionately larger memory and secondary storage requirements, and thus it is not practical in a real-world environment. Hence we see that there is a need to find an optimal sector width (or tolerance value in the case of an R-ACM) that balances well with the available storage in the DBMS catalogue. In the T-ACM, in addition to reducing the sector widths, we can also change the slope of the individual trapezoids to obtain an optimal ACM that closely reflects the underlying data distribution. The techniques for finding these optimal build-parameters remains open, although we believe that it is relatively not very difficult in a static database.

II. Optimal Build-Parameters in a Dynamic Database System

Real-world databases are not static. They constantly undergo dynamic changes such as deletions, insertions and updates of records, often changing the pattern of the underlying data distributions. Thus the ACM parameters computed as being optimal today may not be optimal tomorrow. The question then is the following: How do we address the (more realistic) problem of query optimization in a dynamic environment?

In Chapter 7, we discussed that the build-parameters for the R-ACM and the T-ACM can be obtained using the access probabilities of the attribute values in a database system. By considering the various factors influencing the design of an ACM and deriving a suitable cost model, the above work can be extended to incorporate a learning automaton, which can be either deterministic or stochastic. We

can conceivably come up with a scheme which uses learning automata to compute the optimal build-parameters of the ACM models in a dynamic real-world database system environment. These automata should, hopefully, learn the environment (data distributions, types of queries etc) and compute the optimal parameters for generating the ACMs for the attributes in the relations involved. We also believe that the automata can continuously learn the data distribution and improve the behavior of the ACMs in two possible ways listed below.

In the first method, the automaton uses the result size estimations from the queries posed by the DBMS's users. The errors between the actual result sizes and estimated result sizes from the ACMs are used as a measure to either reward or penalize the learning automata, and to adjust the current "optimal" parameters. This can obviously happen during the day time, in the regular business hours of an enterprise. In the second method, the system manager can *randomly* generate selection and join queries and estimate and compare the result obtained by the learning automata. Again the error values obtained can then be used as a measure to reward or penalize the learning automata, and to thus adjust the current optimal parameters. As opposed to the first method this can be implemented to take place during the idle times of the DBMS, usually during the night.

The current state of the art query estimation techniques are, to the best of our knowledge, relatively static. This is also true of the techniques introduced in Chapter 3 and Chapter 4. The information maintained in the DBMS catalogue, although updated during major changes to the database, is never improved on a continuous manner based on the current estimation errors. Hence one promising direction of research may be that of extending the work presented in this thesis to incorporate adaptive feedback techniques to learn the underlying data distribution.

Even though incorporating adaptive techniques will probably help us to find optimal build-parameters for the ACM, as mentioned earlier, practical considerations such as the storage requirements of the ACMs often conflict with the appropriate choice of these parameters. Hence when constructing the ACMs, we need to wisely balance these two conflicting goals: *optimality*, so that generated estimates have the least error, and *practicality*, so that the ACMs can be constructed and maintained

efficiently.

III. Optimal Build-Parameters for Queries with Any Relational Operator
Our theoretical analysis of both the R-ACM and the T-ACM only considered the relational operations, *select* and *join* and the design of the ACMs was mainly aimed at reducing the estimation errors for queries containing only these operators. Even though *select* and *join* are the most frequently encountered operators, other operators such as aggregate operators (*avg, count* etc) also occur in relational queries. Hence a tolerance value which results in acceptable estimation error for certain types of queries may produce an unacceptably large estimation errors for another type of queries. Hence another avenue for further research is to try to find a tolerance value that would produce acceptable estimation errors for a wide range of queries, or at least for the type of queries which are mainly used in a given database system. One can also study whether *learning automata, neural networks* or any other machine learning scheme has the capacity to sufficiently learn such a complex environment (data distributions and type of queries) and so as to efficiently tackle the problem.

IV. Adaptive Techniques Using Hybrid Attribute Cardinality Maps
Another avenue of research may include an adaptive technique which combines the superior properties of both the R-ACM and the T-ACM to form hybrid structures during the process of optimization. These structures could be extremely useful whenever the storage is a premium and the user-queries are predominantly heavy for certain parts of the attribute value domain. At every step of the generation of a hybrid-ACM, the adaptive technique, possibly using a learning automaton, can choose to use either an R-ACM or a T-ACM, depending on the current knowledge of the underlying data distribution and the query types. Although we do have some initial results about this, we opt to rather leave this as a possible avenue for further research.

