Systematic Literature Review:
Cost Estimation in Relational Databases

Author:
Rohith Kumar Kurella

March 26, 2018

Advisors:
Prof. Dr. rer. nat. habil. Gunter Saake
M.Sc Andreas Meister

Reviewer:
Dr.-Ing. Sandro Schulze
Department of Technical and Business Information Systems
Kurella Rohith Kumar:
Systematic Literature Review: Cost Estimation in Relational Databases
# Contents

List of Figures .................................................. v
List of Tables .................................................. vi

1 Introduction .................................................. 1
   1.1 Motivation .................................................. 1
   1.2 Goal of this thesis ......................................... 2
   1.3 Structure of the thesis ....................................... 2

2 Background .................................................. 3
   2.1 Query processing ............................................ 3
   2.2 Query optimization .......................................... 4
   2.3 Cost estimation .............................................. 6
   2.4 Cardinality estimation ....................................... 6
   2.5 Basics of SLR ................................................. 7

3 Specifics of SLR .............................................. 9
   3.1 Planning the review ........................................... 9
       3.1.1 Primary Search Phase .................................. 10
           3.1.1.1 Key Terms ....................................... 10
           3.1.1.2 Sources for specific areas of cost models .......... 10
           3.1.1.3 References included or excluded and rationale for selec-
in ........................................................... 11
       3.1.2 Secondary Search Phase ................................ 11
   3.2 Implementing the review ..................................... 11

4 Results of Systematic literature review .............. 12

5 Selectivity estimation .................................... 17
   5.1 Estimating the size of operators in a query plan .......... 17
   5.2 Approaches of selectivity estimation ....................... 20
   5.3 Comparison of approaches .................................. 25
   5.4 Advantages and disadvantages of approaches ................ 26
   5.5 Limitations ................................................ 28
   5.6 Scope of further research .................................. 29

6 Cost estimation ............................................. 30
   6.1 Estimating the Cost of operators in a query plan: .......... 30
   6.2 Classical approach of Cost estimation ...................... 34
   6.3 Cost model based on the underlying hardware technology .... 36
   6.4 Cost estimation in commercial database systems ............ 37
       6.4.1 Oracle ................................................ 37
       6.4.2 Microsoft SQL server .................................. 39
       6.4.3 PostgreSQL ........................................... 40
       6.4.4 DB2 .................................................. 41
## Contents

6.4.5 SQL anywhere ........................................... 42  
6.5 Comparison of approaches ................................. 42  
6.6 Advantages and disadvantages of the approaches .......... 43  
6.7 Limitations .................................................. 43  
6.8 Scope of further research ................................... 44  

7 Conclusion ...................................................... 45  

8 Future work .................................................... 46  

Bibliography ...................................................... 47
List of Figures

2.1 Query Processing Steps ........................................ 4
2.2 Query Optimization ............................................. 5

4.1 Summary of Primary Search Results-Stage 1 ............... 13
4.2 Summary of Primary Search Results-Stage 2 ............... 14
4.3 Summary of Primary Search Results-Stage 3 ............... 15
4.4 Summary of Secondary Search Results ...................... 16
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Parameters for size estimation in selection operation</td>
<td>18</td>
</tr>
<tr>
<td>6.1</td>
<td>Cost function parameters in Join operation</td>
<td>34</td>
</tr>
<tr>
<td>6.2</td>
<td>Cost model parameters</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Cost model parameters in Oracle</td>
<td>39</td>
</tr>
<tr>
<td>6.4</td>
<td>Cost model parameters in PostgreSQL</td>
<td>41</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Motivation

Relational databases are used for the storage and retrieval of data. Users input queries for retrieving the required data using SQL (a declarative language). Data is retrieved more often than stored. In most cases, database is accessed by several users at the same time. This creates a need for efficient processing of queries. When users input queries using SQL, a parser parses the query, checks for its syntax and transforms it into a logical plan and then into a physical plan. A plan with lowest cost is selected by the query optimizer and passed onto the query execution engine for execution. To select, cost estimation is needed for estimating the cost of the plans. A classical approach is to estimate the cost by estimating the amount of used resources (i.e. CPU, I/O and memory) for query processing. There has been extensive research in the field of query optimization [AEM05, WZRS99, CY09, L15] since 1970s. This is a huge amount of research with significant findings. These findings focused on certain aspects of query optimization and cost estimation in databases in general [MJB93, PSS00, CE04, MJL04]. It is important to understand which of these findings can provide answers and serve for further research. This creates a need for a trustworthy and reliable methodology which can be used to evaluate and interpret all the relevant reported research. Systematic literature reviews are objective, systematic and transparent in approach. Systematic literature review has been widely used by researchers to summarize the available reports concerning a topic of interest and to find gaps in the focused research that could be further investigated [SS07, ABMB14, AKE15, PARN14, BP13, SNT*08].
1.2 Goal of this thesis

In this thesis, we perform a systematic literature review on cost estimation in relational databases. Cost estimation is an important step in query optimization process. Clear objectives need to be defined for the successful execution of the systematic literature review process. The main objectives of the thesis are

- To provide an overview of the cost estimation in databases with the details of the approaches
- To compare the alternative approaches of cost estimation
- To find the limitations of the approaches
- To understand the scope for further research which contributes towards building better query optimizers.

1.3 Structure of the thesis

The following sections focus on the objectives of the systematic literature. In Section 2, we provide a background of cost estimation in relational databases. It includes an overview of the query processing, query optimization, cost estimation and the basics of systematic literature review (SLR) process. In Section 3, we include the specifics of the performed SLR i.e. search process, used key terms, sources for literature review, inclusion and exclusion criteria. In section 4, we provide results of the systematic literature review. In Section 5, we discuss the selectivity estimation which includes estimating the sizes of the operators in a query plan, different approaches in selectivity estimation. In addition to the details of approaches, a comparison of the approaches, advantages and disadvantages between the approaches, limitations and scope of further research are provided. In Section 6, we discuss the estimation of the costs of operators in a query plan, different approaches in cost estimation. Furthermore, a comparison of approaches in cost estimation, advantages and disadvantages of different approaches, limitations and scope of further research are provided. In Section 7, we conclude the systematic literature review. In Section 8, we discuss future work.
2. Background

2.1 Query processing

Users input queries using SQL to retrieve required information from relational databases. A parser parses the query, checks for its syntax and transforms it into a logical plan and then into a physical plan. Query optimization is performed to minimize the resource consumption needed for query execution which involves a logical and a physical optimization. Logical optimization is used for determining the order of performing operations in a query and physical optimization is used for selecting the right algorithms for performing operations to minimize the execution costs. There are different methods for accessing data from tables like sequential scan or random or full table scan. Once the operators are selected, it is important to define the flow of the intermediate results to subsequent operations [SRV95, SS07, MM96, Rad02]. Figure 2.1 on the following page depicts the query processing steps [Ben11].

Results for queries are generated by accessing the database and extracting only the necessary data by applying the conditions mentioned in the query. Retrieving the data for a query is complex as it depends on the data storage. Moreover, there can be different alternative ways to execute a query. This means that the time required for processing the queries vary with the way data is extracted. This variation in time can be from a fraction of a second to several hours [Ben11]. This brings us to the main point that it is essential to find out the optimal way of processing a query in minimum time. Finding out the optimal plan out of several alternatives is time intensive. To solve this problem, query optimization is used. The objectives of query optimization can be either the time required for selecting an optimal query plan or the quality of the query plan. Comparing and selecting an efficient plan is possible only when values of the costs involved in query plans are available [Joh04, ETKK92, FYP99, SPM02]. Practically, it is not possible to consider estimating the cost of every plan. Costs of query plans are estimated and the plan which has the least cost is picked and passed onto the query execution engine. Query execution is the final step towards achieving the query results [E96, Vik16]. In general, query plans are built in the form of a tree like structure with nodes at each level. These plan nodes represent the operations needed for query execution.

The tree structure of the query plans facilitates the flow of results from the nodes at the bottom level to nodes at the top level. There can be either one or more nodes at each level. The results flow from bottom towards the node at the top level [FYP99, PND02]. Query plans can have join or sort operations and others. There are different ways of accessing data i.e. index access and full table scan. Query plans are a result of the combination of these approaches. This makes the search space quite large [Goe93, SNT+08, MJ84, WSD12].
2. Background

There are too many plans to consider and accurately estimating the costs of these is quite difficult. Accurate estimation of the cost of a plan involves estimation of values of various parameters. These parameters are divided into three categories i.e. properties of the data, properties of the query components and properties of the run-time environment. Properties of data include cardinalities of tables and distributions of the values. Sizes of selectivities of predicates are part of the properties of the query components. Properties of the run-time environment are amount of available memory, processor speed, access characteristics of secondary storage.

2.2 Query optimization

Query optimization is a process of minimizing the total cost or the total response time for the execution of a query [AS02, SN14, Chr01, WRMC92a, PV13, ACF94, DICA10, EW06, Zeh07, Sur98].

A tree structure is used for computing costs i.e. costs from the operators at the lower level are added to get the cost of the next higher level of operators [S+02, MM13, LG94, PK14, SK99, Bin79]. The same kind of approach applies for estimating the cardinality of operators in higher level. Figure 2.2 on the next page depicts the process of query optimization [TP11] with a cost model as an input. There are often several assumptions considered during query optimization, for example there is uniform distribution of data. This does not hold true in most cases. These assumptions can lead to cost estimations which are inaccurate. To avoid such inaccurate estimations, more advanced, histograms, sampling, etc are used.
For the sake of completeness, the step of join ordering in query optimization is mentioned. Join ordering [HF11, LRH14, KJOFO5, MJMA97] is a complex operation in query optimization and there has been extensive research since the seventies. It is a process of calculating the optimal join order i.e. the order of joining tables for the execution of a query [HMCKL91, JG93, SAL00]. The number of joins can increase the size of the search space as there many possible ways of executing a query depending upon the number of tables joined [Ben11, MGA97, JCKL93, AB94]. Dynamic programming algorithm proposed by IBM’s System R database project is used by most of the query optimizers to solve the problem of join order [E96, GT08, SSR98]. A major challenge to join-order optimization [MMC96, TG00] is that the cost of an operation depends on properties of its intermediate inputs as the cost of a join is a function of the size of its inputs [AGJ13, NSJR15].
2.3 Cost estimation

The cost-based optimization does not follow predefined set of ranked rules but rather compares the overall cost of execution of the statement as the main selection criteria for the optimal plan. The plan which involves less cost is selected and executed [NS98, MRCM99, NAD+14, MMM13, HSL02]. Cost models are used for improving the efficiency of the query optimization in database systems. Cost models estimate the various costs involved in query execution. But the question of why estimation is needed in executing a query arises. Cost models provide the inputs (statistics about the distribution of data and the algorithms for executing operators in a query plan) needed for estimating the costs of query plans and thereby facilitating in the selection of optimal plans [FD95, NSJ13]. An important assumption which underlies the execution of a query plan is that it consumes less resources and/or minimum time when less data has to be processed [YET00, MJBS06, PWMR05, YET97]. It is not always clear, which sequence of operations need less amount of data and time for query execution. Statistics can help to resolve this constraint. In addition to the statistics, costs of executing the algorithms and the costs of accessing data are computed. The cost of a query plan is calculated as the summation of all these costs [S+02, SPL02, GFZH96a, HGA98a, DIA12a].

