Tuning SQL Statements by Implementing Mobile Agent Transparent Interface Layer

تنغيم أوامر لغة الاستفسار المهيكلة باستحداث طبقة

وسطية شفافة للوكيل المتنقل

By

Imad Hasan Saleh

Supervisor

Prof. Dr. Alaa Hussein Al-Hamami

This Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Computer Science.

Department of Computer Science

Graduate College of Computing Studies

Amman Arab University for Graduated Studies

May, 2009

Amman Jordan

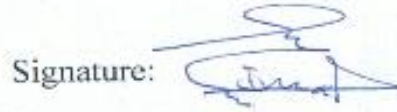
المنارة للاستشارات

www.manaraa.com

AUTHORIZATION OF DISSEMINATION

I, the undersigned "Imad Hasan Saleh" authorize hereby Amman Arab University for Graduated Studies to provide copies of this thesis to libraries, institutions, agencies, organizations, individuals and any other parties upon their request.

Name: Imad Hasan Saleh

Signature: 

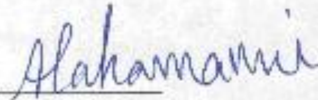
Date: 5/8/2009

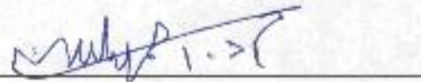
المنارة للاستشارات
www.manaraa.com

APPROVAL

This thesis titled "Tuning SQL Statements by Implementing Mobile Agent Transparent Interface Layer", has been successfully defended and approved by the examining committee on 5 / 8 / 2009.

Prof. Dr. Reyad Farhan Al-Shalabi, Chair.

Prof. Dr. Alaa Hussein Al-Hamami, Member and Supervisor. 

Prof. Dr. Ahmad Al-Jaber, Member. 

المنارة للاستشارات
www.manaraa.com

Acknowledgments

First of all, my prayers and thanks to ALLAH, Who helped me complete this thesis. I would like to thank my supervisor Prof. Dr. Alaa Hussein Al-Hamami for his support, patience, and good directions that guided me toward a qualitative methodology.

I would like to thank the examining committee, all my colleagues and relatives for supporting me from the beginning. Consequently many thanks go to all lecturers, and the administrative staff of Amman Arab University for their support.

All thanks to my mother for her prayers and courage, and special thanks to my adored wife who from the beginning had confidence in my abilities not only to attain a degree, but also to attain it with excellence. Also a lot of thanks to my brothers and sisters.

المنارة للاستشارات
www.manaraa.com

Dedication

I dedicate this work to my patient mother.

To my adored wife.

To my lovely children.

Imad

المنارة للاستشارات

www.manaraa.com

تنغيم أوامر لغة الاستفسار المهيكلة باستحداث طبقة

وسطية شفافة للوكيل المتنقل

إعداد

عماد حسن صالح

إشراف

الاستاذ الدكتور علاء حسين الحمامي

الملخص

Arabic Summary

يعتمد أداء أنظمة قواعد البيانات بشكل كبير على أوامر ال SQL المكتوبة داخل هذه التطبيقات. ولذلك يترتب على المبرمجين ان يكتبوا أوامر SQL محسنة ليتم كسب قدر اكبر من الاداء. ولسوء الحظ بعض المبرمجين هم مبرمجين مبتدئين وبعضهم الاخر غير مهتم بكتابة اوامر محسنة. لتحسين الاداء يجب فتح البرامج من قبل المبرمجين والتغير على الاوامر واعادة كتابتها.

هذا البحث يقدم الوكيل كطبقة وسيطة بين واجهة التطبيق من جهة و قاعدة البيانات من جهة اخرى. مسؤليات الوكيل هي التقاط الاوامر المبعوثة من واجهة التطبيق قبل وصولها الى قاعدة البيانات وفحص هذه الاوامر، وتصحيح الاخطاء ان وجدت، ثم اعادة كتابة هذه الاوامر بطريقة محسنة اذا لم تكن مكتوبة بطريقة محسنة. هذه الاطروحة سوف تركز على اعادة كتابة أوامر SQL بدون التغير على المنظومة.

هذا البحث يعرض التصميم والتطبيق للوكيل كحل لتحسين اداء المنظومات. فالبحث يحتوي على مكونات الوكيل الاساسية والأبتدائية، وكيفية تواصل هذه المكونات مع بعضها البعض لتحقيق الغاية المطلوبة.

المنارة للاستشارات

www.manaraa.com

في هذا البحث يوجد ايضا تحليل لأداء الوكيل، ومقارنة بين بيئة تعمل بدون وجود الوكيل وبيئة اخرى مع وجود الوكيل من خلال ادوات قياس واحصـــائيات يتم جمعها من كل بيئة، مثل الوقت الفعلي المستغرق لتنفيذ الأمر، كمية القراءة من قاعدة البيانات، كمبة القراءة من الذاكرة، واحصائيات اخرى.

معيار الفحص الناتج عن قياس اداء الوكيل مرتبط بشـكل كبير وايجابي مع وجود الوكيل في هذه البيئة مقارنة مع عدم وجود الوكيل في نفس البيئة. حيث ان وجود الوكيل يسرـــع الاداء بمقدار يزيد عن 99% مقارنة مع عدم وجود الوكيل في المنظومة.

ان وجود الوكيـل يزيـل فكرة اي تغير ســـوف يتم على المنظومـة المكتوبـة، بـل بـالعكس فـانه يســتخدم نفس الميزات الموجودة في قاعدة البيانات، ولهذا فان وجود الوكيل ســوف يقلل التكلفة المنفقة على تحسين اداء قاعدة البيانات.

المنـــارة للاستشارات
www.manaraa.com

Tuning SQL Statements by Implementing Mobile Agent Transparent Interface Layer

Prepared by

Imad Hasan Saleh

Supervisor

Prof. Dr. Alaa Hussein Al-Hamami

ABSTRACT

Database applications performance depends on embedded SQL commands written inside the application. So application developers have to write optimized SQL commands to gain performance. Unfortunately some of application developers are novice programmers and the others do not care about performance. To gain performance, application source code should be opened by programmers and SQL commands should be written again.

This research presents an agent as a middle layer between application interface front end and database back end. The responsibilities of the agent are catching the SQL commands sent by application before reaching the database then examining these commands, and correcting them if they have errors then rewrite them in a tuned format if the SQL commands are not tuned. This thesis will focus on rewriting the SQL commands without application modification.

المنارة للاستشارات

www.manaraa.com

This research presents the design and implementation of the agent as a proposed solution for SQL command tuning. It includes the basic building blocks of the agent, and how these blocks communicate with each other to achieve the requirements.

In this thesis, the researcher describes the solution for each reason that makes the SQL command slow, designs an agent to receive poor commands and translates them to efficient commands, implements the agent in real word environment, and tests the results by comparing the old commands with new translated commands. All of this work is done without opening the application source code and modifying it, the agent will do all of the work transparently without affecting the running application.

In this research, there is also a representation of performance analysis for the agent, comparing the SQL command with no agent environment by SQL command after implementing the agent through considering performance measurements metrics and statistics, such as real time, physical reads, logical reads and others.

The performance measurements benchmark test results of the proposed agent were highly and positively correlated with acceptance levels, and amazing achievement to score more than 99% of performance enhancement compared with no agent environment.

Mobile agent solution eliminates any development changes on the existing system, and uses existing features of the database, so the agent reduces the financial cost of application tuning.

المنارة للاستشارات

www.manaraa.com

Table of Contents

Dedication....V
Arabic Summary....VI
ABSTRACT....VIII
Table of Contents....X
List of Tables....XV
Chapter one Introduction....1
1.1. Overview....1
1.2 Managing Performance....3
1.3 Managing Factors....4
1.4 Mobile Agent....6
1.5 Overview of SQL language....9
1.5.1 Query....11
1.5.2 Data Manipulation....13
1.5.3 Transaction Control....14
1.5.4 Data Definition....14
1.5.5 Data Control....15
1.5.6 Test Case Database....15
1.6 Overview of SQL Statement Tuning....15
1.6.1 Overview....16
1.6.2 Sequential Search versus Binary Search....17
1.7 Extensible Markup Language....19
1.7.1 XML Elements....20
1.7.2 XML Attributes....21
1.7.3 XML and HTML....22
1.8 The Problem Statement....23
1.9 Thesis Contribution....25
1.10 Thesis Organization....26
Chapter two Background and related work....28
2.1 Overview....28
2.2 Literature Survey....29
2.2.1 Query Rewrite Using Materialized Views....29
2.2.2 Query Rewrite and XML....32
2.2.3 Enhancing Optimizer Capabilities....34

المنارة للاستشارات
www.manaraa.com

2.2.4 Mobile Agent 35
2.2.5 Miscellaneous 38
2.3 Detailed Problem Statement 40
2.4 Problem Still Exists 45
Chapter Three Proposed System Design 46
3.1 Overview 46
3.2 Optimizer 46
3.3.1 Parse Phase 50
3.3.2 Bind Phase 50
3.3.3 Execute Phase 51
3.3.4 Fetch Phase 51
3.3.5 Query Execution Plan 51
3.4 The Proposed System Design 54
3.5 Detailed System Design 57
3.5.1 Login Use-case 62
3.5.2 Validate Dictionary Use-case 62
3.5.3 Check Syntax Use-case 63
3.5.4 Check for Rewrite Use-case and Rewrite Use-case 64
3.5.5 Compare Results Use-case 65
3.5.6 Save Results to History Use-case 66
3.6 Test Case Design 67
Chapter Four Implementation and Experimental Work 70
4.1 Overview 70
4.2 Metrics and Statistics 70
4.3 Access Paths 72
4.4 Use-case Implementation 75
4.4.1 Login Use-case 75
4.4.2 Validate Dictionary Use-case 78
4.4.3 Check Syntax Use-case 81
4.4.4 Check for Rewrite Use-case and Rewrite Use-case 86
4.4.4.1 Check for Calculation 86
4.4.4.2 Check for Type Conversion 90
4.4.4.3 Check for Index Usage 93
4.4.4.4 Check for Join 96
4.4.5 Compare Results Use-case 102
4.4.6 Save Results to History Use-case 106
Chapter Five Agent Performance Analysis 108

المنارة للاستشارات

www.manaraa.com

5.1 Overview 108
5.2 Performance Analysis Measurements 108
5.3 Calculations, Type Conversion, and Index Usage Benchmarks 111
5.3.1 Experimental Scenarios 112
5.3.2 Variables Discipline 115
5.3.3 Measurement Benchmark 115
5.3.4 Results Comparison 118
5.3.4.1 Real Time Test Results Comparison 118
5.3.4.2 CPU Cost Test Results Comparison 120
5.3.4.3 IO Cost Test Results Comparison 121
5.4 Join Benchmarks 123
5.4.1 Experimental Scenarios 123
5.4.2 Variables Discipline 125
5.4.3 Measurement Benchmark 126
5.4.4 Results Comparison 128
5.4.4.1 Real Time Test Results Comparison 128
5.4.4.2 Consistent Gets Test Results Comparison 130
5.5 Overhead Has No Influence 131
5.6 Statistics Limitations 132
Chapter Six Conclusion and Future Work 133
6.1 Overview 133
6.2 Thesis Conclusion 133
6.2.1 Performance 133
6.2.2 Performance Enhancements 134
6.2.3 Using Agent 134
6.2.4 Performance Analysis 135
6.3 Recommendations for Future Work 135
6.3.1 Development of Proposed Solution Field 135
6.3.2 Performance Analysis of Proposed Solution Field 136
References 137

المنارة للاستشارات

www.manaraa.com

Acronyms and Abbreviations

ANSI	American National Standards Institute
BLOB	Binary Large Object
CLOB	Character Large Object
CPU	Central Processing Unit
DBA	DataBase Administrator
DBMS	DataBase Management System
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DSS	Decision Support System
HTML	HyperText Markup Language
IO	Input Output
IT	Information Technology
JDBC	Java DataBase Connectivity
MA	Mobile Agent
MAMDAS	Mobile Data Access System framework
MS	MicorSoft
MV	Materialized View
OCI	Oracle Call Interface
OLAP	OnLine Analytical Processing
OLTP	OnLine Transactional Processing
QEP	Query Execution Plan
RDBMS	Relational DataBase Management System
SAS	Security Annotated Schema
SEQUEL	Structured English QUEery Language
SQL	Structured Query Language

المنارة للاستشارات
www.manaraa.com

SQO	Semantic Query Optimization
SQR	Secure Query Rewrite
SSN	Social Security Number
TCP	Transmission Control Protocol
VARCHAR	Variable Character
W3C	World Wide Web Consortium
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

المنارة للاستشارات
www.manaraa.com

List of Tables

Table (4-1): Tables Summary 73
Table (5-1): Metrics for Different Row Number 88
Table (5-2): Statistics for Different Row Number 88
Table (5-3): Metrics for Join 95
Table (5-4): Statistics for Join 96

المنارة للاستشارات
www.manaraa.com

LIST OF FIGURES

Figure (1-1): Mobile Agent Movement 6
Figure (1-2): Mobile Agent 7
Figure (1-3): XML Document 15
Figure (1-4): Parts of database applications 17
Figure (2-1): EMPLOYEES table Structure 30
Figure (2-2): Results of first SQL command 30
Figure (2-3): Results of second SQL command 31
Figure (3-1): Optimizer Operations 35
Figure (3-2): SQL Statement Processing Phases 36
Figure (3-3): Nested loops join 38
Figure (3-4): Sort-Merge joins 39
Figure (3-5): Proposed System 40
Figure (3-6): System Use-Case Diagram 42
Figure (3-7): System Class Diagram 44
Figure (3-8): Login Sequence Diagram 45
Figure (3-9): Validate Dictionary Sequence Diagram 46
Figure (3-10): Check Syntax Sequence Diagram 46
Figure (3-11): Check for Rewrite Sequence Diagram 47
Figure (3-12): Compare Results Sequence Diagram 48
Figure (3-13): Save Results to History Sequence Diagram 49
Figure (3-14): Database Design for Conference System 50
Figure (4-1): Metrics and Statistics 52
Figure (4-2): Login Dialog 56
Figure (4-3): Login Flowchart Diagram 57

المنارة للاستشارات
www.manaraa.com

Figure (4-4): Dictionary XML File 58
Figure (4-5): Validate Dictionary Flowchart Diagram 59
Figure (4-6): Agent Interface 61
Figure (4-7): Agent Online Syntax Checking 61
Figure (4-8): Agent Offline Syntax Checking 62
Figure (4-9): Check Syntax Flowchart 63
Figure (4-10): Simple Operation Rewriting 65
Figure (4-11): Complex Operation Rewriting 65
Figure (4-12): Check for Calculation Flowchart 66
Figure (4-13): Type Conversion Rewriting 68
Figure (4-14): Check for Type Conversion Flowchart 69
Figure (4-15): Index Usage Rewriting 70
Figure (4-16): Check for Index Usage Flowchart 71
Figure (4-17): Check for Join Rewriting 74
Figure (4-18): Check for Join Flowchart 75
Figure (4-19): Comparison in Progress 77
Figure (4-20): Comparison is Finished 77
Figure (4-21): Compare Results Flowchart 78
Figure (4-22): Comparison XML File 79
Figure (4-23): Save Results Flowchart 80
Figure (5-1): Logarithmic Chart for Real Time Results Comparison 89
Figure (5-2): Logarithmic Chart for CPU Cost Results Comparison 90
Figure (5-3): Logarithmic Chart for IO Cost Results Comparison 91
Figure (5-4): Logarithmic Chart for Consistent Gets Results Comparison 92
Figure (5-5): Logarithmic Chart for Real Time Results Comparison 97
Figure (5-6): Logarithmic Chart for Consistent Gets Results Comparison 98

المنارة للاستشارات

www.manaraa.com

Chapter one

Introduction

1.1. Overview

Over the last half century DataBase Management System (DBMS) has been very mass-produced and successful, even about $15 billion in 2005 of the worldwide market was for DBMS software with a 10% estimated annual growth. Database applications become a core component in most organizations [27]. These systems are becoming increasingly complex and the task of management to ensure acceptable performance for all applications is very difficult. In recent years, this complexity has approached a point where even DataBase Administrators (DBAs) and other highly skilled information technology professionals are unable to comprehend all aspects of a DBMS's day-to-day performance and manual management has become virtually impossible [13].

Nowadays it is very common to deal with large size databases containing millions and even trillions of records. It is not rare to see databases with gigabyte size or even with terabyte size. Query language can be used to access these databases and as we know query language should be declarative, so we can write alternative formulas to perform same query. And different query formulas provide variation in performance.

It is widely accepted to have a large number of users accessing the database at the same time, so different type of

المنارة للاستشارات
www.manaraa.com

queries submitted to database to perform with minimal response time. Sometimes number of simultaneously connected users exceeded tens of millions.

Also database must support different types of applications starting from OnLine Transactions Processing (OLTP) applications to data ware housing and Decision Support System (DSS) applications. This variation of applications need different storage types starting from simple numbers and characters towards images, sounds and videos, even so documents and files.

Large database size, large number of users, and variation of applications can put the database in a difficult situation, even so in critical situation and this may cause Relations DataBase Management System (RDBMS) hanging and jamming. In spite of that we need good performance and acceptable response time, so we have to overcome all of these situations.

Database Application efficiency and performance strongly depends on SQL commands written by programmers, so programmers have to write SQL commands on optimized way. Unfortunately some programmers are novice and may have misspelling commands, even experts do not care about performance. To gain some performance, application should be edited and SQL commands should be written again. This thesis will focus on rewriting tuned SQL commands without application modification.

المنارة للاستشارات
www.manaraa.com

Application tuning is one of the major areas for database administrator and application developers, and needs a lot of efforts and it is time consuming. Anyone involved in tuning should follow a tuning methodology to achieve maximum performance. Tuning SQL statements is an important step that is least expensive when performed at its proper moment in the methodology [21]. In addition to tuning at the right time, you should also have a good understanding of the issues involved in performance management and the types of performance problems that might arise.

1.2 Managing Performance

Performance Tuning can be achieved in several stages and implemented in all system development life cycle faces. Tuning spans through system development life cycle, starts from analysis stage and extends to deployment stage, but focus in tuning increased in design and development stages. Application tuning is team dependent, and can be achieved by joining the force of database administrators, system administrators, analysts and programmers. So these parties can define their objectives very clearly and can put measurable tasks, because specific tuning tasks can be achieved and completed in a short time in contrast with general problems. A tuning baseline will come up to scene and a comparison can be done to see the amount of gain or lost from previous tuning tasks. A database of tuning problems and solutions can be registered and managed to be as a knowledge base for performance tuning tasks, and this knowledge base can be very helpful for anyone dealing with database tuning. Also this knowledge base can help us to monitor the changes and to measure the influence of these changes to performance.

المنارة للاستشارات
www.manaraa.com

Typically, 20% of database transactions (including select statement) account for 80% of the system usage. The transactions that make up this 20% are the ones that must be tuned, and tuning anything else is probably not worth your time and energy. You can supplement this phenomenon with the following corollary: 50% of the utilization comes from 5% of database transactions, which means that most of database resources are consumed by a few statements. Tuning these few statements reduces the amount of resources consumed, thereby performance will be increased [21].

1.3 Managing Factors

Performance management can be divided into the following areas. Although the areas are separate, they are also interdependent and require different skill sets [21].

Schema Tuning: Schema tuning deals with the physical structure of the data. If an application has inadequate or inappropriate data design, then tuning the physical allocation, providing indexes or rewriting programs will not overcome the problem.

Application Tuning: Application tuning deals with such business functions as 24/7 availability, OLAP, OLTP, as well as the program modules or applications that implement the functions. Tuning the procedural code for the type of application and tuning the embedded SQL statements are also included in this area. If an application is well designed, it may still perform badly. A common reason for this is badly written SQL.

المنارة للاستشارات
www.manaraa.com

Instance Tuning: Each database caches its data in server memory to speed next access of the same data, the cache location is called an instance and it dramatically affects the application performance.

Database Tuning: Database tuning deals with managing the physical arrangement of data on disk. If data is distributed in multi disk drive, performance can be increased by parallel reading and writing instead of sequential use.

User Expectations: Users expect consistent performance. They can cope with slower application functions if they understand why the application is slower than usual. The project team should try to build a realistic user expectation of performance, possibly including application messages to warn operators that they are requesting operations that are resource-hungry. The best time to do this is before the design and build phases and as part of the transition phase.

Hardware, Network Tuning: It deals with performance issues arising from the CPU and from network traffic, the main hardware components are listed below:

CPU: There can be one or more CPUs, and they can vary in processing power from simple CPUs found in hand-held devices to high-powered server CPUs on system.

Memory: Database and application servers require considerable amounts of memory to cache data and avoid time-consuming disk access.

المنارة للاستشارات

www.manaraa.com

I/O Subsystem: The I/O subsystem can vary between the hard disk on client PC and high performance disk arrays. Disk arrays can perform thousands of I/O each second and provide availability through redundancy in terms of multiple I/O paths and hot pluggable mirrored disks.

Network: The primary concerns with network specifications are bandwidth (volume) and latency (speed).

1.4 Mobile Agent

Mobile Agent (MA) systems have for some time been seen as a promising paradigm for the design and implementation of distributed applications. A mobile agent is a program that can autonomously migrate between various nodes of a network and perform computations on behalf of a user. Some of the benefits provided by MAs for creating distributed applications include reduction in network load, overcoming network latency, faster interaction and disconnected operations [8].

Mobile agent is defined as computer software and its data which has the ability to move from one machine to another transparently to continue its work in the second machine. Also mobile agent has the ability to learn from previous experience as shown in Figure (1-1).

المنارة للاستشارات
www.manaraa.com

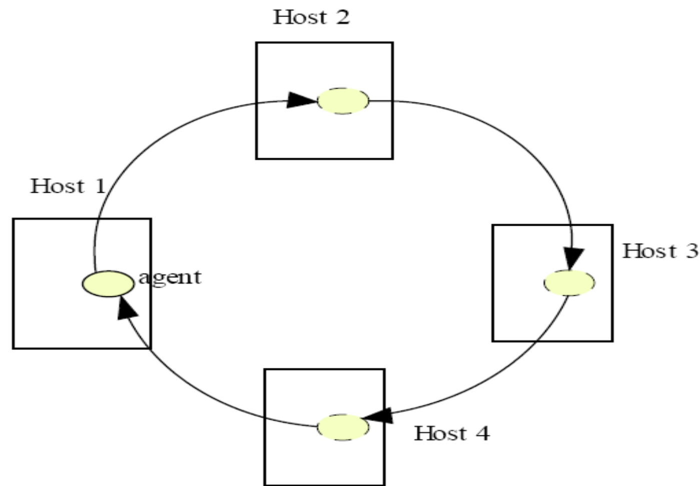


Figure (1-1) Mobile Agent Movement

A suitable application for mobile agents is the electronic commerce. Mobile agent technologies can be used to automate several of the most time consuming stages of the buying process. Unlike "traditional" software, mobile agents are personalized, and autonomous. MA move around the network searching for a user specified product across different shops. With the MA moving to the shops, the number of information exchange is local and is not over the network, thus saving network latencies and load. Using the mobile agent technology client specific queries could be executed at the shops site. Qualities inherent to MAs are conducive for optimizing the whole buying experience and revolutionizing e-commerce over then net. Rahul Jha [23] believes that the effective use of mobile agents can dramatically reduce transaction cost in general.

Client server model versus mobile agent model. Most of new technologies prefer to use mobile agent instead of traditional client server model to overcome the client server limitations such as

المنارة للاستشارات
www.manaraa.com

flexibility, scalability, cost, fault tolerance, and of course portability. All of these features exist in client server architecture but can be enhanced and improved if we switched to mobile agent model. In client server model data processed in pipelined fashion at server side and all client information has to be transferred to server throw network. In mobile agent model, the agent will move from client to client and process the information locally at client side and the architecture will be as multi server model. So instead of moving data between machines to process it, the program (agent) moves to data location to process it locally as shown in Figure (1-2) [15]. This approach leads us to Grid technology.

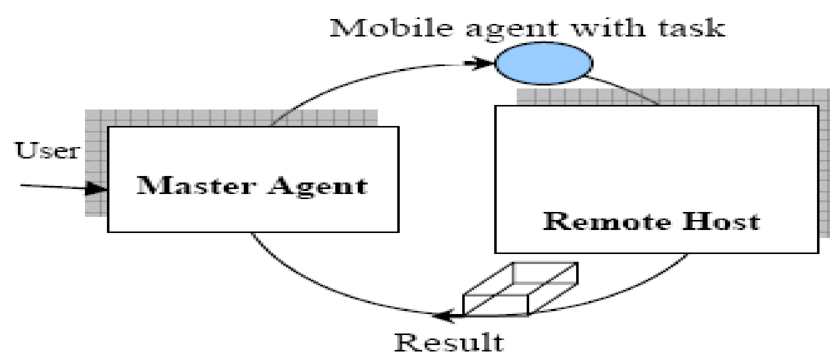


Figure (1-2) Mobile Agent [15]

By using mobile agent model, processing task can be partitioned into multiple lightweight agents. This family of agents is distributed among the cluster and competes with computing resources. This approach of computation is advantageous in that the system operates as an autonomous entity. Agents execute as a collaborative team, working around node failures and system bottlenecks. Additional computing resources can be added and exploited dynamically, enhancing both the system flexibility and scalability [15].

المنارة للاستشارات
www.manaraa.com

One of the best justifications for using mobile agent is that the paradigm is specially appropriate for computing on network devices. This is so because, with proper implementation, mobile agents [15]:

1- Allow efficient and economical use of communication channels which may have low bandwidth, high latency, and may be error-prone.
2- Enable the use of portable, low-cost, personal communications devices to perform complex tasks even when the device is disconnected from the network.
3- Another attractive property of the mobile agent paradigm is that it allows an application to be truly distributed, as much as the tasks involved in an application,

1.5 Overview of SQL language

The name SQL is derived from Structured Query Language. Originally, SQL was called SEQUEL (for Structured English QUEery Language) and was designed and implemented as IBM research and as the interface for an experimental relational database system called SYSTEM R. SQL is now the standard language for commercial relational DBMSs. A joint effort by ANSI (American National Standards Institute) and ISO (International Standards Organization) has led to standard version of SQL (ANSI 1986) called SQL-86 or SQL1. A revised and much expanded standard called SQL2 also referred to as SQL-92 was subsequently developed. The next version of the standard was originally called SQL3, but it is now called SQL-99 [24].

The SQL language may be considered one of the major reasons for the success of relational database in the commercial

المنارة للاستشارات
www.manaraa.com

world. It became a standard for relational databases; so users were less concerned about migration of their database applications from other types of database systems –for example, network or hierarchical systems- to relational systems. The reason is that even if users became dissatisfied with the particular relational DBMS product they chose to use, converting to another relational DBMS product would not be expected to be too expensive and time-consuming, since both systems would follow the same language standards. In practice, of course, there are many differences between various commercial relational DBMS packages but they still have the same standard, so each vender can put his flavor and enhancements to enhance productivity, as an example we may find statements in ORACLE database that not do exist in MS-SQL database, but they both still implement the standard SQL language. Another advantage of having such a standard is that users may write statements in a database application program that can access data stored in two or more relational DBMSs without having to change the database sublanguage (SQL) if both relational DBMS support standard SQL.

SQL is a comprehensive database language: It has statements for Data Definition (DDL), Data Manipulation (DML), and Data Control (DCL). In addition it has facilities for defining views and synonyms on the database. The latest SQL-99 standard is divided into a (core) specification plus optional specialization (packages). The core is supposed to be implemented by all RDBMS venders. The packages can be implemented as optional modules to be

المنارة للاستشارات
www.manaraa.com

purchased independently for specific database applications such as data mining, spatial data, temporal data, data warehousing, on-line analytical processing (OLAP), multimedia data, and so on.

1.5.1 Query

The most common operation in SQL databases is the query, which is performed with the declarative SELECT keyword. SELECT retrieves data from a specified table, multiple related tables in a database or the result of an expression. While often grouped with Data Manipulation Language (DML) statements, the standard SELECT query is considered separate from SQL DML, as it has no persistent effects on the data stored in a database. Note that there are some platform-specific variations of SELECT that can persist their effects in a database, such as the SELECT INTO syntax that exists in some databases to put the selected value into a variable, or SELECT FOR UPDATE to lock the selected record from update or delete until releasing this record by COMMIT or ROLLBACK command.