V. Optimizing ACMs Using Access Probabilities of Attribute Values
The optimization strategies presented in Sections 7.3.1 and 7.3.3 for the T-ACM made the implicit assumption that all the attribute values are accessed equally. This

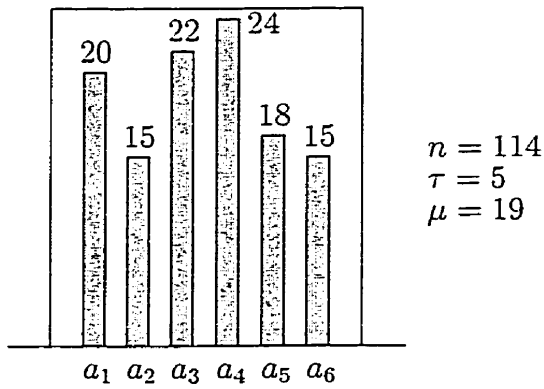


Figure 8.4: Frequency Estimation Using an R-ACM

assumption holds when the queries posed to the system are generated in a truly random manner. But when the ACMs are actually used in a database environment, the above assumption is no longer valid as they are subject to the way the attribute values are themselves accessed by the user-queries. In other words, in designing an ACM for the purpose of query optimization, it is beneficial to improve the estimation accuracy of the set of attribute values which are accessed more frequently even at the sacrifice of the estimation accuracy of the ones which are accessed less frequently. This ensures an even more superior performance for the overall query optimization system.

We briefly describe below a proposed avenue of research showing how an R-ACM and a T-ACM can be optimized based on the access probabilities of the attribute values of underlying data distribution.

The estimation accuracy of a Rectangular ACM is dependent on the tolerance value, τ, used to partition its sectors. Using such a tolerance value to partition an attribute value domain ensures that the frequency values within a given R-ACM sector are close to each other and more importantly close to the mean frequency value of the sector. We propose a new avenue of research for developing an optimization strategy to improve both the average and worst-case estimation errors of an R-ACM based on the access probabilities of the underlying attribute value domain.

Let us consider a hypothetical R-ACM sector with sector width $l = 6$ and tolerance value $\tau = 5$, depicted in Figure 8.4. Let us assume that during a sample session of

100 queries, the attribute values, a_1, a_2, a_3, a_4, a_5 and a_6 are accessed 5, 5, 40, 40, 5, and 5 times respectively. Hence the estimate of the access probabilities of these attribute values are $0.05, 0.05, 0.4, 0.4, 0.05$ and 0.05 respectively. Since the attribute values a_3 and a_4 are accessed more frequently than the other attribute values, an estimation strategy which provides a frequency estimation that is closer to the actual frequencies of these attribute values, at the sacrifice of some estimation accuracy to the less accessed attribute values, is obviously better than a strategy which treats every attribute value equally.

The R-ACM scheme which was discussed in Chapter 3 returns the mean frequency (in this case, $\mu = 19$) of the sector as the frequency estimation for every attribute value within a given sector. Obviously in the above example, the overall estimation accuracy can be improved by returning a frequency estimation which is closer to the actual frequencies of the attribute values a_3 and a_4 (in this case 22 and 24 respectively), than merely returning the running mean frequency. In practice, the access probabilities can be obtained by logging the access information over a long period of time. The estimation errors for individual attribute values can be usually obtained from the query processor module. This is done by comparing the estimated result size from the query optimizer and the true result size generated during the actual execution of a query. Indeed, if $\delta = \sum_{i=1}^{l} p_i \epsilon_i$, it can be maintained for each R-ACM sector, along with the total number of tuples, and periodically updated to reflect the current values of access probabilities.

Considering the optimization of a T-ACM, the optimization strategy based on the least squares error method requires the frequencies of every attribute values in a T-ACM sector in order to compute the sum of the squares of estimation errors. In practice, it is very difficult to maintain the true frequencies of every attribute value. Thus one approach is to use the frequencies of only those attribute values which are accessed more frequently. This requires finding the optimal slope of the straight line which minimizes the least squares errors for *only* those points which represent the frequencies of the attribute values which are accessed more frequently.

Similarly, the optimization strategy based on the average error method can also be improved using the access probabilities of the underlying attribute values.