Cost is the result of the cost of the access method and the cardinality. To derive the cumulative costs and cardinalities estimates for a query plan two important cost factors are considered [TA13, M98, AS02, WRMC92b, KYR90, AG13]. First, the total cost i.e. the cost in seconds to execute an operator and retrieve the complete set of results. Second, the cardinality i.e. the estimated result cardinality of that operation. Cost formulas consider the details of CPU usage and I/O cost [AS03, Sum98, NJ98, KABA99]. Cost formulas have variables that need to be instantiated to determine cost. This would include the cardinality of the input streams to the operator and statistics about the data to which the operator is being applied [DMAA09, HPPMM04, HJA16, N01, BS05, YKK07]. Cost is used as a most important measure in the SQL statement optimization process by developers and DBAs. Cost can be treated as an internal measure that is used in the process of selecting an optimal plan. It does not mean that lower cost means faster execution and vice versa [AFA+10].

2.4 Cardinality estimation

Cardinality estimation is a problem within the query optimization process. The time taken for executing a query plan depends on the size of the base relations and that of the intermediate relations generated in the process. Details of the size of the base relations are available but it is not possible to compute the size of the intermediate results without executing a query plan. For this reason, cardinality estimation is needed for estimating the size of intermediate results [GM08, AGJ13, AFS+97]. Cardinality estimation depends on the selectivities of underlying selections and joins in a query.

Cardinality of the input streams is either computed using cost formulas for the input operators or using statistics if the input is a base table. Therefore, statistics are crucial for any cost-based optimizer. Statistics include information about collections, i.e. the base cardinality and about attributes i.e. information about the distribution
of data values. At the core lies a set of statistics that describe the data and in the
next step, these statistics feed cost formulas to compute selectivity estimates, CPU
and I/O costs. Finally, in the outer layer, operator costs are computed from the
cost formulas and these operator costs would ultimately result in the plan costs
[TMF99]. There are assumptions [NAD+14, MMVM01, PSJ06, CCCFS05], for
element, cardinality estimation assumes attribute value independence assumption
(i.e. predicates of different attributes are independent) for computing selectivity
of predicates in a query. Selectivity of a conjunction of predicates with several
attributes is estimated by multiplying the selectivities of the individual predicates.
Selectivity can be understood as the number of records that satisfy the requirement
of a given query (the number of rows in a table that take part in an operation). In
order to estimate cardinality, it is essential to understand the selectivity of predicates
in query statement [RLCG06, JFA90]. A common approach towards selectivity
computation is to consider the number of distinct values present in a column and
take the reciprocal value of that count. The value of selectivity lies between 0.0 and
1.0. The value of 0.0 does not consider any rows while the value 1.0 include all rows
in operator execution in a query plan.
One dimensional histograms are not efficient in gathering the details about the
correlated relations i.e multi-dimensional distributions. Several techniques such as
multidimensional histograms and graphical models have been proposed to solve this
problem. These techniques suffered limitations as the maintenance of the number
of interactions between different attributes is complex are more than the count of
the attributes. Maintaining statistics is possible only for few interactions. Moreover,
these techniques do not provide any information on the uncertainty of the cardinality
estimates as it is essential factor behind selection of a query plan.

2.5 Basics of SLR

Systematic literature reviews involve a systematic search process to identify the stud-
ies which address the research objectives and present the findings of search results
transparently. This is achieved by following an inclusion and exclusion criteria in
the review which is explicitly stated and consistently implemented. This approach
would help minimize bias by allowing the readers to understand the process fol-
lowed and how the findings are assimilated to reach to conclusions or discussions.
New findings can be integrated by the researchers in future using this methodology
[ZLV+16, LMJ+14]. The use of systematic literature reviews in research has started
in medical field. It has gained a lot of attention from researchers in other domains
as well. This can be linked to its exhaustive approach towards selecting the relevant
studies, collating and summarizing the findings of these studies. Both qualitative
and quantitative analysis of the studies can be performed and there are different
tools provided for the same. Most of the researchers find it difficult to summarize
the findings of the relevant studies and to understand the progress of the existing
research and the scope for further research. Systematic Literature reviews can fa-
cilitate in understanding the gaps in research and provide the right direction for
researchers.
Several standards have been proposed to make it more flexible. The process of SLR involves three main phases [ABMB14, AKE15, PARN14, BP13]:

- Planning the review
- Implementing the review
- Reporting the review

The phases are explained as below:

1. Planning the review: This is the initial phase which helps in defining the steps to be followed for performing the SLR. The first step is to identify the research objectives which define the need for SLR. After analyzing the need for SLR, a search for any existing SLRs is carried out on different databases and repositories. The process is streamlined by preparing a list of databases and repositories and the details of the inclusion and exclusion criteria.

2. Implementing the review: This phase of SLR involves identifying the primary studies and preparing a data schema for the extracting data from the search process. The search process must be in line with the research aim and needs to be checked if the article targets the right audience. The methodology used should be able to address the stated research objectives and if any assumptions persist, then they need to be accounted. This facilitates in finding the gaps in the existing research and provides a direction for researchers. The data sources used by the study include online databases, research journals, conferences and grey literature. Search process is divided into primary search and secondary search. In primary search, databases and repositories are searched for topic related papers. Secondary search involves finding out citations within the papers found in primary search. To facilitate the search process, key terms are identified.

3. Reporting the review: Post implementation of the review, as a concluding step the details of the SLR are documented. This is the final phase that summarizes the findings as well as the results of all the steps performed during the SLR.
3. Specifics of SLR

Defining a research aim is a crucial step towards defining the need for performing a systematic literature review. This systematic literature review is performed

- To provide an overview of the cost estimation in relational databases with the details of the approaches
- To compare the alternative approaches in cost estimation in relational databases.
- To find out the limitations of the approaches.
- To provide the scope for further research which contributes towards building better query optimizers.

The research has been carried out based on the guidelines proposed by [BP13], which are widely accepted in software engineering. Based on these guidelines, the review process has been divided into three phases: Planning, Implementing and reporting [DP06, ABMB14, LMJ+14, BP13].

3.1 Planning the review

The planning activity deals with developing the search phases, search terms, sources for the literature search, inclusion and exclusion criteria. The overall search phase is divided into two phases to search the vast majority of published work which addresses the research aim [ZLV+16, LMJ+14]:

- Primary Search
- Secondary Search
3.1.1 Primary Search Phase

The search process for relevant studies is carried out in online repositories, journals, search engine, theses and conferences.

3.1.1.1 Key Terms

Selecting the relevant search terms is a crucial step in systematic literature review. It helps to find as many potentially relevant articles as possible for inclusion. The procedure is summarized as below:

1. Search terms are selected from the research objectives:
   - cost estimation in relational databases
   - cost models in relational databases
   - cost parameters in relational databases
   - cost factors relational in databases
   - cost components in relational databases
   - logical cost in relational databases
   - physical cost in relational databases
   - temporal cost in relational databases
   - algorithmic cost in relational databases
   - cost based query optimization in Oracle
   - cost based query optimization in DB2
   - cost based query optimization in SQL server
   - cost based query optimization in PostgreSQL
   - cost based query optimization in SQL anywhere.
   - cost based query optimization in relational databases

2. Keywords found within articles also become relevant terms
   - cardinality estimation in relational databases
   - selectivity estimation in relational databases
   - database statistics in relational databases

3.1.1.2 Sources for specific areas of cost models

- Repositories - ACM Digital library, IEEE Xplore, Springer Link
- Online Search Engine - Google scholar
3.2 Implementing the review

3.1.1.3 References included or excluded and rationale for selection

Inclusion criteria and exclusion criteria is required for selecting the relevant papers.

**Inclusion Criteria**

1. All papers that are published in English language
2. Papers that focuses on cost models in databases, approaches in cost estimation in databases
3. All published papers that have the potential of answering at least one of the objectives of the review.

**Exclusion Criteria**

1. Papers that are not published in English language.
2. Duplicate papers – papers which are repeated within the search results
3. Papers that do not have any link with the objectives of the review.
4. In the Secondary search phase, relevant papers cited within the literature are searched.

3.1.2 Secondary Search Phase

In the secondary search phase, relevant papers cited within the literature are searched. This kind of search provides papers which not only have relevance but also facilitates in finding the most of relevant work published.

3.2 Implementing the review

The next phase of systematic literature review involves extracting the data from the literature found through the search process. The implementation phase focuses on extracting the data from the literature searched. Finally, the last phase describes how the final report has been elaborated [ZLV+16, PARN14, AKE15]. Here, the data extraction has to be specific and clear as there are significant numbers of papers obtained from the literature search. For this purpose, a data schema has been created, which is stated as below:

- **Title and author**
- **Details regarding the publication, whether it is book, journal, conference or technical paper.**
- **Year in which the study has been published**
- **Methodology or the approach on which the research is based and the advantages and disadvantages of other alternatives**
- **Any related work which lists the studies cited within the research study.**
- **Citations of the study in other research studies**
- **Information about the scope of future work related to the research objectives.**
4. Results of Systematic literature review

The results of the systematic literature review process are provided below with the details of each step.

Primary search has been carried out on online repositories and search engine using the key terms mentioned above. In the first stage of filtration, studies which are not in the English have been excluded. In Chapter 4 on the facing page the summary of primary search results of stage is provided.

In Stage 2, studies which were found to be repeated between the search results of different sources have been eliminated. Those details are provided in Chapter 4 on page 14.

In Stage 3, the rest of the papers have been searched for its relevance and any link with the objectives of the review. The studies included topics relating to relational databases, temporal databases, query optimization, join-order optimization, object-oriented databases, with more number of papers on XML Databases, object-relational databases and spatio-temporal databases. Studies which focus on the cost estimation in relational databases have been considered. The number of relevant and irrelevant papers can be found in Chapter 4 on page 15.