SQL queries allow the user to specify a description of the desired result set, but it is left to the devices of the database management system (DBMS) to plan, optimize, and perform the physical operations necessary to produce that result set in as efficient a manner as possible. An SQL query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk "*" can also be used as a "wildcard" indicator to specify that all available columns of a table (or multiple tables) are to be returned [30]. SELECT is the most complex statement in SQL, with several optional keywords and clauses, including:

المنارة للاستشارات

www.manaraa.com

1- The FROM clause which indicates the source table or tables from which the data is to be retrieved. The FROM clause can include optional JOIN clauses to join related tables to one another based on user-specified criteria.
2- The WHERE clause includes a comparison predicate, which is used to restrict the number of rows returned by the query. The WHERE clause is applied before the GROUP BY clause. The WHERE clause eliminates all rows from the result set where the comparison predicate does not evaluate to true.
3- The GROUP BY clause is used to combine, or group, rows with related values into elements of a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregate functions or to eliminate duplicate rows from a result set.
4- The HAVING clause includes a comparison predicate used to eliminate rows after the GROUP BY clause is applied to the result set. Because it acts on the results of the GROUP BY clause, aggregate functions can be used in the HAVING clause predicate.
5- The ORDER BY clause is used to identify which columns are used to sort the resulting data, and in which order they should be sorted (options are ascending or descending). The order of rows returned by an SQL query is never guaranteed unless an ORDER BY clause is specified.

The following is an example of a SELECT query that returns a list of rich employees. The query retrieves all rows from the Employees table in which the Salary column contains a value

المنارة للاستشارات
www.manaraa.com

greater than 5000.00. The result is sorted in ascending order by Name. The asterisk (*) in the select list indicates that all columns of the Employees table should be included in the result set.

```
SELECT      *
FROM        Employees
WHERE       Salary > 5000.00
ORDER BY    Name
```

The example below demonstrates the use of multiple tables in a join, grouping, and aggregation in an SQL query, by returning a list of Departments and the number of Employees working in each department. Note the alias of the count(*) column (No_Of_Employees) used after AS keyword, also the table Department alias (d), Employee table alias (E).

```
SELECT      d.Name, count(*) AS No_Of_Employees
FROM        Departments d, Employees E
WHERE       d.Deptno = E.Deptno
GROUP BY    d.Name
```

1.5.2 Data Manipulation

The standard DML commands used for entering data to table using INSERT command, updating old values by new values by using UPDATE command and removing data from table by DELETE command. The example below demonstrates the three commands:

```
INSERT INTO Employees (Id,Name,Salary,Deptno)
VALUES (109,'Basem',3000.00,10)
```


المنارة للاستشارات
www.manaraa.com

```
UPDATE   Employees
SET      Salary = 3500.00
WHERE    Id = 109
DELETE FROM Employees
WHERE    Id = 109
```

The interesting thing is that we can put a SELECT statement in a DML command if we want to insert data from table to another table, updating records retrieved by select statement, or deleting record retrieved by select statement. In SQL:2003 a new command came out, the MERGE command is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements.

1.5.3 Transaction Control

The transaction is the group of commands ended by COMMIT or ROLLBACK command. Once the COMMIT statement has been executed, the changes cannot be rolled back. In other words, it is meaningless to have ROLLBACK executed after COMMIT statement and vice versa

1.5.4 Data Definition

Data definition language (DDL) consists of five commands (CREATE, ALTER, RENAME, DROP, TRUNCATE) which allow to control the structure of the table and to change the definition of the table.

CREATE is used to present an object such as table in the database. ALTER is used to modify the structure of existing table, such as adding column, modifying existing column type, removing column from table. RENAME is used to change the table name. DROP is

المنارة للاستشارات
www.manaraa.com

used to remove the object structure and its data. TRUNCATE is used to remove the data but remaining the table structure (fast delete).

1.5.5 Data Control

The two data control language (DCL) commands (GRANT, REVOKE) are used to control the privileges about an object. With GRANT command, we can permit other users to access or manipulate our objects and with REVOKE command we forbid others from accessing or manipulating our objects.

1.5.6 Test Case Database

The test database used to execute the SQL commands will be relational database management system and one of the databases that dominates the market place with a lot of capabilities and features. The relations created in this database will hold starting from two million records in a relation, to 10 million records in a relation.

1.6 Overview of SQL Statement Tuning

Putting any application in production phase can reveal a lot of factors and problems of how transactions and queries use the database. Resource utilization and database query optimization can be discovered. Most of relational database management systems RDBMSs provide facilities to see the execution plan for any SQL statement. And from these execution plans we can see the problem of the command and why it takes more time that it should take. A poor command can be discovered by monitoring some facts such as, the command which makes a lot of disk access or makes full table scan to read a small number of data. Another fact is the execution plan which shows that the SQL command does not use indexes related to selected table.

المنارة للاستشارات
www.manaraa.com

1.6.1 Overview

The optimizer of any RDBMS will take the decision of how the command will be executed depending on statistics gathered for the target tables and related indexes. So if the SQL command received by the optimizer has poor conditions, the optimizer will generate poor execution plan. From this fact SQL command has to fit some rules like [21]:

- Driving table has the best filter.
- Fewer numbers of rows are being returned to the next step.
- The join method is appropriate for number of rows being returned.
- Views are used efficiently.
- There are no unintentional cartesian products.
- Each table is being accessed efficiently.
- Predicates in the SQL statement and the number of rows in the table.
- A full table scan does not mean inefficiency.

Data can be fetched from database in several ways. The slower way is reading the data sequentially from table (full table scan). But the faster way is reading the data using ordered index, because the index contains the required column in order and the physical location of the record for that column (pointer), so a binary search on the specified column can be done using this index to get the physical location of the record (pointer), then fetching the data using this pointer. Most of RDBMSs provide facilities to see the usage of indexes and the steps of executing these indexes.

المنارة للاستشارات
www.manaraa.com

The only solution that most of RDBMSs provide for programmers and database administrators to change the execution plan for the SQL command is rewriting the command again in efficient way. And they put some guidelines to do that, some RDBMSs provide more facilities to store the execution plans and statistics for any command to be exported for another database. Some guidelines are mentioned below [21]:

- Compose predicates using AND and =.
- Avoid transformed columns in the WHERE clause.
- Avoid mixed-mode expressions and beware of implicit type conversions.
- Write separate SQL statement for specific tasks.
- Use EXISTS rather than IN for subqueries.
- Control the access path and join order with hints.
- Remove nonselective indexes to speed the DML.
- Index performance-critical access path.
- Reorder columns in existing concatenated indexes.
- Add columns to the index to improve selectivity.
- Consider index-organized tables.

1.6.2 Sequential Search versus Binary Search

Sequential search is a search algorithm, also known as linear search. It operates by checking every element one at a time in sequence in the list and compares it with required value. Linear search runs in O(n) which means that the worst case will need number of comparisons as the number of elements in the set, in database system most of the cases are the worst cases because

المنارة للاستشارات
www.manaraa.com

we may have the same data inserted more than once like names, salaries, jobs and so on, so if we find the required data we have to continue to the end to find the other value. Sequential search in large amount of data may have a long time, we are not talking about couple of seconds; search may take minutes and some times hours [31].

Binary search is a technique to locate a value in an ordered list. The method makes progressively better guesses, and closes in on the location of the required value by selecting the middle element in the list (because the list is in sorted order, is the center value), comparing its value to the target value, and determining if it is greater than, less than, or equal to the target value. A guessed index whose value turns out to be too high becomes the new upper bound of the list, and if its value is too low that index becomes the new lower bound. Only the sign of the difference is inspected. Pursuing this strategy iteratively, the method reduces the search list by a factor of two each time, and soon finds the target value or else determines that it is not in the list at all. A binary search is an example of divide and conquer search algorithm. Binary search runs in O(logN) and for sure it is faster than sequential search [32].

An index is a database object that is logically and physically independent of the table data. Any RDBMS may use an index to access data that is required by a SQL statement, or it may use indexes to enforce integrity constraints. You can create and drop indexes at any time after creating the table object. Indexes can be unique or non unique. Unique indexes guarantee that no two index

المنارة للاستشارات
www.manaraa.com

entries have the same value. But duplicate can exist in a non unique index. A composite index (also called a concatenated index) is an index that is created on multiple columns in a single table. Columns in a composite index which can appear in any order and need not be adjacent in the table. For standard indexes, RDBMS uses B*-tree indexes that are balanced to equalize access times. B*-tree indexes which normally make the SQL command to be executed use binary search instead of using sequential table search, and this is the reason for creating indexes in database applications.

1.7 Extensible Markup Language

XML is the abbreviation for Extensible Markup Language. XML documents look similar to Hypertext Markup Language (HTML) documents, although they are very different. HTML is a markup language, primarily used for formatting and displaying text and images in a browser. XML is a markup language for structuring data rather than formatting information.

We use XML to create a document that contains structured data that can be used or interpreted by other applications. The format or structure is straightforward and can be used by any person or program that can read text. Unlike HTML, the tags in XML are extensible, and so we can create our own tags as we need them. HTML has a set of predefined formatting tags that we can use, but we cannot create our own. XML is a markup language that provides a universal format for structured documents and data on the web and it is part of the World Wide Web Consortium (W3C) standards [9].

المنارة للاستشارات
www.manaraa.com

1.7.1 XML Elements

The example illustrated in Figure (1-3) is a sample of XML document that uses nested elements to describe the database dictionary. XML file must start with a line (<?xml version="1.0" ?>) as an XML declaration line and the document file is recognized as XML file. Elements are identified by tag names, such as **DBdictionary**, **tables**, **table**, **column** and so on. Tag names are distinguishable as markup, rather than data, because they are surrounded by angle brackets (< and >) and they are case-sensitive. In XML, an element includes the start tag (<**DBdictionary**>), end tag (</**DBdictionary**>), called root element and all the markups and character data contained between the tags inside the root element tag.

المنارة للاستشارات
www.manaraa.com

```
<?xml version="1.0" ?>
- <DBdictionary>
  - <tables>
    + <table name="CLIENTS">
    + <table name="CONFERENCES">
    + <table name="CONFERENCE_DETAILS">
    + <table name="COUNTRIES">
    - <table name="DEPT">
      - <column id="0">
          <name>DEPTNO</name>
          <type>NUMBER</type>
          <length>2</length>
          <scale>0</scale>
        </column>
      - <column id="1">
          <name>DNAME</name>
          <type>VARCHAR2</type>
          <length>14</length>
          <scale>null</scale>
        </column>
      - <column id="2">
          <name>LOC</name>
          <type>VARCHAR2</type>
          <length>13</length>
          <scale>null</scale>
        </column>
      </table>
    + <table name="EMP">
    </tables>
  </DBdictionary>
```

Figure (1-3) XML Document

1.7.2 XML Attributes

Attributes are simple name-value pairs that are associated with a particular element. XML attributes must be specified after the start tag of an element. As an example from pervious document, the **name** attribute in the **table** tag (<table **name**="DEPT" >). Attribute names are case-sensitive and follow the naming rules that apply to

المنارة للاستشارات
www.manaraa.com

element names. In general, spaces are not used, but are allowed on either side of the equal sign. The attribute values must always be in matching quotes, either single or double quotes. Attributes provide additional information about the XML document's content or other XML elements. Attributes can be used to describe how the XML document data is encoded or represented, indicate where the links or external resources are located, specifying an element instance in the document for facilitating a rapid search.

1.7.3 XML and HTML

The key difference between XML and Hyper Text Markup Language (HTML) is that XML is a markup language for describing data but HTML is a markup language for formatting data, XML contains user defined markup tags but HTML contains predefined markup tags, XML can be displayed as a document tree in the web browser but HTML will be formatted in the web browser, XML is extensible language and we can add elements as we need but HTML is not extensible, XML confirms to rules for will-formed document but HTML do not confirm to rules of will-formed document.

The World Wide Web Consortium (W3C) has worked on defining the Extensible HyperText Markup Language (XHTML) as a successor to HTML. XHTML is designed to conform to XML standards and well-formed document rules, and provide a way to reproduce, subset, and extend HTML documents. An XHTML document is a particular XML document instance intended for processing by a Web browser. Not all browsers support XHTML document and different browsers often process XHTML documents in different ways [9].

المنارة للاستشارات
www.manaraa.com

1.8 The Problem Statement

The researcher's aim is to execute SQL commands faster than original commands; this can be achieved by implementing many things like: rewriting the SQL command again with new structure to use existing database object efficiently, using mobile agent that navigates through existing servers to minimize network overheads and roundtrips, minimize the incorrect SQL commands written by naïve users by automatically correct them before reaching the RDBMS.

As we know any database application has two parts as shown in Figure (1-4):

1. The first part (Application) which contains code and application logic, in this part programmers write the SQL commands.
2. The second part (RDBMS) is the physical structure of the application which contains tables, relations, indexes, views and other objects. In fact this part will receive the SQL commands from the first part and then executes these commands to return results back.

المنارة للاستشارات
www.manaraa.com

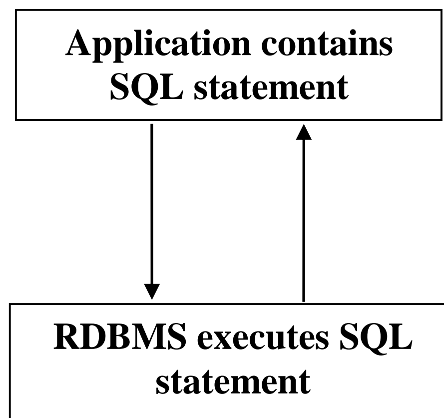


Figure (1-4) Parts of database applications

The application may work slow and may have to be revised for the following reasons:

1- Some programmers are novice so they write SQL commands inefficiently and incorrectly, the commands may contain mistakes, syntax errors, also SQL commands may need more time than what it should take, and need a lot of I/O roundtrip.
2- Expert programmers may write a very complex SQL commands but their concern is to get the correct results and they do not care or give any attention to performance and efficiency.
3- Some SQL commands may work efficiently in development phase, but when we move to production, they work very inefficiently.
4- Number of users working on the application and the amount of data entered and retrieved from the application may make some SQL commands work inefficiently.
5- Queries may be generated programmatically.

المنارة للاستشارات
www.manaraa.com

These facts show us that most of programmers and database users can write SQL command in a correct syntax but they do not have the required experience and knowledge to know how the RDBMS optimizer will choose the execution plan to execute the command. So if SQL command written in a professional way, this may help the optimizer to choose to best execution path for this command.

1.9 Thesis Contribution

Most of work done is to enhance the performance of SQL commands in the optimizer level. This means that the RDBMS will receive the SQL command and let the database optimizer to discover the best execution plan for this statement. But in most cases we can simplify the optimizer work by sending good written SQL commands, so the optimizer can work efficiently and will take the best rule for executing the commands.

In this thesis, we describe the solution for each reason that makes the SQL command slow (as mentioned in pervious section), design an agent to receive poor commands and translates them to efficient commands, implement the agent in real word environment, and test the results by comparing the old commands with new translated commands. All of this work done without opening the application source code and modifying it, the agent will do all of the work transparently without affecting the running application.

The agent mobility will help the existing system to run faster because when agent moves from machine to machine holding its information about existing system (system metadata) will reduce

المنارة للاستشارات
www.manaraa.com

network messages and roundtrips between servers. Correcting syntax errors of the SQL commands written by naïve users on the fly (before reaches the RDBMS) will reduce the communication between machines. Rewriting the SQL command in efficient way then sending it to RDBMS also will enhance the performance of the existing system. RDBMS always receives efficient, good written, error free SQL commands will not suffer from contentions and bottlenecks.

1.10 Thesis Organization

In this thesis the researcher focuses on each of the above problems, first describing each problem against the background of previous research. He then proposes solutions to the above problems. The solutions emphasize increasing performance and decreasing access time. In addition to introductory chapter, there are five chapters in this thesis.

Chapter 2 will present some researches in the area of query rewriting with different types of methodologies. Some researches focused on rewriting SQL command using materialized views technique, others focused on enhancing database optimizer capabilities, and other techniques used in the area of database optimization. Also this chapter focuses on the researches done in mobile agent field. The last sections of the chapter describe the research problem in details and how it still exists in spite of all researches done in this area.

المنارة للاستشارات
www.manaraa.com

Chapter 3 proposes the system that will solve the research problem. Also this chapter will discuss the details of the system step by step and component by component. The last section of this chapter will present the test database and schema that the research will use as a test case to do the measurements.

Chapter 4 will implement the research problem and will do the experiments needed to test the performance of the agent and how the agent solves the research problem. This chapter will present the metrics and statistics needed to be measured and discussed to make performance analysis. In this chapter an implementation of each system building block will be discussed in details including flowcharts and pseudo code for each system component. Also a screen shots will be taken to show the agent performance.

Chapter 5 will analyze the agent performance, present comparisons between old and new statements done to see the achievement of the agent. Different scenarios will be discussed and implemented to see different behaviors of the agent. Also measurement benchmarks will be presented for all types of metrics and statistics. This chapter will include result comparisons between old and new statements done by the agent.

Chapter 6 includes the conclusion and future work. This chapter presents the conclusion extracted from each bit and piece discussed in this research. Also it puts the recommendations for future work that will be focused on the area of SQL tuning.

المنارة للاستشارات
www.manaraa.com

Chapter two
Background and related work

2.1 Overview

Complex queries and long running statements have gained plenty of research attention regarding the application is On-Line Transaction Processing (OLTP), On-Line Analytical Processing (OLAP), or Decision Support System (DSS) due to emphasis of increasing query throughput and decreasing response time. From a point of view, much of the researches focus on minimizing query time and provide feedback more quickly by using a physical data structure created on the database called materialized view, and then rewrite the SQL command to use the materialized views instead of original tables.

Section 2.2 will present some of researches in the area of query rewriting. This section is divided according to the area that researchers focused on. Section 2.2.1 focused on rewriting using materialized views, section 2.2.2 describes research efforts in XML area. Another approach focuses on speeding up the query and gain more time by making the database optimizer more intelligent and efficient to decide what is the best execution plan for variety of SQL commands, section 2.2.3 will briefly present some researches in this area. Section 2.2.4 will present some researches about agents, mobile agents and how to communicate with databases. Some approaches focused on distributed queries and expecting future queries to enhance search results, section 2.2.5 will present some researches in this area. In section 2.3 we will describe the thesis problem in details and what is achieved before, what is left to do?

المنارة للاستشارات

www.manaraa.com

2.2 Literature Survey

A lot of studies and researches presented and focused in query rewriting using materialized views, XML, enhancing optimizer capabilities, and so on. But few of them focus on the core of this research which is rewriting the original SQL command. The following sections will describe related efforts in this area.

2.2.1 Query Rewrite Using Materialized Views

In recent years the need for large and long running queries increased dramatically especially when On-Line Analytical Processing (OLAP) and Decision Support System (DSS) became very common and appeared in commercial market. In order to overcome this problem, the technique of using Materialized View (MV) becomes popular. Materialized view is a physical structure holds data and reserves space like a table, but it can be created from joining more than one table, nested select, grouping select, or any type of select statement. The main difference between table and materialized view is the way of populating data into this object, normally we insert data into table but materialized view gets its data from another table. So by creating materialized views we can execute queries faster because our query well get data from materialized views instead of joining or grouping data from original tables. So we eliminate joining, grouping, or any calculation time by using materialized view because data already prepared inside it.

A materialized view stores both the definition of a view and the rows resulting from the execution of the view. Like a view, it uses a query as the basis. However, the query is executed at the time the view is created, and the results are stored in a table

المنارة للاستشارات
www.manaraa.com

(materialized view). You can define the materialized view with the same storage parameters that any other table has. You can also index the materialized view table in the same way that you index other tables to improve the performance of queries executed against them. When a query can be satisfied with data in a materialized view, the RDBMS optimizer transforms the query to reference the view rather than the base tables. By using a materialized view, expensive operations such as joins and aggregations do not need to be executed.

A lot of researches have been done in this area. Chang-Sup, Myong and Yoon [6] proposed a new method for rewriting a given OLAP query using various kinds of materialized aggregate views which already exist in data warehouses. They defined the normal forms of OLAP queries and materialized views based on the lattice of dimension hierarchies, the Symantec information in data warehouses. John [14] introduced the aggregating concept and the query rewriting using materialized views on Oracle Database. He introduced the Oracle requirements to use the query rewrite option in Oracle 9.2. Oracle [20] introduced the data warehouse concepts, objects and encouraged optimization on star queries. She described the creating and maintaining materialized views and dimensions to enhance ad-hoc query performance. She introduced the summary advisor tool for data warehouse design recommendations. Priya Vennapusa [21] introduced a tuning methodology and tuning roles, he described the SQL statement processing phases in Oracle Database and then he introduced the Oracle SQL analyzer and the deferent types of optimizers and how the optimizer rewrites the SQL using materialized views.

المنارة للاستشارات

www.manaraa.com

A.Y. [1] surveyed the state of the art on the problem of answering queries using views, and synthesized the disparate works into a coherent framework. He described the different applications of the problem, the algorithms proposed to solve it and the relevant theoretical results. The problem has recently received significant attention because of its relevance to a wide variety of data management problems. In query optimization, finding a rewriting of a query using a set of materialized views can yield a more efficient query execution plan. To support the separation of the logical and physical views of data, a storage schema can be described using views over the logical schema. Finally, the problem arises in data integration systems, where data sources can be described as precomputed views over a mediated schema.

Shukla [3] introduced the requirement of fast interactive multidimensional data analysis, database systems precompute aggregate views on some subsets of dimensions and their corresponding hierarchies. However, the problem of what to precompute is difficult and intriguing. The leading existing algorithm, BPUS, has a running time that is polynomial in the number of views and is guaranteed to be within (0.63 - f) of optimal, where f is the fraction of available space consumed by the largest aggregate. Unfortunately, BPUS can be impractically slow, and in some instances may miss good solutions due to the coarse granularity at which it makes its decisions of what to precompute. They studded the structure of the precomputation problem and showed that under certain broad conditions on the multidimensional data, an even simpler and faster algorithm.

المنارة للاستشارات

www.manaraa.com

Chirkova [22] focused on the problem of automatically reformulating a database in a way that reduces query processing time while satisfying strong storage space constraints. In previous work they have investigated database reformulation for the case of unary databases. In this paper they extended this work to arbitrary arty, while concentrating on databases with conjunctive rules. The main result of the paper is that the database reformulation problem is decidable for conjunctive databases.

2.2.2 Query Rewrite and XML

Internet database applications are designed to interact with users through web pages and the common method to build web page is through using hyperlink documents. The most popular language to build hyperlink documents is HTML (Hypertext Markup Language), but HTML is not suitable for structuring data extracted form database. New language named XML (Extended Markup Language) rises as a standard language for structuring and exchanging data over the web. XML is used to provide information about the structure and the data in the web pages rather than specifying just the formatting of the data.

XML is one of the most extensively used data representation and data exchange formats. Much of the researches on XML have focused on developing efficient mechanisms to store, query and manage XML data either as a part of a relational database or using native XML stores. However, hiding sensitive data is as important as making the data efficiently available, as has been emphasized and studied for decades in relational databases.

المنارة للاستشارات
www.manaraa.com

There are many researches that focus on this area. Maxim [18] introduced the techniques of rewriting a XML query into equivalent one that can be executed faster. He devoted a comprehensive discussion on XQuery rewriting in the presence of data schema. Muralidhar [19] presented the XML Query Rewrite technique used in Oracle XML DB. This technique integrates querying XML using XPath embedded inside SQL operators and SQL/XML publishing functions with object relational and relational algebra. He used a common set of algebraic rules to reduce both XML and object queries into their relational equivalent.

Sriram [26] adopted and extended a graph editing language for specifying role-based access constraints in the form of security views. He proposed a security annotated schema (SAS) as internal representation for security views. He proposed secure query rewrite (SQR) as a set of roles that can be used to rewrite the XPath to equivalent XQuery expression against the original data. Deutsch [2] stated and solved the query reformulation problem for XML publishing in a general setting that allows mixed (XML and relational) storage for the proprietary data and exploits redundancies (materialized views, indexes and caches) to enhance performance. The correspondence between published and proprietary schemas is specified by views in both directions, and the same algorithm performs rewriting-with-views, composition-with-views, or the combined effect of both, unifying the Global-As-View and Local-As-View approaches to data integration.

المنارة للاستشارات
www.manaraa.com

2.2.3 Enhancing Optimizer Capabilities

A powerful feature of relational query languages is that identities of relational algebra may be used to transform query expressions to enhance efficiency of evaluation. Some transformations are always valid, but whether they enhance or degrade efficiency depends upon characteristics of the data. Other transformations are such that their very validity depends on characteristics of the instance. So the database optimizer has to take a very important decision to select what is the best execution plan for any SQL command, sometimes the decision may be difficult or not the best choice.

The optimizer is the part of the RDBMS that creates the execution plan for a SQL statement. An execution plan is a series of operations that are performed to execute the statement. The optimizer uses various pieces of information to determine the best path such as hints supplied by the developer, statistics, information in the dictionary, WHERE clause of the SQL statement. The optimizer usually works in the background, however, with diagnostic tools provided by the RDBMS vender we can see the decisions that the optimizer makes.

The optimizer determines the least-cost plan (most efficient way) to execute a SQL statement after considering many factors related to the objects referenced and the conditions specified in the query. This determination is an important step in the processing of any SQL statement and can greatly affect execution time. Because of that a lot of researches have been done to cover this area.

المنارة للاستشارات
www.manaraa.com

Bryan [5] presented a thorough analysis of research into semantic query optimization (SQO). He identified three problems which inhibit the effective use of SQO in relational database management systems (RDBMS). He then proposed solutions to these problems and described first steps towards the implementation of an effective SQO for relational databases. Chris [7] approach governs whether the optimization yield more efficient query processing. He used approximate functional dependencies as the conceptual basis for this decomposition and develops query rewriting techniques to exploit it. He presented experimental results leading to a well-defined class of queries which improve processing time. Elmasri and Navathe [24] introduced algorithms form query processing and optimization and how the SQL represented as query tree also as a query graph, then the query optimizer will select the best execution plan. Then they discussed how to improve database performance through database tuning.

2.2.4 Mobile Agent

It is simply wrong to assume that mobile agents generally improve performance. Sometimes they do, sometimes they do not. However, the major benefits are modularity, mobility and reuse. Mobile agents introduce an innovative approach to designing distributed systems. They allow creating mobile components of software programs which act autonomously. This enables decentralization, load-balancing, self-organization and many other new concepts. A huge effort was made in this field.

Danny [8] described the benefits of mobile agents and considered it as a promising paradigm for the design and implementation of distributed applications. Because of mobile agent

المنارة للاستشارات
www.manaraa.com

nature, MA can do the computations transparently from the user. And this feature can reduce network load, latency, and failed operations. Rahul [23] project contributes towards an evaluation and implements of an e-commerce application using mobile agents. The project quantitatively evaluates various implementation strategies and identifies various application parameters that influence application performance. The project also provides qualitative and quantitative comparison across three Java based mobile agent framework. Voyager, Aglets, Concordia, for e-commerce applications.

William [29] discussed achievable security goals for mobile agents, and he proposed architecture to achieve these goals. The architecture models the trust relations between the principals of mobile agent systems. A unique aspect of the architecture is a "state appraisal" mechanism that protects users and hosts from attacks via state modifications and that provides users with flexible control over the authority of their agents. Baumann [12] described the basic concepts of a mobile agent system, i.e., mobility, communication and security, and he discussed different implementation techniques, presented the decisions made in Mole and gave an overview of the system services implemented in Mole. Mole is the first mobile agent system that has been developed in the Java language. The first version was finished in 1995, and since then Mole has been constantly improved. Mole provides a stable environment for the development and usage of mobile agents in the area of distributed applications.

المنارة للاستشارات
www.manaraa.com

Jiao [28] applied the Mobile Agent technology in a Mobile Data Access System framework (MAMDAS) using the Summary Schemas Model as the underlying multi database platform. This approach provides better performance by reducing the network traffic and higher degree of autonomy by allowing agents to execute without the owners interference. As witnessed by their experimental results, the MAMDAS exhibits the following advantages compared to the first SSM prototype: it is about 6 times faster, it supports larger number of concurrent queries, and it demonstrates greater scalability, portability, and robustness.