VI. Discretization of Continuous Attribute Value Domains

Our research work in this thesis is only applicable to discrete attribute value domains. Consequently, many continuous attribute value domains (such as a person's weight, or blood glucose level etc.) that are encountered in real-world database applications need to be discretized before they can make use of the attribute cardinality maps. Many recent research works in the field of Artificial Intelligence have resulted in a number of techniques that can be used to efficiently partition continuous attribute value domains.

Among the many techniques from the AI field, a method proposed by Fayyad and Irani [37] based on classification learning for multi-interval discretization of continuous-valued attributes is particularly appealing for the above purpose. Their method is based on a so called *Minimum Description Length* (MDL) principle [114], where the discretization rule is viewed as a classifying theory in itself, that in turn uses a single attribute and that associates a class with each sub-interval. Another related potential avenue of future research may include combining the above discretization (or classification) technique and an adaptive technique such as the use of a learning automaton that improves the estimation accuracy using a feed-back mechanism.

8.4 Summary

In summary, in this thesis we have introduced some novel techniques for estimating the result sizes of an important subset of query operators in database systems. These techniques are both highly efficient and significantly more accurate than the state-of-the-art estimation techniques currently in use, and can be implemented with minimal cost. Due to these reasons, we anticipate they could prove to be standard tools for query result size estimation in future database systems.

Bibliography

[1] Gennady Antoshenkov. Random sampling from pseudo-ranked $B+$ trees. In *Proceedings of the 18th Conference on Very Large Databases*, pages 375–382, 1992.

[2] M. M. Astrahan and D.D. Chamberlin. Implementation of a structured English query language. In *Communication of the ACM*, volume 18, October 1975.

[3] M.M. Astrahan, M. Schkolnic, and K. Whang. Approximating the number of unique values of an attribute without sorting. In *Information Systems*, volume 12, pages 11–15, 1987.

[4] C.K. Baru and O. Frieder. Database operations in a cube-connected multicomputer system. In *IEEE Transactions on Computers*, volume 38, page 920, 1989.

[5] B. Baugsto and J Greipsland. Parallel sorting methods for large data volumes on a hypercube database computer. In *Proceedings of the 6th International Workshop on Database Machines*, Deauville, France, June 1989.

[6] R. Bayer and E. McCreight. Organization and maintenance of large ordered indexes. In *Acta Inf.*, volume 1, pages 173–189, 1972.

[7] M. Beck, D. Bitton, and W.K. Wilkinson. Sorting large files on a backend multiprocessor. In *IEEE Transactions on Computers*, volume 37, July 1988.

[8] J.L. Bentley and J.H Friedman. Data structures for range searching. In *ACM Computing Survey*, volume 11, pages 397–409, December 1979.

[9] P.A. Bernstein, Goodman. N, Wong. E., Reeve. C.L., and Rothnie. J. B. Query processing in a system for distributed databases. In *ACM Transactions on Database Systems*, volume 6, pages 602–625, December 1981.

[10] D. Bitton-Friedland. *Design, analysis, and implementation of parallel external sorting algorithms.* PhD thesis, University of Wisconsin - Madison, 1982.

[11] M.W. Blasgen and K.P. Eswaran. On the evaluation of queries in a relational database systems. In *IBM Research Report*, volume RJ 1745, 1976.

[12] M.W. Blasgen and K.P. Eswaran. Storage and access in relational databases. In *IBM System Journal*, volume 16, pages 363–377, 1977.

[13] A. Bolour. Optimal retrieval for small range queries. In *SIAM Journal of Computing*, volume 10, pages 721–741, 1981.

[14] H. Boral. Parallelism in Bubba. In *Proceedings of the International Symposium on Databases in Parallel and Distributed Systems*, page 68, Austin, TX, 1988.

[15] H. Boral, W. Alexander, L. Clay, G. Copeland, S. Dansforth, M. Franklin, B. Hart, M. Smith, and P. Valduriez. Prototyping Bubba, A Highly Parallel Database System. In *IEEE Transactions in Knowledge and Data Engineering*, volume 2, 1990.

[16] H. Boral and D.J. DeWitt. Database machines: An idea whose time has passed? In *Proceedings of the International Workshop on Database Machines*, Washington, DC., 1983.

[17] R. Krishnamurthy H. Boral and C. Zaniolo. Optimization of nonrecursive queries. In *Proceedings of the 12th International Conference on Very Large Databases*, pages 128–137, Kyoto, 1986.

