Chapter-2

LITERATURE SURVEY AND SCOPE OF RESEARCH

2.1 CHAPTER OVERVIEW

This chapter covers extensive literature review of existing materials, documents and tools. The statement of problem is described with objective and scope of research. To identify and understand the demand and expectations of IT industry people, the survey has been conducted. The survey has been prepared and distributed online to the experienced IT persons of renowned companies. The questionnaire has been prepared on the basis of existing research work. Detailed analysis has been done to check the feasibility of a proposed tool. Hypothesis testing is also done for the various parameters. The research methodologies which have been followed are data collection, feasibility study and statistical analysis. Hypothesis are tested to check dependency between parameters and to check feasibility.

2.2 STATEMENT OF PROBLEM

In distributed database management system, there is performance degradation due to inefficient execution of transactions [1]. One of the reasons is poor design of physical structures such as index [55][57]. Index may improve or degrade the transaction performance [71]. There is a need to analyze and suggest proper index for the tables. Many index tuning tools are available but they are designed for homogeneous databases. Index tuning tool for multiple heterogeneous distributed databases is not available. If this type of tool would be developed, it would be very useful to the IT field where distributed transactions take place. For ex., many companies develop e-commerce sites in which transactions are executed on distributed database. These types of applications require tool which can recommend proper index for the database. One of the Ahmedabad based
organization named Indylogix Solutions facing the problem of performance degradation in the e-commerce transaction processing developed by them for different clients. The owner of the company Mr. Pankaj Wagh has requested to investigate the problem. The statement of problem is “Improving concurrent transaction execution in multiple heterogeneous distributed databases”. It is resolved and achieved by developing a tool to evaluate existing table indexes and by recommending better indexes.

2.3 OBJECTIVE OF RESEARCH

Recent is an era of the Internet and mobile applications where huge amount of structured and unstructured data is distributed and stored on different sites. The structured data are stored and processed in relational database management system [15]. Users at different sites use heterogeneous database environment to manage these data [71].

To access, process and restore these data to and from the database, powerful hardware and software are required. Performance of concurrent transaction processing in distributed heterogeneous environment is degraded because of poor index selection or designing, improper partitioning, lack of query optimization [20], inconsistent data updation on different sites, improper mapping of local data dictionary to global data dictionary, etc.[2][71].

Most of the research has been done for transaction improvement in distributed homogeneous databases. The area of transaction performance in distributed heterogeneous databases is not explored much by researchers and there are very less publications available regarding the same.

Nowadays, most of the e-commerce applications contain terabytes of data which are stored on different machines. There are users who are accessing data concurrently. When there are many requests send to the database, the performance degrades. The objective of the research is to study the cause of performance degradation in multiple heterogeneous distributed databases and to propose a technique which helps to improve the performance. The research in this area would find the solution which will help IT industry and academia.
2.4 SCOPE OF RESEARCH

In distributed database environment, physical database is distributed over various sites through fragmentation and replication [71]. When different sites are accessing parts of fragments, they may use different hardware and software and even different DBMSs. The distributed environment in which hardware and software are different on all the sites is known as heterogeneous distributed environment. It is very difficult to maintain good performance during data access and transaction processing in this kind of environment because there are many applications which are using and updating the same data [71].

Many algorithms, matrix and models are available to fragment, replicate, allocate and handle concurrent transactions in homogeneous distributed environment, but still there is a wide scope of improvement in transaction processing by revising and creating better models, algorithms and matrix which focus on physical designing such as indexing, partitioning and materialized views as new technologies are emerging day by day with changing environment [3][14][16][53][63][72][78]. Ample of research has been done to improve performance of transactions [94], but limited or no research has been done in the area of optimization of distributed transaction processing for multiple heterogeneous databases [54][58][59][60][61][77]. Thus, there is a scope of research in this area.

In this research work, existing methods to improve performance of transactions in distributed and non-distributed databases are reviewed [58][59][60][61]. The tools which are available in market for index tuning are studied in detail with matrix affecting the performance such as response time, CPU time and cost. During the research, a tool named Table Index Evaluator and Recommender (TIER) has been developed which would help to improve performance of transactions in multiple heterogeneous distributed database environment. The comparative study of existing table indexes and new indexes (recommended by TIER) is also performed.