Brewington [4] discussed the strengths of mobile agents and argued that although none of these strengths are unique to mobile agents, no competing technique shares all of them. Next he examined one specific information retrieval application searching distributed collections of technical reports and considers how mobile agents can be used to implement this application efficiently and easily. He described two planning services that allow mobile agents to deal with dynamic network environments and information resources.

Moizumi [16] studied several planning problems that arise in mobile agent information retrieval and data-mining applications. The general description of the planning problems is as follows: We are given sites at which a certain task might be successfully performed. Each site has an independent probability of success associated with it. Visiting a site and trying the task there requires time (or some other cost matrix) regardless of whether the task is completed

المنارة للاستشارات
www.manaraa.com

successfully or not. Latencies between sites, that is, the travel time between those two sites also have to be taken into account. If the task is successfully completed at a site then the remaining sites need not be visited. The planning problems involve finding the best sequence of sites to be visited, which minimizes the expected time to complete the task. He named the problems Traveling Agent Problems due to their analogy with the Traveling Salesman Problem. This Traveling Agent Problem is NP complete in the general formulation. A polynomial-time algorithm has been successfully developed to solve the problem by adding a realistic assumption to it. The assumption enforces the fact that the network consists of subnetworks where latencies between machines in the same subnetwork are constant while latencies between machines located in different subnetworks vary.

Manwade [15] introduced a methodology for the creation of parallel applications on the network. The proposed mobile-Agent parallel processing framework uses multiple Java mobile Agents. Each mobile agent can travel to the specified machine in the network to perform its tasks. He also introduced the concept of master agent, which is Java object capable of implementing a particular task of the target application. Master agent dynamically assigns the task to mobile agents. A prototype application has been developed and tested.

2.2.5 Miscellaneous

Some researchers focus on distributed queries and expect future queries to enhance search results. Distributed queries fetch data from multiple databases at the same time by using database

المنارة للاستشارات
www.manaraa.com

links to link databases with each other. Others tried to enhance the execution of query by expecting the pattern of SQL command submitted by the application. A lot of researches focused on this field.

Graham, Peter and Gray [10] addressed the problem of processing complex queries including quantifiers, which have to be executed against different databases in an expanding heterogeneous federation. They transformed queries within wrappers to make best use of the query processing capabilities of external databases. Their approach was based on pattern matching and query rewriting. Ivan [11] noticed that streams of relational queries submitted by application to database contain patterns that can be used to predict future requests. He presented scalpel system to detect these patterns and optimized request streams using context-based predictions of future requests. Rosie [25] introduced the notion of query substitution by generating a new query to replace a user's original query for web searchers. He defined a scale for evaluating query substitution and showed that their method performs well at generating new queries related to the original queries.

Popa [17] introduced the traditional query optimizers that assume a direct mapping from the logical entities modeling the data (e.g. relations) and the physical entities storing the data (e.g. indexes), each physical entity corresponding precisely to one logical entity. This assumption is no longer true in non-traditional applications (object-oriented and semi-structured databases, data integration), which often exhibit a mismatch between the logical

المنارة للاستشارات
www.manaraa.com

view and the actual storage of data. In addition, there is an increased amount of redundancy, even at the logical level, that can greatly enhance optimization opportunities, if exploited. To deal with all this, he proposed a novel architecture for query optimization, in which physical optimization is leveraged at the level of query rewriting. As a consequence, the other important aspect of query optimization, semantic optimization (that takes advantage of the redundancy at the logical level), can be naturally incorporated. The optimizer can then make global decisions based on both semantic and physical knowledge, leading to plans of higher quality than those obtainable by a traditional two-level approach.

2.3 Detailed Problem Statement

Reading previous sections gives us and indicator of how much work done in the area of SQL tuning. Some researchers focused on building physical structure to enhance the query time as materialized views, others focused on enhancing database optimizer capabilities. But instead of sending a poor SQL statement to database and let the RDBMS to do the entire job of parsing, deciding what is the best execution plan for this command, executing the command, fetching the data. Of course there are a lot of advantages of sending a good structured and tuned SQL statement to database, so the main idea is writing a tuned, error free command and send it to RDBMS. Our effort will focus on receiving any SQL command and rewriting this command to new structure then sending the new command to RDBMS. The new command has to be written in a way to minimize the work of database, minimize database optimizer effort, and minimize the network roundtrips.

المنارة للاستشارات
www.manaraa.com

One good example in our case, is the method of fetching data from database, as we all know any database can get data from a table in two ways, first way is sequential search (full table scan) which means that the RDBMS will read the table record by record until finding the required record, of course this method will take a lot of time and I/O can take hours. The second way is binary search which means that the RDMBS will split the table into two parts and see the value fits in which part, then split the specified part into two parts and see the value fits in which part, and so on, until it reaches the required value. It is very obvious that the second way (binary search) will take a very shorter time than the first one (sequential search).

So, we can write two statements to get the same results but the database optimizer will decide to use sequential search for the first statement and binary search for the second statement, although the two commands will get the same result. The following example will explain this idea.

Supposing that the following table 'EMPLOYEES' will be used as in Figure (2-1), also there is an index created on the column 'SALARY' to make a binary search on this column instead of sequential search.

المنارة للاستشارات
www.manaraa.com

Name	Null?	Type
SSN	NOT NULL	VARCHAR2(20)
FNAME		VARCHAR2(20)
MINIT		VARCHAR2(20)
LNAME		VARCHAR2(20)
BDATE		DATE
ADDRESS		VARCHAR2(100)
SEX		VARCHAR2(1)
SALARY		NUMBER(10,3)
SUPERSSN		VARCHAR2(20)
DNO		NUMBER(3)

Figure (2-1) EMPLOYEES table Structure

The test will use same SQL command with different predicate (where clause), but the two SQL commands will get the same result.

1- The first predicate will be (WHERE SALARY*12 = 60012)

SSN	FNAME	MINIT	LNAME	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
4901	fname4901	minit4901	lname4901	20/01/08	amman jordan	m	5001	4901	10

Elapsed: 00:00:03.00

Execution Plan

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2639 Card=131 Byte s=19519)
1	0	TABLE ACCESS (FULL) OF 'EMPLOYEES' (TABLE) (Co st=2639 Card=131 Bytes=19519)

المنارة للاستشارات
www.manaraa.com

Statistics

0	recursive calls
0	db block gets
11893	consistent gets
10767	physical reads
0	redo size
952	bytes sent via SQL*Net to client
512	bytes received via SQL*Net from client
2	SQL*Net roundtrips to/from client
0	sorts (memory)
0	sorts (disk)

1 rows processed

Figure (2-2) Results of first SQL command

The results show that we have employee with SSN 4901 which matches the predicate, the time consumed for this query is 3 seconds as shown in Figure (2-2). The interesting thing in execution plan is that the RDBMS optimizer dose not decide to use the SALARY index and makes full table scan, so a sequential table search is used. Statistics says that we have 10767 physical reads, which is the amount of data that the RDBMS processed and used from hard disk.

2- The second predicate will be (WHERE SALARY = 60012/12)

المنارة للاستشارات
www.manaraa.com

SSN	FNAME	MINIT	LNAME	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
4901	fname4901	minit4901	lname4901	20/01/08	amman jordan	m	5001	4901	10

Elapsed: 00:00:00.00

Execution Plan

0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=4 Card=1 Bytes=149)
1	0	TABLE ACCESS (BY INDEX ROWID) OF 'EMPLOYEES' (TABLE) (Cost=4 Card=1 Bytes=149)
2	1	INDEX (RANGE SCAN) OF 'EMP_SALAR Y_INDEX' (INDEX) (Cost=3 Card=1)

Statistics

0	recursive calls
0	db block gets
5	consistent gets
0	physical reads
0	redo size
956	bytes sent via SQL*Net to client
512	bytes received via SQL*Net from client
2	SQL*Net roundtrips to/from client
0	sorts (memory)
0	sorts (disk)

1 rows processed

Figure (2-3) Results of second SQL command

The results show that we have employee with SSN 4901 which is the same result of first command, the time consumed for this query is 0 second (less than 0.0001 second) as shown in Figure (2-3). Execution plan shows that the RDBMS optimizer used the SALARY index to get the data using a binary search method. Statistics say that we have 0 physical reads so the amount of data processed is less than one block. Comparing the two results ensures that there is a performance problem that can be avoided if the SQL command is written in proper way.

المنارة للاستشارات
www.manaraa.com

The previous example shows that the same data can be populated from database with different ways of writing SQL statement especially the predicate (WHERE clause) of the statement. Also the predicate can be used in any SQL statement like DML commands (INSERT, UPDATE, DELETE), so the enhancement will reach all types of SQL statements submitted from application to RDMBS to handle. Rewriting the SQL statements to use the capabilities of the database and to use the indexes built in the database efficiently for sure will enhance the overall performance of the application.

2.4 Problem Still Exists

From the previous sections we can notice that most researches focused on enhancing the performance of SQL commands without affecting the command itself, they accepted the SQL command as it is with its weakness and poor writing, then started to enhance the performance of executing these commands. The core of this thesis is different, in this thesis we will dig the SQL command and its structure to see the area of weakness, then rewrite it again eliminating the weakness areas before sending it to RDBMS to execute it.

المنارة للاستشارات
www.manaraa.com

Chapter Three
Proposed System Design

3.1 Overview

The goal of this thesis is to control the impact that the execution of large queries has on the performance of database. The proposed approach is to find a better execution plan for SQL statement before accessing the database, and if there is another statement that gives better execution plan than the previous one which will be replaced by the new one. As we know each database system has a built in feature called optimizer, the purpose of the optimizer is selecting the best execution plan for any statement. The proposed approach focuses on breaking up the SQL statement into pieces and concentrates on the WHERE clause to see if we can rewrite the WHERE clause in a different way with better execution plan before sending the original statement to database optimizer.

3.2 Optimizer

The optimizer is a part of any database system that creates the execution plan for a SQL statement. An execution plan is a series of operations that are performed to execute the statement. The optimizer uses various pieces of information to determine the best path:

- Hints supplied by the developer.
- Statistics.
- Information in the dictionary.
- WHERE clause.

The optimizer usually works in the background. However, with diagnostic tools provided by database vender, you can see the decisions that the optimizer makes.

المنارة للاستشارات
www.manaraa.com

The optimizer determines the least-cost plan (most efficient way) to execute a SQL statement after considering many factors related to the objects referred to and the conditions specified in the query. This determination is an important step in the processing of any SQL statement and can greatly affect execution time. The optimizer may not make the same decisions from one database to another or from one version of the same database to another version. In recent versions, the optimizer may make different decisions because better information is available [21].

Optimizer Operations: For any SQL statement processed by the database server, the optimizer performs the following operations as shown in Figure (3-1) [21]:

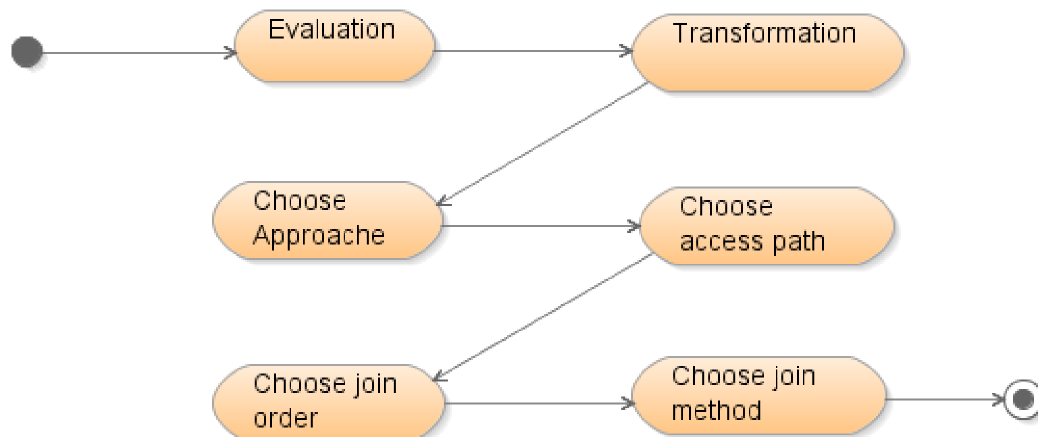


Figure (3-1) Optimizer Operations

- **Evaluation of expressions and conditions:** The optimizer first evaluates expressions and conditions containing constants as fully as possible.

المنارة للاستشارات
www.manaraa.com

- **Statement transformation:** For complex statements involving, for example, correlated subqueries or views, the optimizer might transform the original statement into an equivalent join statement.
- **Choice of optimizer approaches:** The optimizer determines the goal of the optimization.
- **Choice of access paths:** For each table accessed by the statement, the optimizer chooses one or more of the available access paths to obtain table data.
- **Choice of join orders:** For a join statement that joins more than two tables, the optimizer chooses which pair of tables is joined first, then which table is joined to the result, and so on.
- **Choice of join methods:** For any join statement, the optimizer chooses an operation to use to perform the join.

To choose an execution plan for a join statement, the optimizer must make these interrelated decisions:

- **Access paths:** For simple statements, the optimizer must choose an access path to retrieve data from each table in the join statement. Access paths are: full table scans, rowid scans, index scans, cluster scans, hash scans
- **Join method:** To join each pair of row sources, database optimizer must perform a join operation. Join methods include nested loop, sort merge, cartesian, and hash joins.
- **Join order:** To execute a statement that joins more than

المنارة للاستشارات
www.manaraa.com

- two tables, database optimizer joins two of the tables and then joins the resulting row source to the next table. This process is continued until all tables are joined into the result.

3.3 SQL Statement Processing Phases

A good understanding of SQL processing is essential for writing optimal SQL statements. In SQL statement processing, there are four important phases: parsing, binding, executing, and fetching. As shown in Figure (3-2) [21]:

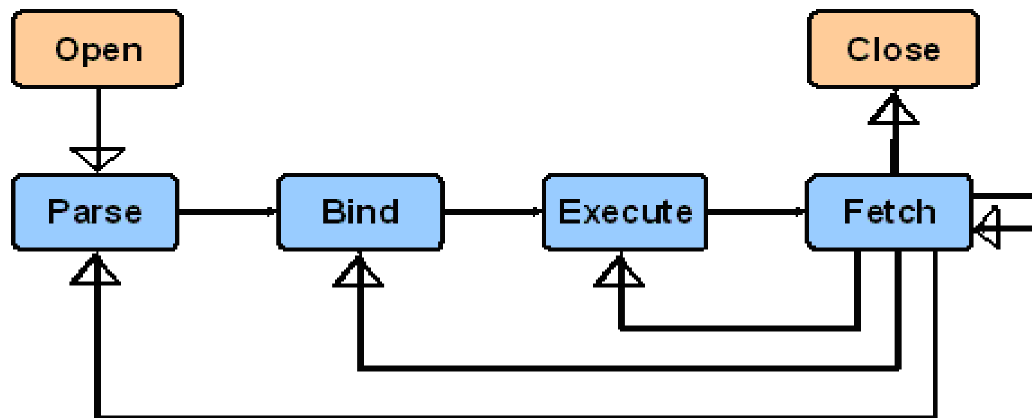


Figure (3-2) SQL Statement Processing Phases

The reverse arrows indicate processing scenarios (for example, Fetch—(Re)Bind—Execute—Fetch). The Fetch phase applies only to queries and DML statements with a returning clause.

المنارة للاستشارات
www.manaraa.com

3.3.1 Parse Phase

Parsing is the first stage in the processing of a SQL statement. When an application issues a SQL statement, the application makes a parse call to the RDBMS. During the parse call, the RDBMS:

1- Checks the statement for syntactic and semantic validity.
2- Determines whether the statement has privileges to run it.
3- Allocates a private SQL area in memory for the statement.

The RDBMS first checks whether there is an existing parsed representation of the statement in the cache. If so, it uses this parsed representation and runs the statement immediately. If not, the RDBMS generates the parsed representation of the statement, and allocates a shared SQL area for the statement in the cache and stores its parsed representation there. A parse operation allocates a shared SQL area for a SQL statement. After a shared SQL area has been allocated for a statement, it can be run repeatedly without being reparsed. Both parse calls and parsing can be expensive relative to execution, so they should be minimized. Ideally, a statement should be parsed once and executed many times rather than reparsing for each execution.

3.3.2 Bind Phase

During the bind phase, The RDBMS checks the statement for references of bind variables, then assigns or reassigns a value to each variable. When bind variables are used in a statement, the optimizer assumes that SQL sharing is intended and that different invocations should use the same execution plan. If different

المنارة للاستشارات
www.manaraa.com

invocations of the SQL would significantly benefit from different execution plans, then using bind variables may adversely affect the performance of the SQL statement.

3.3.3 Execute Phase

The RDBMS uses the execution plan to identify the required rows of data from the data buffers. Multiple users can share the same execution plan. The RDBMS performs physical reads or logical reads/writes for DML statements and also sorts the data when needed. Physical reads are disk reads; but logical reads are blocks already in memory of the server. Physical reads are more expensive because they require I/O from disk.

3.3.4 Fetch Phase

The RDBMS retrieves rows for a SELECT statement during the fetch phase. Each fetch typically retrieves multiple rows, using an array fetch because there is minimum number of records to be selected in each fetch and it is equal to RDBMS block size, this may enhance the performance because the same data may be selected next time by the same user or different users.

3.3.5 Query Execution Plan

The output of a query optimizer for a declarative query statement is called a Query Execution Plan (QEP). The structure of a QEP determines the order of operations for query execution. The QEP is typically represented using a tree structure where each node represents a physical database operator (e.g. nested loop join, table scan etc). Multiple plans may exist for the same query and it is a query optimizer's top priority to choose an optimal plan. To supplement the QEP, most query optimizers produce performance related information such as cost information, predicates, selectivity

المنارة للاستشارات
www.manaraa.com

estimates for each predicate and statistics for all objects referenced in the query statement. Figure (3-3) shows an example of nested loop join statement.

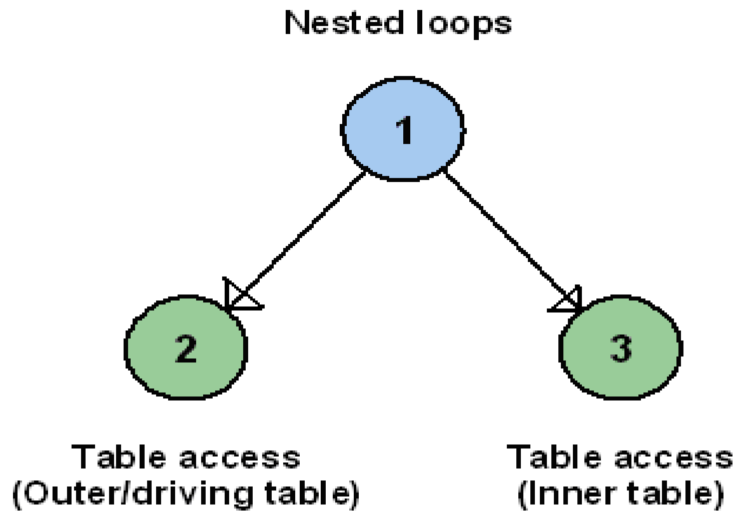


Figure (3-3) Nested loops join

In nested loop joins one of the tables defined as outer table (driving table), the other table called inner table, and for each row in the outer table all matching rows in the inner table are retrieved. Another way to join two tables is a sort-merge joins. Figure (3-4) shows an example of sort-merge joins.

المنارة للاستشارات
www.manaraa.com

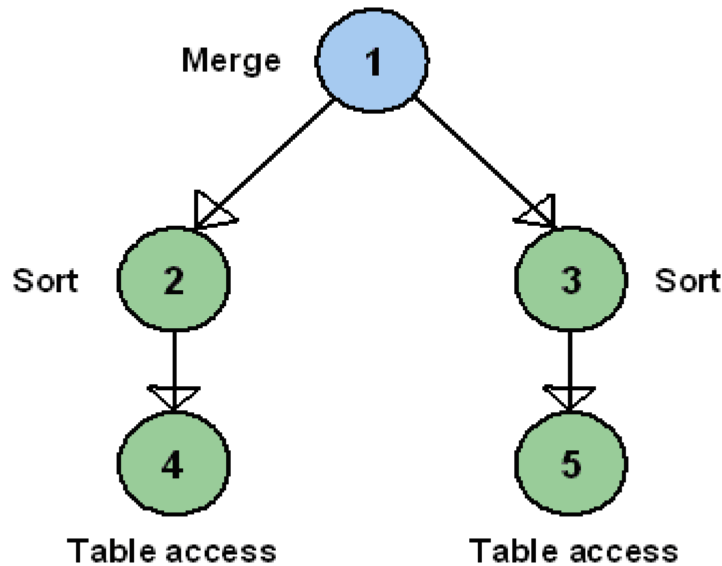


Figure (3-4) Sort-Merge joins

In sort-merge joins, the two row sources are sorted on the values of the columns used in the join predicate. If a row source has already been sorted in a previous operation, the sort-merge operation skips the sort on that row source. Sorting could make this join technique expensive, especially if sorting cannot be performed in memory. The merge operation combines the two sorted row sources to retrieve every pair of rows that contain matching values for the columns used in the join predicate.

Nested loop joins used when we have small number of rows that have a good driving condition between the two tables. But a sort-merge joins used for large amounts of data or the join condition between two tables is not an equijoin. From these two examples we find that, we can rewrite the SQL statement (change the original statement from old syntax to new syntax) and use optimizer hints

المنارة للاستشارات

www.manaraa.com

before sending these statements to optimizer. If we rewrite the statement in proper way, we can make the database optimizer take a good decision and the best execution plan for executing the statement.

3.4 The Proposed System Design

Designing a large system to handle real time problems is a challenge and needs a lot of attention, because you have to compromise between the needs of each requirement. Designing large systems always starts with decomposing it into sub-systems to provide related services, and establishing a framework for sub-system control and communication, so this decomposition may has negative effect on performance. And in contrary for real time systems that puts performance and response time in the first stage by building large-grain rather than fine-grain components to reduce the number of controllers between sub-systems and eliminate communication time between sub-systems, so being in the middle is always the hardest.

Another issue arises in the scene, mobility of the system; we need this system to be an intelligent mobile agent, so it has to be independent, self-governing and self-determining to solve the problems. This mobility forces the agent to own his dictionary and Meta data (repository) about running database, and hold it when traveling from system to another. So this Meta data has to be dynamic, light, easy to manage and with minimal overheads.

المنارة للاستشارات
www.manaraa.com

The proposed agent will be implemented using JAVA language as an interface and will be tested by using ORACLE database as a back end; the implementation will follow these steps, as in the Figure (3-5) taking into consideration that the repository is already built and will be configured:

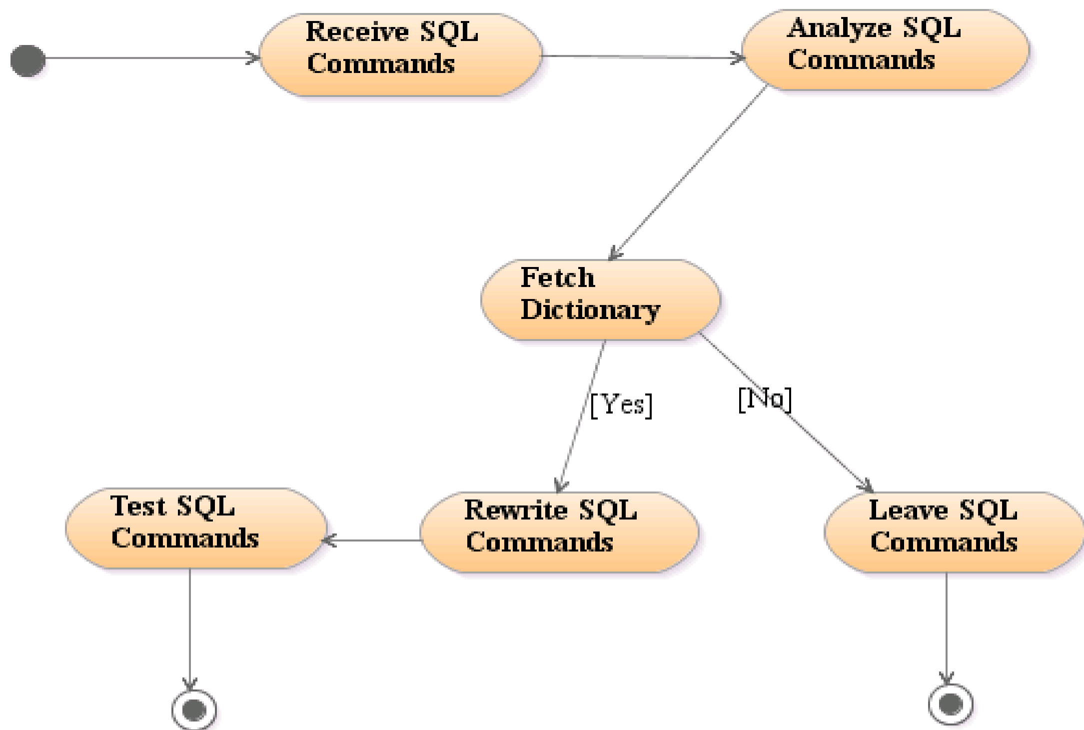


Figure (3-5) Proposed System

1. First the agent connects to target database and consolidate its repository with target database to get any changes or modifications, then the agent waits for any SQL command from application layer, it expects that the command is not error free and there may be syntax problems with command. The agent checks the syntax of the command to report any error if

المنارة للاستشارات
www.manaraa.com

2. the command has any thing conflicting with the dictionary that has been validated with running database. If the command is validated correctly it will be passed to the second stage.
3. The agent analyzes SQL command to see which recommendation it will take from recommendation dictionary to tune this command. Also the agent may find that the SQL command does not need tuning in this step.
4. The agent fetches the recommendation dictionary; basically this dictionary contains rules for writing the SQL command in tuned syntax. If there is any rule that satisfies the SQL command the agent will send the command to rewrite stage, else the agent will leave the command as it is.
5. The agent will leave the SQL command as it is if there is no rule or recommendation that satisfies this command.
6. Rewriting stage is the most complex stage because the agent is going to restructure the SQL command with new syntax taking two things in its consideration, the first one is rewriting the command with better performance syntax after getting the help from recommendation dictionary, the second one is resulting the same data (output) as previous old command.
7. The agent will test the new SQL command and compare it with old one to see difference in time, IO, network round-trips, execution time and many. These results will be saved in a statistics dictionary to measure the overall performance of the agent in a real live system.

المنارة للاستشارات
www.manaraa.com

3.5 Detailed System Design

In this section we will go deeply in agent design describing each component, its functionality and how these components communicate with each other. Neither large-grain components nor fine-grain components are going to be used in agent design, Agent components are going to be in the middle range as possible. Three types of diagrams are going to be used in this section; first, use-case diagram will describe the interaction between actors with use-cases. Second, sequence diagrams are used to add information to a use-case and show how the actors are involved in the interaction. Third, class diagram shows the structure of the agent as classes and relations between these classes.

The system (agent) use-cases and its interaction with actors are shown in Figure (3-6).