In the secondary search phase, relevant papers cited within the literature are searched. In Chapter 4 on page 16, details of the papers found within the studies of the primary search are included.
<table>
<thead>
<tr>
<th>S.no</th>
<th>Key terms</th>
<th>ACM Digital library</th>
<th>IEEE Xplore</th>
<th>Springer Link</th>
<th>Google scholar</th>
<th>Non English</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>selectivity estimation in relational databases</td>
<td>343</td>
<td>264</td>
<td>218</td>
<td>1477</td>
<td>13</td>
<td>2289</td>
</tr>
<tr>
<td>2</td>
<td>cost estimation in relational databases</td>
<td>141</td>
<td>164</td>
<td>124</td>
<td>1017</td>
<td>3</td>
<td>1443</td>
</tr>
<tr>
<td>3</td>
<td>cost parameters in relational databases</td>
<td>32</td>
<td>30</td>
<td>28</td>
<td>240</td>
<td>1</td>
<td>329</td>
</tr>
<tr>
<td>4</td>
<td>cost components in relational databases</td>
<td>27</td>
<td>25</td>
<td>25</td>
<td>279</td>
<td>0</td>
<td>356</td>
</tr>
<tr>
<td>5</td>
<td>logical costs in relational databases</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>physical costs in relational databases</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>52</td>
<td>0</td>
<td>67</td>
</tr>
<tr>
<td>7</td>
<td>temporal cost factors in relational databases</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>algorithmic costs in relational databases</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>14</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>cardinality estimation in relational databases</td>
<td>81</td>
<td>48</td>
<td>58</td>
<td>412</td>
<td>0</td>
<td>599</td>
</tr>
<tr>
<td>10</td>
<td>database statistics in relational databases</td>
<td>44</td>
<td>49</td>
<td>50</td>
<td>543</td>
<td>0</td>
<td>686</td>
</tr>
<tr>
<td>11</td>
<td>cost based query optimization in Oracle</td>
<td>26</td>
<td>16</td>
<td>23</td>
<td>199</td>
<td>1</td>
<td>263</td>
</tr>
<tr>
<td>12</td>
<td>cost based query optimization in DB2</td>
<td>32</td>
<td>18</td>
<td>12</td>
<td>167</td>
<td>1</td>
<td>228</td>
</tr>
<tr>
<td>13</td>
<td>cost based query optimization in SQL server</td>
<td>18</td>
<td>9</td>
<td>11</td>
<td>127</td>
<td>1</td>
<td>164</td>
</tr>
<tr>
<td>14</td>
<td>cost based query optimization in PostgreSQL</td>
<td>16</td>
<td>4</td>
<td>13</td>
<td>91</td>
<td>2</td>
<td>122</td>
</tr>
<tr>
<td>15</td>
<td>cost based query optimization in SQL anywhere</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>cost based query optimization in relational databases</td>
<td>72</td>
<td>52</td>
<td>92</td>
<td>339</td>
<td>4</td>
<td>551</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>836</td>
<td>687</td>
<td>666</td>
<td>4992</td>
<td>26</td>
<td>7155</td>
</tr>
</tbody>
</table>

Figure 4.1: Summary of Primary Search Results-Stage 1
<table>
<thead>
<tr>
<th>S.no</th>
<th>Key terms</th>
<th>ACM Digital library</th>
<th>IEEE Xplore</th>
<th>Springer Link</th>
<th>Google Scholar</th>
<th>Count of papers after stage 1</th>
<th>Count of duplicate papers</th>
<th>Count of unique papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>selectivity estimation in relational databases</td>
<td>88</td>
<td>76</td>
<td>54</td>
<td>1004</td>
<td>2289</td>
<td>1222</td>
<td>1067</td>
</tr>
<tr>
<td>2</td>
<td>cost estimation in relational databases</td>
<td>68</td>
<td>56</td>
<td>38</td>
<td>571</td>
<td>1443</td>
<td>733</td>
<td>710</td>
</tr>
<tr>
<td>3</td>
<td>cost parameters in relational databases</td>
<td>21</td>
<td>23</td>
<td>19</td>
<td>145</td>
<td>329</td>
<td>208</td>
<td>121</td>
</tr>
<tr>
<td>4</td>
<td>cost components in relational databases</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>113</td>
<td>356</td>
<td>142</td>
<td>214</td>
</tr>
<tr>
<td>5</td>
<td>logical costs in relational databases</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>15</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>physical costs in relational databases</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>67</td>
<td>23</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>temporal cost factors in relational databases</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>17</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>algorithmic costs in relational databases</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>16</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>cardinality estimation in relational databases</td>
<td>40</td>
<td>27</td>
<td>26</td>
<td>284</td>
<td>599</td>
<td>377</td>
<td>222</td>
</tr>
<tr>
<td>10</td>
<td>database statistics in relational databases</td>
<td>18</td>
<td>6</td>
<td>13</td>
<td>214</td>
<td>686</td>
<td>251</td>
<td>435</td>
</tr>
<tr>
<td>11</td>
<td>cost based query optimization in Oracle</td>
<td>17</td>
<td>15</td>
<td>15</td>
<td>145</td>
<td>263</td>
<td>192</td>
<td>71</td>
</tr>
<tr>
<td>12</td>
<td>cost based query optimization in DB2</td>
<td>19</td>
<td>15</td>
<td>8</td>
<td>137</td>
<td>228</td>
<td>179</td>
<td>49</td>
</tr>
<tr>
<td>13</td>
<td>cost based query optimization in SQL server</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>96</td>
<td>164</td>
<td>128</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>cost based query optimization in PostgreSQL</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>72</td>
<td>122</td>
<td>90</td>
<td>32</td>
</tr>
<tr>
<td>15</td>
<td>cost based query optimization in SQL anywhere</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>cost based query optimization in relational databases</td>
<td>62</td>
<td>48</td>
<td>82</td>
<td>192</td>
<td>551</td>
<td>384</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>370</td>
<td>285</td>
<td>287</td>
<td>3014</td>
<td>7155</td>
<td>3956</td>
<td>3199</td>
</tr>
</tbody>
</table>

Figure 4.2: Summary of Primary Search Results - Stage 2
<table>
<thead>
<tr>
<th>S.no</th>
<th>Key terms in relational databases</th>
<th>Total after stage 2</th>
<th>Irrelevant papers</th>
<th>Relevant papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>selectivity estimation</td>
<td>1067</td>
<td>999</td>
<td>68</td>
</tr>
<tr>
<td>2</td>
<td>cost estimation</td>
<td>710</td>
<td>680</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>cost parameters</td>
<td>121</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>cost components</td>
<td>214</td>
<td>211</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>logical costs</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>physical costs</td>
<td>44</td>
<td>42</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>temporal cost factors</td>
<td>14</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>algorithmic costs</td>
<td>10</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>cardinality estimation</td>
<td>222</td>
<td>217</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>database statistics</td>
<td>435</td>
<td>421</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>cost based query optimization in Oracle</td>
<td>71</td>
<td>74</td>
<td>-3</td>
</tr>
<tr>
<td>12</td>
<td>cost based query optimization in DB2</td>
<td>49</td>
<td>44</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>cost based query optimization in SQL server</td>
<td>36</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>cost based query optimization in PostgreSQL</td>
<td>32</td>
<td>29</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>cost based query optimization in SQL anywhere</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>cost based query optimization in relational databases</td>
<td>167</td>
<td>137</td>
<td>30</td>
</tr>
</tbody>
</table>

| Total | 3199 | 3036 | 163 |

Figure 4.3: Summary of Primary Search Results-Stage 3
<table>
<thead>
<tr>
<th>Serial No</th>
<th>Database Name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACM Digital library</td>
<td>46</td>
</tr>
<tr>
<td>2</td>
<td>IEEE Xplore</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Science Direct</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Springer Link</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Google scholar</td>
<td>89</td>
</tr>
<tr>
<td>6</td>
<td>CiteSeer</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td><strong>200</strong></td>
</tr>
</tbody>
</table>

Figure 4.4: Summary of Secondary Search Results
5. Selectivity estimation

This section starts with the estimation of sizes of operators in a query plan and proceeds further with the details of the approaches in selectivity estimation, comparison of approaches, advantages and disadvantages of different approaches, limitations of approaches and finally scope of further research in selectivity estimation in relational databases.

5.1 Estimating the size of operators in a query plan

The size of intermediate relations has a significant influence on the cost estimation. These intermediate relations are the result of operators involved within the query plan. Statistics are needed for estimating the size of the operators. There are three basic elements which make up the statistics [GM08]:

1. Tuples: Data needed for query execution denoted by $T(M)$, where $M$ is a relation

2. Blocks: Blocks are required for storing the tuples of a relation in a query, denoted by $b$.

3. Distinct values: The number of distinct values in a column of a relation, denoted by $V(M,i)$, where $i$ is one of the attributes of relation $M$.

The details of estimating the size of different operators such as projection, selection, join and union are discussed below:

- **Projection operation**: A projection operation results in fewer columns of a relation i.e. depending on the attributes mentioned in the projection, columns are retrieved [ME05]. An extended projection can be used to create new components by combining attributes. The size of the result in case of an extended projection of bags (bags are type constructors and they allow occurrence of an element more than once but in an unordered manner) can be computed easily. The extended projection provides the flexibility of choosing the components within the tuples.
Selection operation: A selection operation selects tuples out of a relation, which satisfy the select condition. A select condition has several possibilities i.e. equality, AND, OR and inequality condition. These conditions are discussed in detailed as follows [ME05]:

1. Equality condition: When an attribute is equal to a constant i.e. equality condition can be denoted [ME05] as

\[ S = \sigma_{I=c}(M) \]  

(5.1)

In Table 5.1 is the list of the parameters for size estimation in selection operation.

The size of the result can be estimated when the number of distinct values in M are known. The estimate can be computed [ME05] as

\[ T(S) = \frac{T(M)}{V(M,i)} \]  

(5.2)

where T(S) is the size estimate,

2. Inequality condition: Inequality condition can be denoted [ME05] as

\[ S = \sigma_{I \neq c}(M) \]  

(5.3)

The size of the result can be estimated by removing those fraction of tuples i.e. \(1/V(M,i)\) that have attribute values equal to the constant, then the estimate can be computed [ME05] as:

\[ T(S) = \frac{T(M) \cdot (V(M,i) - 1)}{V(M,i)} \]  

(5.4)

or can be assumed that all the tuples would satisfy the condition which leads to a value of 1 for T(S).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>an attribute of M</td>
</tr>
<tr>
<td>c</td>
<td>a constant</td>
</tr>
<tr>
<td>T(M)</td>
<td>number of tuples in M</td>
</tr>
<tr>
<td>T(S)</td>
<td>size estimate</td>
</tr>
<tr>
<td>V(M,i)</td>
<td>number of distinct values in a particular column in a relation M</td>
</tr>
<tr>
<td>i</td>
<td>one of the attributes of relation M</td>
</tr>
</tbody>
</table>

Table 5.1: Parameters for size estimation in selection operation

3. AND condition: Whenever an AND is encountered, estimated size of the result can be computed by multiplying the size of the initial relation with the selectivity factor of individual conditions [ME05].
4. OR condition: OR condition can be denoted [ME05] as

\[ S = \sigma_{c_1} \ OR \ \sigma_{c_2} \]  \hspace{1cm} (5.5)

If M has y tuples and n1 number of tuples satisfy C1 and n2 number of tuples satisfy C2, then

\[ \text{Estimated number of tuples} = y \cdot (1 - \left(1 - \left(\frac{n_1}{y}\right)\right) \cdot (1 - \left(\frac{n_2}{y}\right)) \]  \hspace{1cm} (5.6)

where 1 - n1/y is the number of tuples that do not satisfy C1 and 1 – n2/y is the number of tuples that do not satisfy C2. Multiplying the above values would give the number of tuples in M that are not part of S and the number of tuples in S is 1 minus the value of this product.

- Join operation: Estimating the size of join is quite complex as there are many possible combinations of joining tables [ME05]. Considering an equality condition between two tables M (R, S) and N (S, T). Depending on how the values of S are related in M, N, the number of tuples in a join vary. If the two relations have disjoint set of S values, then number of tuples in join, \( T(M, N) = 0 \). If S is a key of M and also a foreign key for N [ME05], then

\[ T(M \bowtie N) = T(M) \]  \hspace{1cm} (5.7)

where T(M) denotes the number of tuples in M. When all the tuples in M and N have same S values, then join operation has a value equal to the product of the number of tuples in both tables [ME05],

\[ T(M \bowtie N) = T(M) \cdot T(N) \]  \hspace{1cm} (5.8)

where T(N) denotes the number of tuples in N. A standardized formula has been provided in [GH08], the number of estimated tuples in \( T(M \bowtie N) \) is the product of number of pairs of tuples in M, N with the probability value of having a common S value [ME05].

\[ T(M \bowtie N) = \frac{T(M) \cdot T(N)}{\max(V(M, S), V(N, S))} \]  \hspace{1cm} (5.9)

V (M, S) denotes distinct values in a column of a relation M and V (N, S) denotes distinct values in a column of a relation N.