2.5 LITERATURE SURVEY

The extensive literature survey is done before proposing the TIER model. Various tools of database tuning are studied to find out their limitations.
2.5.1 Research Background

The concept of distributed database exists for many years. Extensive research is done in this area to resolve the problems which may occur in DDBMS. People are doing intensive literature survey has been done to identify the performance issues which occur during concurrent transaction execution in multiple heterogeneous distributed database [51][54]. The survey was initiated with overall study of performance issues and eventually ends up in identification of an issue “Index Automation Problem” which may improve or degrade the performance of transactions in distributed database [71]. Many RDBMS provide built-in tuning tools which also include index tuning [4][25][39][40][48][49][62][97]. Therefore, further literature survey has been done to study various indexing tuning tools which are available with RDBMS or independently as third party tools. Following sections describe survey of performance issues in distributed database transactions, comparison of existing methods of index tuning and limitations of existing index tuning tools.

2.5.2 Performance Issues in Distributed Database Transactions

When physical database is fragmented and stored on different machines or sites through communication network, it is called distributed database. After the database is distributed, data is accessed and processed by any machine. Performance of transactions is mainly based on data localization and parallel execution of queries. Data localization means the most frequently accessed data by any site is stored on that site or near to that site. Query execution may be inter-query or intra-query. In inter-query execution, many queries are executed in parallel at the same time. In intra-query execution, a single query is divided into many subqueries which are then executed on different sites and then results obtained from all the sites are joined together to display the output of the query [71].

M. Ozsu and P. Valduriez state that there are algorithms given for partitioning or fragmentation. In internet-based applications data comes in the form of audio, video, documents and other formats which are called stream data. Real time data comes in the form of stream (unbounded sequence) from internet. These data are distributed across many machines, which are accessed by many users from their own machines [71].
D. Wang has proposed Cluster-and-Conquer algorithm [99] for optimizing distributed join [65] over database federation with efficiently considering run-time conditions. Cluster-and-Conquer algorithm is motivated from real world observation in which author has proposed to view the whole database federation as clustered system, and provide each cluster of data sources with its cluster mediator. Finally author has implemented the prototype federation system with the proposed architecture and optimization algorithm. The experimental results showed the capabilities and efficiency of Cluster-and-Conquer algorithm and gave the target environment where the algorithm performs better than other related approaches. Currently the prototype system has two levels of mediators, but it is necessary to extend the system in order to support multi-level mediators whenever the environment demands. Another possible extension is to employ this algorithm to other distributed systems, such as distributed databases and grid computing systems. The philosophy of cluster-and-conquer is expected to be useful for large-scale distributed computing environments. So the algorithm can be extended for the processing of other types of operations, like aggregate (such as group-by, max and min), top-K, etc.[99]

R. Taylor proposed a cost model that allows inter-operator parallelism opportunities to be identified within query execution plans [93]. This allows the response time of a query to be estimated more accurately. The author has merged two existing centralized optimization algorithms DPccp and IDP1 to create a practically more efficient algorithm IDP1ccp. He proposed the novel Multilevel optimization algorithm framework that combines heuristics with existing centralized optimization algorithms. The distributed multilevel optimization algorithm (DistML) proposed in this paper uses the idea of distributing the optimization phase across multiple optimization sites in order to fully utilize the available system resources. The future work on this cost model could be done to make it capable of handling pipelining between operators, which means that one operator feeds its output tuples directly into a parent operator when they become available without writing them to disk [93].