المنارة للاستشارات
www.manaraa.com

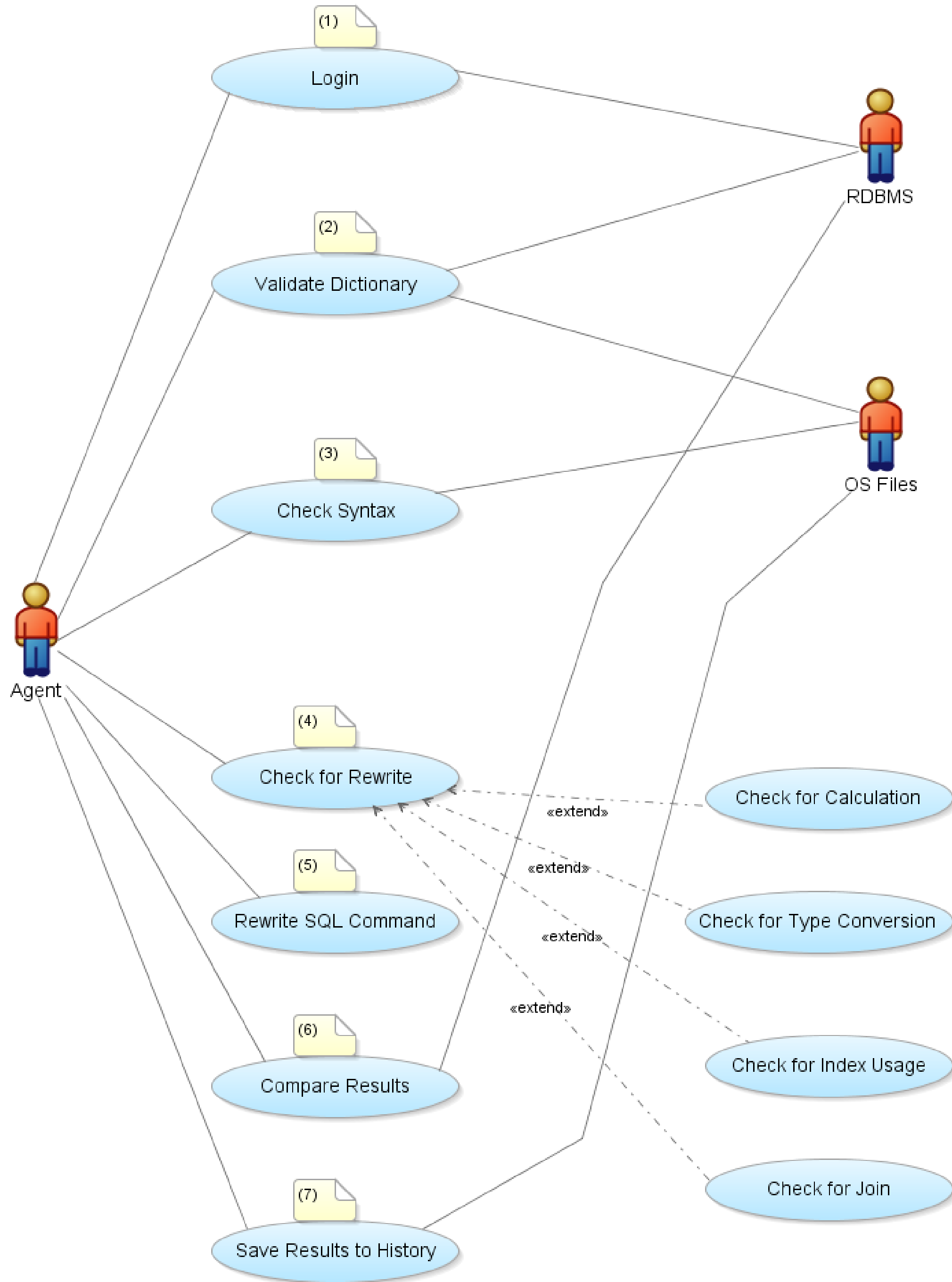


Figure (3-6) System Use-Case Diagram

المنارة للاستشارات
www.manaraa.com

Figure (3-6) shows us three actors:

1- Agent actor as a primary player in the system which is going to instantiate most of the system use-cases, all of system operations are going to start from this actor.
2- RDBMS actor to represent the database that the agent is going to handle, then the agent will execute SQL commands in this database before and after rewriting.
3- OS Files actor to represent the agent dictionary that will be used as a Meta data for the agent wherever the agent travels between hosts.

Actors communicate with use-cases; use-cases are described as follows:

1- Login use-case to establish connection between our system (agent) and database using authorized username, password and correct connection string.
2- Validate dictionary use-case is going to check if the agent has the correct and matched dictionary as the database, if the agent has dictionary (Meta data) different than database, this use-case will remove and create the dictionary again then save this dictionary in the operating system files.
3- Check syntax use-case will check if the SQL command received has any syntax error before proceeding for tuning this command, and it will report an error if the command is not valid.
4- Check for rewrite use-case will examine the SQL command to see if there is any condition that exists for rewriting the command. These conditions are described in the use-cases that extend from this use-case like checking if the SQL

المنارة للاستشارات
www.manaraa.com

5- command has mathematical operations, the SQL command has implicit type conversion, the SQL command does not use indexes, or the join condition is not written in a correct format. All of these conditions will make the SQL command run slowly, it needs to be rewritten.
6- Rewrite use-case will rewrite the SQL command if any command has any condition described in previous use-case.
7- Compare results use-case will get all of metrics and statistics for old SQL command (before rewriting) and new SQL command (after rewriting) to show the difference between them and highlight the gain that the agent made by rewriting the SQL command.
8- Save results use-case will store the metrics resulted from pervious use-case to file system for future analysis.

Once the interactions between the software system that is being designed and the system's environment have been defined, we can use this information as a basis for designing the system architecture. From this information we can introduce a class diagram to show the overall system structure as shown in Figure (3-7):

المنارة للاستشارات

www.manaraa.com

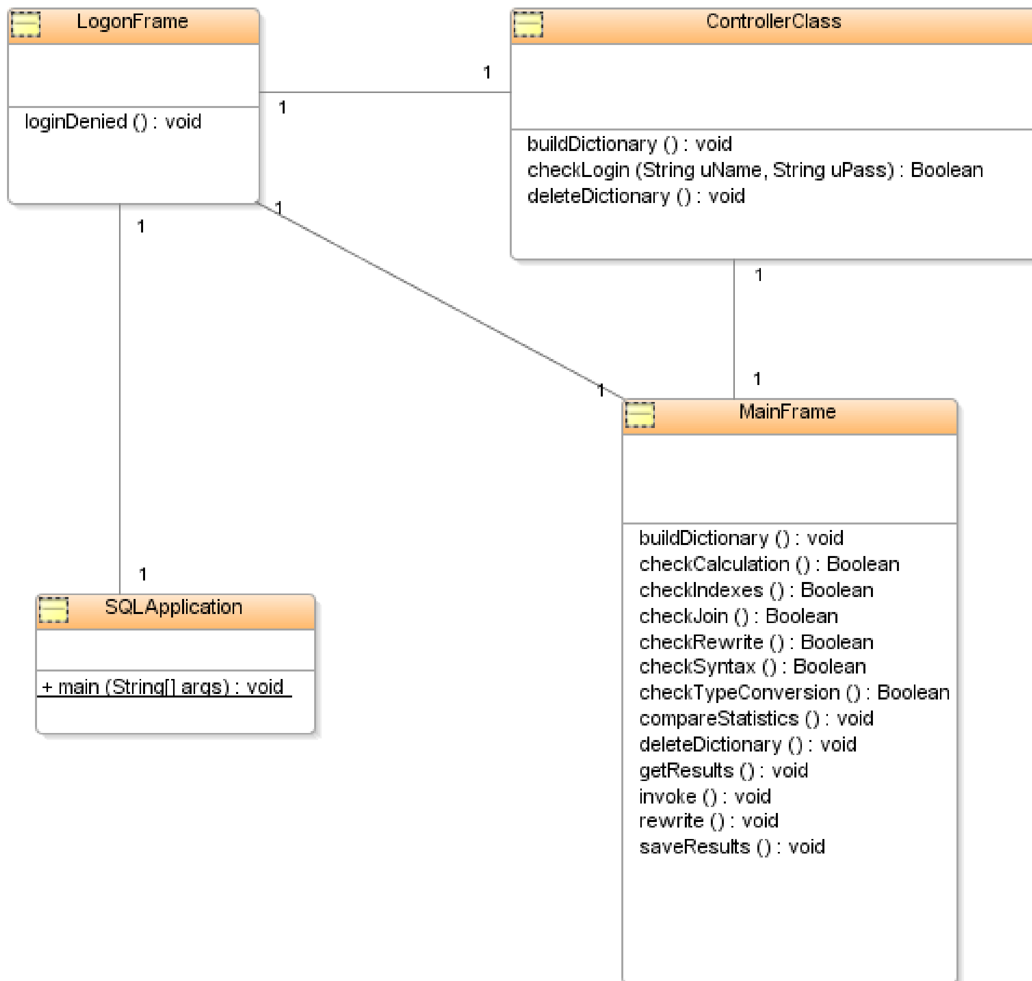


Figure (3-7) System Class Diagram

System (agent) starts from **SQLApplication** class which is the initiation method to start the agent. Form **LogonFrame** class the agent gets user name and password from administrator and tries to connect to database using the **ControllerClass** class which controls the communication with database. The **MainFrame** class has as a major role in the system because it is going to handle most of the agent work hand by hand with **ControllerClass**.

المنارة للاستشارات
www.manaraa.com

Sequence diagrams are models of the use-cases, for each mode of interaction between actor and use-case, a sequence diagram takes place. The following diagrams will illustrate the interaction for each use-case described before.

3.5.1 Login Use-case

Figure (3-8) shows the interaction of the first use-case (login use-case). As we can see agent starts the login process by putting username and password to **LogonFrame**, then this class asks the **ControllerClass** to connect to database and check if this login is valid or not, if it is not valid an error message will come into sight else the **LogonFrame** class will send the control to **MainFrame** class to continue agent process.

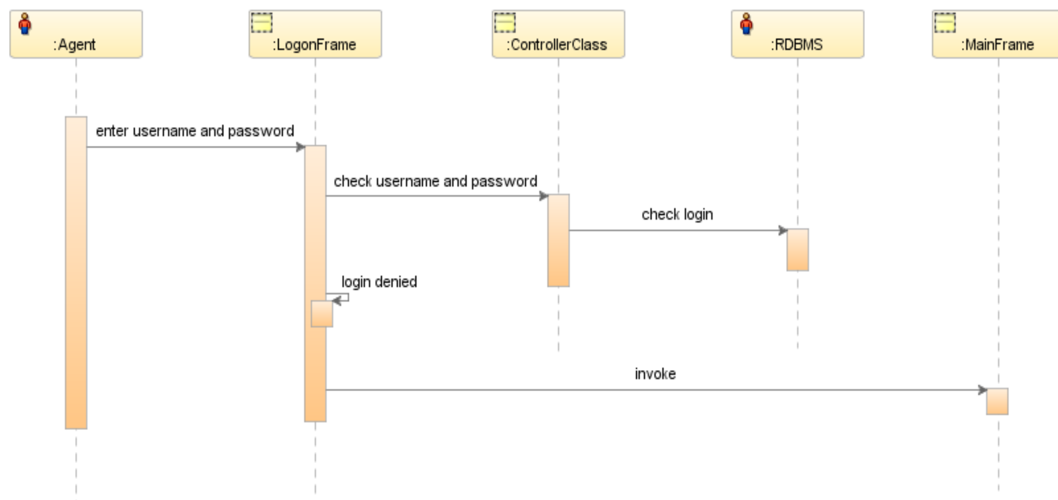


Figure (3-8) Login Sequence Diagram

3.5.2 Validate Dictionary Use-case

Figure (3-9) shows the interaction of the second use-case (validate dictionary use-case). In this use-case the agent mainly does two things, first one is building the dictionary by asking this from **MainFrame** class, then **MainFrame** class sends this request

المنارة للاستشارات
www.manaraa.com

to **ControllerClass** which in turn gets the dictionary information from database and saves it to operating system files to be traveled with agent wherever the agent goes. The second issue is deleting the dictionary by **ControllerClass** if there are any new changes in the database to be reflected to agent dictionary when the agent builds it again.

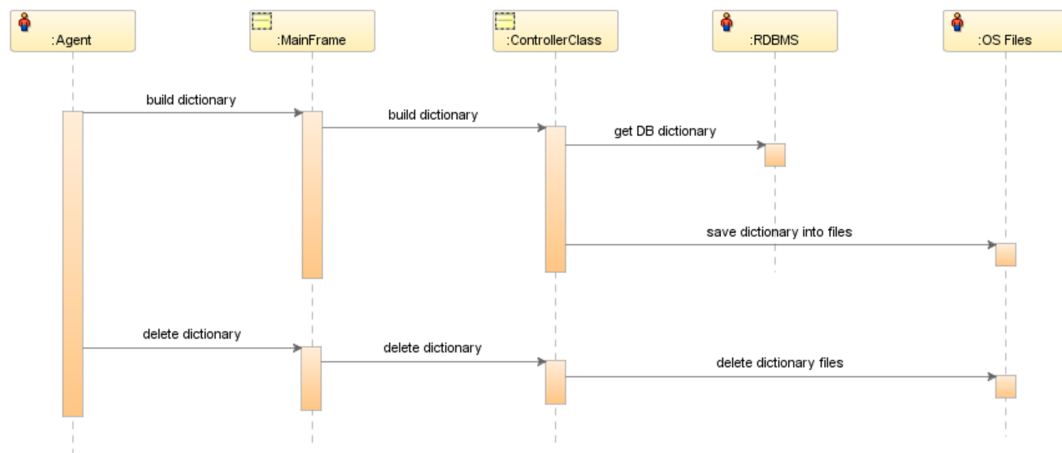


Figure (3-9) Validate Dictionary Sequence Diagram

3.5.3 Check Syntax Use-case

Figure (3-10) shows the interaction of the third use-case (check syntax use-case). In this use-case the agent sends check syntax request to **MainFrame** class, then **MainFrame** class reads the SQL statement and checks if it has correct tables name, columns name, and if it has correct SQL statement structure, if it is not, an error message will be shown, and if the SQL statement has no errors the agent will pass it to next use-case (check for rewrite).

المنارة للاستشارات
www.manaraa.com

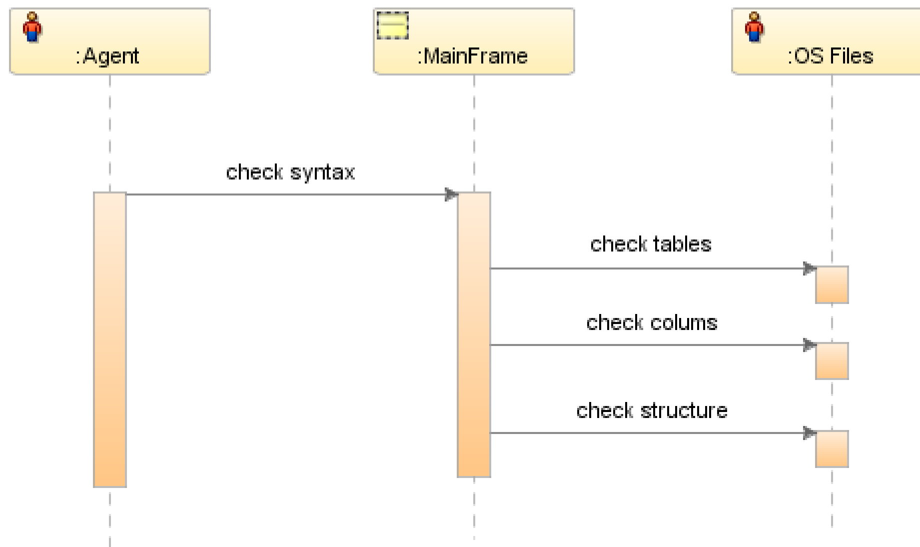


Figure (3-10) Check Syntax Sequence Diagram

3.5.4 Check for Rewrite Use-case and Rewrite Use-case

Figure (3-11) shows the interaction of the fourth and fifth use-cases (check for rewrite use-case and rewrite use-case). In check for rewrite use-case the agent sends check for rewrite request to **MainFrame** class, and then this class does the job. It checks if:

- The SQL statement has mathematical operations that prevent index usage
- SQL statement has implicit or explicit type conversion.
- Does SQL statement use the indexes properly?
- The SQL statement has proper written join condition.

If the statement has one of these conditions, the agent will decide to rewrite it so the agent sends request to **MainFrame** class to rewrite the SQL statement and remove the condition that makes the SQL statement slow.

المنارة للاستشارات
www.manaraa.com

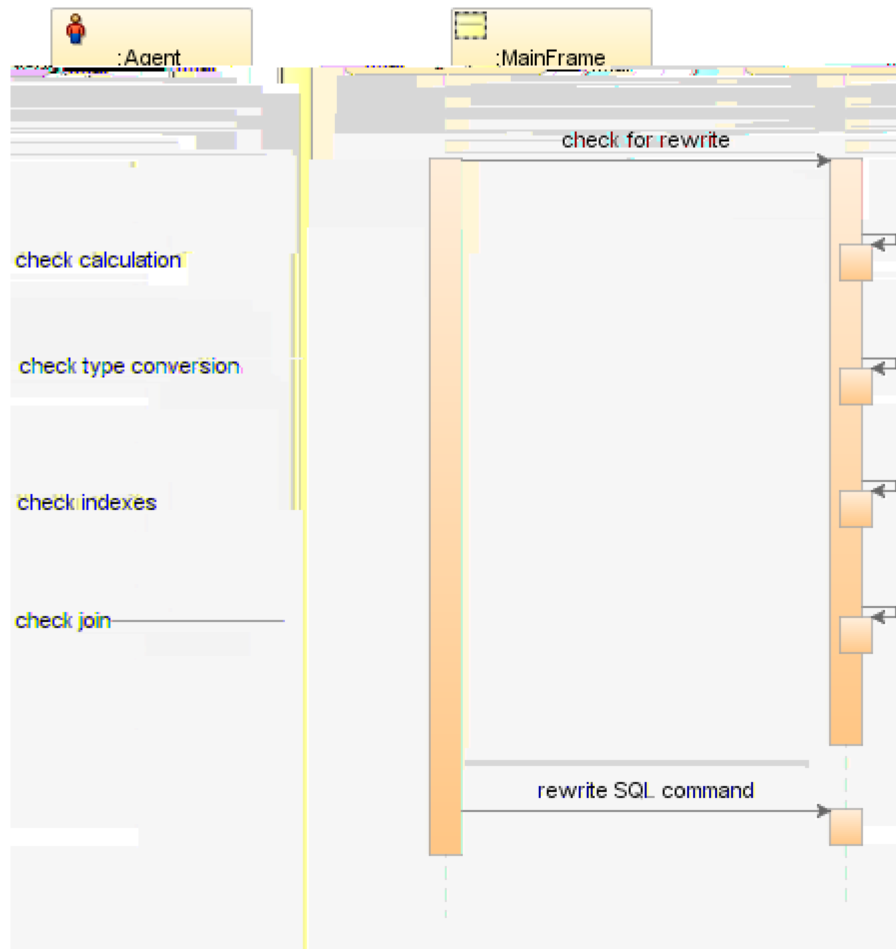


Figure (3-11) Check for Rewrite Sequence Diagram

3.5.5 Compare Results Use-case

Figure (3-12) shows the interaction of the sixth use-case (compare results use-case). In this use-case the agent asks the **MainFrame** class to get statistics about old SQL statement (before rewriting) and statistics about new SQL statement (after rewriting), and compare these statistics with each other to see the difference between these results. Statistics contain a lot of metrics about SQL statements like estimated time to execute the statement, actual time for executing the statement, number of bytes and blocks read from database, number of network round-trips between client and server

المنارة للاستشارات
www.manaraa.com

, cost of statement, execution plan for statement, and others.

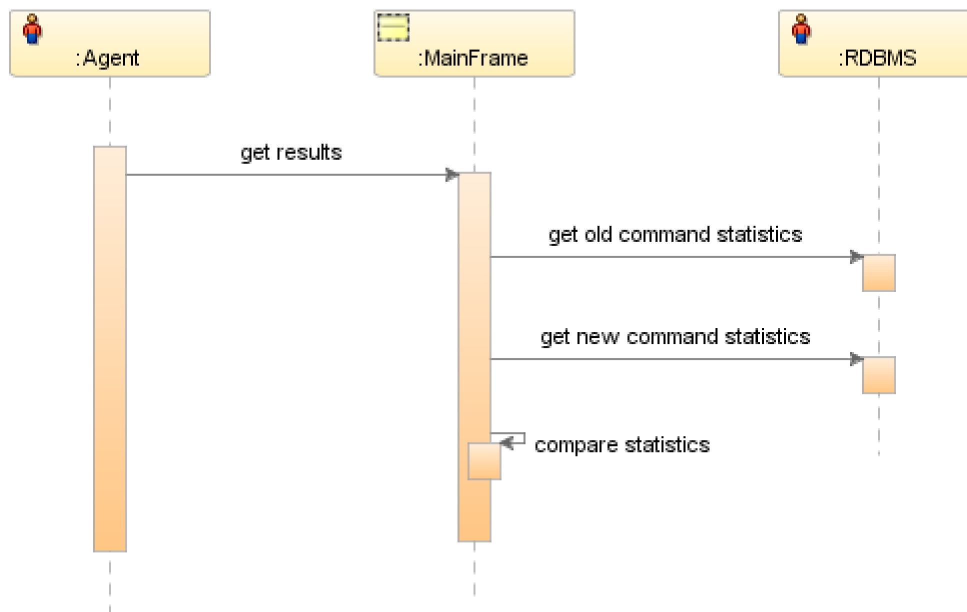


Figure (3-12) Compare Results Sequence Diagram

3.5.6 Save Results to History Use-case

Figure (3-13) shows the interaction of the seventh use-case (Save results to history use-case). In this use-case the agent asks the **MainFrame** class to save the statistics about the old and new SQL statements in operating system files to be used in the future and to be used for statistical reports generated from the agent. The statistics are saved into files because the agent may need them when they travel to another machine.

المنارة للاستشارات
www.manaraa.com

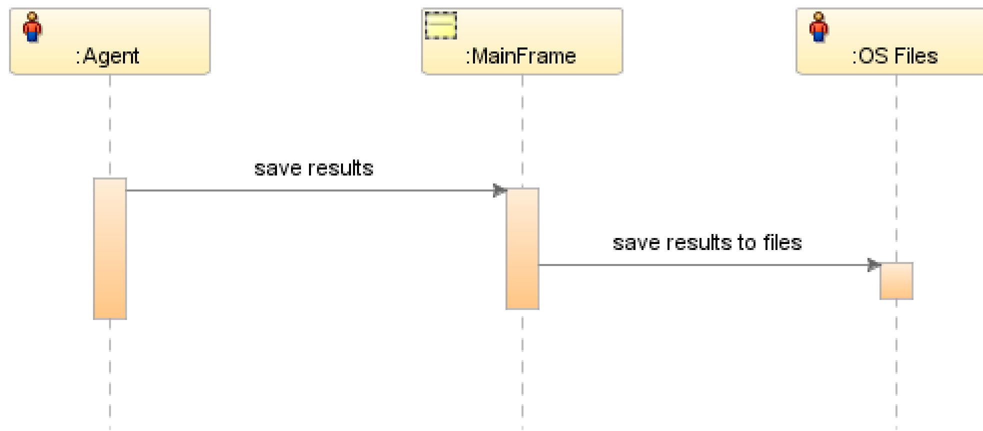


Figure (3-13) Save Results to History Sequence Diagram

3.6 Test Case Design

The ultimate goal of this thesis is to put the agent in real life environment and get the results that we expect. But first we have to put it in a test environment and see how the agent behaves. So we created a test case for this agent which includes an information system with a lot of heavy and complex SQL commands. The test case inspired from systems that can handle a large number of concurrent users efficiently, and can store, retrieve information competently like conference systems that serve public users from the internet as Skype system or Yahoo Messenger system, these systems do not serve thousands or few of millions of concurrent users but it handles tens of millions and sometimes more of concurrent users. Also the system has to store all details about conferences, information sent between conference parties, and statistics about each conference.

المنارة للاستشارات
www.manaraa.com

Figure (3-14) shows database design for a conference system that will be used to test the agent.

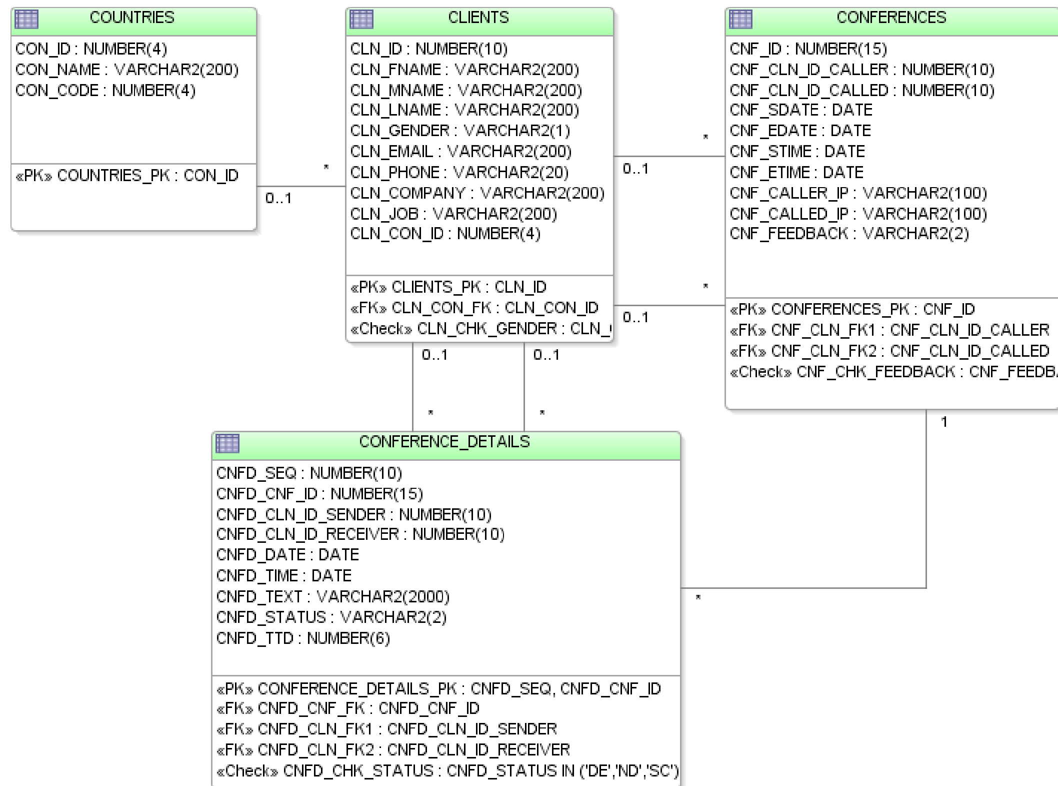


Figure (3-14) Database Design for Conference System

As shown in Figure (3-14), we have **COUNTRIES** relation to store the ids, names and codes of client country with country id as a primary key. The **CLIENTS** relation to store information about clients like name, gender, phone, and more, the **CLIENTS** relation has client id as primary key and country id as foreign key to **COUNTRIES** relation. The **CONFERENCES** relation stores information about each conference done by any client like conference start date and time, conference end date and time, caller id, and called id. The **CONFERENCES** relation has conference id as primary key and it has two foreign keys to **CLIENTS** relation, first

المنارة للاستشارات
www.manaraa.com

one for caller id and second one for called id. The **CONFERENCE_DETAILS** relation stores information about conference lines as the text sent between each clients, status of each message sent, and if the message received or not. This relation has a sequence as primary key; it has foreign key to **CONFERENCES** relation and two foreign keys to **CLIENTS** relation one as sender and the other as receiver.

المنارة للاستشارات

www.manaraa.com

Chapter Four
Implementation and Experimental Work

4.1 Overview

In the previous chapter, the researcher developed agent design and use-cases to implement the scenarios that may happen when receiving SQL commands from clients. In this chapter, the researcher will present the architecture of the implemented query rewrite mobile agent, where those use-cases are applied. The mobile agent was developed using Java language as an interface and Oracle database as a back end to receive the SQL commands. The main features of the agent are efficiency, ease of use and adaptability with various types of operating systems with different kinds of databases running on those machines.

4.2 Metrics and Statistics

Monitoring for performance requires certain information that goes beyond statistics. To determine whether a particular statistic is important, the researcher needs to know how much it has changed over a certain period of time. To be proactive, the researcher needs to be notified when certain conditions exist (for example when system response time approaches the agreed maximum). To diagnose performance issues, the researcher needs to know what has changed. Metrics and statistics provide this information. A **metric** is a timed rate of change in a cumulative statistic (for example, physical reads per second, logical reads per second). **Statistics** are counters of events that happen in the database in a raw data format.