[18] L. Breiman, W. Meisel, and E. Purcell. Variable kernel estimates of multivariate densities. In *Technometrics*, volume 19, pages 135–144, 1977.

[19] S. Christodoulakis. Estimating selectivities in data bases. In *Technical Report CSRG-136*, Computer Science Dept, University of Toronto, 1981.

[20] S. Christodoulakis. Estimating block transfers and join sizes. In *ACM SIGMOD 83, Proceedings of the Annual Meeting*, pages 40–54, San Jose, CA, 1983.

[21] S. Christodoulakis. Estimating record selectivities. In *Information Systems*, volume 8, 1983.

[22] S Christodoulakis. Implications of certain assumptions in database performance evaluation. In *ACM Transactions on Database Systems*, volume 9, pages 163–186, 1984.

[23] S.E. Clausen. Optimizing the evaluation of calculus expressions in a relational database system. In *Information Systems*, volume 5, pages 41–54, 1980.

[24] S. Cluet and G. Moerkotte. On the complexity of generating optimal left-deep processing trees with cartesian products. In *Proceedings of the International Conference on Databases Theory*, Prague, 1995.

[25] IBM Corporation. Introduction to IBM direct-access storage devices and organization methods. In *Programming Manual*, pages 1649–06, 1966.

[26] D. Daniels and P. Ng. Distributed query compilation and processing in R*. In *IEEE Database Engineering*, volume 5, 1982.

[27] H.W. Davis and L.E Winslow. Computational power in query languages. In *SIAM Journal of Computing*, volume 11, pages 547–554, 1982.

[28] W. Davison. Parallel index building in Informix OnLine 6.0. In *Proceedings of the ACM-SIGMOD Conference*, page 103, 1992.

[29] L. Devroye. A note on the L_1 consistency of variable kernel estimates. In *Annals of Statistics*, volume 13, pages 1041–1049, 1985.

[30] D. DeWitt, J. Naughton, and D. Schneider. Parallel sorting on a shared nothing architecture using probabilistic splitting. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, 1991.

[31] D.J. DeWitt, S. Ghandeharizadeh, and D. Schneider. A performance analysis of the GAMMA database machine. In *Proceedings of the ACM-SIGMOD Conference*, page 350, 1988.

[32] D.J. DeWitt, S. Ghandeharizadeh, D. Schneider, A. Bricker, H.I. Hsiao, and R. Rasmussen. The GAMMA database machine project. In *IEEE Transactions in Knowledge and Data Engineering*, volume 2, page 44, 1990.

[33] R. Elmasri and S. Navathe. *Fundamentals of Database Systems.* Benjamin/Cummings Publishing Co, Redwood City, CA, 1994.

[34] S. Englert, J. Gray, R. Kocher, and P. Shah. A benchmark of nonstop SQL release 2 demonstrating near-linear speedup and scaleup on large databases. In *Tandem Computers Technical Report*, number 89.4, 1989.

[35] R. Epstein and M Stonebraker. Analysis of distributed data base processing strategies. In *Proceedings of the 6th International Conference on Very Large Data Bases*, pages 92–101, Montreal, 1980.

[36] Christos Faloutsos, Yossi Matias, and Avi Silberschatz. Modeling skewed distributions using multifractals and the 80-20 law. In *Technical Report*, Dept. of Computer Science, University of Maryland, 1996.

[37] U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, San Francisco, 1993. Morgan Kaufmann.

[38] J Fedorowicz. Database evaluation using multiple regression techniques. In *Proceedings of the ACM-SIGMOD Conference*, pages 70–76, Boston, MA, 1984.

[39] J Fedorowicz. Database performance evaluation in an indexed file environment. In *ACM Transactions on Database Systems*, volume 12, pages 85–110, March 1987.

[40] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for database applications. In *Journal of Computer and System Sciences*, volume 31, pages 182–209, 1985.

[41] S. Fushimi, M. Kisuregawa, and H. Tanaka. An overview of the system software of a parallel relational database machine GRACE. In *Proceedings of the International Conference on VLDB*, page 209, Kyoto, Japan, 1986.

[42] Sumit Ganguly, Philip B. Gibbons, Yossi Matias, and Avi Silberschatz. Bifocal sampling for skew-resistant join size estimation. In *Proceedings of the ACM-SIGMOD Conference*, pages 271–281, 1986.