Blanas has evaluated the join methods on a 100-node system and shown the unique tradeoffs of these join algorithms [11] in the context of MapReduce. They have also explored how their join algorithms can benefit from certain types of practical preprocessing techniques. The valuable insights obtained from their study can help an optimizer select the appropriate algorithm based on a few data and manageability characteristics. The proposed methods can be evaluated for multi-
way joins, exploring indexing methods to speedup join queries, and designing an optimization module that can automatically select the appropriate join algorithms. Another important future direction is to design new programming models to extend the MapReduce framework for more advanced analytic techniques. For computations of cost from the optimization process, the optimizer must consult the data sources involved in an operation to find the cost of that operation. The mentioned analytical process in [89] indicate that, in many cases, especially when the physical database design is known to the optimizer, this query optimization algorithm [67] works very well. But in absence of physical database design, more aggressive optimization techniques must be required.

In [105], through the research on query optimization technology, based on a number of optimization algorithms commonly used in distributed query [88], a new algorithm is designed, and experiments show that this algorithm can significantly reduce the amount of intermediate result data, effectively reduce the network communication cost, to improve the optimization efficiency. As a future work, the algorithm can be extended for distributed file system.

It is very difficult to find an ideal optimum solution for the distributed data. To obtain optimum solution, the cost of network, resources, response time, access time, memory usage, processing time, etc. should be minimized which could be done with the use of better algorithms for different principles of DDBMS, materialized views, on-the-fly schema, caching of frequently used data, etc.

### 2.5.3 Comparison of Existing Methods of Index Tuning

Database index is a physical structure which is created on table columns to improve data retrieval during transaction execution [71]. Good designed index improves performance of transactions and bad designed index worsen the performance. As index is a physical structure, it occupies memory. Once the index is defined, it requires constant monitoring. Change in workload affects performance of index. Therefore, the database needs a tool which analyze the existing indexes, create a new index or rebuild the existing and if required drop the index. There are built-in tools available with database management systems which analyze the table indexes for specific types of queries. Many third party tools are also available for transaction monitoring and tuning. These tools are not effective for complex queries. The model proposed in this paper will consider tables which are stored in heterogeneous distributed database [41], various types queries fired on this
tables, workload [12], hints from DBA, etc. and will give suggestions to build/rebuild or drop index for the improvement of transaction performance. The survey has been conducted to find feasibility before proposing the model. The survey states that proposed index evaluator and recommender will be highly useful in the IT industry. The paper focuses on comparative study of existing index analyzers, related research, literature review, survey analysis and discussion, proposed model and its features, future work and conclusion.

Index is created on a single table. There are various types of indexes. According to the usage, the appropriate index is created on the table. The database management systems provide built-in programs or tools to help design/redesign the index. Generally, DBMSs automatically create index on the primary key field irrespective of its usefulness. Also, from inputted queries, hints and explain plan commands; suggestions are generated by DBMS. This is limited up to certain types of queries. Various Index Advisor tools [43] are developed for a single database. MS SQL Server provides SQL Profiler [26] and Oracle provides SQL Tuning Advisor to improve physical design and performance of the database.

Reference [79] presented a cost model to identify indexes automatically at the run time for a specific query before its execution which is developed in DB2 and Java. Candidate indexes are identified on the basis of SELECT, WHERE, GROUP BY and ORDER BY clauses. The assumption is made that indexes are stored in a separate area in cache memory namely “Index Pool”. The heuristic is developed on the basis of difference in cost of old and new indexes. There are certain disadvantages of this model. It suggests index only for the current query. There is extra overhead because for each query, old indexes are dropped. Performance of query with new index is evaluated without implementing new index physically, which could lead to wrong decision. The maintenance cost i.e., cost of index updating (INSERT, UPDATE, DELETE) of new indexes has not been considered. The cost model of DB2 is used instead of developing new one. Future workload [12] is not considered while building new index.

Reference [82] developed COLT which is a framework implemented in PostgreSQL that suggests candidate indexes for incoming batches of 10 queries. The Extended Query Optimizer is developed which is replaced with traditional query optimizer to choose optimal execution plan based on execution time.
Reference [75] is a tool named AISIO with the heuristic implemented in PostgreSQL to recommend index. The recommended index’s details are stored in a parameter to compare it with the index generated from the benchmark DBT-2. This tool recommends index on the basis of columns used in SQL which appears in where, order by, group by and join. Columns used after having clause are not considered.