المنارة للاستشارات
www.manaraa.com

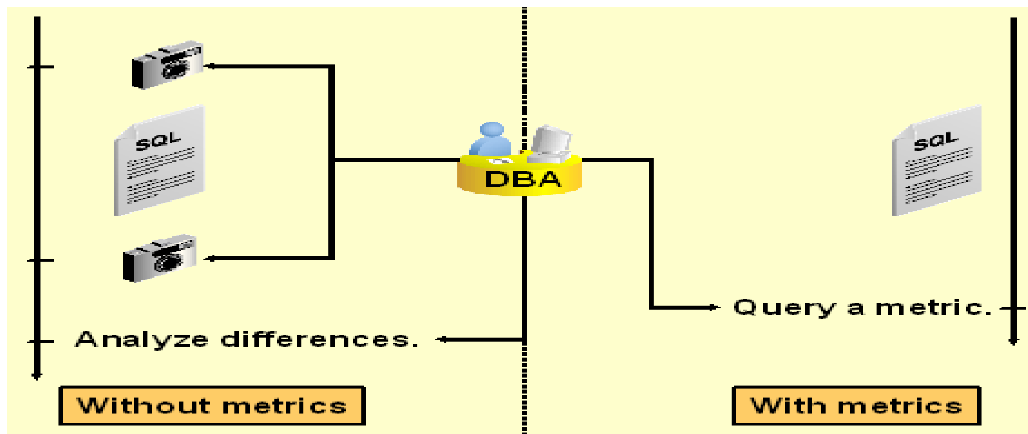


Figure (4-1) Metrics and Statistics

The main benefit of keeping metrics is that the data is readily available when a component needs to compute the rate of change of some activity. But with statistics only the researcher had to capture statistics before and after running SQL command to compute the changed rate for a particular base statistic. With metrics, all the researcher needs to do is to run the SQL command and select the corresponding metrics as illustrated in Figure (4-1).

Although the metrics can give you an idea of the trend for particular statistics, but they do not tell you if a particular bottleneck is affecting the whole system or if it is just localized. As an example, you can observe a high metric rate, but this sudden increase could be localized to only one or two sessions in your system. In this case, it might not be worth investigating the issue. However, if the sudden increase is generalized to the whole system, you need to investigate further. The metrics can alert you to potential problems. By drilling down using statistics, you can clearly determine whether there really is a problem[21].

المنارة للاستشارات
www.manaraa.com

4.3 Access Paths

Access paths are the ways in which data is retrieved from the database. Any row can be located and retrieved with one of the methods determined by RDBMS optimizer. Access paths are revealed and used by database optimizer, access paths can be one of these:

1- **Full table scan:** the database will access the table row by row, this operation may take a long time and a lot of physical reads, but it is useful when SQL statement access a large portion of the table like decision support system (DSS) that access a large number of rows in each statement to generate statistical reports.

2- **ROWID scan:** each record in a table has a physical pointer called ROWID, which contains the physical address of the record as hard disk number, file number, partition number, block number and record number. If database optimizer has the ROWIDs for certain records, it will use them to get the specified rows.

3- **Index scan:** using the index to get ROWIDs of the selected records may make the SQL statement very fast and efficient especially when SQL statements retrieve a small subset of table rows. OLTP applications which consist of short running SQL statements with high selectivity are characterized by using index scan access path.

The database optimizer chooses the access path based on the following factors:

- Available access paths for the statement
- Estimated cost of executing the statement, using each access path or combination of paths

المنارة للاستشارات

www.manaraa.com

To choose an access path, the optimizer first determines which access paths are available by examining the conditions in the statement's WHERE clause and it's FROM clause. The optimizer then generates a set of possible execution plans using available access paths and estimates the cost of each plan, using the statistics for the index, columns, and tables accessible to the statement. Finally, the optimizer chooses the execution plan with the lowest estimated cost. Choosing the access path, the optimizer is influenced by the following:

- **Optimizer hints:** The optimizer's choice among available access paths can be overridden with hints
- **Old statistics:** For example, if a table has not been analyzed since it was created, and if the table is small, then the optimizer uses a full table scan.

Full table scan reads all rows from a table and filters out those that do not meet the selection criteria (WHERE clause). During a full table scan, all blocks in the table are scanned. Each row is examined to determine whether it satisfies the statement's WHERE clause. When performing a full table scan, the blocks are read sequentially. Full table scans are cheaper than index range scans when accessing a large fraction of the blocks in a table (more than 75% of the table). They are cheaper because full table scans can use larger I/O calls; making fewer large I/O calls is cheaper than making many smaller I/O calls. The database optimizer uses a full table scan in each of the following cases:

المنارة للاستشارات
www.manaraa.com

- **Lack of index:** If the query is unable to use any existing indexes, then it uses a full table scan. For example, if there is a function used on the indexed column in the query, the optimizer is unable to use the index and instead uses a full table scan.
- **Large amount of data:** If the optimizer thinks that the query will access most of the blocks in the table, then it uses a full table scan, even though indexes might be available.
- **Small table:** If the entire table contains few numbers of blocks, then a full table scan might be cheaper because this can be read in a single I/O call.

The ROWID of a record specifies the physical location of the row (data file, data block, as well as the location of the row in that block). Locating a row by specifying its ROWID is the fastest way to retrieve a single row, because the exact location of the row in the database is specified. To access a table by ROWID, the optimizer first obtains the ROWIDs of the selected rows, either from the statement's WHERE clause or through an index scan of one or more of the table's indexes. The database then locates each selected row in the table based on its ROWID. This is generally the second step after retrieving the ROWID from an index. The table access might be required for any columns in the statement that are not present in the index. Access by ROWID does not need to follow every index scan. If the index contains all the columns needed for the statement, then table access by ROWID might not occur.

المنارة للاستشارات
www.manaraa.com

In index scan method, the indexed column values specified by the statement are used to retrieve the row. An index scan retrieves data from an index based on the value of one or more columns in the index. To perform an index scan, the database searches the index for the indexed column values accessed by the statement using binary search method. If the statement accesses only columns of the index, then the database reads the indexed column values directly from the index rather than from the table. The index contains not only the indexed value but also the ROWIDs of rows in the table having that value. Therefore, if the statement accesses other columns in addition to the indexed columns, then the database can find the rows in the table by using a table access by ROWID.

4.4 Use-case Implementation

In this section the researcher will describe each use-case mentioned in the previous chapter in details, flowchart to describe the implementation, algorithms used to implement them, and also the pseudo code used to get deep understanding of each use-case.

4.4.1 Login Use-case

The login use-case will handle the connection between the agent and the database, in our case the agent will be implemented using Java language and the database will be Oracle database. Oracle supports many types of Java connection such as Thin connection, OCI (Oracle Call Interface) connection, and Lite connection. Thin connection is the most popular type of connections because it is portable, operating system independent, and fully implemented in Java. OCI connection is not portable connection and you have to install OCI driver in each machine you are going to use, and this is against the agent portability feature. OCI driver has a

المنارة للاستشارات
www.manaraa.com

feature that it can connect to database using network protocols other than TCP protocol, but our agent interested in TCP protocol because it is the protocol which dominates other network protocols. The Lite connection is implemented by Oracle to support Oracle Lite Database which is a type of database installed in mobile devices like PDAs and Palms. So the agent decided to use Thin connection because it is the most connection type suitable for it.

Thin connection requires three variables, the first one is the user name of database schema, the second one is the password, and the third one is the JDBC (Java DataBase Connectivity) which consists of database server IP address or host name, the network port is used by database server to open the connection, and the database name to connect to, because the database server may contain more than one database. Figure (4-2) shows the connection dialog used by the agent which contains three input fields as described above.

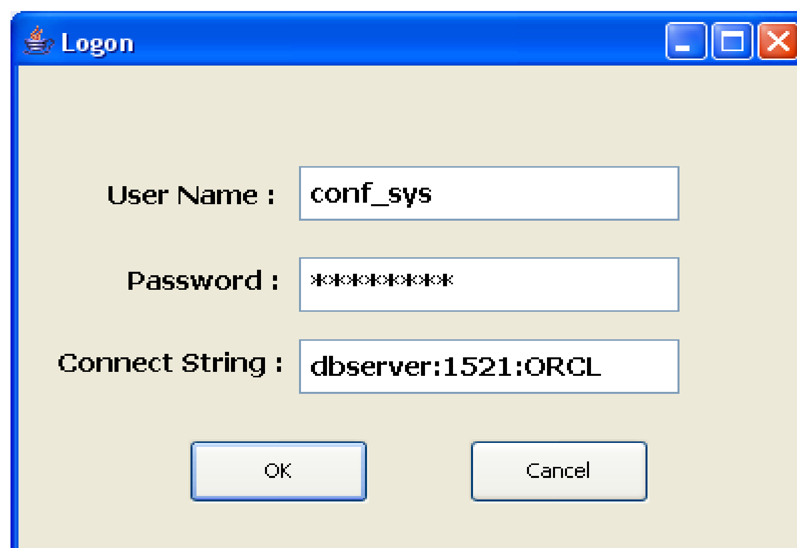


Figure (4-2) Login Dialog

المنارة للاستشارات
www.manaraa.com

The agent will receive the connection information (Username, Password), then the agent will try to connect to database using these variables, if it succeeds, the control will be passed to MainFrame screen, otherwise an error message will be displayed to inform that we have invalid username or password. As illustrated in Figure (4-3) the login sequence is described in flowchart diagram.

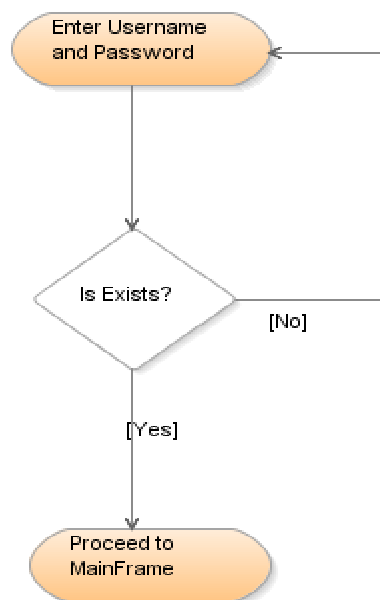


Figure (4-3) Login Flowchart Diagram

The Pseudo code illustrated below shows how the login use-case works:

get Username

get Password

connect to database

if successful

    go to MainFrame

else

المنارة للاستشارات

www.manaraa.com

show error message

end if

4.4.2 Validate Dictionary Use-case

Agent must be portable with fast response, so all information needed for it must be packaged with the agent. Dictionary of the database used in SQL commands is highly used and needed to check the syntax of the command and to do the rewrite of the required commands. So when starting up the agent and after login to database, the agent checks if the dictionary exists, if not, the agent builds the dictionary then loads it to agent buffer. If the dictionary exists just load it to agent buffer.

Dictionary must travel with agent as a Meta data for existing database, so the best way to insure portability of this Meta data is to build it and store it in XML format. The XML file shown in Figure (4-4) gives detailed information about the dictionary of the test schema the researcher is going to use in this thesis. As shown in Figure (4-4) the root element of the XML file is <DBdictionary> which contains all other elements, then the element <tables> which contains all tables of the schema, and each <table> element contains <column> element to store the name, type, length, and scale of the column using <name>, <type>, <length>, and <scale> elements in sequence. Some elements has attributes like <table> element has "name" as an attribute to store the name of the table, and <column> element has "id" as an attribute to store column identifier.

المنارة للاستشارات
www.manaraa.com

```
<?xml version="1.0" ?>
- <DBdictionary>
  - <tables>
    - <table name="CLIENTS">
      + <column id="0">
      - <column id="1">
          <name>CLN_CON_ID</name>
          <type>NUMBER</type>
          <length>4</length>
          <scale>0</scale>
        </column>
      - <column id="2">
          <name>CLN_EMAIL</name>
          <type>VARCHAR2</type>
          <length>200</length>
          <scale>null</scale>
        </column>
      + <column id="3">
      + <column id="4">
      + <column id="5">
      + <column id="6">
      + <column id="7">
      + <column id="8">
      + <column id="9">
      </table>
    + <table name="CONFERENCES">
    + <table name="CONFERENCE_DETAILS">
    + <table name="COUNTRIES">
    + <table name="DEPT">
    + <table name="EMP">
    </tables>
  </DBdictionary>
```

Figure (4-4) Dictionary XML File

To generate the dictionary XML file, the agent looks for this file first if it exists or not, if the file exists the agent will read this file and load the buffer with Meta data. But if the file does not exist, the agent will read the database to get tables names and columns names and other information to generate the XML file and load it in the buffer. Figure (4-5) illustrates the flowchart of the use-case.

المنارة للاستشارات
www.manaraa.com

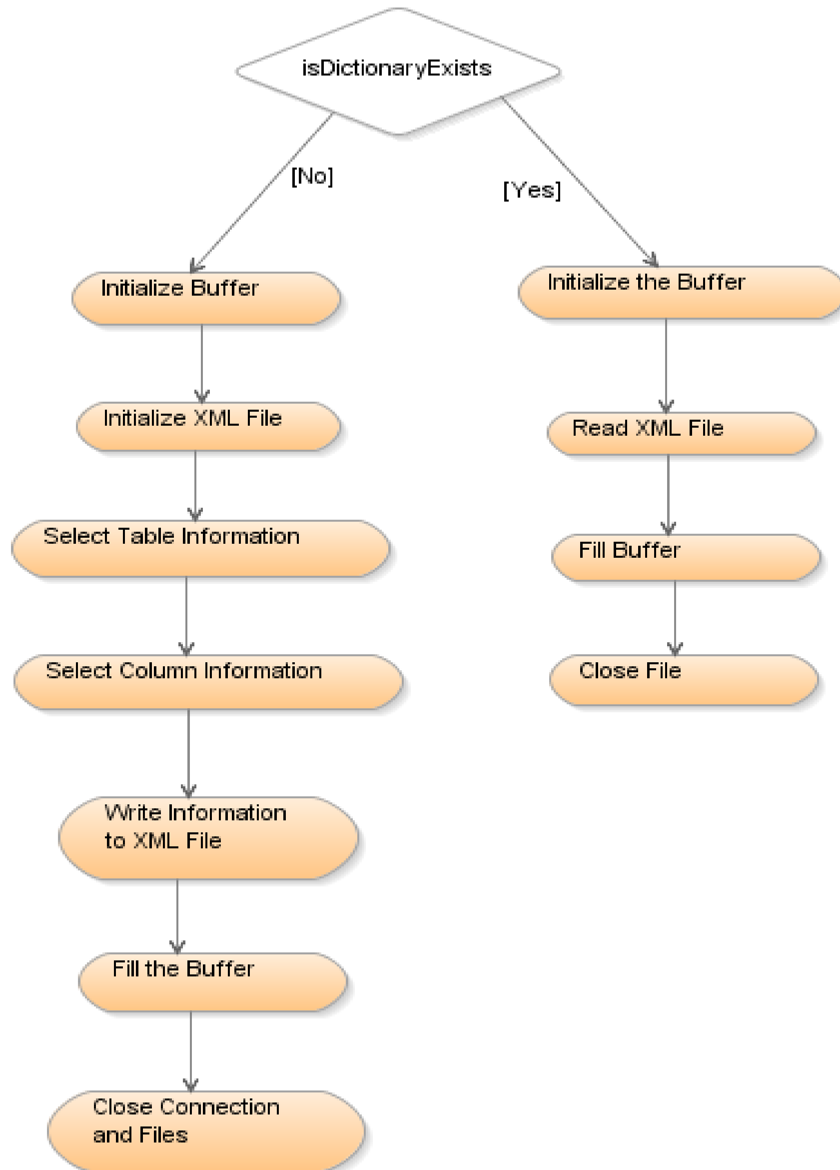


Figure (4-5) Validate Dictionary Flowchart Diagram

The Pseudo code for validating dictionary use-case is shown as follows:

If dictionary does not exist

Initialize table array

Initialize column array

Open XML file

Select table's information

المنارة للاستشارات
www.manaraa.com

Select column's information

Write information to XML file

Fill information to arrays

Close connection and XML file

else

Initialize table array

Initialize column array

Read XML file

Fill information to arrays

Close connection and XML file

end if

4.4.3 Check Syntax Use-case

Rewriting SQL statement and getting better execution time requires that the SQL statement must be written with correct syntax, so before sending SQL statement to rewrite process, the agent has to insure that the statement has no syntax errors. Agent has two types of error checking, the explicit type (offline) which requires the user to ask for syntax checking, and implicit type (online) which is done automatically without user intervention.

To see the difference between online and offline syntax checking we have to take a few description of the agent interface. The interface is built to see the behavior of the agent in real word and get output numbers from SQL command processed by the agent. Agent has some options to interact with it in spite of all processes that are done internally. As seen in Figure (4-6) we have two tabs SQL Command and SQL Results. The first tab (SQL Command) is divided into two parts, the left part for old SQL

المنارة للاستشارات
www.manaraa.com

command with its own metrics and the right part with new SQL command and new metrics, metrics are shown below the SQL command area like (Real Time, Position, Cost, …). The second tab (SQL Results) shows us the execution output of the old command compared with the new command just to be sure that rewriting the command did not change the original output of the command. The toolbar of the interface contains three buttons, the first one to check the syntax, the second one to rewrite the SQL command, and the third one to compare the old command with new command and fill out the metrics for both commands.

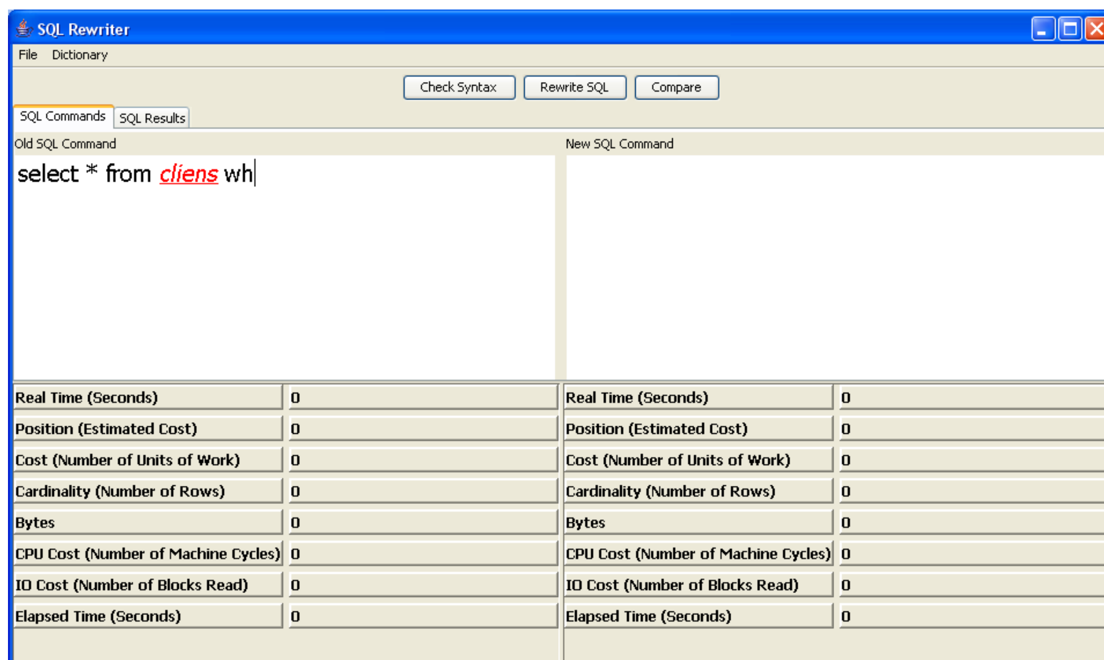


Figure (4-6) Agent Interface

From Figure (4-6) the researcher can see the online syntax checking. If a command with wrong table name is written, the agent will instantly display the table name in ***red*** after proceeding to next word, as in Figure (4-6) the user wrote the table name (***cliens***) instead of (clients), so the agent immediately formatted the error.

المنارة للاستشارات
www.manaraa.com

Figure (4-7) illustrates online syntax checking but this time with wrong column name (***cl_id***) instead of (cln_id) which is the correct column name.

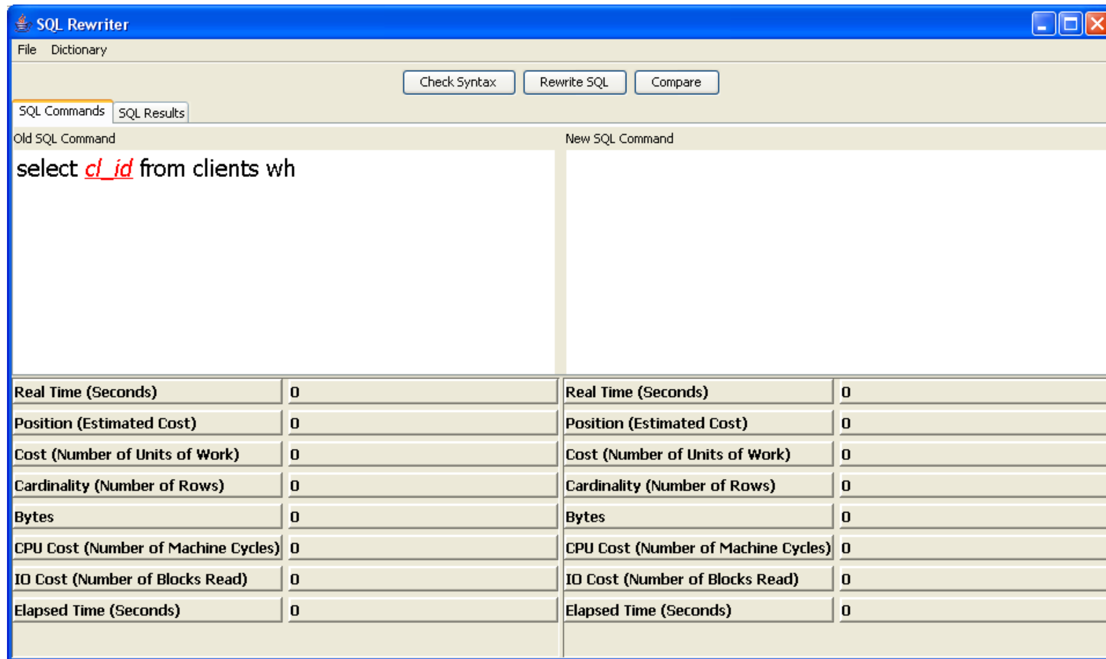


Figure (4-7) Agent Online Syntax Checking

From Figure (4-8) we can see the offline syntax checking. If we wrote a command then we can check the syntax by pressing the button (Check Syntax) and a message will be displayed to inform us if the syntax is correct or not.

المنارة للاستشارات
www.manaraa.com

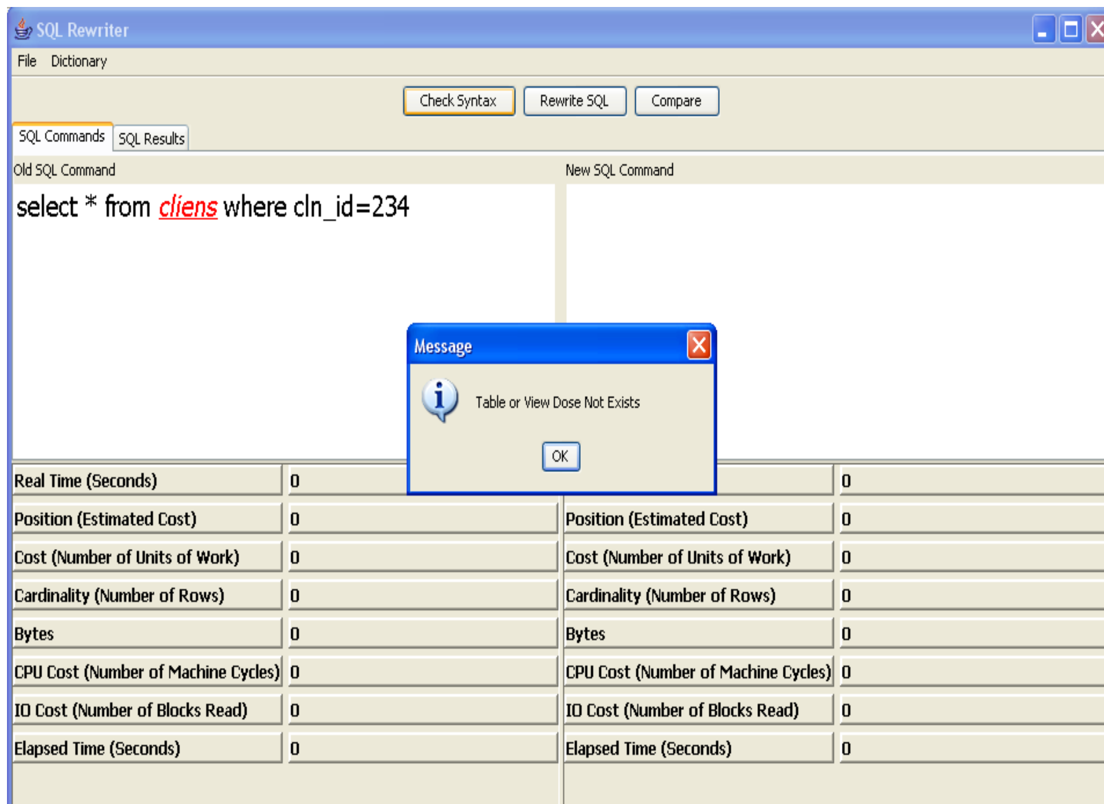


Figure (4-8) Agent Offline Syntax Checking

To check the syntax of the command, the agent has to tokenize the statement into tokens then check each token if it is in correct position and spelling like reserved words such as (select, from, where), and to check for columns names also the tables names if exist in the dictionary which is built in previous process. Figure (4-9) illustrates a flowchart of the processes followed by pseudo code that implements this use-case.

المنارة للاستشارات
www.manaraa.com

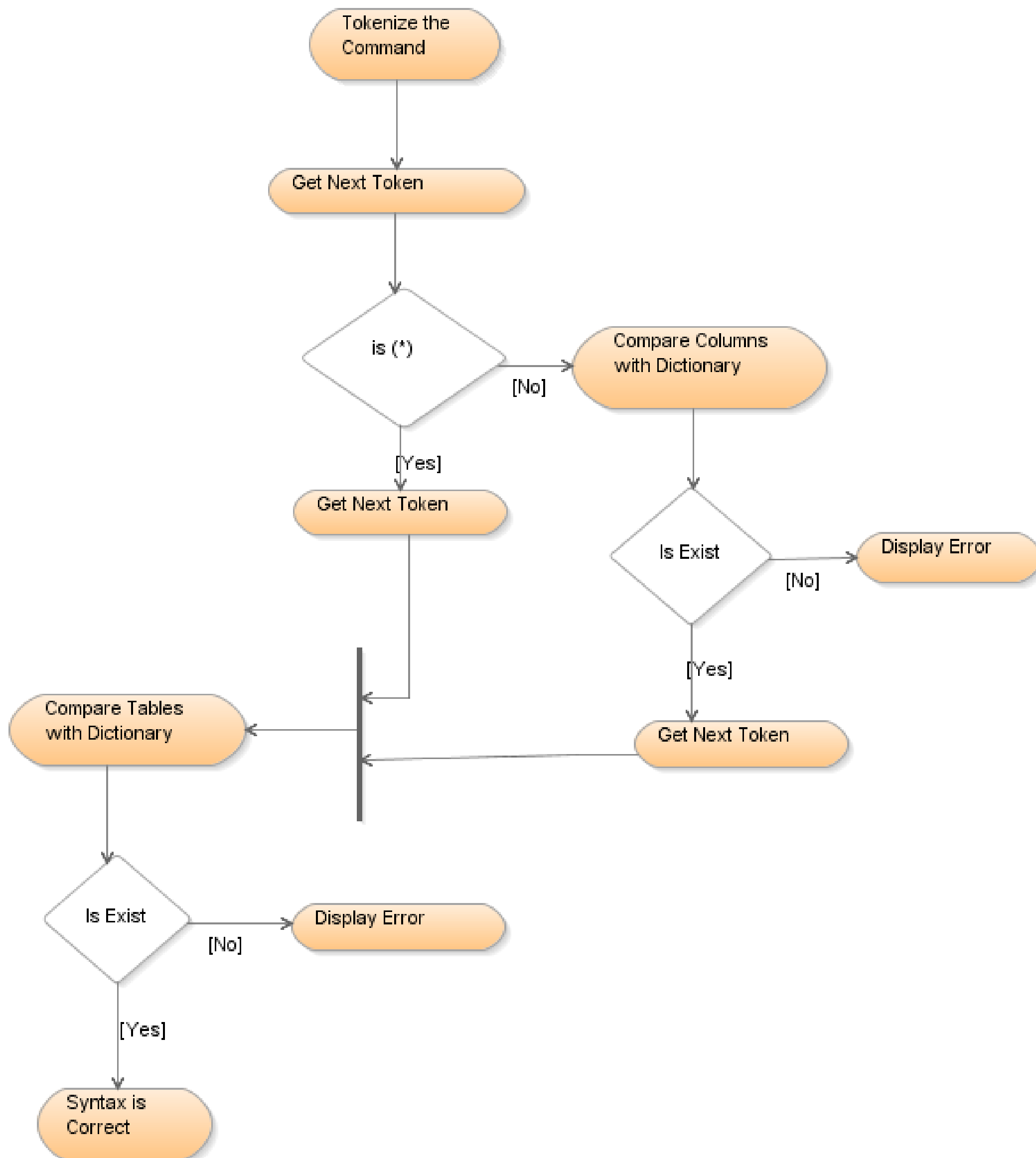


Figure (4-9) Check Syntax Flowchart

The Pseudo code for checking syntax use-case is show as follows:

Tokenize the statement

Get next token

If the token is (*)

Get next token

else

المنارة للاستشارات
www.manaraa.com

```
            Separate each column by (,)
            Compare column with dictionary
            If column not exist
                  Display error message
            end if
            Get next token
            Separate each table by (,)
            Compare table with dictionary
            If table not exist
                  Display error message
            end if
      end if
```

4.4.4 Check for Rewrite Use-case and Rewrite Use-case

Both use-cases are considered as the heart of the agent, because they are the most important processes which have to be done by the agent. So the researcher divided these processes to sub-processes to be easier in implementation and tracking. As mentioned in chapter 3 there are four use-cases which extend check for rewrite use-case and the researcher is going to describe the implementation of each one in details.