[43] E. Gelenbe and D Gardy. The size of projections of relations satisfying a functional dependency. In *Proceedings of the 8th International Conference on Very Large Data Bases*, pages 325–333, Mexico City, 1982.

[44] M.G. Gouda and U. Dayal. Optimal semijoin schedules for query processing in local distributed database systems. In *Proceedings of the ACM-SIGMOD Conference*, page 164, 1981.

[45] G. Graefe. An extensible and parallel dataflow query processing system. Technical report, Oregan Graduate Center, 1989.

[46] G. Graefe. Encapsulation of parallelism in the Volcano query processing system. In *Proceedings of the ACM-SIGMOD Conference*, page 102, 1990.

[47] G. Graefe. Parallel external sorting in Volcano. Technical Report 459, University of Colorado, Boulder, 1990.

[48] G. Graefe. The cascades framework for query optimization. In *IEEE Data Engineering Bulletin*, volume 18, pages 19–29, September 1995.

[49] N.D. Griffeth. Nonprocedural query processing for databases with access paths. In *Proceedings of the ACM-SIGMOD International Connference on Management of Data*, pages 160–168, Austin, Texas, 1978.

[50] L.M. Haas, P.G. Selinger, E. Bertino, D Daniels, B. Lindsay, G. Lohman, Y. Masunaga, C. Mohan, P Ng, P. Wilms, and R Yost. R*: A research project on distributed relational database management. In *Tech. Report*, IBM Research Division, San Jose, CA, 1982.

[51] P.A.V. Hall. Optimization of a Single Relational Expression in a Relational Database. In *IBM Journal of Research and Development*, volume 20, pages 244–257, May 1976.

[52] M.Z. Hanani. An optimal evaluation of Boolean expressions in an online query system. In *Communication of the ACM*, volume 20, pages 344–347, May 1977.

[53] W. Hasan and H. Pirahesh. Query rewrite optimization in starburst. Technical Report RJ 6367, IBM Research Division, Alamaden Research Center, 1988.

[54] L.M. Hass, W. Chang, G.M. Lohman, J. McPherson, P.F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita. Starburst mid-flight: As the dust clears. Technical Report RJ 7278, IBM Research Division, Alamaden Research Center, 1990.

[55] Peter Hass and Arun Swami. Sequential sampling procedures for query size estimation. In *Proceedings of the ACM-SIGMOD Conference*, pages 341–350, 1992.

[56] Peter Hass and Arun Swami. Sampling-based selectivity estimation for joins using augmented frequent value statistics. In *Proceedings of the IEEE Conference on Data Engineering*, pages 522–531, 1995.

[57] W. Hong and M. Stonebraker. Optimization of parallel query execution plans in XPRS. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, 1991.

[58] W. Hong and M. Stonebraker. Optimization of parallel query execution plans in XPRS. In *Distributed and Parallel Databases*, volume 1, 1993.

[59] Wen-Chi Hou, G. Ozsoyoglu, and E. Dogdu. Error constrained count query, evaluation in relational databases. In *Proceedings of the ACM-SIGMOD Conference*, pages 278–287, 1991.

[60] K.A. Hua and C Lee. Handling data skew in multicomputer database computers using partition tuning. In *Proceedings of the International Conference in VLDB*, page 525, Barcelona, Spain, 1991.

[61] T. Ibaraki and T. Kameda. On the optimal nesting order for computing n-relational joins. In *ACM Transactions on Database Systems*, volume 9, pages 482–502, September 1984.

[62] Yannis Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *Proceedings of the ACM-SIGMOD Conference*, pages 268–277, 1991.

[63] Yannis Ioannidis and Stavros Christodoulakis. Optimal histograms for limiting worst-case error propagation in the size of join results. In *ACM TODS*, 1992.

[64] Yannis Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *ACM-SIGMOD Conference*, pages 233–244, 1995.

[65] Y.E. Ioannidis and Y.C. Kang. Randomized algorithms for optimizing large join queries. In *Proceedings of the ACM-SIGMOD Conference on Management of Data*, pages 312–321, 1990.

[66] Y.E. Ioannidis and E. Wong. Query optimization by simulated annealing. In *Proceedings of the ACM-SIGMOD Conference on Management of Data*, pages 9–22, 1987.

[67] B. R. Iyer and D.M. Dias. System issues in parallel sorting for database systems. In *Proceedings of the IEEE Conference on Data Engineering*, page 246, 1990.