Reference [97] is a tool named “AISIO” with the heuristic implemented in PostgreSQL to recommend index. The recommended index’s details are stored in a parameter to compare it with the index generated from the benchmark DBT-2. This tool recommends index on the basis of columns used in SQL which appears in where, order by, group by and join. Columns used after having clause are not considered.

Reference [97] is an advisor for DB2 which recommends index on the basis of SQL queries fetched from SQL cache with their frequencies. The algorithm which is used in development of GUI named “Index SmartGuide” is based on knapsack algorithm. This advisor is very good as it is based on a mathematical model.

Reference [18] is an online algorithm to continuously monitor changes in the workload and revise the indexes. It also addresses and resolves the problem of oscillation.

Reference [2][3] says that indexes and materialized view both affect and improve the performance as they are physical structures. One can’t isolate them. The algorithm developed by authors considers both these structures to identify candidate materialized view. The algorithm is included as a part of AutoAdmin tool of MS SQL Server 2000. Cost is one of the performance matrixes.

Reference [15] explains the method MDI which has been developed on the basis of genetic algorithm for finding appropriate indexes for tables using group of queries. It is different than other methods in a way it considers group of queries, not individual query. According to authors, there are no other methods available which consider group of queries.

Reference [52] is a tool “Kaizen” which is a semi-automatic tuning tool for indexes. It combines features of online and offline, therefore called semi-automatic. It is an interactive tool which considers feedback and votes from DBA. It is highly scalable because it has capability to analyze large datasets. It can run on top of any database management system and hence it is portable. Kaizen uses divide-and-conquer approach. The core algorithm of Kaizen proposes new index on the basis of cost of total work done by any database management system. The limitation of Kaizen is it does not consider future workload.
Reference [21] proposed a tool SCISSOR for single column which reduces workload execution time. Authors have used algorithm to identify best candidates for indexes on the basis of complex queries.

Reference [95] is an index advisor for replicated database. It is based on divergent design tuning which suggests different indexes for replicas stored at different locations.

Reference [52] is based on the paradigm which analyzes the workload as it comes and executed, i.e. it analyzes online queries while they are running. Database administrator gives the feedback anytime in between regarding indexes which could be considered for further index suggestions. The tool named COLT [82] is developed on the basis of this paradigm and WFIT algorithm.

Most of these approaches are suggested and implemented for single relational database management systems. Few support multiple databases and replicas distributed on different sites. The tool for index analysis and design to improve concurrent transaction execution in heterogeneous distributed database is proposed in this paper [94].

### 2.5.4 Limitations of existing Index Tuning Tools

Comparison of existing tuning tools is shown in Table 2.1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Tool Name</th>
<th>Author(s)</th>
<th>Paradigm/ Tool Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Kaizen [52]</td>
<td>Ivo Jimenez</td>
<td>Semi-automatic</td>
</tr>
<tr>
<td></td>
<td>DTA (Database Tuning Advisor) [4]</td>
<td>S. Agrawal, S. Chand Hu, L. Kollar, A. Marathe, V. Narasayya, M. Syamala</td>
<td>Automatic</td>
</tr>
<tr>
<td></td>
<td>RITA (Replication-Aware Index Tuning Advisor) [95]</td>
<td>Q. trung tran, I. Jimenez, Rui Wang, Neoklis Polizotis, Anastasia Ailamaki</td>
<td>Automatic</td>
</tr>
<tr>
<td></td>
<td>SCISSOR(Single Column Index Suggestor) [21]</td>
<td>S. Chasins, P. Koonce</td>
<td>Automatic</td>
</tr>
</tbody>
</table>