- Union: Depending on the type constructors i.e. bag or set, the estimated size of the result varies. In case of a bag union, the size is the sum of the argument sizes whereas the estimated result size of a set union may be smaller than the sum of the argument sizes[ME05].
5.2 Approaches of selectivity estimation

Understanding a given data distribution is a very important in selectivity estimation of relational operations in a query. Different techniques have been proposed which store information about the database. They are categorized into Parametric, Histograms, Sampling, Single value decomposition, Discrete cosine transform, Wavelets, Probabilistic counting, Kernel density estimation [HC11, SK07, CEF05, FPH09]. Details of the techniques are provided below:

1. Parametric techniques: Parametric techniques follow a parameterized mathematical distribution for approximating the value distribution. Parametric method uses different model function (i.e. distribution functions which are predefined with a polynomial function) and compares them with the actual data distribution. If the actual data distribution has a near match with model distribution, then estimates can be generated. This technique has less overhead, but its approximations are inaccurate as the mathematical approach does not hold for real time data [CEF05, FPH09].

2. Histograms:

Most of the commercial database systems use histograms for statistics purposes i.e. for approximating the frequency distribution of values. This is due to fact that histograms have no run time overhead and consumes less storage space. There has been considerable research for developing different types of histograms. As histograms are used for selectivity estimation, their selection depends on two factors i.e. errors have to be minimum with respect to the computed estimates, maintaining and constructing the histograms has to be efficient.

Histograms play a very important role in cardinality estimation. Histogram-based cardinality estimation approaches are classified into two categories namely, proactive approaches and reactive approaches [VVE99, L16, EV95]. In proactive approach, histograms are constructed and updated by periodical data scans. Whereas data scan is avoided in the reactive approach, alternatively, query feedback records (QFRs) are collected to construct and update histograms. Due to time-consuming algorithms such as the effective QFR set calculation, the hole drilling algorithm and the iterative scaling algorithm used by the reactive approaches make it inefficient [BYV97, VBA+17, XA01]. A histogram for an attribute is constructed based on a partition rule for partitioning the data distribution into several subsets which are referred to as buckets. Using the statistical information on the number of tuples and also the number of distinct values in within each bucket, selectivity estimation is computed.

Partition rule is further divided into four elements, which are explained as follows:

- Partition Class: This property provides the information of any restrictions on the buckets. For example, in serial class, buckets follow a sequential flow of arrangement with respect to the frequency of the attribute values. There is a subclass of serial class i.e. end biased which focus on the lowest and highest frequencies of the attribute values.
• Sort parameter: Attribute values, frequency and area are used as sort parameters. Serial histograms are sorted based on these parameters. These parameters have a value which are derived from the data distribution.

• Source parameter: Spread, area and frequency are the source parameters. These parameters are crucial in selectivity estimations as they capture the property of distribution of data.

• Partition constraint: For identifying a single histogram in the partition class, a mathematical constraint on the source parameter is applied. Examples of these partition constraints include equi-sum, v-optimal, compressed and maxdiff. The success of histograms in approximating data distribution depends on how source parameters are grouped. Histogram are a result of a combination of partition constraint (p), sort (s) and source (u) parameters and denoted as p(s, u). Spreads (S), attribute values (V), frequencies (F), area (A) and cumulative frequencies (C) are the values which are used for sort (s) and source (u) parameters.

There are assumptions with respect to construction of histograms depending on the approximations i.e. value or frequency [Yan03]. Value approximation involves approximations of the values in a bucket of histogram. An assumption in this case is continuous value and uniform spread. Both these assumptions consider a uniform data distribution within the range of the buckets. Continuous value assumption is independent of the number of the values. Frequency approximations involves approximations of frequencies of values within the buckets of the histogram. Uniform distribution is another underlying assumption. Histograms partition the data into different buckets. Histograms estimate the values and frequencies of the attributes based on information in these buckets.

There are different types of histograms which are as follows [CCG10, VJEJ96]:

- Variable range histograms (Equi-depth) – The interval between minimum and maximum attribute values of a column are divided into subintervals. In equi-depth, the sum of the frequencies of values in a bucket remain same and independent of the number of values. Equi-depth histograms have generated less errors in estimation when compared to equi-width histograms [Yan03].

- Variable count (Equi-width) histograms - Data is partitioned into subintervals with equal length. These subintervals are based on the average frequency of the values. In Equi-width, the number of values in each bucket is same and independent of the frequencies of the values. This results in storage of more information and facilitates accurate estimations [Yan03].

- Serial histograms: In serial histograms, frequencies of the attribute values are grouped into buckets in a serial fashion. There are three types of histograms within the serial histograms with varying source parameters [Yan03]:
* Equi Sum: In equi-sum histogram, sum of the source values in every bucket takes a value of $1/n$ times the sum of all source values in a histogram (where $n$ is the number of buckets).

* V-optimal: V-optimal histograms tries to reduce the variance of the source parameter i.e. area. But these histograms are cost intensive to construct [Yan03].

* Spline-based: The main aspect of this type of histograms is to minimize the maximum absolute difference of a source value compared to the average of the source values in the bucket [Yan03].

* End-biased histograms: It is a subclass of serial histograms, wherein the highest and lowest frequencies are captured in separate buckets and the rest of the middle frequencies are assigned a single bucket. This means less number of buckets are needed. End-biased histograms require less storage when compared to serial histograms [Yan03].

  - Maxdiff histograms: Maxdiff has a bucket boundary which is dependent on the difference between two source parameter values. This difference has to be the largest among the one in $b-1$ ($b$ is the number of buckets) [FPH09, VE97].

  - Compressed histograms: The highest frequency values are placed in separate buckets and for the rest of the data equi width histograms are used [Yan03].

The main reason for using Maxdiff and compressed histograms is to eliminate the possibility of combining attribute values with different source parameters in to the same bucket.

3. Sampling: Maintenance of histograms is cost intensive in certain cases and may not provide correct approximations. Sampling [YLDA01, VBA+17, RLCG06, JFA90, FD95] can be used to execute a query on a sample and collect details of the statistics. Sampling technique is deployed in run time, wherein it takes random samples of data and computes the estimates. However, it provides accurate estimates in environment requiring higher number of updates. This technique has an added advantage that it does not store statistics in the database. A balance between sample size and estimation accuracy has to be achieved, when one considers using this technique.

Three different approaches have been proposed for sampling [JFA90], i.e. adaptive sampling, two phase sampling and sequential sampling. Adaptive sampling [JFA90] uses a small sample size for collecting details of the statistics. Two phase sampling [JFA90] is an extension of adaptive, where in it uses a small sample to derive a rough picture of the statistics and improves it in a second step. In sequential sampling [JFA90], the process of using sample samples is continued until clear details of the statistics are obtained. When there are secondary indices available for conjunctive query involving several queries, the predicate which satisfies the condition of smallest number of tuples is selected. The index is used to read those tuples and if they satisfy the remaining predicates then these tuples are considered for the result of the query. Adaptive, two phase and sequential sampling compute the interval based on the
errors of the previous samples and terminate sampling if the interval is small. Random Sampling in another technique in sampling [BS05], which bases its estimation by taking random samples of the database at query execution time. These samples are of a fixed size generally of the order of few hundred tuples. There are two ways of capturing i.e. record level and block level sampling. In record level sampling, random samples are taken from individual tuples uniformly. Block level sampling scans the complete set of tuples in a disk block. Random sampling does not consider the attribute independence assumption which has been a major reason for estimation errors. Random sampling can be implemented for different types of predicates i.e. equality, range predicates, predicates involving substring matches, arithmetic expressions etc. Sampling based approach for the maintenance of equi-depth histograms has been proposed. This approach uses a split and merge technique for keeping histograms updated [BYV97]. The details of the selectivities of range queries are obtained from the feedback of the query execution engine. Buckets having large counts are split and that of near-equal frequencies are merged. This approach works better for multi-dimensional data distributions having low or moderate skew. Another feedback-based approach has been proposed for partitioning the multi-dimensional data into buckets in a flexible manner [JHDHCW99].

4. Probabilistic counting is proposed for selectivity estimation which estimates the number of tuples in a given set of data by taking the count of its distinct values [BS05].

5. Wavelets: Wavelets technique do not use buckets for partitioning of the data distribution but provides a summary of the data. Wavelets are mathematical tools for decomposing functions hierarchically. Function can be a curve, an image or a surface. Wavelets provide an effective way of representing functions depending on the level of detail required [YSM98, XJ09, YSM00]. Wavelets based histogram is built in three steps:

(a) Preprocessing: An extended cumulative data distribution of the attribute in consideration is created using random sample of the original data.

(b) Wavelet decomposition: The wavelet is decomposed into a set of wavelet coefficients.

(c) Pruning: Depending on the storage, only the most significant coefficients are considered. The values and corresponding indices of these coefficients serve as histogram for reconstruction of the approximate data distribution.

Selectivity estimation is computed using the statistics provided by the wavelet based histograms.

6. Discrete Cosine Transform (DCT): Discrete cosine transform [JHDHCW99, Yan03] is used for compressing histogram information. This approach follows a uniform partitioning of the entire data distribution and compresses the bucket information using discrete cosine transform. Compressing the bucket information leads to less storage overheads and error rates. Selectivity estimation using DCT is accomplished in the following way. Firstly, finding out
an efficient sampling method for selecting low frequency coefficients with high values. Secondly, the constraint for compressing the histogram information. Third, reflecting the data updates into statistics dynamically. Finally, computation of selectivity estimation.

(a) Sampling method: Low error rate for higher dimensional data is possible if size of the buckets is small. As a result, the count of DCT coefficients required for transformation increases and all the coefficients cannot be considered. Only those low frequency DCT coefficients which have large values are selected for computation using geometric zonal sampling technique.

(b) Data distribution: The constraint for data distribution is that there must be low frequency coefficients with large values and high frequency coefficients with small values.

(c) Data Updates: As DCT is a linear transform, data insertion or deletion can be reflected dynamically. Computing values of the DCT coefficients and adding it to the existing coefficients for data insertions and subtracting coefficients for data deletions.

(d) Selectivity estimation: Using inverse DCT, the histogram buckets which are in the range of the query are obtained and then computing selectivity using the histogram information. Another method of computation of selectivity is to take integral of the inverse of the DCT function. Discrete cosine transform facilitates recovery of the compressed information using integral of the inverse function of DCT. This has an added advantage of less space requirement as well as reduced estimation time.

7. Single value decomposition: An approach for handling multi-dimensional data has been proposed which works on the singular value decomposition technique [JHDHCW99] of linear algebra. It approximates the joint data distribution into smaller individual data distributions [VE97]. Although, it is best suited for two dimensions, but its accuracy is dependent on its one-dimensional histograms. One thing that is common among the histograms is that every bucket contains the number of distinct values and also the average frequency of those values. Singular value decomposition technique that has been proposed for selectivity estimation for range clauses works only with real number.