[68] M. Jarke and J.W. Schmidt. Evaluation of first-order relational expressions. In *Technical Report 78*, Fachbereich Informatik, Universitaet Hamburg, Hamburg, FRG, 1981.

[69] N. Kamel and R. King. A model of data distribution based on texture analysis. In *Proceedings of ACM-SIGMOD Conference*, pages 319–325, May 1985.

[70] W. Kim. On optimizing an SQL-like nested query. In *ACM Transactions on Database Systems*, volume 7, pages 443–469, September 1982.

[71] M. Kitsuregawa and Y. Ogawa. Bucket spreading parallel hash: A new, robust, parallel hash join method for skew in the super database computer (SDC). In *Proceedings of the International Conference on VLDB*, page 210, Brisbane, Australia, 1990.

[72] M. Kitsuregawa, H. Tanaka, and T Motooka. Application of hash to database machine and its architecture. In *New Generation Computers*, volume 1, page 63, 1989.

[73] M. Kitsuregawa, W. Yang, and S. Fushimi. Evaluation of 18-stage pipeline hardware sorter. In *Proceedings of the 6the International Workshop on Database Machines*, Deauville, France, June 1989.

[74] A. Klug. Access paths in the ABE statistical query facility. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 161–173, Orlando, Florida, 1982.

[75] D.E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.

[76] R. P. Kooi. *The optimization of queries in relational databases.* PhD thesis, Case Western Reserve University, 1980.

[77] Erwin Kreyszig. *Advanced Engineering Mathematics.* John Wiley & Sons, New York, 6th edition, 1988.

[78] A. Kumar and M. Stonebraker. The effect of join selectivities on optimal nesting order. In *ACM-SIGMOD Record*, volume 16, pages 28–41, March 1987.

[79] M.S. Lakshmi and P.S. Yu. Effect of skew on join performance in parallel architectures. In *Proceedings of the International Symposium on Databases in Parallel and Distributed Systems*, page 107, Austin, TX, 1988.

[80] M.S. Lakshmi and P.S. Yu. Effectiveness of parallel joins. In *IEEE Transactions on Knowledge and Data Engineering*, volume 2, page 410, 1990.

[81] R.S.G. Lanzelotte, P. Valduriez, and M. Zait. On the effectiveness of optimization search strategies for parallel execution spaces. In *Proceedings of the 19th VLDB Conference*, pages 493–504, Dublin, Ireland, 1993.

[82] Lecoutre, Jean-Pierre. The L^2-Optimal Cell Width for the Histogram. In *Statistics and Probability Letters*, volume 3, pages 303–306, 1985.

[83] Yibei Ling and Wei Sun. An evaluation of sampling-based size estimation techniques for selections in database systems. In *Proceedings of the IEEE Conference on Data Engineering*, pages 532–539, 1995.

[84] Richard Lipton and Jefferey Naughton. Estimating the size of generalized transitive closures. In *Proceedings of the 15th VLDB Conference*, pages 165–172, 1989.

[85] Richard Lipton and Jefferey Naughton. Query size estimation by adaptive sampling. In *Proceedings of the 9th ACM Symposium on Priciples of Database Systems*, pages 40–46, 1990.

[86] Richard Lipton, Jefferey Naughton, and Donavan Schneider. Practical selectivity estimation through adaptive sampling. In *Proceedings of the ACM-SIGMOD*, pages 1–11, 1990.

[87] L. Mackert and G Lohman. R* optimizer validation and performance evaluation for distributed queries. In *Proceedings of the 12th International Conference on Very Large Databases*, Kyoto, Japan, August 1986.

[88] L. Mackert and G Lohman. R* optimizer validation and performance evaluation for local queries. In *Proceedings of the ACM-SIGMOD Conference*, pages 84–95, Washington, DC, 1986.

[89] A. Makinouchi, M. Tezuka, H. Kitakami, and S Adachi. The optimization strategy for query evaluation in RDB/V1. In *Proceedings of the 7th International Conference on Very Large Databases*, pages 518–529, New York, September 1981.

[90] M.V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. In *ACM Computing Surveys*, volume 20, pages 192–221, 1988.

[91] A.W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and Its Applications.* Academic Press, New York, 1979.

[92] W.J. McKenna. *Efficient Search in Extensible Database Query Optimization: The Volcano Optimizer Generator.* PhD thesis, University of Colorado, Boulder, 1993.