[TABLE 2.1 Comparison of various tuning tools]
<table>
<thead>
<tr>
<th><strong>Inputs</strong></th>
<th>Current workload, feedback</th>
<th>Set of multiple databases, SQL statements, feature set, alignment constraints, Partial configuration, Storage and Time constraint</th>
<th>Workload, Set of features (index, materialized query table, materialized view), partition, multi-dimensional cluster, Diskspace constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outputs</strong></td>
<td>Recommend configuration (set of indexes)</td>
<td>Recommends horizontal partition, indexes, materialized views</td>
<td>Recommendaiton for subset of indexes, materialized views, partitions, multi-dimensional clustering of tables</td>
</tr>
<tr>
<td><strong>Algorithm/Heuristic/Technique used</strong></td>
<td>WFIT (based on WFA)</td>
<td>What-if analysis, Greedy search strategy, Integrated approach</td>
<td>Hybrid approach (combination of iterative and integrated approaches), Enumeration algorithm</td>
</tr>
<tr>
<td><strong>Works with DBMS</strong></td>
<td>MySQL, PostgreSQL, DB2</td>
<td>MS SQL Server 2005</td>
<td>DB2</td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td>Partial CUI and GUI</td>
<td>Both CUI and GUI</td>
<td>CUI</td>
</tr>
</tbody>
</table>
Chapter-2 Literature survey and Scope of Research

<table>
<thead>
<tr>
<th>Features</th>
<th>Analyzes queries, generate candidate indexes, candidates are converted into non-interacting (non-overlapping fields) subsets, tracks benefits of these subsets by taking feedback from DBA</th>
<th>Allows to specify manageability requirements, provides scalability through workload compression, reduced statistics creation, use test server(for recommendation implementation) to reduce load on production server</th>
<th>Built-in workload compression tool, Tool recommends and evaluates on the basis of strong and weak dependency between features</th>
<th>Handles replica failures, Balance load</th>
<th>Reduces workload execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limitations</td>
<td>Clustered indexes are not considered, only secondary indexes are considered</td>
<td>Provides recommendation for single node horizontal partitioning, Supports single column range partition</td>
<td>Combined search space of all the features(physical design structures) becomes very large</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.6 RESEARCH METHODOLOGY USED

Research methodology means approach to solve the research problem. The following research methodologies are used to solve the problem stated in section 2.2.

2.6.1 Experience Survey

The experience survey is conducted to know point of views of different people who have working experience in the area on which the researcher is doing the research. The experience survey helps research to define the research problem more precisely and concisely. Through experience survey,
researcher can collect new ideas from the respondents and with the help of this formulate the working research hypothesis.

Therefore, for this research problem, the experience survey has been conducted through online questionnaire, analyzed and used to formulate research hypothesis. The detailed questionnaire has been prepared and distributed to the IT industry people who have reasonable experience in database transaction processing. The questionnaire is attached as Appendix.

The experience survey covered questions related to person’s working experience, usage of database and indexes, tools used for index improvement and suggestions to design index evaluator and recommender with good features. The samples have been collected from the people working in the IT industry in different 38 cities of 9 countries. 46.9% people who have filled up the survey have more than 6 years of working experience in the IT industry.

Figure 2.1 shows percentage of people using Paid and/or Open Source software. 59% of the respondents use both Open Source and Paid software and only 10% use only Paid software.

![FIGURE 2.1 Percentage of People using Paid and/or Open Source Software]

About 21% of the respondents were facing transaction performance issues related to data read, write or processing in the applications and 97% of these respondents believe that bad/no indexing is the cause of transaction performance.

Figure 2.2 shows that 87% of the respondents have used one or other database. Out of these 87%, 42% of the respondents believe that no indexing or bad indexing affects transaction performance.
From 42% (87%*48%) who feel that no/bad indexing affect transaction performance, only 28% currently use a tool which gives suggestion on index to be created on tables, which means there is a potential for the rest 72% to start using such a tool if it is readily available in the market.

![Figure 2.2 Percentage of people using database and their views about effect of indexing on transaction performance]

Figure 2.3 shows that out of these 42% (87%*48%) of respondents mentioned in Figure 2.2; 34% feel there should be automatic index suggestions for the database. 26% feel there should be analysis of the current index and 23% feel there should be Suggestion to improve table structure/datatypes/constraints.

To check the feasibility of proposed model, four different chi-square tests are performed on data.