8. Kernel Density estimation: Kernel density estimation [BDB99] is a tool which is used in statistical applications for estimating probability distribution of a given sample of data. Using kernel density estimator, selectivity is estimated by using a random sample taken as an average of the probability distribution. Here, the word kernel is used to categorize those values which are in the center of the random sample. There are advantages with respect to its application provided kernel density estimators make use of GPU for computational purposes [MMV15].
5.3 Comparison of approaches

There are different kinds proactive approaches of histogram used to approximations of data Equi depth histogram, v-optimal histogram, histogram with frequency as the sort parameter, maxdiff histogram, compressed histogram, equi-width histogram [JHD+10, EV99, CCG10, VJEJ96, EW89, KBM96, LMS03, YSM98]. These approaches suffered drawbacks as data updates must be executed periodically to make histograms consistent with underlying data. There is database overload due to periodical data updates and its effect on the performance of routine queries. Moreover, real time incorporation of data updates into a histogram is not possible leading to large errors of cardinality estimation and reducing the number of buckets in a histogram for database performance can negatively impact estimation accuracy.

A choice between accurate histograms or minimizing the cost of maintaining histograms can be made depending on the requirement. Histograms constructed based on frequency as the sort parameter provided optimal results when compared to value as sort parameter. Histograms are preferred mostly in single dimensional cases as the storage overhead and errors can increase with dimensions [JHDHCW99]. Maxdiff and compressed histograms are partition constraints which do not allow grouping of values of different source parameters into a bucket [VJEJ96, AM01].

The type of information stored by Maxdiff and Wavelets are different. Maxdiff stores largest value of the attribute, number of distinct values and average frequency of the elements in the bucket. Whereas wavelet-based histograms store the value and index of the coefficient [JHDHCW99].

The way skew (i.e. non-uniform distribution of data) is handled differs among the commercial database systems. Oracle uses Frequency and Height balanced histograms [DJM+15]. In Frequency histograms there is one bucket for each distinct value, storing exact cardinalities. They are limited only to certain number of distinct values (actually 254) otherwise the space used for storing the histogram would be too big. SQL server differentiates statistics based on its usage [Ben11]. Three different types of statistics used in SQL server, histogram, density information, and the string statistics. Histograms in most scenarios provide information about data distribution but not the functional dependencies between different columns. Density information and string statistics are used for this purpose.

Probabilistic counting technique is used mostly for estimating the size of projections [BS05].
5. Selectivity estimation

5.4 Advantages and disadvantages of approaches

Advantages of different approaches:

Database systems use histograms, as they are precomputed and occupy less space i.e. less additional overhead. Whereas sampling is efficient with small samples but using this technique for complex queries requires large number of samples.

Histograms can provide reasonably better estimations most of time [Yan03]. Histograms are useful in determining the size of joins in join operations. This is an added advantage as updating histograms perfectly is not possible all the time. Whereas sampling technique is cost intensive and not practical as it operates only in run time. End-biased histograms has less storage requirements as the singleton buckets takes less space when compared to the buckets with multiple attribute values [VJEJ96]. Parametric methods have less storage overhead and can provide accurate estimations.

Maxdiff and v-optimal are considered to more efficient as they are based on the value and area parameters. Area parameter (A) captures skew in values and frequencies and V captures the values. Maxdiff is most cost effective when compared to v-optimal as bucket boundaries are dependent on the maximum difference of source values. Maxdiff is effective for high values of skew as there are very few frequencies that are high .

Sampling based approach for maintaining histograms follows a Zipfian distribution i.e. only few of the records may get updated leaving other records unchanged. Sampling do not involve precomputation and storage of statistical information which provides an advantage of less storage overhead. Moreover, they can guarantee probabilistic accuracy

Probabilistic counting technique can provide estimates with a one-time scan of the data [BS05, APFK96].

Kernel density estimators do not need additional rules (like bucketing in histograms) for maintaining samples and can be used instantly for estimation purposes once samples are taken [BDB99].

With DCT being a linear transform, data changes are immediately reflected into the statistics for selectivity estimation. Discrete cosine transform facilitates recovery of the compressed information using integral of the inverse function of DCT. This has an added advantage of less space requirement as well as reduced estimation time [JHDHCW99]. In addition, DCT can be used for multi-dimensional selectivity estimation.
Disadvantages of different approaches:

An important requirement of parametric methods is that the actual data distribution must be known before choosing a model function. If the shape of data distribution is different compared to the available model functions, then approximations of data is not possible [CEF05].

Most of the commercial database systems try to recompute histograms on a periodic basis. This could lead to errors in estimations when histograms are utilized between two updates. Histograms have space constraints as they are deployed in main memory. Moreover, it is cost intensive to recompute histograms for large relations by scanning the entire relation.

Histograms store statistical information in the database profile for computing estimates and frequent updates on data makes it cost intensive. It is not possible to compute new histogram for every database update. Rather, schedules for timely update of histograms needs to be employed.

Multidimensional histograms occupy more space and require more time for approximating data distribution. This complexity further increases with number of dimensions in an exponential manner.

V-optimal, Maxdiff and compressed histograms are accurate when compared to other histograms but are more cost intensive to construct [HC11].

Equi-depth histograms is suitable for range queries only when there is minimum skew in data distribution [VJE96].

Techniques which consider attribute value assumption lead to high errors in estimating selectivity for predicates having multiple attributes [VE97].

Sampling technique works well for low-relational queries, but there is too much variance with queries having several joins. Block level sampling is much faster when compared to the record level sampling as it scans the entire set of tuples in a disk block. But block level sampling is not effective as there are possibilities of correlations between values of data in different tuples of a disk block.

Kernel density estimators are expensive to implement as complete sample needs to be scanned for estimation purposes. Also, there is no possibility of adjusting the bandwidth for taking the samples [BDB99, MMV15].

Using DCT for compressing histogram for higher dimensional data is effective only if the size of buckets is small as only the low frequency DCT coefficients are used for computation purposes [JHDHCW99].
5.5 Limitations

The selectivity of a join operation is calculated as the inverse of the maximum of the two join column cardinalities. Here the assumption is that domain of one column is a subset of another. This is true only when referential integrity constraints exist [JVJ12, Dan16]. Errors with respect to the size of the query results can increase exponentially for joins and also with the complexity of the query [EV95]. Cardinality estimation of query subexpressions are crucial towards estimating the cost of the plans. There has been significant work in this domain but still there exists inaccuracies in providing the statistics. The problem of correlations is another issue, even if the statistics are estimated accurately [VAA+15]. It is quite evident from the fact that if the queries are complex involving more tables, then there are more chances of selectivity and cardinality estimation errors Further leading to selection of sub optimal plans and inaccurate cost estimations.

Even though histograms have been the choice of major commercial database systems, there are issues with respect to maintenance of histograms. An approach in this regard has been proposed which is termed as self-tuning histogram [JHD+10]. This approach works on the information of the cardinalities of the range queries from the feedback of query execution engine. Using the cardinality information which reflects the updated data, buckets are maintained incrementally. Even though this approach seems to address the problem yet suffers from limitations. As feedback is available only on the queries executed, buckets are not updated in entirety. Moreover, maintaining the histograms based on the feedback is an additional overhead.

Most of the histogram techniques consider a uniform frequency and uniform spread assumption. Uniform frequency assumption requires storage of the average frequency of individual bucket and uniform spread requires storing the low and high values in individual bucket along with the number of distinct values of attributes [VJEJ96]. There is an assumption during run time that the distribution of values in the tables are uniform, which doesn’t hold to be true always. There is a possibility of existence of functional dependency between different columns. Calculating selectivities for each predicate individually and multiplying them without considering the functional dependency could lead to errors. When these assumptions are invalid or incorrect then there are inaccurate cardinality estimates [Yan03].

Another assumption made by histograms is that data distributions of individual attributes of a relation are considered to be independent. Also that the joint data distributions can be obtained from one dimensional histograms [VE97].

To address the problem of functional dependency in tables, multi-dimensional histograms are used. But they are limited to local predicates and not for join predicates or aggregations etc. The present database systems have thousands of tables with more columns in each table. With this kind of complexity, it is not possible to maintain multi-dimensional histograms for understanding the functional dependencies between different columns. Moreover, database characteristics are considered to be stable and that the statistics reflect the current state of the database.
5.6 Scope of further research

Approaches for selectivity estimation has been discussed in detailed in the previous sections. Histograms are being used by most of the present-day database systems. This can be attributed to the research which lead to new developments in maintaining histogram [BYV97]. But there are underlying assumptions such as attribute value independence, uniform spread and uniform frequency assumption. These assumptions doesn’t hold true in reality. Most of the histogram techniques are concerned about the frequencies rather than the attribute values as the focus on the spread. Frequencies in a bucket are considered to be constant and less information needs to be stored for approximations. As mentioned earlier that area as a source parameter has most relevance as it is the product of frequency and spread. Fixed and predefined approach of approximating data distributions may not provide the best results in all cases. Rather, approaches which provide flexibility in terms of choosing an optimal approximation which are specific to dimensions in individual buckets must be investigated.

Errors with respect to cardinality estimation have a direct influence on cost estimation. There are many techniques which have been proposed for cardinality estimation. But most of the techniques make assumptions (discussed in earlier sections) on the underlying data distribution. Due to these assumptions that there are significant errors with respect to cardinality estimations and these errors further propagate into cost estimation. Different approaches have been proposed to minimize the effect of cardinality errors such as use of feedback loop [MMVM01] and an approach of incremental execution [TA13]. Use of feedback loop has a major disadvantage that feedback is possible only after query execution. Nevertheless, subsequent queries can benefit from this approach. Incremental approach tries to optimize those parts of the query where it identifies that cardinality estimation errors can occur and tries to re optimize them. But this technique has some additional costs involved. This technique can be used only in situations when necessary and not on the complete query. However, [TA13] proposed that a combination of feedback loop approach [MMVM01] and incremental execution approach can be used. There still exists scope for future work towards development of new techniques which can minimize the cardinality estimation errors.

Another promising area of research is to combine histograms with other techniques which can provide better approximations. Techniques such as sampling have been used for histogram construction, but they suffer from incorrect approximations for multi-dimensional data [JFA90]. Techniques which can differentiate both the dependency and independency among the dimensions in a multi-dimensional data have to be investigated.

There is an interesting area of work on extending the capabilities of the kernel density estimators for selectivity estimation. The use of kernel density estimators for selectivity estimation has been proposed in [BDB99]. This approach has not received much attention as it is expensive to implement and maintain. But there are advantages with respect to its application provided the kernel density estimators make use of the GPU [MMV15] for computational purposes. Moreover, there is need of research on making kernel density estimators sensitive to the database changes.
6. Cost estimation

The basic cost estimation framework has been derived from the System-R approach. Each of query plans which is part of the search space are assigned with a cost by the cost model. The size of the output of the operators are estimated using the statistics of data distribution, selectivity estimates of the predicates as well as the formulas for estimating the CPU and I/O costs of executing query operators. Number of distinct values in a column, pages in an index and number of data pages in a relation are all part of the statistics. Even the order of the data is important as it can reduce the cost of certain operations like sort merge join [DIA12a, FYP99, HSL02, YKK07, Rob13, TMF99]. The cost models use the above details and starts the estimation from the child nodes. The intermediate results of the operations flow from child nodes to the parent node. This way, cost of a plan is deduced by adding all the costs of the operator nodes.

Most of the statistics are computed on a periodic basis as they do not change frequently. Usually, this computation process is triggered automatically at defined intervals or after any updates. Computation of statistics can be initiated in situations where query optimizers [NR05] chooses poor query plans. It would be very expensive to compute statistics for an entire relation. The best way would be to consider only a sample i.e. fraction of data. Generally, estimators use these statistics for costing purposes. It is for this reason statistics need to be updated periodically. This section starts with the estimation of costs of operators in a query plan and proceeds further with the details of the approaches, comparison of approaches, advantages and disadvantages of different approaches, limitations of approaches and finally scope of further research in cost estimation in relational databases.