[93] Merrett, T.H. Database Cost Analysis: A Top-Down Approach. In *Proceedings of the ACM-SIGMOD Conference on Management of Data*, pages 135–143, 1977.

[94] Merrett, T.H., and Otoo, E. Distribution Models of Relations. In *5th International Conference on Very Large Database*, pages 418–425, 1979.

[95] Merrett, T.H., Otoo, E.J., Thiyagarajah, M., Valdivia-Martinez, A., Zhao, X.Y. Inter-operation of Heterogeneous GIS via Database Communication: A Prototype. In *8th International Symposium on Spatial Data Handling (SDH'98)*, 1998.

[96] A.I. Montgomery, D.J. D'Souza, and S.B Lee. The cost of relational algebraic operations in skewed data: Estimates and experiments. In *Information Processing*, volume 83, pages 235–241, 1983.

[97] D. Moore and J. Yackel. Consistency properties of nearest neighbor density function estimates. In *Annals of Statistics*, volume 5, pages 143–154, 1977.

[98] M. Muralikrishna and David J Dewitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of ACM-SIGMOD Conference*, pages 28–36, 1988.

[99] B. Muthuswamy and L. Kerschberg. A DDSM for relational query optimization. In *Proceedings of the ACM Annual Conference*, October 1985.

[100] National Basket Ball Association. NBA Players Performance Statistics. ftp:olympos.cs.umd.edu, 1992.

[101] P.M. Neches. The Ynet: An interconnect structure for a highly concurrent data base computer system. In *Proceedings of the 2nd Symposium on the Frontiers of Massively Parallel Computation*, Fairfax, VA, 1988.

[102] K.E. Niebuhr and S.E. Smith. N-ary joins for processing Query by Example. In *IBM Technical Disclosure Bullettin*, volume 19, pages 2377–2381, 1976.

[103] Frank Olken and Doron Rotem. Simple random sampling from relational databases. In *Proceedings of the 12th International Conference on Very Large Data Bases*, August 1996.

[104] K. Ono and G.M. Lohman. Measuring the complexity of join enumeration in query optimization. In *Proceedings of the 16th VLDB Conference*, pages 314–325, Brisbane, Australia, 1990.

[105] B. John Oommen and Murali Thiyagarajah. Benchmarking Attribute Cardinality Maps for Database Systems Using the TPC-D Specifications. Technical Report TR-99-07, School of Computer Science, Carleton University, Ottawa, Canada, Feb 1999.

[106] B. John Oommen and Murali Thiyagarajah. The Rectangular Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. In

International Database Engineering and Applications Symposium, IDEAS'99, Montreal, Canada, August 1999.

[107] B. John Oommen and Murali Thiyagarajah. The Rectangular Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. Technical Report TR-99-01, School of Computer Science, Carleton University, Ottawa, Canada, Jan 1999.

[108] B. John Oommen and Murali Thiyagarajah. The Trapezoidal Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. Technical Report TR-99-04, School of Computer Science, Carleton University, Ottawa, Canada, Feb 1999.

[109] M.T. Ozsu and P. Valduriez. Distributed database systems: Where are we now. In *IEEE Computer*, volume 24, August 1991.

[110] M.T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall, Englewood Cliffs, N.J, 1991.

[111] F.P. Palermo. A database search problem. In *Proceedings for the 4th Symposium on Computer and Information Science*, pages 67–101, Miami Beach, Florida, 1972. AFIPS Press, Reston, VA.

[112] Gregory Piatetsky-Shapiro and Charles Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proceedings of ACM-SIGMOD Conference*, pages 256–276, 1984.

[113] P Richard. Evaluation of the size of a query expressed in relational algebra. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 155–163, Ann Arbor, Michigon, 1981.

[114] J Rissanen. A universal prior for integers and estimation by minimum description length. In *Annals of Statistics*, volume 11, pages 416–431, 1983.

[115] A. Rosenthal and D Reiner. An architecture for query optimization. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 246–255, Orlando, Florida, 1982.

[116] G.M. Sacco and M Schkolnick. A mechanism for managing the buffer pool in a relational database system using the hot set model. In *Proceedings of the 8th International Conference on Very Large Databases*, pages 257–262, Mexico City, 1982.

[117] W. Samson and A Bendell. Rank order distributions and secondary key indexing (extended abstract). In *Proceedings of the 2nd International Conference on Databases*, Cambridge, England, 1983.