**Test-1**

Figure 2.4 displays the chi-square test of independence to test association between the variables gender and features required from proposed tool.

\[ H_0 : \text{Features required from index advisor tool is not dependent on the gender of respondents.} \]

\[ H_a : \text{Features required from index advisor tool is dependent on the gender of respondents.} \]

Alpha: 0.05

Computed p-value = 0.509
Conclusion: Since, the p-value >0.05, we fail to reject null hypothesis. Therefore, it is concluded that the features required from index advisor tool is not dependent on the gender of respondents.

![Figure 2.3 Percentage of people who have suggested features to be included in proposed tool]

**Test-2**

Figure 2.5 displays the chi-square test of independence to test association between the variables total number of year of experience of a person and features required from proposed tool.

- $H_0$: Features required from index advisor tool is not dependent on years of experience of respondents.

- $H_a$: Features required from index advisor tool is dependent on years of experience of respondents.

**Alpha: 0.05**

**Computed p-value= 0.998**

Conclusion: Since, the p-value >0.05, we fail to reject null hypothesis. Therefore, it is concluded that the features required from index advisor tool is not dependent on no. of years of experience of respondents.
Figure 2.6 displays the chi-square test of independence to test association between the variables cause of bad transaction performance and whether the respondents have used database systems.

\[ \text{FIGURE 2.4 Chi-square test to find dependency between gender and features required from proposed tool} \]

Test-3

Figure 2.6 displays the chi-square test of independence to test association between the variables cause of bad transaction performance and whether the respondents have used database systems.

\[ H_0: \text{The cause of bad transaction performance is independent of whether the respondents have used database systems.} \]

\[ H_a: \text{The cause of bad transaction performance is dependent on whether the respondents have used database systems.} \]

\[ \text{Alpha: 0.05} \]

\[ \text{Computed p-value= 0.00003232} \]
Conclusion: Since, the p-value < 0.05, we reject null hypothesis. Therefore, the cause of bad transaction performance is dependent on whether the respondents have used database systems.

[FIGURE 2.5 Chi-square test to find dependency between total no. of years of experience and features required from proposed tool]

[FIGURE 2.6 Chi-square test to find dependency between cause of bad transaction performance and whether the respondents have used the database system]
**Test-4**

This test is performed in PHstat to check demand of Table Index Evaluator and Recommender tool by the people who are already using other tool for index tuning. Figure 2.7 and Figure 2.8 shows the details of this test.

- **H₀**: There is no association between satisfaction level of usefulness of present indexing tool used by respondent and a new proposed tool with advanced features.

- **H₁**: There is association between satisfaction level of usefulness of present indexing tool used by respondent and a new proposed tool with advanced features.

**Alpha**: 0.05  
**Computed p-value** = 0.483225087

Conclusion: Since the p-value > 0.05, the null hypothesis is not rejected which means that people who are using an index tuning tool will use the proposed tool if it is provided with advanced features.

Figure 2.7 shows that at least one of the cells in data is less than 5, so we have to apply Yate's correction. Figure 2.8 shows the result after applying Yate's correction manually. This result also states that do not reject the null hypothesis.

Thus, the survey says that approximately 93.8% people are interested in the proposed Index evaluator and recommender and strongly supports the model.

**2.6.2 Data Collection**

The data collected to test the hypothesis should be standard data. Hypothesis should be tested for first for secondary data and then for the primary data. Secondary data means the data which is already available and research has been done on such data, while primary data means the data on which is not used by any researcher to test the hypothesis. This research problem is based on the specific case and it follows experimental research methodology, the secondary data which is used is “Inventory Data”. The primary data used is provided the company itself which are live data of one of their clients. As the data is highly confidential, the data set has not been disclosed and the name of the client has not been mentioned anywhere in the thesis. But photocopy of the certificate
of the company has been shown as a proof. Apart from this, the secondary data is taken from TPC-H benchmark which proves the authenticity of hypothesis testing.