6.1 Estimating the Cost of operators in a query plan:

Physical optimization involves building physical query plans using operators which form the steps of a plan. These physical operators implement operations which belong to relational algebra. Apart from these operators, there are physical operators for scanning purposes i.e. loading data from the disk to main memory. The choice of these physical operators is dependent on the estimated cost of their execution. Therefore, selection of an optimal physical query plan has to consider physical operators associated with least cost. The cost is computed in terms of I/O costs, i.e. number of disk I/O’s. Apart from the physical operator costs in query plan, writing the result of the query to disk involves I/O costs.
6.1. Estimating the Cost of operators in a query plan:

Cost estimation involves estimating the cost of the operators in a physical plan, which is computed using the following selections\[YKK07\]:

1. Selection of order for joins, intersections and unions.
2. Selection of an algorithm for executing every operator in the logical plan.
3. Selection of other operators like scan, sort etc. which are not mentioned in the logical plan but necessary for executing the operators.
4. Selecting the possible ways of passing arguments from one operator to another. For instance, using iterators for passing one tuple at a time or storing the intermediate results on the disk. The physical plan which has the least estimated cost of execution is selected and passed on to the query execution engine.

The details of estimating the cost of different operators such as scans, selections and joins are discussed below:

- **Scan operators**
  - The number of disk I/O’s of a clustered relation (tuples occupy fewer number of blocks) for table scan takes a value of B i.e. size of a block.
  - The number of disk I/O’s for table scan of an un-clustered relation (tuples are scattered over different blocks) takes a value of T (tuples) because every tuple is read as a block.
  - The value of sort scan is the same as that of table scan for clustered and un-clustered relations.
  - Index scan would require more number of I/O’s as the relation as well as its index has to read \[ME05\].

- **Selection operator**: Several algorithms are available for implementing select operation. These algorithms search the data on a disk and retrieve those records which satisfy the select condition. These algorithms are termed as file scans. In case index is used, then it is called as index scan. Cost functions of these algorithms are computed in terms of the number of blocks transferred between disk and memory. The details of the algorithms and their cost functions are explained as follows\[ME05\]:
  - Linear Search: All the file blocks are searched for records which satisfy the select condition. Cost is equal to the number of blocks searched i.e. \( \text{Cls} = b \), where Cls is the cost of linear search and b is the number of blocks for tuple storage \[ME05\].
  - Binary search: If an equality condition exists on a key attribute of a file which is ordered, then binary search is used. The number of blocks accessed using binary search for b number of blocks is equal to \( \log(b) \) blocks. Taking into account the selectivity and blocking factor, cost of Binary search (Cbs) is computed \[ME05\] as,
\[ C_{bs} = \log b + \frac{S_f}{bf} - 1 \] (6.1)

Where \( S_f \) is the selectivity factor, \( bf \) is the blocking factor i.e. the number of records per block for the file.

- When a primary index is used for retrieving single record, one disk block is retrieved for every index level with an additional disk block from the file. So, the cost \( C_{pi} \) is equal to the number of index levels plus the additional disk block i.e. \( C_{pi} = In + 1 \) where \( In \) is the number of index levels.

- When a hash key is used for retrieving single record, one disk block is required i.e. cost function \( C_{chk} \) takes a value of 1 i.e. \( C_{chk} = 1 \).

- When a clustering index is used for retrieving records, address of the file disk block in the cluster is available with one disk block access at every index level. Then cost \( C_{ci} \) is equal to the number of index levels plus the number of file blocks having the selected records from the cluster of file blocks [ME05].

\[ C_{ci} = In + \frac{Sc}{bf} \] (6.2)

Where \( Sc \) is the selection cardinality of the attribute with an index. For an equality condition, if \( Sc \) number of records satisfies the condition, then \( Sc/bf \) number of file blocks have the selected records from the cluster of file blocks.

- Join operation There are different join algorithms being used i.e. nested loop join, sort merge join [HIMSS97, ME05].

1. Nested loop join: Nested Loops Join (NLJ) is the most basic join operator which finds matching tuples in the right argument for each tuple retrieved from its left argument. In Table 6.1 on page 34 is the list of the cost function parameters in join operation. If \( X \) and \( Y \) are the two relations to be joined, then the cost function [ME05] is computed as the cost of number of block accesses and the cost of writing the result to disk. \( C_{nl} \) denotes the cost function of nested loop join computed as follows [ME05]:

\[ C_{nl} = bx + (bx \cdot by) + \left( \frac{jt \cdot |X| \cdot |Y|}{bfxy} \right) \] (6.3)

\( bx + (bx \cdot by) \) is the number of block accesses required for implementing the join method.

\( \left( \frac{jt \cdot |X| \cdot |Y|}{bfxy} \right) \) is the cost of writing the result to disk.

2. Single loop join: In continuation of the previous example in nested loop join, consider an index on a join attribute \( A \) of \( Y \) exists with \( Ia \) index levels. Only those records in \( X \) and \( Y \) are retrieved which satisfy the join condition [ME05]. For secondary index, cost function [ME05] is computed
6.1. Estimating the Cost of operators in a query plan:

as the number of block accesses using the index, taking into account the selection cardinality of the relation. The total cost of single loop join includes the cost for writing the result to the disk. Then, cost function is defined as below [ME05]:

\[
C_{sl} = bx + ((|X| \cdot (Ia + 1 + Sa)) + \frac{jt \cdot |X| \cdot |Y|}{bfxy})
\] (6.4)

In case the attribute is not a key field, then clustering index is used. In this case, the number of block accesses is computed by taking into account a factor of \( \frac{Sa}{bfxy} \). Then the number of block accesses is computed as the product of the number of tuples in relation \( X \) with a factor which takes into account both the index levels and selection cardinality in attribute \( A \) in \( Y \) [ME05]

\[
C_{sc} = bx + ((|X| \cdot (Ia + \frac{Sa}{bfxy})) + \frac{jt \cdot |X| \cdot |Y|}{bfxy})
\] (6.5)

In case the attribute is a key field, then primary index is used. In this case, the number of block accesses is computed as product of the number of tuples in \( X \) to that the index levels in \( Y \) [ME05]

\[
C_{sp} = bx + ((|X| \cdot (Ia + 1)) + \frac{jt \cdot |X| \cdot |Y|}{bfxy})
\] (6.6)

For hash key existing for one among the two join attributes, the number of block accesses is computed as product of the number of tuples in \( X \) to value of the average number of block accesses required to retrieving a record. Then the cost function is defined as below [ME05]:

\[
C_{sh} = bx + ((|X| \cdot hk) + \frac{jt \cdot |X| \cdot |Y|}{bfxy})
\] (6.7)

hk assumes a value of 1 for static and linear hashing and 2 for extendible hashing.

3. Sort-Merge Join: Another approach in join operation is to sort each of its arguments on the join attribute and then merge the results. While sorting is algorithmically preferable, its overhead often makes this method inferior to that of plain NLJ. The information on order properties of the underlying access paths can be utilized instead of the sorting. In case the sorting of files is already done on join attributes, then cost function [ME05] is computed as the number of block accesses of relations in consideration and the cost of writing the result to the disk, which is defined as follows [ME05]:

\[
C_{sm} = bx + by + \frac{jt \cdot |X| \cdot |Y|}{bfxy}
\] (6.8)
### 6.2 Classical approach of Cost estimation

A most common approach towards deriving total cost is to add I/O costs and CPU costs (with a weight factor) \( [S^02, GMD^88] \)

\[
C = C_{I/O} + w \cdot C_{cpu}
\]  

(6.9)

where \( w \) is the weight which depends on the system in use i.e if the system is heavily dependent on CPU for query execution, then \( w \) is increased and decreased if system needs more I/O operations.

The goal of the approach can be to minimize the response time or to minimize the resource consumption. In most cases, the goal of cost estimation is to minimize the time needed for query execution. A classic approach towards cost estimation of a plan is to estimate the costs of operators in a plan separately and add them \( [LJM12] \). In a traditional centralized system, only the CPU and I/O costs are estimated. The CPU costs include the time for locking and accessing the tuples, fetching the tuples for comparisons or projections, processing of the operations in a query i.e. join, sort. The I/O costs are estimated using the seek, latency, prefetch capabilities and transfer costs. Reading and writing of data between secondary disk storage and main memory buffers involves disk I/O costs. Access structures on files also effect the costs for data retrieval \( [WRMC92a] \).

Intermediate results of the operators in a query needed to be stored for further processing. These results are stored on disk. Such costs are termed as disk storage costs.

Distributed systems involve several terminals and each terminal may have several computers and data is integrated from multiple local databases. At local level, cost estimation can be accomplished using the local cost models \( [FSP^+97] \). When queries are optimized on a global level, then local information must be available to
the global query optimizer. There is an additional communication cost involved in addition to I/O CPU costs. When users input queries, both sending instructions to the database and receiving the results from the database involves communication cost. They are categorized as per their changing frequency. A list of such factors [ME05, NR05, MM13, RR13, NJMH11] are mentioned i.e. amount of the data sent, packet size and communication protocol and CPU cost of packing and unpacking the messages at transmitting and receiving.

Cost models in distributed databases consider total time or response time for query execution. Cost model in distributed can be considered as an extension of that of the centralized database system. That is, in addition to I/O, CPU costs, cost for sending and receiving the data is to be accounted. In Table 6.2 is the list of cost model parameters. A formula for the total time taken for query execution in a distributed database is given below [LRH14]:

\[
TotalTime = TCPU \cdot N\text{insts} + TI/O \cdot NI/Os + TMSG \cdot N\text{msgs} + TTR \cdot N\text{bytes}
\]

Minimizing the total time would mean more usage of resources such as CPU, I/O and communication channels. When total time for query execution is reduced, number of queries processed can be increased.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCPU</td>
<td>time required by a CPU instruction</td>
</tr>
<tr>
<td>Ninsts</td>
<td>number of CPU instructions</td>
</tr>
<tr>
<td>TI/O</td>
<td>time required for a disk I/O</td>
</tr>
<tr>
<td>NI/Os</td>
<td>number of input/output operations</td>
</tr>
<tr>
<td>TMSG</td>
<td>fixed time of initiating and receiving a message</td>
</tr>
<tr>
<td>Nmsgs</td>
<td>number of messages</td>
</tr>
<tr>
<td>TTR</td>
<td>time taken for transmitting a data unit from one site to another</td>
</tr>
<tr>
<td>Nbytes</td>
<td>number of bytes transmitted</td>
</tr>
</tbody>
</table>

Table 6.2: Cost model parameters
6.3 Cost model based on the underlying hardware technology

The main focus of the classic model of cost estimation is to estimate the amount of resources needed for query execution. It is useful if the queries consume few resources. This model has a disadvantage when the goal is to minimize the response time of query execution. As this needs more usage of resources for executing the queries in parallel [ASL+99, IK89, ASY95, TJ85, MM86, Mar87, E96, S+02]. An example has been provided in [Don00] which illustrates the difference between these two goals. In this example the costs of join processing at 2 different sites were considered. In one scenario, join processing involves less resource consumption but in another scenario the usage of additional resources is more in order to minimize the response time.