4.4.4.1 Check for Calculation

In this sub-process the agent searches for mathematical operations done by SQL statement, the agent is interested in the mathematical operations done in the left side of the WHERE clause of the SQL statement, because if we use any mathematical operation with the column the database management system will ignore the index created in this column and will decide to use

المنارة للاستشارات
www.manaraa.com

sequential search. But if the statement has any operation in the right side, the agent will ignore it and will consider it as a passed statement from this sub-process.

The algorithm of this sub-process is simple, the agent will divide the SQL statement into tokens if it is not tokenized before and will search at the token after the WHERE clause, if there is any mathematical operation like addition, subtraction, multiplying, division (+, -, *, %) the agent will search for the equal sign (=), and any operation in the left side of the equal sign will be moved to the right side of the equal sign after converting it to opposite operation so the addition in left side will be subtraction in right side, multiplying will be division and so on, for example if we have WHERE clause like this:

WHERE (((CLN_ID + 5 / 2) – 3) * 4) = 232323

It will be converted to this:

WHERE CLN_ID = ((((232323 / 4) + 3) * 2) - 5)

Figure (4-10) shows us how the agent converts a simple mathematical operation, and Figure (4-11) shows us a more complex operation, we can see the differences be comparing the old section of the SQL statement with the new section of SQL statement.

المنارة للاستشارات
www.manaraa.com

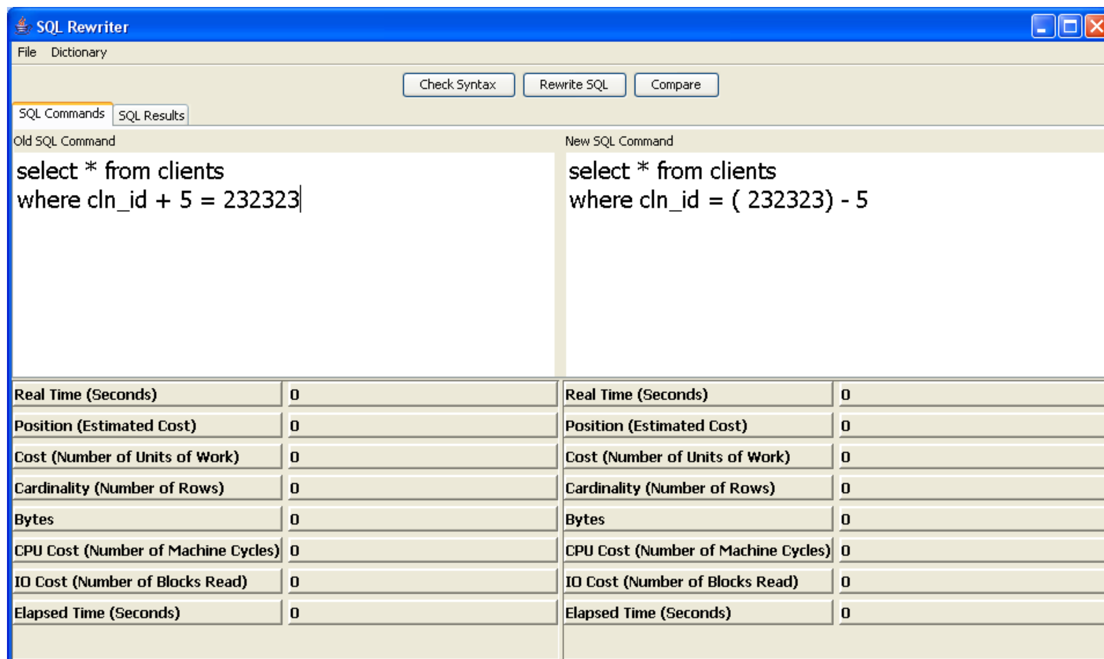


Figure (4-10) Simple Operation Rewriting

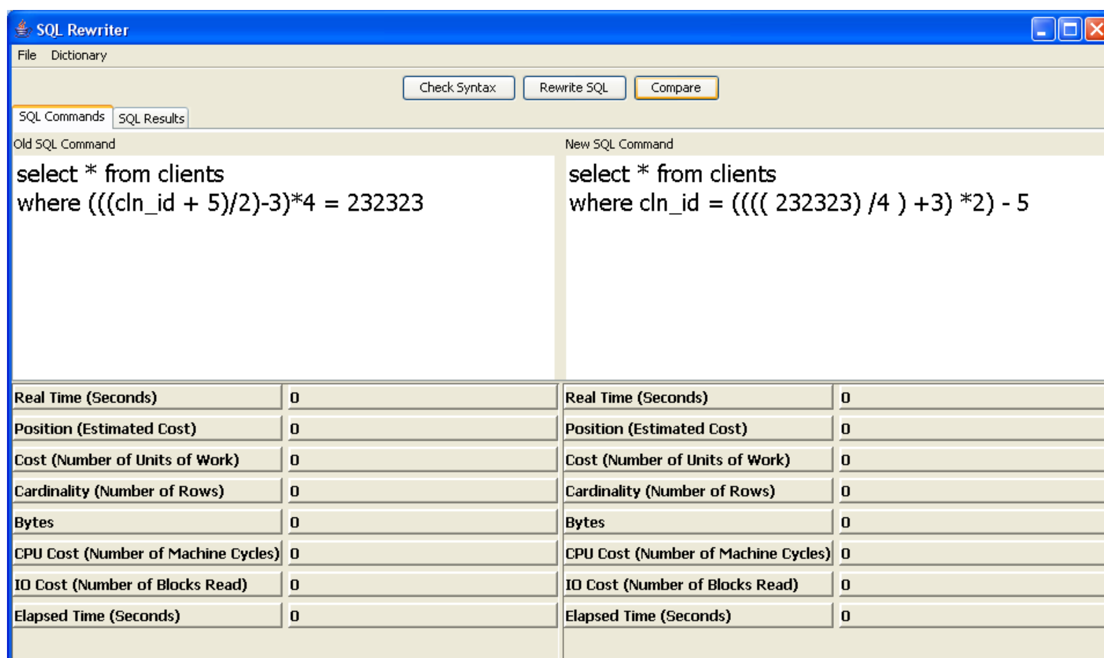


Figure (4-11) Complex Operation Rewriting

Figure (4-12) illustrates a flowchart of this sub-presses followed by the pseudo code to implement it.

المنارة للاستشارات
www.manaraa.com

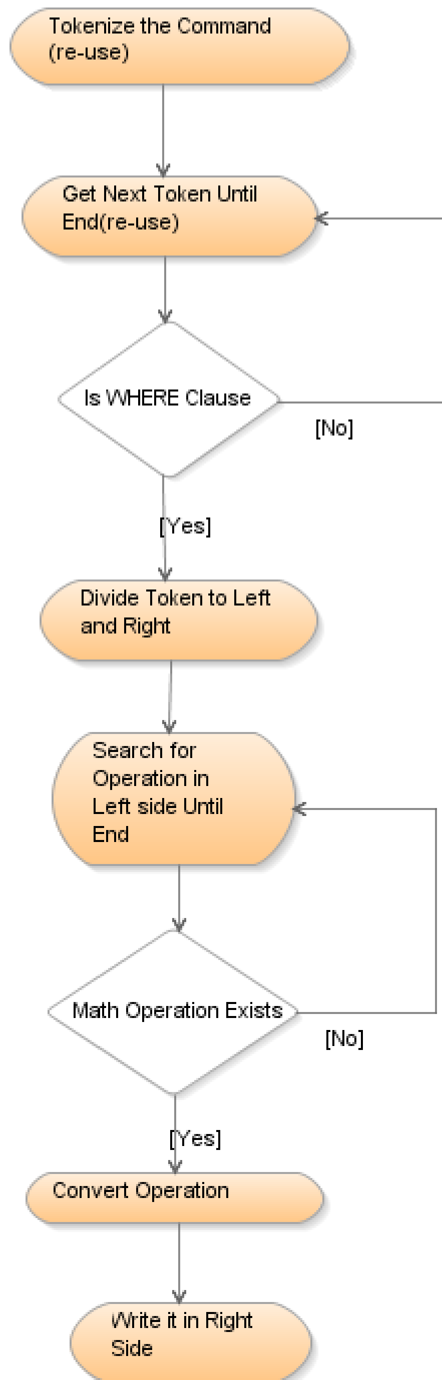


Figure (4-12) Check for Calculation Flowchart

The Pseudo code is shown as follows:

Tokenize the statement

Get next token

If the token is (WHERE)

المنارة للاستشارات
www.manaraa.com

```
        Get next token
        Get left side
        While (left side) loop
            If operation exists
                Convert the operation
                Write it to right side
            End if
        End loop
    End if
```

4.4.4.2 Check for Type Conversion

One of the steps that has to be done in designing the application is defining the type of each column in each table used in the schema, the main data types in database are (NUMBER, VARCHAR, VARCHAR2, DATE, CLOB, BLOB) and others, each type is used in a different way for example when using NUMBER type the value will be mentioned without any extra letters, but when using VARCHAR, VARCHAR2 or DATE the value should be surrounded by single quote ('). For example the WHERE clause will be like this if we use number (WHERE cln_id = 232323) but if we use names the WHERE clause should be like this (WHERE cln_fname = 'ali').

Some times we need to convert the data from type to type, like converting the number to character or character to date and vise versa, the conversion process can be done by using predefined function in database like (TO_CHAR, TO_NUMBER, TO_DATE) and others. When we use any function to convert the type in WHERE clause, the index created in this column will be ignored and

المنارة للاستشارات

www.manaraa.com

the database will make sequential search in the table to get the required row, the interesting thing is some times the database management system do implicit type conversion if there is mismatch between the two parts of the WHERE clause, for example if the WHERE clause is written like this (WHERE cln_phone = 9615811581) and the cln_phone column is defined in database as VARCHAR2, the database management system will convert it to (WHERE TO_NUMBER(cln_phone) = 9615811581) and this step will prevent using the index created in this column, but if the WHERE clause is written like this (WHERE cln_phone = '9615811581') before reaching the database, the database will use the index created in cln_phone column.

The agent has to track this possibility of type conversion and rewrite the SQL command to be free of type conversion functions. The algorithm used in this process is searching for WHERE clause used after tokenizing the statement if not tokenized before, then find the column type in the WHERE clause from agent dictionary, and if the column type does not match with the right side of the WHERE clause, the agent will match it, Figure (4-13) shows us an example of rewriting the statement because cln_phone is defined in database as VARCHAR2 and used as a NUMBER in the WHERE clause (without single quotes), so the agent surrounds the right side value with single quotes to be matched with cln_phone type.

المنارة للاستشارات

www.manaraa.com

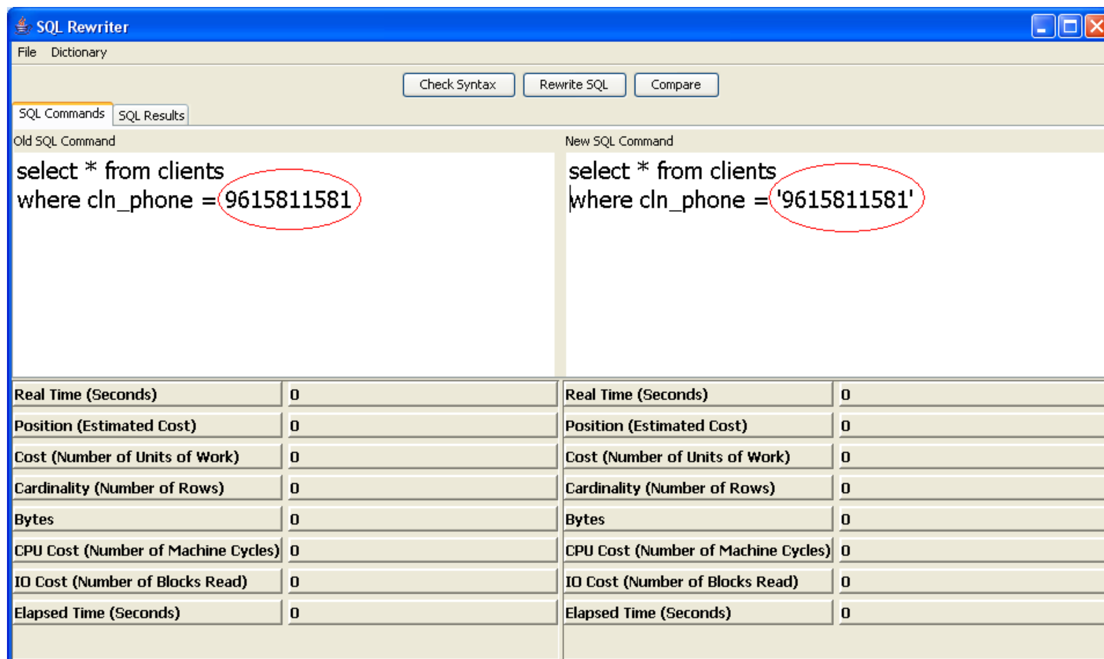


Figure (4-13) Type Conversion Rewriting

Figure (4-14) illustrates a flowchart of type conversion sub-process and the pseudo code to implement it.

The Pseudo code is show as follows:

```
Tokenize the statement
Get next token
If the token is (WHERE)
    Get next token
    Get left side and right side
    While (left side has column) loop
        Get column type
        If (type not matches)
            Convert the right side
        End if
    End loop
End if
```


المنارة للاستشارات
www.manaraa.com

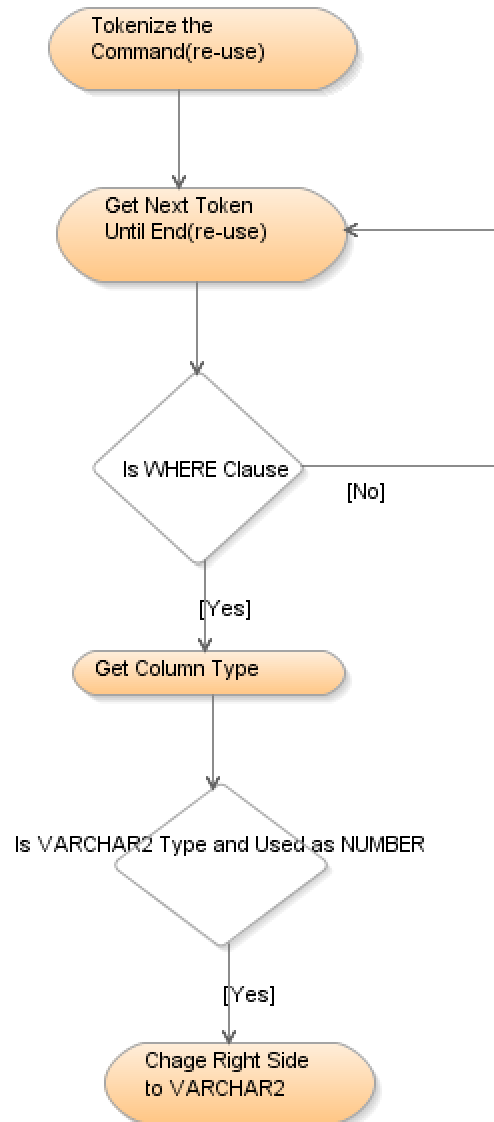


Figure (4-14) Check for Type Conversion Flowchart

4.4.4.3 Check for Index Usage

Using functions are very common procedure when writing SQL statements especially with new versions of database that comes with rich predefined functions and built-in procedures that help the programmer to immediately use these options, but using function in WHERE clause of the statement will prevent index usage and will make sequential search. The agent will try to catch these functions and remove it from the left side of the WHERE clause then

www.manaraa.com

change the right side to match the original statement. For example if the WHERE clause of the statement use the function TO_CHAR to change the date format of a column, the index of this column will not be used as shown in left side of Figure (4-15), but if we change the default format of the date by the command (alter session) then we can write the statement without using the function TO_CHAR and this will make use of the index.

The agent will try to do this process, the algorithm is similar to previous section but with small changes, the agent will tokenize the statement and search for WHERE clause, then search if there are functions used with any column then try to change the statement without this function after setting the default date format.

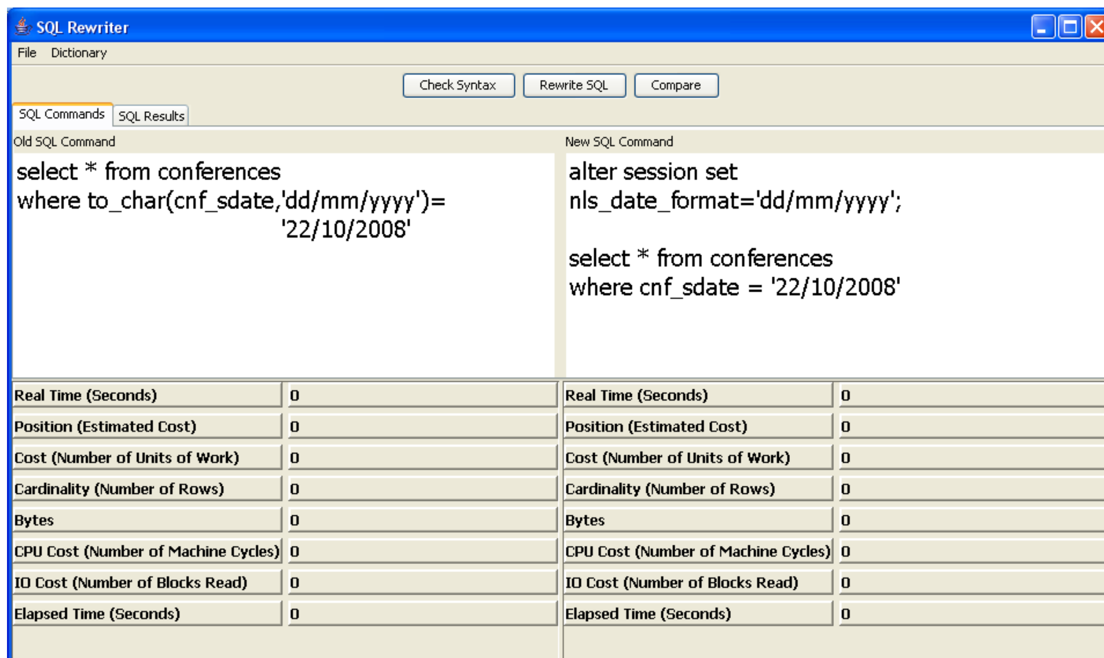


Figure (4-15) Index Usage Rewriting

المنارة للاستشارات
www.manaraa.com

Figure (4-16) illustrates a flowchart of index usage sub-process followed by pseudo code to implement it.

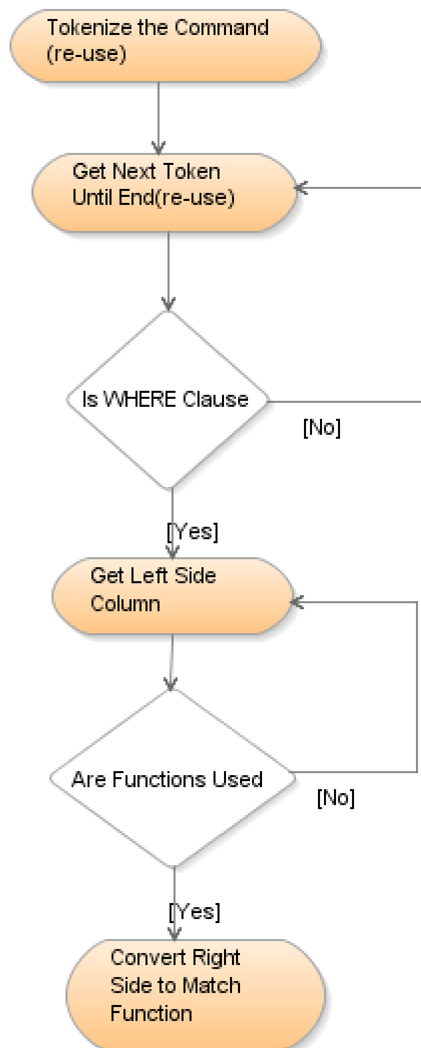


Figure (4-16) Check for Index Usage Flowchart

The Pseudo code is shown as follows:

Tokenize the statement

Get next token

If the token is (WHERE)

Get next token

Get left side and right side

While (left side has function) loop

المنارة للاستشارات
www.manaraa.com

Remove the function

Change session settings

Write command with new settings

End loop

End if

4.4.4.4 Check for Join

The RDBMS can join only two row sources at a time. Join operations (such as nested loops and sort-merge) are used as building blocks if the join statement contains more than two tables. Except for the limitations that are inherent in each kind of join statement, there are no restrictions on the combination of join operations involved in a join with more than two tables.

A join statement is a select statement with more than one table in the FROM clause. A ***join predicate*** is a predicate in the WHERE clause that combines the columns of two of the tables in the join. A ***non join predicate*** is a predicate in the WHERE clause that references only one table. As in the example below the WHERE clause (WHERE CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER) and (AND CNF_CNF_ID = CNFD.CNFD_CNF_ID) are join predicate but (AND CLN_CLN_ID = 2323) is non join predicate.

```
SELECT   *
FROM     CLIENTS CLN,
         CONFERENCES CNF,
         CONFERNCE_DETAILS CNFD
WHERE    CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER
AND      CNF.CNF_ID = CNFD.CNFD_CNF_ID
AND      CLN.CLN_ID = 2323
```


www.manaraa.com

Determining the sequence of joining more than one table is a very important decision, joining two tables like making a nested loop, so each record in the first table (outer or driving table) will be matched with each record in the second table (inner or drive table). The important thing in joining is placing the non join predicate in the first of join order. By making a non join predicate table the driving table of a join operation, the RDBMS effectively reduces join operation. For example, for a nested loop join, the main loop is reduced. The non join predicate results in less rows (or no rows at all), so the inner loop is executed less (or not at all).

Let us take an example with numbers to see how much important selecting the driving table first is. For previous SELECT statement let us consider these facts for three tables as summarized in Table (4-1):

Table (4-1) Tables Summary

	CLIENTS (CLN)	CONFERENCES (CNF)	CONFERENCE_DETAILS (CNFD)
Number of Rows	100,000	1,000,000	10,000,000
Rows for Client ID (2323)	1	10,000	100,000

If the database starts by joining CNF table with CNFD table the join results by (1,000,000 * 10,000,000) 1E+13 loops, in spite of that not all data in CNF and CNFD tables belong to CLIENT number

المنارة للاستشارات
www.manaraa.com

2323 as in the SELECT statement, so the database joining the two tables for all clients then the database will join the resulting data (1E+13) row with CLN table after applying the non join predicate (AND CLN_CLN_ID = 2323) which will result for one record, the total loops will be (1E+13 * 1) for joining the three tables.

But if the database starts by joining CLN table with CNF table after applying the non join predicate (AND CLN_CLN_ID = 2323), the join results by (1 * 10,000) loops, then joining the third table CNFD, the number of loops will be (10,000 * 100,000) 1E+9 loops which is of course less than the previous method.

To make database optimizer take the decision of starting by non join predicate table, the programmer has to send a hint for optimizer. Hints can be written with any SQL command after the first word of the SQL statement, also optimizer hints must be started with (/*+) and end by (*/). As an example of using hints to make the database optimizer starts by joining CLN table and CNF table using nested loop operation as follows:

```
SELECT    /*+USE_NL (CLN CNF)*/    *
FROM      CLIENTS CLN,
          CONFERENCES CNF,
          CONFERNCE_DETAILS CNFD
WHERE     CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER
AND       CNF.CNF_ID = CNFD.CNFD_CNF_ID
AND       CLN.CLN_ID = 2323
```


المنارة للاستشارات
www.manaraa.com

The agent will try to find any SQL statement with join predicate and examine if the statement has non join predicate, the agent will rewrite the statement and send a hint to database optimizer to start with non join predicate table as shown in Figure (4-17):

SQL Rewriter

File Dictionary

Check Syntax | Rewrite SQL | Compare

SQL Commands | SQL Results

Old SQL Command

```
select * from
clients  cln,
conferences  cnf,
conference_details  cnfd
where cln.cln_id = cnf.cnf_cln_id_caller
and cnf.cnf_id = cnfd.cnfd_cnf_id
and cln.cln_id = 2323
```

New SQL Command

```
select /*+use_nl(cln cnf) */ * from
clients  cln,
conferences  cnf,
conference_details  cnfd
where cln.cln_id = cnf.cnf_cln_id_caller
and cnf.cnf_id = cnfd.cnfd_cnf_id
and cln.cln_id = 2323
```

Real Time (Seconds)	18465.562	Real Time (Seconds)	22.592
Position (Estimated Cost)	460035	Position (Estimated Cost)	1030382
Cost (Number of Units of Work)	460035	Cost (Number of Units of Work)	1030382
Cardinality (Number of Rows)	24567975347	Cardinality (Number of Rows)	24567975347
Bytes	47219648616934	Bytes	47219648616934
CPU Cost (Number of Machine Cycles)	2461317156257	CPU Cost (Number of Machine Cycl...	2472584016376
IO Cost (Number of Blocks Read)	37106	IO Cost (Number of Blocks Read)	605517
Elapsed Time (Seconds)	5521	Elapsed Time (Seconds)	12365

Figure (4-17) Check for Join Rewriting

The agent will try to do this process, so the agent will tokenize the statement and searches for WHERE clause, then it will search if there are join predicates and non join predicates. If they exist the agent will add a hint to SQL statement to start with non join predicate. Figure (4-18) illustrates a flowchart of check for join sub-process and the pseudo code to implement it.