<table>
<thead>
<tr>
<th>Observed Frequencies</th>
<th>Column variable (Interest shown in proposed tool)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>Satisfied</td>
<td>10</td>
</tr>
<tr>
<td>Not Satisfied</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expected Frequencies</th>
<th>Column variable (Interest shown in proposed tool)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>Satisfied</td>
<td>8.8</td>
</tr>
<tr>
<td>Not Satisfied</td>
<td>7.2</td>
</tr>
<tr>
<td>Total</td>
<td>16</td>
</tr>
</tbody>
</table>

Level of Significance: 0.05
Number of Rows: 3
Number of Columns: 2
Degrees of Freedom: 2

Results
- Critical Value: 5.991464547
- Chi-Square Test Statistic: 1.454545455
- p-Value: 0.483225087

Do not reject the null hypothesis

[FIGURE 2.7 Chi-square test to find association between satisfaction level of usefulness of present indexing tool used by respondent and a new proposed tool with advanced features.]

<table>
<thead>
<tr>
<th>Column variable (Interest shown in proposed tool)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Satisfied</td>
</tr>
<tr>
<td>Not Satisfied</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Observed Frequency (O)</th>
<th>Expected Frequency (E)</th>
<th>(O-E)^2</th>
<th>(O-E)^2/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfied, Required</td>
<td>10</td>
<td>8.8</td>
<td>0.7</td>
</tr>
<tr>
<td>Not Satisfied, Required</td>
<td>6</td>
<td>7.2</td>
<td>-1.7</td>
</tr>
<tr>
<td>Satisfied, Not Required</td>
<td>1</td>
<td>2.2</td>
<td>-1.7</td>
</tr>
<tr>
<td>Not Satisfied, Not Required</td>
<td>3</td>
<td>1.8</td>
<td>0.7</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td>4</td>
<td>2.042929293</td>
</tr>
</tbody>
</table>

P 3.84146

[FIGURE 2.8 Computed p-value after applying Yate’s correction manually]

2.6.3 Statistical Analysis

Statistical analysis includes various statistical methods to test the hypothesis. The method using which the research hypothesis has been tested is chi-square method. The statistics are collected
from the results of experiments done on the data and then chi-square method is applied to test the hypothesis. The detailed statistical analysis is discussed in the chapter related to analysis of results obtained from experiments.

2.6.4 Experimental

The research presented in this thesis is empirical or experimental because it is based on the experiments done. The experiments have been done to improve concurrent transactions performance [90] [96] in multiple heterogeneous distributed database system by doing comparison of before index suggestion and after index suggestion. The tool named TIER-Table Index Evaluator and Recommender has been developed which would suggest new indexes for the tables. These new indexes are applied on the tables and their results are compared with the tables before index recommendation. The experiment is done on the standard dataset generated by the database benchmark TPC-C. The environment of distributed database system has been created to run the distributed concurrent transactions [46].

2.6.5 Benchmarking

Benchmark is a standard which is used for performance comparison [10]. TPC is a Transaction Processing Performance Council which has given benchmarks for database. To compare the results generated by a tool developed during this research work, the TPC-C and TPC-H benchmarks [101] are used. TPC-C benchmark is a benchmark for OLTP (Online Transaction Processing). TPC-H benchmark is a benchmark for decision support and concurrent transaction execution. Both these benchmarks are used to generate high volume of data, standard set of queries and result comparison. Benchmarks are very useful in experimental research.

2.7 RESEARCH HYPOTHESIS FORMULATION

From the experience survey and analysis, the model for table index evaluation and recommendation has been proposed and developed to improve the concurrent transaction performance in multiple heterogeneous distributed database system. The following working research hypothesis has been formulated from the survey analysis (discussed in section 2.6.1), which is tested after development of the proposed tool named TIER.
H0 : There is no improvement in performance after implementing indexes recommended by tool Table Index Evaluator and Recommender.

H1 : There is improvement in performance after implementing indexes recommended by tool Table Index Evaluator and Recommender.

Testing of the above hypothesis is done in Chapter-5.

2.8 CHAPTER SUMMARY

The chapter covered research problem, objective and scope of the research, detailed literature review, research methodologies used during research, analysis of experience survey and formulation of working research hypotheses from the experience survey.