The approach towards cost estimation changes with the underlying hardware properties (i.e. storage technology).

There are new advancements in I/O technology such as the introduction of Flash SSDs, NV-Memories technologies. These new technologies exhibit different characteristics compared to Hard Disk Drives (HDD).

In contrast [DIA12b, NJMH11] to the traditional spinning disk storage, there are key differences such as asymmetric read/write performance, low latencies and reads are faster than writes. These features directly affect the I/O costs involved in cost-based query optimization.

Most of the current database systems are built around traditional hardware technologies. With the inception of new technologies having fundamentally different characteristics, the cost functions used by the traditional query optimizers need to be changed accordingly. As traditional spinning disks exhibit symmetric read and write characteristics, their I/O costs are accounted by weighing the number of sequential and random storage accesses with different factors. As the complexity of the operating a database is increasing, additional cost factors need to be considered [DIA12b, NJMH11].

There is asymmetry in the read and write operation for sequential and random I/O and this has to be accounted for accurate cost estimation. Four I/O configuration parameters are considered instead of the two. Earlier, only sequential and random I/O were considered. These are further divided into sequential read and write, random read and write. Cost functions of sort and hash join algorithms are modified to include these four factors. As sort and hash join algorithms perform both read and write operations [DIA12b, NJMH11].

Main memory size and CPU speed has improved significantly in the recent past. But the memory access latency has not paced accordingly. Cache memories have been introduced to reduce this gap. There are different cascading levels of cache memories to cope up with the usage patterns [ALS08].

As disks are used for persistent data storage, it is needed for initial data access. Later all the operations and their intermediate results are stored in memory. This has to some extent reduced the I/O costs. Most of the traditional cost models have considered the memory access as uniform during the cost computations [S+02, SPL02, PJP00]. It is because data access is independent of the order of access and memory address. But with cascading levels of cache in place, both the above assumptions do not hold true and memory access costs need to be included.
in cost calculations. Memory access costs are dependent on three factors namely, Latency, Bandwidth, Address translation. Latency is the time needed for the data to be made available to CPU. Bandwidth is the volume of data transferred between CPU and memory in a second. Address translation is defined as the translation of the logical virtual memory addresses used by application into physical memory addresses in main memory. Memory access cost are computed by calculating the cache misses and multiplying them with corresponding latency [ALS08, PJP00].

6.4 Cost estimation in commercial database systems

A brief explanation of cost estimation approaches in commercial database systems is provided below. Below is the list of the popular databases currently in use:

1. Oracle
2. Microsoft SQL server
3. PostgreSQL
4. DB2
5. SQL anywhere

6.4.1 Oracle

Oracle RDBMS is an object-relational database management system provided by Oracle Corporation. Earlier, rule-based approach was used for optimizing queries in Oracle. In the rule-based approach, execution plans are selected based on the ranking assigned to operations (access paths). An efficient approach has a lower ranking, which is assigned to table access using ROWID and higher ranking is assigned to full table scan. This rule-based approach is no more used in Oracle [ME05, Jon06, Ora15]. Currently all the database systems use cost-based query optimization. Using cost-based technique, Oracle selects an execution plan of lowest cost out of several alternatives. As observed in the research on cost estimation, the analogy of cost implies I/O costs, CPU and memory usage for performing query operations. This also applies to Oracle cost-based query optimization. But there are few underlying changes which differentiate the commercial systems from that in research. Depending on the applications being supported by these commercial database systems, there is a variation in usage of database statistics. Accurate estimation of statistics is crucial towards cost calculations.

Statistics

Oracle uses two types of statistics i.e. object and system statistics [DJM+15]. Object statistics are further divided into logical and physical statistics. Information about the number of rows, average row length, column histograms are stored in logical statistics. As the name states, physical statistics include details of the format in which the tables are stored. Details about the number of CPUs, throughput of I/O operations and available memory are available with the system statistics. Logical
6. Cost estimation

Object statistics are useful in cardinality estimation of various operations. Costs of the operations are estimated using these cardinalities, physical statistics and system statistics. For accurate cost estimation, statistics must be up to date. Uniform distribution of data is not true in most cases. To deal with skew, Oracle uses histograms. Histograms are stored in the data dictionary during the statistics update. Histograms are used in calculations of selectivity i.e. filtering factor – FF. Density in selectivity calculation is taken into account rather than the number of distinct values. Frequency and Height balanced histograms are two types of histograms used in Oracle. For each distinct value, a bucket is used in frequency histograms. But there is limitation in the number of distinct values (i.e 254) which can be stored. In case there are more distinct values, height balanced histograms can be used which divide the column values into different buckets. Endpoints are regarded as maximum values and the special zero bucket as minimum value.

There are several factors considered by the Oracle optimizer in determining the execution plan with the least cost [Ora11]. Estimating the number of rows resulting from each of the operations. For cardinality both table and column level statistics are used. Even when these statistics are up to date there are factors (such as data skew, complex query expressions) which can lead to inaccurate estimation. The method of accessing data i.e table scan or index access. Algorithms such as nested loop join, merge sort join, and hash join for joining tables and the type of join operation i.e. outer join or semi join or anti join. This depends on the cardinality estimates and the access paths used. Accessing only the necessary partitions to answer a query i.e. partition pruning is the simplest way of improving performance [HNS11].

The decisions made by an Oracle optimizer for selecting an optimal plan can be understood using the explain statement. The explain plan gives information on the cardinality estimations, access methods and join methods used in plan selection. There is a possibility of specifying hints to the optimizer in Oracle. Hints can be an access path for a table, join order or a specific join operation [ME05].

In Oracle 10g and 11g, Cost based optimization uses the formula mentioned below [Jon06, Rob13]: Costs are computed as the time needed for single block reads, multiple block reads and CPU time expressed in terms of units of single block read time. In Table 6.3 on the next page, details of the cost model parameters in Oracle are provided.

\[
\text{Cost} = \left( \frac{Sbr \cdot srt + Mbr \cdot mrt + (CC/CS))}{srt} \right) 
\]  
(6.11)
6.4. Cost estimation in commercial database systems

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sbr</td>
<td>number of single block reads</td>
</tr>
<tr>
<td>Mbr</td>
<td>number of multi block reads</td>
</tr>
<tr>
<td>CC</td>
<td>number of CPU Cycles</td>
</tr>
<tr>
<td>srt</td>
<td>read time of single block in milliseconds</td>
</tr>
<tr>
<td>mrt</td>
<td>read time of multi block in milliseconds</td>
</tr>
<tr>
<td>CS</td>
<td>CPU cycles per second</td>
</tr>
</tbody>
</table>

Table 6.3: Cost model parameters in Oracle

6.4.2 Microsoft SQL server

Microsoft SQL Server is a relational database management system provided by Microsoft. The query optimizer in SQL server is based on the extensible Cascades Framework architecture [Ben11, SRJ+12]. The cost estimation approach followed by the SQL server optimizer is similar to that of other commercial cost-based optimizers. Costs are estimated using the details of the resources like I/O, CPU and memory. Apart from the resources, the optimizer needs statistics of the internal storage format and data distribution of the values in the tables, operator and row estimations.

Statistics

There are three types of statistics used by SQL server optimizer namely the histogram, density information, and the string statistics. These statistics are part of the cardinality estimation process [Ben11]. Histograms and string statistics are created for the first column of a statistics object. Density information is created if the column is of a string data type. Histograms distribute the values of the columns into several buckets. There is a limit of 200 buckets in SQL server. Maxdiff histogram is used for this purpose. SQL server features a new set of statistics namely, filtered indexes and statistics. Filtered statistics are created when filtered indexes (an index defined with a WHERE clause) are available. They are useful with queries accessing specific subsets of data and also in situations like correlated columns. If the columns used in a query are correlated, then the cardinality estimation would be incorrect. Density information on multi-column statistics can improve the quality of execution plans. Columns are assumed to be independent by default and in case there is a functional dependency between columns then multi-column statistics can be helpful. Density information is of help for filters and GROUP BY operations [Ben11]. Another feature of SQL server optimizer is that it performs automatic matching of computed columns. For queries with scalar expressions, computed columns are used [Ben11]. Sampling is being used for speeding up the recomputation of histograms. This approach has been used in SQL Server, wherein a random sample is extracted from the relation and then using this sample a histogram is computed. These samples act as backing sample so that histograms need not be computed from scratch and no overheads during update.

Errors with respect to cardinality estimation can lead to selection of bad query plans. Cardinality estimation error on one operator can impact the cardinality estimation of all other operators using this data in a plan.
These errors can be minimized to some extent if the query can be partitioned into several steps and store the intermediate results in temporary tables. Statistics on these intermediate results can be created and used in selection of query plans. Hints can be used in SQL server to override decisions made by the query optimizer for improving the performance of query execution [Ben11].

6.4.3 PostgreSQL

PostgreSQL, is an object-relational database management system (ORDBMS). PostgreSQL is a great example of an open source database. PostgreSQL follows the same process of query optimization discussed in the earlier sections. The goal of the optimizer is to select a query plan associated with lowest cost. Cost estimation depends on the statistics of the data stored in tables. PostgreSQL optimizer follows a sequence of steps for cost estimation [KS03, Tob10, HNS11, VAA+15]. Firstly, to estimate input and output cardinality and compute the CPU cost based on these estimates. Second step is to estimate the number of pages accessed according to the cardinality estimates and compute the I/O cost based on these estimates.

Finally, the total cost is calculated by adding the CPU and I/O costs [CB13, KS03, VAA+15]. The cost model of PostgreSQL optimizer involves five cost factors represented as vectors. These five cost factors include the I/O cost (for sequential and random access) along with CPU costs (for processing tuples using index and performing hash or aggregation operation). Table 6.4 on the facing page is the list of cost model parameters in PostgreSQL.

\[ c = (cs, cr, ct, ci, co)^T \]  

(6.12)

The cost CO of an operator O in a query plan is then calculated as a linear combination of cs, cr, ct, ci, and co.

\[ CO = (n)^T c = ns \cdot cs + nr \cdot cr + nt \cdot ct + ni \cdot ci + no \cdot co. \]  

(6.13)

The values \( n = (ns, nr, nt, ni, no)^T \) represent the number of pages sequentially scanned, the number of pages randomly accessed etc during the execution of the operator O. The total cost is the summation of the costs of the individual operators in a query plan. The estimation accuracy of CO depends on both the accuracy of the c’s and the n’s. Cardinality estimation is crucial towards determining accurate values for these quantities.

Statistics

There are two statistics used in PostgreSQL i.e. height balanced histogram and most common values. Random sampling is used as a collection algorithm in PostgreSQL.

Cost models of optimizers are inaccurate due to usage of incorrect cardinalities in the query plan. Cost modelling errors can be minimized by selecting a cost model with low modelling. Optimizer designers design a default cost model for PostgreSQL which needs further tuning according to the underlying hardware systems [pan15, HNS11, KS03, VAA+15]. There are difficulties in tuning certain parameters like random page cost, which is the cost of accessing pages randomly.
6.4. Cost estimation in commercial database systems

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>I/O cost of sequentially accessing a page</td>
</tr>
<tr>
<td>cr</td>
<td>I/O cost of randomly accessing a page</td>
</tr>
<tr>
<td>ct</td>
<td>CPU cost of processing a tuple</td>
</tr>
<tr>
<td>ci</td>
<td>CPU cost of processing a tuple via index access</td>
</tr>
<tr>
<td>co</td>
<td>CPU cost of performing hash or aggregation operation</td>
</tr>
<tr>
<td>n</td>
<td>number of pages accessed</td>
</tr>
</tbody>
</table>

Table 6.4: Cost model parameters in PostgreSQL

6.4.4 DB2

IBM DB2 has been developed by IBM. DB2 is a relational database which also supports object-relational model.