The Pseudo code is shown as follows:

Tokenize the statement

Get next token

If the token is (WHERE)

Get next token

المنارة للاستشارات
www.manaraa.com

```
        Search for join predicate
        If (exists)
                Search for non join predicate
                If (exists)
                        Add hint
                End if
        End if
End if
```


المنارة للاستشارات
www.manaraa.com

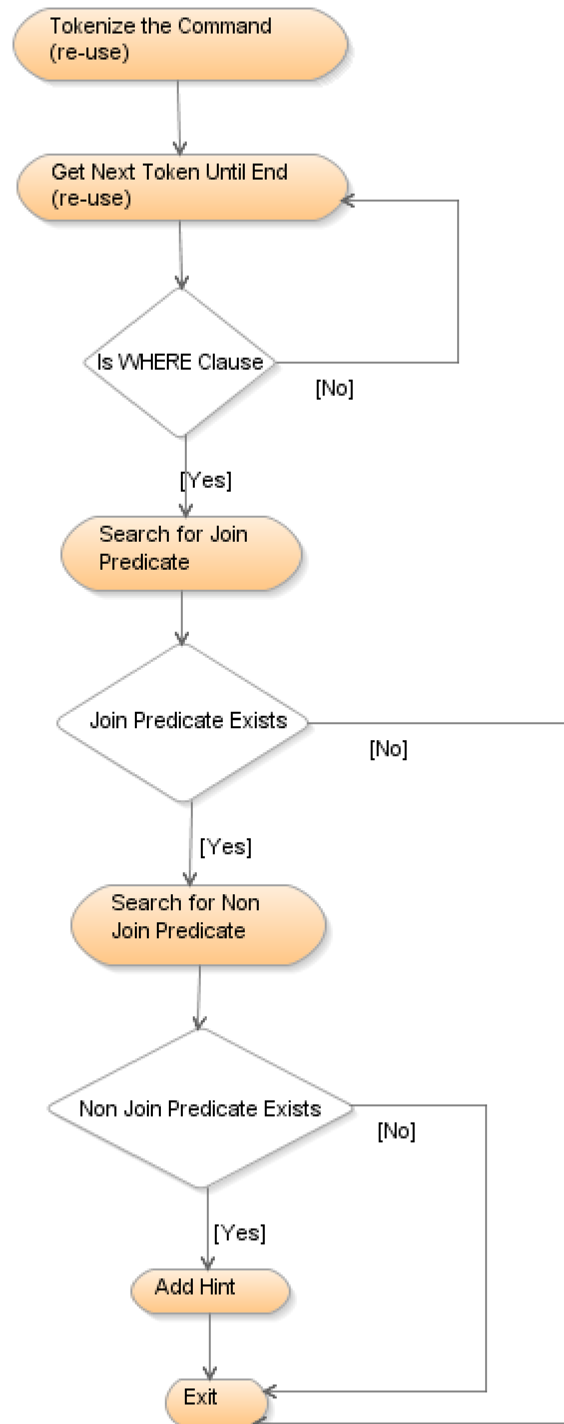


Figure (4-18) Check for Join Flowchart

المنارة للاستشارات
www.manaraa.com

4.4.5 Compare Results Use-case

To measure the efficiency of the agent we have to see the differences between before and after rewriting process. There are many measurement variables that can be extracted from database to evaluate the SQL statement and its behavior, section 5.2 will discuss these variables in details. So the responsibility of this use-case is to read these variables for each SQL statement (before and after rewriting process) and display them at agent interface to be able to save these variables at next stage.

Some metrics can be evaluated without executing SQL command by estimating the time and the number of reads in database, but others can not be evaluated until we execute the command and get real numbers about the command. To get correct results for each command (before and after) the agent executes both commands at the same time to be sure that the two commands have the same environments, because if the agent runs each command separately one after the other there is no guarantee that we have the same environments for both commands, because they may for another task be run when one of the commands is under execution.

To execute two commands in same time, agent has to create two threads – thread is parallel execution line created from the main execution line - then execute two threads in the same time. Each thread will handle one of the commands such as executing the command, reading the results, and evaluating the statistical information and metrics resulting from the command. The button

المنارة للاستشارات
www.manaraa.com

(Compare) in the agent interface will execute the threads and wait for the lowest thread to be finished then the results will be displayed on the screen. Figure (4-19) shows us two SQL statements with comparison process still working (Real Time of the left side is increasing but in the right side is finished), and all other metrics still not filled until the left side SQL statement is finished as shown in Figure (4-20), so all other metrics such as (Position, Cost, Cardinality and etc) are now filled with their values.

SQL Rewriter

File Dictionary

Check Syntax | Rewrite SQL | Compare

SQL Commands | SQL Results

Old SQL Command		New SQL Command	
select * from clients where cln_id + 5 =232323		select * from clients where cln_id = (232323) - 5	
Real Time (Seconds)	12.0	Real Time (Seconds)	0.32
Position (Estimated Cost)	0	Position (Estimated Cost)	0
Cost (Number of Units of Work)	0	Cost (Number of Units of Work)	0
Cardinality (Number of Rows)	0	Cardinality (Number of Rows)	0
Bytes	0	Bytes	0
CPU Cost (Number of Machine Cycles)	0	CPU Cost (Number of Machine Cycles)	0
IO Cost (Number of Blocks Read)	0	IO Cost (Number of Blocks Read)	0
Elapsed Time (Seconds)	0	Elapsed Time (Seconds)	0

Figure (4-19) Comparison in Progress

المنارة للاستشارات
www.manaraa.com

SQL Rewriter

File Dictionary

Check Syntax | Rewrite SQL | Compare

SQL Commands | SQL Results

Old SQL Command	New SQL Command
select * from clients where cln_id + 5 =232323	select * from clients where cln_id = (232323) - 5

	Old		New
Real Time (Seconds)	135.465	Real Time (Seconds)	0.32
Position (Estimated Cost)	88888	Position (Estimated Cost)	3
Cost (Number of Units of Work)	88888	Cost (Number of Units of Work)	3
Cardinality (Number of Rows)	200098	Cardinality (Number of Rows)	1
Bytes	27813622	Bytes	139
CPU Cost (Number of Machine Cycles)	7888070127	CPU Cost (Number of Machine Cycles)	36137
IO Cost (Number of Blocks Read)	87533	IO Cost (Number of Blocks Read)	3
Elapsed Time (Seconds)	1067	Elapsed Time (Seconds)	1

Figure (4-20) Comparison is Finished

In this use-case the agent starts by initializing all variables and metrics to start its comparison, the first step is getting the time which will be considered as start time for both statements, then the two threads will start at the same time, each thread will execute its SQL command and wait for results to be retrieved, after retrieving the results the agent will register the time which will be considered as finish time for each statement, now the agent has the start time and finish time so it can easily evaluate the time spent by each SQL statement. After that the agent will select the statistics generated by each command and finally display all of the results (the spent time and statistics) for both commands. Figure (4-21) illustrates a flowchart of compare results use-case followed by pseudo code that implements it.

المنارة للاستشارات
www.manaraa.com

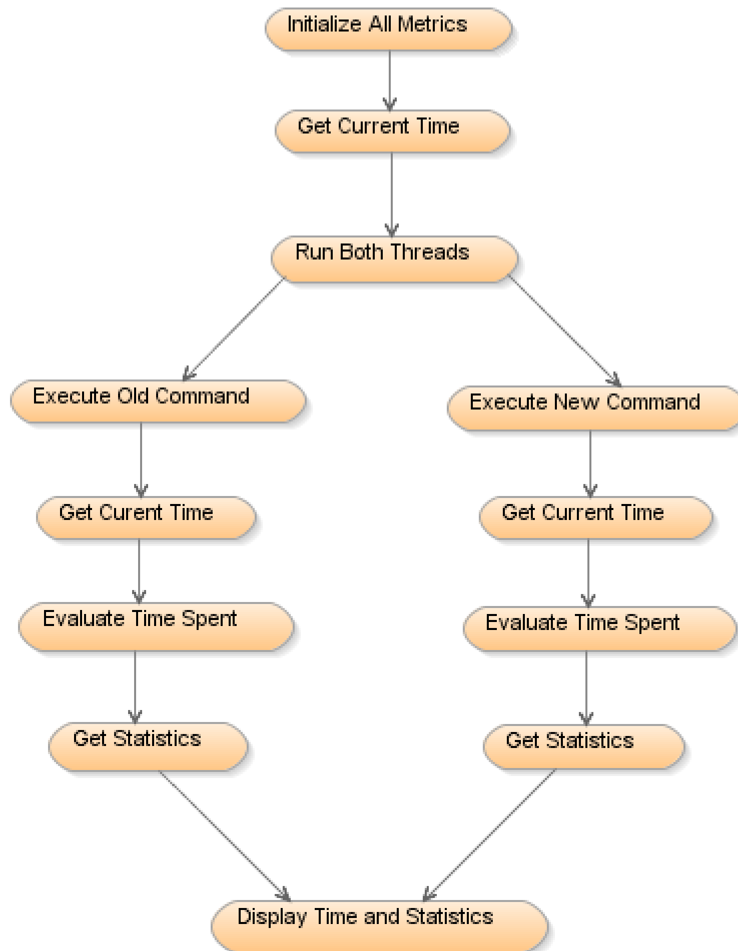


Figure (4-21) Compare Results Flowchart

The Pseudo code is show as follows:

Initialize all variables

Get start time

Run thread procedure (one)

Run thread procedure (two)

Display time spent and statistics

Thread procedure

> **Execute SQL command**
>
> **Get end time**
>
> **Time spent = end time – start time**

المنارة للاستشارات
www.manaraa.com

Select statistics

4.4.6 Save Results to History Use-case

To enhance the reusability and efficiency of the agent, the agent saves all metrics and statistics generated by compare use-case in a file system to be referenced in the future. All of the results will be saved in XML file format, so we can reference these files any time we need. The XML file shown in Figure (4-22) illustrates the statistics saved by a comparison process.

```
  <?xml version="1.0" ?>
- <comparison id="37">
  - <oldSQL>
      <command>select * from clients where cln_id + 5 = 232323</command>
      <real_time>147.512</real_time>
      <position>88888</position>
      <cost>88888</cost>
      <cardinality>200098</cardinality>
      <bytes>27813622</bytes>
      <cpu_cost>7888070127</cpu_cost>
      <io_cost>87533</io_cost>
      <time>1067</time>
    </oldSQL>
  - <newSQL>
      <command>select * from clients where cln_id = (232323) - 5</command>
      <real_time>0.892</real_time>
      <position>3</position>
      <cost>3</cost>
      <cardinality>1</cardinality>
      <bytes>139</bytes>
      <cpu_cost>36137</cpu_cost>
      <io_cost>3</io_cost>
      <time>1</time>
    </newSQL>
  </comparison>
```

Figure (4-22) Comparison XML File

The agent responsibility in this use-case is simple, the agent has to initialize the XML file then read the statistics which is calculated before then write them to the file and close the connection. Figure (4-23) shows us flowchart of save results use-case followed by pseudo code that implements it.

المنارة للاستشارات
www.manaraa.com

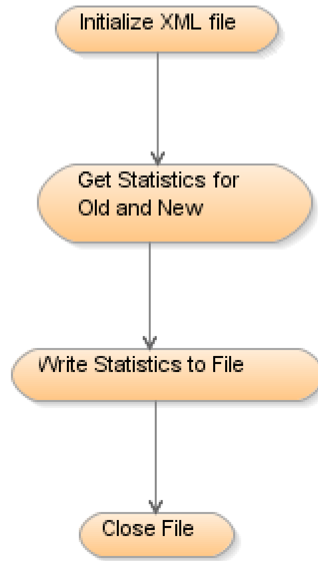


Figure (4-23) Save Results Flowchart

The Pseudo code is shown as follows:

Initialize XML file

Read statistics

Write statistics

Close file

المنارة للاستشارات
www.manaraa.com

Chapter Five
Agent Performance Analysis

5.1 Overview

In this chapter the performance measurement of the agent is analyzed and a comparison between old SQL statements with new SQL will be discussed in details. This chapter will show a technical detail about performance measurement metrics. Then a comparison between old measures with new measures will be shown to see the achievement of the agent.

5.2 Performance Analysis Measurements

To have a good comparison between old statement and new statement the researcher has to select the required measurement metrics to reflect the changes that the agent achieved by rewriting the SQL statement. Performance measurement variables include real time spent for statement execution, position, cost, cardinality, bytes, CPU cost, IO cost, and elapsed time.

Real time: The exact number of seconds needed to execute the statement; it is measured by subtracting the start time of statement execution from finish time.

Cost: The estimate number of standardized inputs/outputs (I/Os) it takes to execute the statement. The standard cost metric measured by the optimizer is in terms of number of single block random reads, so one cost unit corresponds to one single block random read. Also the cost includes CPU costing because in most cases CPU utilization is as important as I/O; often it is the only contribution to the cost.

المنارة للاستشارات
www.manaraa.com

Position: The optimizer's estimated cost of executing the statement. So most of time position value will be equal to cost value.

Bytes: Number of bytes accessed by the operation. The bytes touched by the RDBMS to execute the statement.

CPU Cost: The number of machine cycles required for the operation, it includes CPU cost of query processing (pure CPU cost) and CPU cost of data retrieval (CPU cost of the buffer cache get).

IO Cost: The number of data blocks read by the operation, it includes single-block and multi-block reads.

Elapsed Time: The estimated number of seconds needed to execute the statement, it is an estimation calculated before executing the statement.

Any SQL command generates several statistics; the most important statistics are the following:

Recursive calls: Number of SQL statements done by RDBMS against the database dictionary to enable him to execute the client SQL statement, for example retrieving table names, column names, privileges, data format from dictionary to be able to run the client SQL statement.

DB block gets: Number of logical I/Os for current statement, it is the number of current blocks fetched in the memory. This means that this block has been processed before.

المنارة للاستشارات

www.manaraa.com

Consistent gets: Reads of buffer cache blocks from memory that have undo data, so it is fetched from memory because the modified data is not saved yet, and this process makes the RDBMS gets read consistency image of the data.

Physical reads: Number of blocks read from disk. So it is the first time the RDBMS fetches this block of data.

Redo size: Amount of redo generated (for DML statements). The RDBMS saves the old values for any DML statement to bring it back if the client rolls back his transaction (undo the transaction).

SQL*Net sent: Number of bytes sent from server to client through the network.

SQL*Net received: Number of bytes sent from client to server through the network.

SQL*Net roundtrips: Number of roundtrip packets sent from server to client or client to server through the network.

Sorts (memory): Number of sorts performed in memory if the SQL statement contains order by clause.

Sorts (disk): Number of sorts performed using temporary disk storage, this can be happen if the sort can not be finished in the memory (memory is not enough) and has to be completed in the disk.

المنارة للاستشارات

www.manaraa.com

Rows processed: Number of records that have been fetched from memory or disk, sorted, manipulated, filtered, or the RDBMS make any operation on these rows.

From previous measurement variables and other statistics we can have a good image of agent performance and achievement.

5.3 Calculations, Type Conversion, and Index Usage Benchmarks

This section presents the tests done by the agent and the performance measurements of old SQL statement before rewriting compared with new SQL statement after rewriting. This section will discuss three types of rewriting as mentioned in previous chapters. Rewriting SQL command with new structure taking into consideration to make the three recommendations:

- Remove the mathematical operations from the left side of the statement WHERE clause and rewrite it in the right side of the statement.
- Remove and implicit type conversion so the RDBMS will not do any hidden type conversion. And the RDBMS will execute the statement as it is.
- Try any possibility to make the SQL statement to use index instead of using full table scan.

If we implement the above recommendations, the database optimizer will use the index created on the table instead of making full table scan. And as discussed in Chapter 1 section 1.6.2 implementing the above recommendations will change the execution of SQL statement from sequential search to binary search.

المنارة للاستشارات
www.manaraa.com

5.3.1 Experimental Scenarios

To have a complete view of the difference that the agent has done by converting the command, the researcher has to execute the SQL command before rewriting process, then collect all metrics and statistics related to this command, after that he has to execute the SQL command after the rewriting process to collect the same metrics and statistics, then compare all of these outputs.

The two experimental scenarios were considered to evaluate and compare performance when running SQL commands that uses full table scan before rewriting and SQL commands uses index after rewriting. These scenarios were repeated several times during experiments with different number of rows in the tables.

First Scenario: Before Rewrite

In this scenario the researcher sent SQL statements to the agent with WHERE clause that makes the database optimizer to make full table scan decision, then collect the output generated from the agent, the following SQL statements are just an example of statements sent to agent:

- SELECT * FROM CLIENTS
  WHERE
  CLN_ID + 5 = 232323;

- SELECT * FROM CLIENTS
  WHERE
  ABS(CLN_ID) = 232323;

المنارة للاستشارات

www.manaraa.com

- SELECT * FROM CLIENTS
  WHERE
  CLN_PHONE = 9615811581;

- SELECT * FROM CLIENTS
  WHERE
  TO_NUMBER(CLN_PHONE) = 9615811581;
- SELECT * FROM CONFERENCES
  WHERE
  TO_CHAR(CNF_SDATE,'MM/YYYY') = '10/2008';
- SELECT * FROM CONFERENCES
  WHERE
  TRUNC(CNF_SDATE) = '22/10/2008';

The above SQL statements are very normal and common statements used by programmers, but all of them will prevent the index to be used because of the structure of the WHERE clause.

Second Scenario: After Rewrite

In this scenario the researcher watched the transformed SQL commands generated by the agent which makes the database optimizer to use index search instead of full table scan, then collect the output generated from the agent, the following SQL statements are example of statements generated from the agent:

- SELECT * FROM CLIENTS
  WHERE
  CLN_ID = 232323 - 5;

المنارة للاستشارات
www.manaraa.com

- SELECT * FROM CLIENTS
  WHERE
  CLN_ID = 232323 OR
  CLN_ID = -232323

- SELECT * FROM CLIENTS
  WHERE
  CLN_PHONE = '9615811581';

- SELECT * FROM CLIENTS
  WHERE
  CLN_PHONE = '9615811581';

- ALTER SESSION SET
  NLS_DATE_FORMAT='MM/YYYY';

- SELECT * FROM CONFERENCES
  WHERE
  CNF_SDATE = '10/2008';

- ALTER SESSION SET
  NLS_DATE_FORMAT='DD/MM/YYYY';

- SELECT * FROM CONFERENCES
  WHERE
  CNF_SDATE = '22/10/2008';

المنارة للاستشارات
www.manaraa.com

The above SQL statements are generated from the agent as a result of the SQL statement sent to agent in the first scenario in the order. So we can match the first statement in first scenario with first statement in second scenario and so on to see how the WHERE clause was rewritten to use the index search feature.

5.3.2 Variables Discipline

To control the test, the following variables are disciplined and controlled to make minimum effect on the test setup:

1- System Load: only the SQL statement was executed in the database and it is the only session connected to database. So we insured that the full capabilities of the database were focused in executing the required SQL statement.

2- Network Load: only one client was connected to the database server so the full network bandwidth was free for agent use.

3- Caching: a built in database function used to clear the cache each time the SQL statement submitted to database.

5.3.3 Measurement Benchmark

The benchmark depends on the number of rows that the table holds, so the two scenarios were repeated with different number of rows in the tables. Table (5-1) shows the number of rows in the table when the run is executed and the output metrics are generated for each run. Table (5-2) shows number of rows in the table when the run is executed and the output statistics are generated for each run.

المنارة للاستشارات
www.manaraa.com

Table (5-1) Metrics for Different Row Number

Number of Rows in Table										
Rows	1.0E+06		5.0E+06		1.0E+07		1.5E+07		2.0E+07	
Metrics	Old	New	Old	New	Old	New	Old	New	Old	New
Real Time (Seconds)	7.52	0.16	40.76	0.16	74.98	0.16	103.27	0.17	133.96	0.35
Postition (Estimated Cost)	4213	3	22045	3	44299	3	67467	3	88888	3
Cost (Units of Work)	4213	3	22045	3	44299	3	67467	3	88888	3
Cardinality (Number of Rows)	1	1	1	1	1	1	1	1	200098	1
Bytes	652	652	652	652	652	652	652	652	28(6)	139
CPU Cost (Machine Cycles)	4E+8	36E+3	2E+9	36E+3	4E+9	36E+3	6E+9	36E+3	8E+9	36E+3
IO Cost (Blocks Read)	4144	3	21712	3	43631	3	66438	3	87533	3
Elapsed Time (Seconds)	51	1	265	1	532	1	810	1	1067	1

المنارة للاستشارات
www.manaraa.com

Table (5-2) Statistics for Different Row Number

Number of Rows in Table										
Rows	1.0E+06		5.0E+06		1.0E+07		1.5E+07		2.0E+07	
Statistics	Old	New	Old	New	Old	New	Old	New	Old	New
Recursive Calls (Number of SQL)	4	1	4	1	4	1	4	1	5	1
DB Block Gets (Number of IO)	0	0	0	0	0	0	0	0	0	0
Consistent Gets (Number of Buffer)	18849	4	98859	4	198876	4	303040	4	407205	4
Physical Reads (Number of Blocks)	18216	0	98625	0	198511	0	301894	0	407089	0
Redo Size (Number of Blocks)	0	0	0	0	0	0	0	0	0	0

المنارة للاستشارات
www.manaraa.com

Bytes Sent (Bytes)	1069	977	106 9	977	1069	977	1069	977	1069	977
Bytes Received (Bytes)	385	374	385	374	385	374	385	374	385	374
Net. Roundtrips (Count)	2	1	2	1	2	1	2	1	2	1
Sorts (Memory) (Count)	1	0	1	0	1	0	1	0	1	0
Sorts (Disk) (Count)	0	0	0	0	0	0	0	0	0	0

5.3.4 Results Comparison

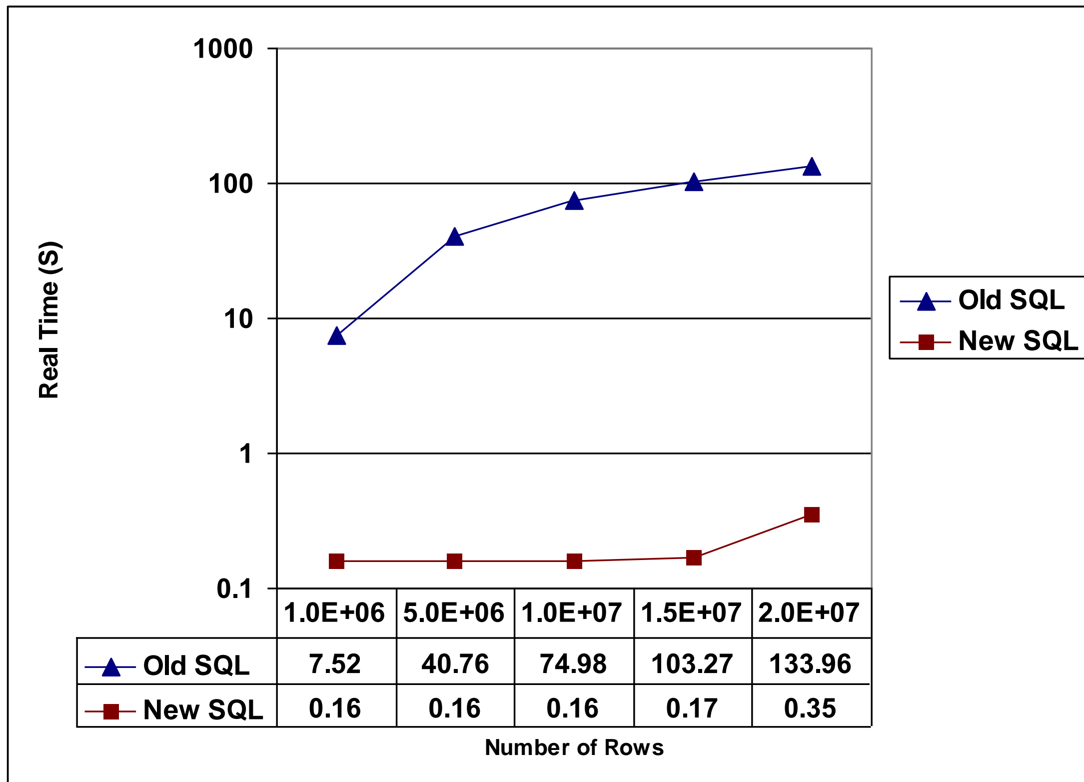
This section presents a comparison of the benchmark tests results by charts and discusses these results for certain measurements such as (Real Time, CPU Cost, IO Cost, and Consistent Gets).

5.3.4.1 Real Time Test Results Comparison

SQL command real time execution is the most important measurement used to evaluate the performance between old command and new command. Figure (5-1) illustrates a logarithmic chart comparison.

المنارة للاستشارات

www.manaraa.com



	1.0E+06	5.0E+06	1.0E+07	1.5E+07	2.0E+07
Old SQL	7.52	40.76	74.98	103.27	133.96
New SQL	0.16	0.16	0.16	0.17	0.35

Figure (5-1) Logarithmic Chart for Real Time Results Comparison

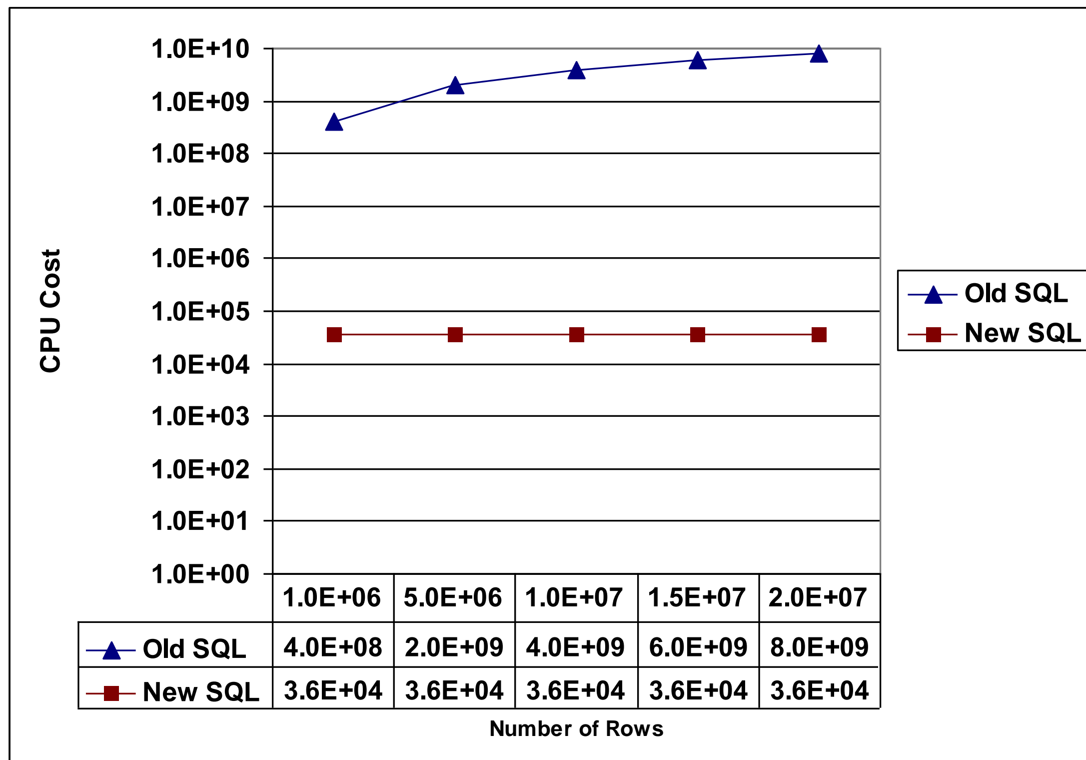
The real time results plotted as logarithmic chart have been analyzed. The comparisons are as follows:

1- The real time coefficient for old SQL commands increased dramatically according to number of rows in the table, because the old SQL commands use sequential search which depends on the size of the table and the RDBMS has to go through each record in the table.

2- The real time coefficient for new SQL commands increased slightly with increasing number of rows, because the new SQL command uses binary search.