[118] K.L Schenk and J.R Pinkert. An algorithm for servicing multi-relational queries. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 10–20, Toronto, Canada, 1977.

[119] W. Scheufele and G. Moerkotte. On the complexity of generating optimal plans with cross products. In *Proceedings of the 23rd International Conference on Very Large Databases*, Athens, 1997.

[120] J.W. Schmidth. Parallel processing of relations: A single-assignment approach. In *Proceedings of the 5th International Conference on Very Large Databases*, pages 398–408, Rio de Janeiro, October 1979.

[121] Scott, David. On Optimal and Data-based Histograms. In *Biometrika*, volume 66, pages 605–610, 1979.

[122] P. Selinger, D.D. Chamberlin M.M. Astrahan, R.A. Lorie, and T.G. Price. Access Path Selection in a Relational Database Management System. In *Proceedings of ACM-SIGMOD Conference*, 1979.

[123] S. Seshadri and J. Naughton. Sampling issues in parallel database systems. In *Extending Database Technology*, pages 328–343, 1992.

[124] Amit Shukla, Prasad Deshpande, Jeffrey F. Naughton, and Karthikeyan Ramaswamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proceedings of the 22nd International Conference on Very Large Databases*, pages 522–531, 1996.

[125] J.M. Smith and P. Chang. Optimizing the performance of a relational algebra database interface. In *Communications of the ACM*, volume 18, pages 568–579, 1975.

[126] M. Stonebraker, P. Aoki, and M. Seltzer. Parallelism in XPRS. Technical Report 89/16, University of California, Berkeley, 1989.

[127] M. Stonebraker, R. Katz, D. Patterson, and J. Ousterhout. The design of XPRS. In *Proceedings of the International Conference on VLDB*, Los Angeles, 1988.

[128] H. Sturges. The choice of class interval. In *Journal of American Statistics Association*, pages 65–66, 1926.

[129] Wei Sun, Yibei Ling, Naphtali Rishe, and Yi Deng. An instant and accurate size estimation method for joins and selections in a retrieval-intensive environment. In *Proceedings of ACM-SIGMOD Conference*, pages 79–88, 1993.

[130] A.N. Swami and A Gupta. Optimization of large join queries. In *Proceedings of the ACM-SIGMOD Conference on Management of Data*, pages 8–17, 1988.

[131] R Tapia and J Thompson. *Nonparametric Probability Density Estimation.* John Hopkins University Press, Baltimore, MD, 1978.

[132] Murali Thiyagarajah and B. John Oommen. Prototype Validation of the Rectangular Attribute Cardinality Map for Query Optimization in Database Systems. In *International Conference on Business Information Systems - BIS'99*, Poznan, Poland, April 1999.

[133] Murali Thiyagarajah and B. John Oommen. Prototype Validation of the Trapezoidal Attribute Cardinality Map for Query Optimization in Database Systems.

In *International Conference on Enterprise Information Systems - ICEIS'99*, Setubal, Portugal, March 1999.

[134] Transaction Processing Council (TPC). TPC-D Benchmark Specification. Feb 1998.

[135] U.S. Census Bureau. U.S. CENSUS Database. 1997.

[136] B. Vander Zander, H. Taylor, and D. Biton. Estimating block accesses when attributes are correlated. In *Proceedings of the 12th International Conference on Very Large Databases*, pages 119–127, Kyoto, Japan, August 1986.

[137] E. Wegman. Density estimation. In *Encyclopedia of Statistical Sciences*, volume 2, 1983.

[138] W. Wertz. *Statistical Density Estimation: A Survey.* Vandenhoeck Ruprecht, Gottingen, 1978.

[139] A.N. Wilschut. *Parallel query execution in main memory database systems.* PhD thesis, University of Tweuk, The Netherlands, 1993.

[140] A.N. Wilschut and P.M.G. Apers. Dataflow query execution in a parallel main memory environment. In *Proceedings of the International Symposium on Databases in Parallel and Distributed Systems*, Dublin, Ireland, 1991.

[141] E. Wong and K. Youseffi. A Strategy for Query Processing. In *ACM Transactions on Database Systems*, volume 1, pages 223–241, 1976.

[142] S. Yao. Optimization of query evaluation algorithms. In *ACM Transactions of Database Systems*, volume 4, June 1979.

[143] G. K. Zipf. *Human Behavior and the Principle of Least Effort.* Addision-Wesley, Reading, MA, 1949.