The query compilation is same as the standard query process being followed by other optimizers. There may be different objectives towards cost estimation, i.e. to minimize the time required for query execution or minimum consumption of the system resources [G93]. The costs are calculated by combining the CPU costs i.e. number of instructions, I/O costs (random and sequential accessing of pages) and the communication costs between nodes or sites in distributed environment. The DB2 optimizer needs details about available buffer, database design (indexes and tablespace configuration) and table statistics.

Statistics

Different statistics are used by the optimizer. Basic statistics include the number of rows in a table as well as the number of pages for a table. Index clustering provides detailed index statistics which can improve the estimates of data page fetches and column group statistics which are collected for more than one column. The performance of the optimizer can be improved by running frequency statistics to understand the skew in data distribution, keycard statistics for correlations and re-evaluating the parameter markers at run time. DB2 maintains two different histograms for estimating the selectivity. For equality predicates (frequency histograms) and range predicates (quantile histogram) [CJS+04, VMV03].

The most interesting feature of the DB2 optimizer is the usage of optimization class (OPTLEVEL) which controls the level of modeling and influences compilation time. A lower number of optimization class means less optimization (and therefore less optimization time, and likely more query execution time), while a higher number means more optimization (and therefore more optimization time and possibly less query execution time). This feature is very useful in adjusting the optimization as per the environment i.e. for OLTP and OLAP systems. Queries in an OLTP system are small and requires less optimization for achieving an optimal plan. Whereas in OLAP systems, queries are complex and take more time for execution. So, more optimization is required.

The default level of 5 is used in most of the cases.

- Level 0, when minimum optimization is required.
6. Cost estimation

- Level 1, here the optimization is roughly equal to Version 1 of DB2
- Level 2 – Higher than level 1, not recommended for complex queries
- Level 3 – Moderate level of query optimization
- Level 5 – Significant optimization. Heuristic rules are used to limit optimization time.
- Level 7 – Significant optimization like level 5, without using heuristic rules
- Level 9 – Maximum optimization for very complex queries on very large tables.

6.4.5 SQL anywhere

SAP SQL Anywhere is a relational database management system (RDBMS) developed by SAP. SQL Anywhere follows the same query process as in other query optimizers. Cost model of SQL Anywhere has additional features when compared to other query optimizers. Just as any other query optimizer, SQL Anywhere relies on statistics for cost estimation. SQL Anywhere relies on selectivity estimates for I/O cost computation. Histograms with more number of buckets are used for this purpose. Buffer residency ratio (BRR) of database objects is another factor that the optimizer of SQL Anywhere takes into account for I/O cost estimation. The buffer residency ratio of a relation is defined as the ratio of the number of buffer resident pages to the total number of pages for a relation [YKK07].

There are estimation errors despite SQL Anywhere uses the buffer residency ratio. Most of these errors are attributed to inaccurate I/O costs estimation. SQL Anywhere also provide the feature of hints which can used to override the decisions of the query optimizers in case of poor query plan selection.

6.5 Comparison of approaches

The major focus of any query optimizer is to reduce its resource footprint. As mentioned earlier, cost-based optimizer selects an execution plan with the least cost, but the quality of selecting a plan is directly proportional to the accuracy of the optimizer’s cost and cardinality estimations. Even if the query optimizer is given the time for analyzing the entire search space, still there are chances that the query optimizer would choose a wrong plan due to cardinality and cost estimation errors. Most of the time cost estimation models make assumptions about the environment and consider only few hardware conditions. This can be reduced to some extent by building optimization capabilities which are specific to a domain. Most of the commercial database management systems provide the option of specifying hints on which access methods and join orders are to be used. These hints can be issued through SQL or catalog tables [AZV+07, MWMN08].

There are more possibilities for a cost model to assume that a query starts with a cold cache i.e. that data is read from disk rather from memory and this would lead to incorrect cost estimation errors[Ben11]. Commercial database systems employ additional features to improve cost-based query optimization. Partitioning, usage of hints [ME05], accounting buffer residency ratio for cost estimations[YKK07], options of setting the optimization levels.
6.6 Advantages and disadvantages of the approaches

Advantages of different approaches:

Cost models are useful in accurately predicting the query execution time, scheduling database query tasks [ZAY14, LJM12, SSAV03] designing new algorithms or improving the performance of available algorithms used in the query execution [S+02]. The factors considered by the commercial optimizers are mostly similar i.e. cardinality estimation, access method, join method, join type, join order with few optimizers using additional resources or techniques [QA96, GFZH96b, TMF99, OAB12, YET98, QYS00, Abd09, AQP04, HGA98b, DIA12b, MC96, ZSR05,QP98a]. The approach towards these factors vary from optimizer to optimizer. Depending upon the complexity of the queries, a balance between optimization time and query execution time can be achieved. Moreover, use of the existing cost models can reduce the consumption of resources in developing new hardware systems [S+02].

Disadvantages of different approaches:

Access paths selected by an optimizer is based on the statistics of the tables and indexes. These results can be inaccurate due to underlying assumptions during execution time. The optimizers need to consider the real filter factors at execution time [TM05, VTS06, KS10, ME05]. Most of the cost models have inaccuracies in cost estimation, which can be attributed to problems in selectivity and cardinality estimations. Several studies [ES91, JFA90, AVD13] have proposed solutions with respect to selectivity estimation. Even if these selectivity estimates are accurate, problems still exist in cost estimation [SVL04, SVS99, MSB97, VE97, MSRV00]. Most of the commercial database do not expose the internal working of their database systems and physical database design to safeguard their intellectual property. This often leads to an incomplete understanding of cost estimation in commercial database systems [KS03, SMP88, JCNG02, JDB97]. The underlying system statistics influence the selection of query plan [VAA+15]. There is problem in accounting other factors which influence the access methods i.e. clustering, data layout and caching, amount of available memory. These errors lead to inaccurate I/O cost estimation. There are new advancements in hardware technology i.e. Flash SSDs has led to changes in the database characteristics and the classical approach of cost estimation considering only the CPU and I/O costs is no more valid.

6.7 Limitations

The present-day query optimizers have received enhancements and improvements over the years. With accurate selectivity estimation techniques in place and CPUs becoming more powerful, cost estimation would be perceived to have reached a better level of performance. Due to the new domains in which databases are operating be it distributed [TP11, Don00] or heterogenous systems [HGA98a, HSL02], still there are limitations that needs attention. There is a possibility of queries being invoked by applications rather than by the user. In such a case, host variables, special registers and parameter markers may occur. As the optimizer is not aware of their values, it would generate random values at run time [M14, FS15]
6. Cost estimation

6.8 Scope of further research

From the previous sections, it is clear that there are multiple factors underlying the estimation of costs in relational databases. There are opportunities for further research to keep on working on this issue, according to the literature review. An interesting area of future work is cost estimation in GPU accelerated DBMS. With GPUs being used extensively for speeding up applications which are data intensive. GPUs are specialized processors programmed for parallel processing. Processing of data in GPU is handled by the CPU. By leveraging the parallel processing capabilities of the GPU, the query execution time could be reduced significantly. However, cost functions with respect to GPUs are needed for accurate cost estimations [MMV15]. Another area of future work is cost estimation in heterogeneous databases. Heterogeneous databases (HDBMS) integrate different databases and work as a single unit. However, estimating the costs involved in executing queries in a heterogeneous environment is complex as costs varies with the type of database. A categorization of the heterogenous databases is provided in [WRMC92a] i.e. proprietary, conforming and non-conforming DBMS. A database which is part of the heterogeneous database environment is referred to as proprietary when it is provided by the same vendor as that of the integrating DBMS (HDBMS) and provides the cost functions along with database statistics for estimation purposes. Conforming DBMS is the one which is provided by a vendor other than that of integrating DBMS and facilitates the availability of cost functions as well as the database statistics. Non-conforming is provided by vendor other than that of the integrating DBMS but does not provide the cost functions as well as databases statistics. As there is no information about the non-confirming DBMS, estimating costs is difficult. This creates a need for development of new techniques for deducing cost model parameters for non-conforming DBMS.
7. Conclusion

Systematic reviews are an efficient way of investigating the relevant studies and to obtain the insights and find gaps in the research. Most importantly, provides the scope of further research [ABMB14]. The review process started with understanding the goal of the review and forming clear objectives. Efforts required were channelized to streamline the review process.

In this systematic review, different approaches in cost estimation in databases have been investigated. With the help of the inclusion and exclusion criteria, the relevant studies were identified. The contents of these papers were further studied and checked if they were able to answer at least any one of the research objectives. Data extraction was performed based on the data schema which provided a summarized view of the paper.

The classical approach of query optimization has been provided in the section overview of approaches. The later sections provided a detailed understanding on how the commercial optimizers work and the extent of variation in their approach towards dealing with cost estimation. With the database systems becoming more complex in terms of scale and functionality, it is more obvious that several factors influence the efficiency of a database system. Whether it is the query optimization time or the quality of a query plan, combination of existing or new methods are needed.

Main objectives of this thesis are to investigate cost estimation approaches in databases and thereby providing a comparison between these approaches. Furthermore, the limitations of these approaches and the gaps within the research have been investigated. Most of the approaches have varying levels of priority of the cardinality estimation and join order techniques with respect to their functioning and in achieving their goals. It is true that no one size fits all but approaches have few things in common i.e. cardinality estimation. It has been observed that cardinality estimation in specific has rather significant influence on the cost estimation in databases and on the query optimization. Most of the papers have highlighted the need for better cardinality estimation. Given the fact that query optimizers have been dependent more on the cardinality estimation for running the queries efficiently in run time. This brings us to the point that the errors with respect to the cardinality estimation have to be minimized.
8. Future work

As this research work is primarily focused only on relational databases, there exists opportunities for future work on research in other databases such as object-oriented databases, XML databases, object-relational databases, temporal databases, spatio-temporal databases. There has been significant research in query optimization and cost estimation in these databases. The database models of these databases are different from the database model of relational databases and the way queries are processed in these databases are also different. This provides an opportunity to work on the techniques used for query optimization and also the underlying cost models.

Another area of future research is to focus on heterogeneous databases. There is a need to analyze query optimization and cost estimation in a heterogenous environment. In a heterogeneous environment, different types of databases work together. Estimating the cost of executing a query in a heterogeneous environment would require an in-depth analysis of the underlying cost functions as well as the database statistics of individual database systems.
Bibliography


International Conference on Artificial Intelligence, Knowledge Engineering Data Bases (WSEAS), page 24. World Scientific and Engineering Academy and Society (WSEAS), 2005. (cited on Page 20 and 27)


[QYS00] Zhu Qiang, Sun Yu, and Motheramgari Satyanarayana. Developing cost models with qualitative variables for dynamic multidatabase environments. In *Proceedings. 16th International Conference on Data*


Jain Swati and Barwal Paras Nath. Performance Analysis of Optimization Techniques for SQL Multi Query Expressions Over Text Databases in RDBMS. *International Journal of Information and


Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 26. March 2018