المنارة للاستشارات
www.manaraa.com

5.3.4.2 CPU Cost Test Results Comparison



	1.0E+06	5.0E+06	1.0E+07	1.5E+07	2.0E+07
Old SQL	4.0E+08	2.0E+09	4.0E+09	6.0E+09	8.0E+09
New SQL	3.6E+04	3.6E+04	3.6E+04	3.6E+04	3.6E+04

Figure (5-2) Logarithmic Chart for CPU Cost Results Comparison

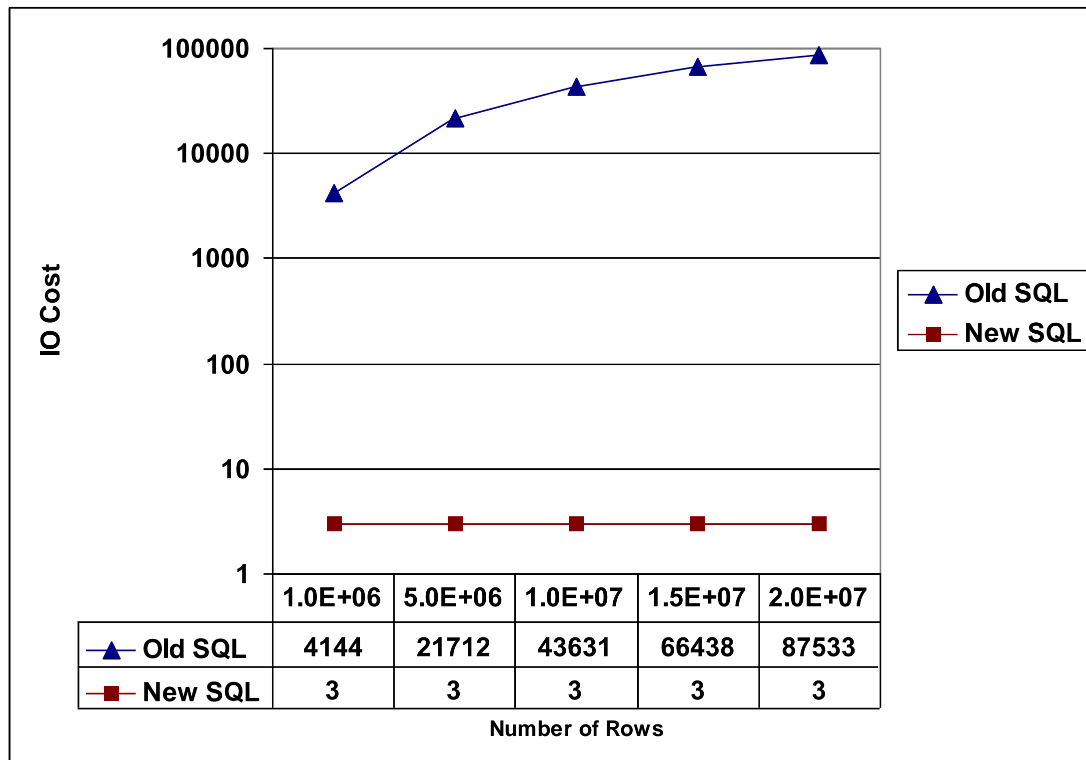
The CPU cost results plotted as logarithmic chart have been analyzed. The comparisons are as follows:

1- Old SQL uses sequential search and this causes an extra overhead on the CPU to process each record in the table, so the CPU cost will be increased by increasing the number of rows in the table.

2- New SQL uses binary search, with binary search there is just one extra process if the data inside the table is duplicated, so the CPU cost will be approximately the same by increasing the data inside the table.

المنارة للاستشارات
www.manaraa.com

5.3.4.3 IO Cost Test Results Comparison



	1.0E+06	5.0E+06	1.0E+07	1.5E+07	2.0E+07
Old SQL	4144	21712	43631	66438	87533
New SQL	3	3	3	3	3

Figure (5-3) Logarithmic Chart for IO Cost Results Comparison

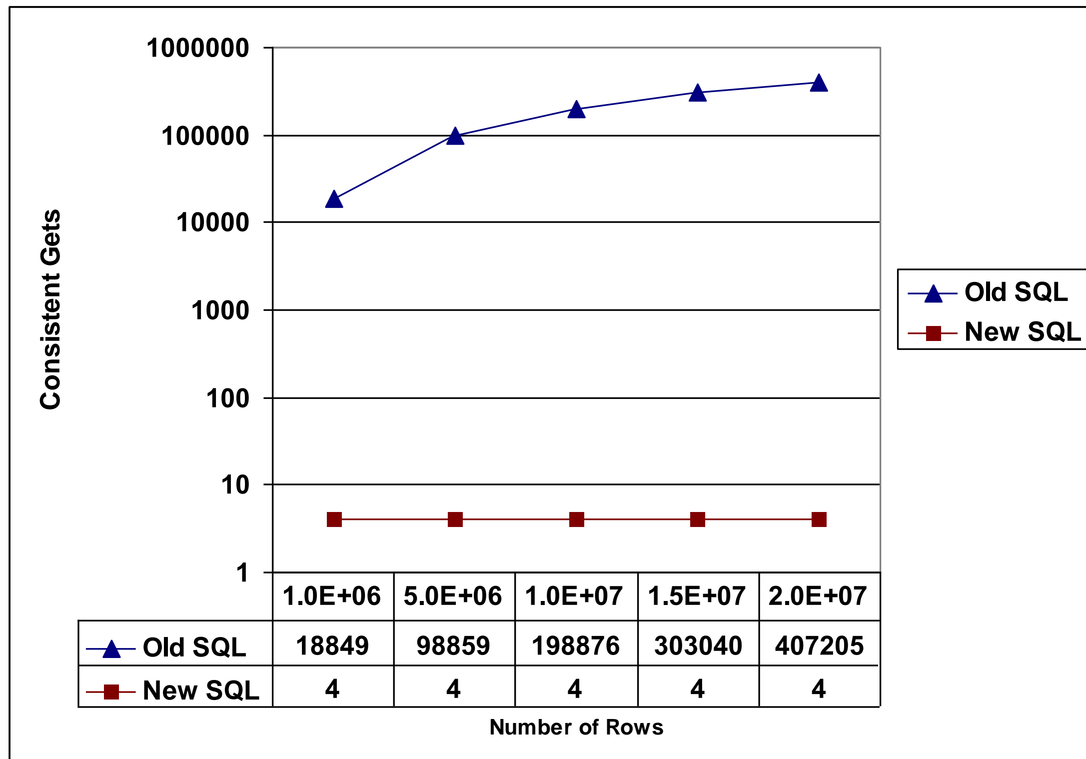
The IO cost results plotted as logarithmic chart have been analyzed. The comparisons are as follows:

1- As CPU cost, the old SQL uses sequential search and this causes extra overhead on the IO to process each record in the table, so the IO cost will be increased by increasing the number of rows in the table.

2- As CPU cost, new SQL uses binary search, so the IO cost will be approximately the same by increasing the data inside the table.

المنارة للاستشارات
www.manaraa.com

5.3.4.4 Consistent Gets Test Results Comparison



	1.0E+06	5.0E+06	1.0E+07	1.5E+07	2.0E+07
Old SQL	18849	98859	198876	303040	407205
New SQL	4	4	4	4	4

Figure (5-4) Logarithmic Chart for Consistent Gets Results Comparison

The consistent gets results plotted as logarithmic chart have been analyzed. The comparisons are as follows:

1- As CPU cost and IO cost, the old SQL uses sequential search, so the consistent gets will be increased by increasing the number of rows in the table.

2- As CPU cost and IO cost, new SQL uses binary search, so the consistent gets will be approximately the same by increasing the data inside the table.

المنارة للاستشارات
www.manaraa.com

5.4 Join Benchmarks

This section presents the tests done by the agent and the performance measurements of old SQL statement before rewriting compared with new SQL statement after rewriting. This section will discuss rewriting of join statements as mentioned in previous chapters. Rewriting SQL command with new structure taking into consideration to make none join predicate table as the driving table.

5.4.1 Experimental Scenarios

To see the achievement done by the agent when converting the command, we have to execute the SQL command before rewriting process, then collect all metrics and statistics related to this command, after that we have to execute the SQL command after the rewriting process to collect the same metrics and statistics, then compare all of these outputs.

The two experimental scenarios were considered to evaluate and compare performance when running join statements without any attention if the none join predicate will be used at the first or not. Then watch the agent how it will convert the join statement to use none join predicate as a driving table.

First Scenario: Before Rewrite

In this scenario we sent join SQL statements to the agent without any attention of none join predicate, the following SQL statements are just an example of this type of statements:

المنارة للاستشارات
www.manaraa.com

- SELECT * FROM
  COUNTRIES CON, CLIENTS CLN, CONFERENCES CNF
  WHERE CON.CON_ID = CLN.CLN_CON_ID
  AND CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER
  AND CON.CON_NAME = 'JORDAN';

- SELECT * FROM
  CLIENTS CLN, CONFERENCES CNF, CONFERENCE_DETAILS CNFD
  WHERE CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER
  AND CNF.CNF_ID = CNFD.CNFD_CNF_ID
  AND CLN.CLN_ID = 2323;

The above SQL statements are very normal and common statements used by programmers, but all of them use none join predicate as the last line of the statement.

Second Scenario: After Rewrite

In this scenario we watched the transformed SQL commands generated by the agent which makes none predicate join as a driving table:

- SELECT /*+USE_NL(CON CLN) */ * FROM
  COUNTRIES CON, CLIENTS CLN, CONFERENCES CNF
  WHERE CON.CON_ID = CLN.CLN_CON_ID
  AND CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER
  AND CON.CON_NAME = 'JORDAN';

المنارة للاستشارات
www.manaraa.com

```
- SELECT /*+USE_NL(CLN CNF) */ * FROM
  CLIENTS CLN, CONFERENCES CNF,
  CONFERENCE_DETAILS CNFD
  WHERE CLN.CLN_ID = CNF.CNF_CLN_ID_CALLER
  AND   CNF.CNF_ID = CNFD.CNFD_CNF_ID
  AND   CLN.CLN_ID = 2323;
```

The above SQL statements are generated from the agent as a result of the SQL statement sent to agent in the first scenario in the order.

5.4.2 Variables Discipline

To control the test, the following variables are disciplined and controlled to make minimum effect on the test setup:

1- System Load: only the SQL statement was executed in the database and it is the only session connected to database. So we insured that the full capabilities of the database were focused in executing the required SQL statement.

2- Network Load: only one client was connected to the database server so the full network bandwidth was free for agent use.

3- Caching: a built in database function used to clear the cache each time the SQL statement is submitted to database.

المنارة للاستشارات
www.manaraa.com

5.4.3 Measurement Benchmark

The benchmark depends on two things:

1- The number of rows that the table holds.

2- The number of rows that will be filtered by none join predicate.

So the two scenarios ware repeated with different numbers of rows for three tables and different numbers of rows filtered by none join predicate. Table (5-3) shows the total number of rows and filtered rows in the three tables when the run is executed and the output metrics are generated for each run. Table (5-4) shows the total number of rows and filtered rows in the three tables when the run executed and the output statistics generated for each run.

Table (5-3) Metrics for Join

Number of Rows for Three Joined Tables						
Total Rows	1E+12		1E+15		1E+18	
Filtered Row	1*100*1000		1*1000*10000		1*10000*100000	
Metrics	Old	New	Old	New	Old	New
Real Time (Seconds)	2.33	1.00	76.17	1.10	18465.56	22.59
Postition (Estimated Cost)	5984	385	41835	3736	460035	1030382
Cost (Units of Work)	5984	385	41835	3736	460035	1030382
Cardinality (Number of Rows)	93966	93966	794461	794461	24E+9	24E+9
Bytes	18E+7	18E+7	15E+8	15E+8	47E+12	47E+12

المنارة للاستشارات
www.manaraa.com

CPU Cost (Machine Cycles)	69E+7	56E+6	44E+8	48E+7	24E+11	24E+11
IO Cost (Blocks Read)	5866	375	41068	3653	37106	37106
Elapsed Time (Seconds)	72	5	503	45	5521	12365

Table (5-4) Statistics for Join

Number of Rows for Three Joined Tables						
Total Rows	1E+12		1E+15		1E+18	
Filtered Rows	1*100*1000		1*1000*10000		1*10000*100000	
Statistics	Old	New	Old	New	Old	New
Recursive Calls (Number of SQL)	0	0	7	7	576	576
DB Block Gets (Number of IO)	0	0	0	0	0	0
Consistent Gets (Number of Buffer)	154116	1656	15E+6	83277	20E+8	826345
Physical Reads (Number of Blocks)	0	0	0	0	1665661	1665661

المنارة للاستشارات

www.manaraa.com

Redo Size (Number of Blocks)	0	0	0	0	0	0
Bytes Sent (Bytes)	1975	1975	10E+7	10E+7	93E+7	93E+7
Bytes Received (Bytes)	374	374	733711	733711	7333711	7333711
Net. Roundtrips (Count)	1	1	66668	66668	666668	666668
Sorts (Memory) (Count)	0	0	2	2	4	4
Sorts (Disk) (Count)	0	0	0	0	0	0

5.4.4 Results Comparison

This section presents a comparison of the benchmark tests results by charts and discusses these results for certain measurements such as (Real Time, Consistent Gets).

5.4.4.1 Real Time Test Results Comparison

SQL command real time execution is the most important measurement used to evaluate the performance between old command and new command. Figure (5-5) illustrates a logarithmic chart comparison.

المنارة للاستشارات
www.manaraa.com

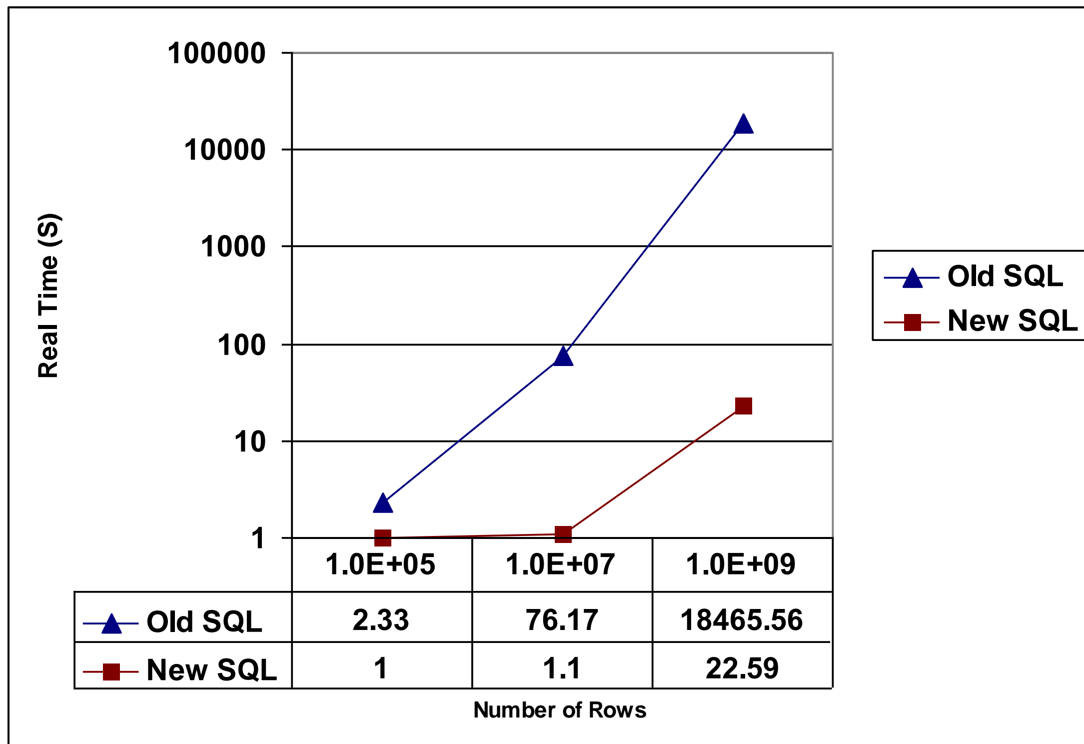


Figure (5-5) Logarithmic Chart for Real Time Results Comparison

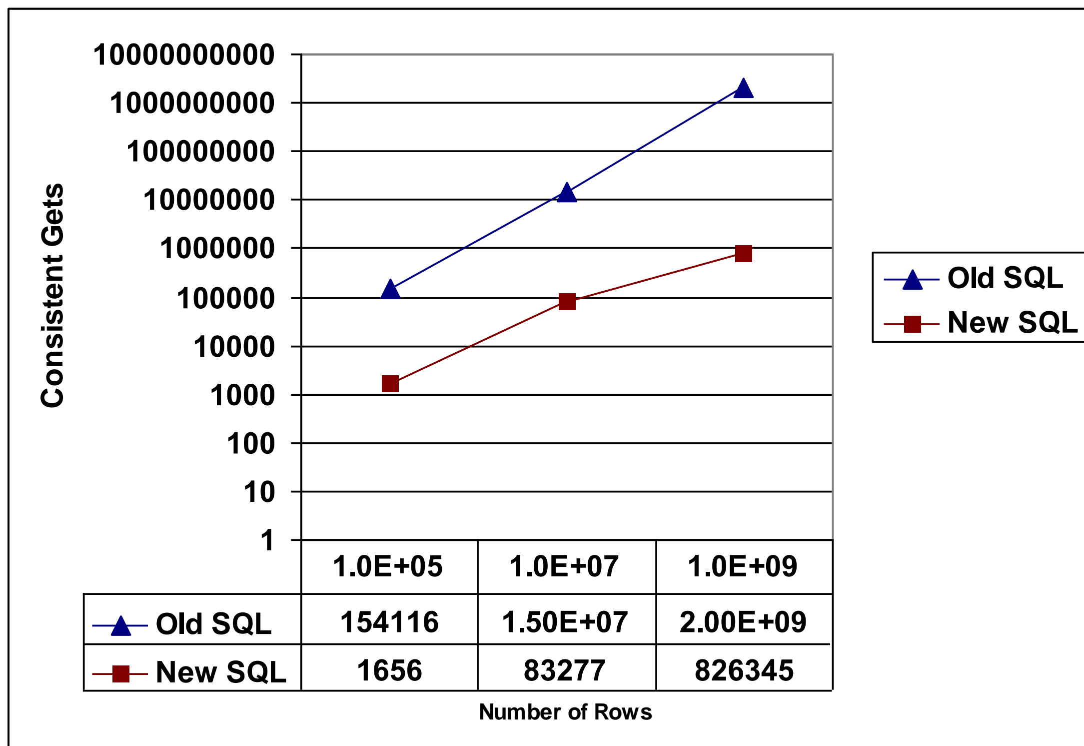
The real time results plotted as logarithmic chart have been analyzed. The comparisons are as follows:

1- The real time coefficient for old SQL commands increased dramatically according to number of rows in the table, because the old SQL will process the multiplication of all rows in the three joined tables.

2- The real time coefficient for new SQL commands increased slightly with increasing number of rows, because the new SQL will process just the filtered rows from three joined tables.

المنارة للاستشارات
www.manaraa.com

5.4.4.2 Consistent Gets Test Results Comparison



	1.0E+05	1.0E+07	1.0E+09
Old SQL	154116	1.50E+07	2.00E+09
New SQL	1656	83277	826345

Figure (5-6) Logarithmic Chart for Consistent Gets Results Comparison

The consistent gets results plotted as logarithmic chart have been analyzed. The comparisons are as follows:

1- The number of rows processed by old SQL statement is very large so the consistent gets will fetch all rows in the three joined tables.

2- The number of rows processed by new SQL statement is small compared to old SQL statement, so the fetch size will be minimized.

المنارة للاستشارات
www.manaraa.com

5.5 Overhead Has No Influence

In this section the researcher will discuss if the agent will do an overhead for running system. There is no doubt that the agent will do some overhead on the system, but the question is whether this overhead will affect the overall performance of the running system or it will not affect at all.

To answer this question, we have to make some tests and calculations to get real numbers of the overhead done by the agent, and compare these numbers with the time that the agent saves by rewriting the SQL statements. If we examine the agent processes we will find that the online operations done for each command are:

1- Check syntax
2- Check for rewrite
3- Rewrite the SQL statements

And the other operations like login, validate dictionary are done off line and once at agent startup.

By testing each online process and how much time it takes, we found that the total time for three steps does not exceed millisecond, so if we compare this with the time saved from rewriting (from Table (5-1) the third column), the time will be 74.82 (74.98 – 0.16) second, this is the time saved by the agent when executing the new command. By doing simple calculation, we need to optimize just one statement from 74820 (74.82 * 1000) SQL statement to be equal. Of course if we do the same calculations in join large data

المنارة للاستشارات
www.manaraa.com

statements (Table (5-3) the third column) we find that we need to optimize just one statement from ((18465.56 – 22.59) * 1000) 18442410 statement.

5.6 Statistics Limitations

As mentioned in previous sections, all metrics and statistics are gathered from the database that the test is performed on. And of course there are some limitations generated from database such as accuracy, dependency, and rounding factors. These limitations can be noticed by examining table (5-1) through table (5-4), and we can find some equal numbers generated for different numbers of rows inside the database tables. For example we have the same real time (0.16 second) for new SQL command from 1.0E+06 until 1.0E+07 row inside the database table as in table (5-1). This constant number should be increased by increasing the number of rows inside the database table, but using binary search will minimize this increment because the RDBMS will exclude the new added data from first hit, and of course this will not happen when using sequential search. Another cause is statistics limitations and number rounding functions that generate the same result for different set of rows.

المنارة للاستشارات
www.manaraa.com

Chapter Six
Conclusion and Future Work

6.1 Overview

The aim of this thesis was to develop an agent to rewrite SQL commands with new syntax with much better performance compared with old syntax.

The proposed agent has been implemented and tested. The performance analysis for new SQL commands are compared with old commands in previous chapters. This chapter includes the conclusion of all parts of the thesis, recommendations for future work and suggestions for studies on the development of performance analysis.

6.2 Thesis Conclusion

This section revises parts of this research including the introduction, background and related works, proposed system design, it includes Furthermore it includes the development and performance analysis of the proposed system solution.

6.2.1 Performance

Performance enhancement acquired a major role in database applications especially when the researcher is talking about data warehouse applications or online applications that require fast user response. Enhancements were previously done in the database side, using different optimizer techniques after accepting the SQL command. But this thesis focuses on the area that comes before the database, so this thesis tries to achieve the performance before the SQL command reaches the database to be executed.

المنارة للاستشارات

www.manaraa.com

6.2.2 Performance Enhancements

Most of the researches focus in depth on the area of SQL command performance enhancements, and most of them reached very good conclusions. Some of them focused on enhancing database optimizer capabilities, others focused on rewriting SQL command by using materialized views, others by using XML features. But all of them focused on the area following capturing the SQL command by the database. This thesis did the opposite; it focused on the area before the SQL command reaches the database. So this thesis tried to enhance the performance of the database by sending well done, error free, and professional SQL commands. Of course when the database receives a good SQL command the result will be good performance.

6.2.3 Using Agent

The system that receives a SQL command and rewrites it in an intelligent way without affecting the resulting data and acquiring better performance has to be intelligent. An agent came into the scene holding the important features to do the job like portability, reliability, and self-maintenance.

The agent acts as a connection layer between the clients and database server, in the clients side the agent receives the SQL commands and starts to study these commands to see if they need to be rewritten or not. In the server side the agent has to keep an online dictionary of the database to be able to make good decisions about the SQL commands received by clients. Linking clients with server has to be secured, so the agent has to maintain security issues and authorizations.

المنارة للاستشارات
www.manaraa.com

6.2.4 Performance Analysis

The comprehensive set of the performance measurements relating to old SQL command and new SQL command are considered and focused on many metrics like real time, CPU cost, IO cost, consistent gets and others.

Several benchmark tests were done for certain performance measurements in which the revised data organized in tables were observed. The benchmark tests results were compared graphically with each other for old and new SQL. The deduction points of these results were discussed in detail to explain the true reasons for the relationship between certain performance measurements and different SQL commands used.

6.3 Recommendations for Future Work

This section offers some suggestions for future work. In addition these recommendations will encourage the researchers to conduct further studies in the research field.

6.3.1 Development of Proposed Solution Field

For future work, the proposed solution presented can be improved by carrying out the following:

1- Develop a solution for other types of functions like (Round, Truncate, Ceil, and Floor).

2- Develop a solution for other types of SQL commands like nested SQL.

3- Develop a solution for SQL set commands like (Union, Intersect, Minus).

4- Develop a solution for XML query language.

المنارة للاستشارات
www.manaraa.com

6.3.2 Performance Analysis of Proposed Solution Field

The presented performance analysis in this research can be readily extended to include:

1- Research for new algorithms to decrease the execution time of SQL commands.

2- Research for new algorithms to make all commands use binary search instead of sequential search.

3- Analyze the effects of the performance measurements when applying the agent to a different type of database like (SQL server).

المنارة للاستشارات

www.manaraa.com

References

[1] A.Y. Halevy, "**Answering queries using views: A survey**" 2001, Department of Computer Science and Engineering, University of Washington, Seattle.

[2] Alin Deutsch and Val Tannen, "**Reformulation of XML Queries and Constraints**" 2003, UC San Diego, deutsc, University of Pennsylvania.

[3] Amit Shukla, Prasad M. Deshpande, Jeffrey F. Naughton, "**Materialized View Selection for Multidimensional Datasets**" 1998, Computer Sciences Department, University of Wisconsin – Madison, Madison.

[4] Brian Brewington, Robert Gray, Katsuhiro Moizumi, David Kotz, George Cybenko and Daniela Rus, "**Mobile Agents in Distributed Information Retrieval**", 1999, Thayer School of Engineering, Department of Computer Science, Dartmouth College, Hanover, New Hampshire.

[5] Bryan Genet, Annika Hinze, "**Open Issues in Semantic Query Optimization in Relational DBMS**", 2003, Department of Computer Science, University of Waikato, New Zealand.

[6] Chang-Sup Park, Myoung Ho Kim, Yoon-joon lee, "**Rewriting OLAP Queries Using Materialized Views and Dimension Hierarchies in Data Warehouses**" 2000, supervised by IITA.

المنارة للاستشارات
www.manaraa.com

[7] Chris M. Giannella, Mehmet M. Dalkilic, Dennis P. Groth, Edward L. Robertson, "**Improving Query Evaluation with Approximate Functional Dependency Based Decomposition**", 2003, NSF Grant IIS-0092407

[8] Danny B. Lange and Mitsuru Oshima, "**Seven Good Reasons for Mobile Agents**", Communications of ACM , vol. 42, no. 3, March 1999.

[9] Glenn Stokol, "**Oracle XML Fundamentals**", 2004, Oracle Corporation.

[10] Graham J.L. Kemp, Peter M.D. Gray and Andreas R.Sj "**Rewrite Rules for Quantified Subqueries in Federated Database**" IEEE 2001.

[11] Ivan T. Bowman, Kenneth Salem, "**Optimization of Query Streams Using Semantic Prefetching**", SIGMOD 2004, ACM 1-58113-859-8/04/06.

[12] J. Baumann, F. Hohl, K. Rothermel and M. StraBer, "**Mole – Concepts of a mobile agent system**", 1998, IPVR (Institute of Parallel and Distributed High-Performance Systems), University of Stuttgart, Breitwiesenstrabe

المنارة للاستشارات
www.manaraa.com

[13] J.O. Kephart, D.M. Chess. "**The Vision of Autonomic Computing**", IEEE Computers,36(1), 2003, pp. 41 – 52.

[14] John P. Mckenna, "**Aggregate Navigation using Materialized Views and Query Rewrite**". 2002, Counterpoint Technologies, Inc.

[15] K. B. Manwade, and G. A. Patil, "**Performance Analysis of Parallel Client-Server Model Versus Parallel Mobile Agent Model**" 2008, ISSN 2070-3740.

[16] Katsuhiro Moizumi, "**Mobile Agent Planning Problems**", 1998, Thayer School of Engineering, Dartmouth College,Hanover, New Hampshire.

[17] Lucian Popa, "**Object/relational query optimization with chase and backchase**" 2001, Institute for Research in Cognitive Science, IRCS Technical Reports Series, University of Pennsylvania.

[18] Maxim Grinev, "**XQuery Optimization Based on Rewriting**" 2004, Vorob'evy Gory, Moscow 11992, Russia.

[19] Muralidhar krishnaprasad, Zhen Hua Liu, Anand Manikutty, James W. Warner, Vikas Arora, Susan Kotsovolos, "**Query Rewrite for XML in Oracle XML DB**" 2005. Oracle Corporation.

[20] Oracle corporation, "**Building Oracle Data Warehouses**" 2004.

المنارة للاستشارات
www.manaraa.com

[21] Priya Vennapusa, "**Oracle Database SQL Tuning Workshop**", 2003, Oracle Corporation.

[22] Rada Chirkova and Michael R. Genesereth, "**Linearly Bounded Reformulations of Conjunctive Databases**" 2000, Stanford University, Stanford CA 94305, USA

[23] Rahul Jha, "**Mobile agents for e-commerce**", 2001, KR School of Information Technology Indian Institute of Technology, Bombay.

[24] Ramez Elmasri, Shamkant B. Navathe, "**Fundamentals of Database Systems**" forth edition, 2004, Pearson Addison Wesley.

[25] Rosie jones, Benjamin Rey, Omid Madani, Wiley Greiner, "**Generating Query Substitutions**" 2006, IW3C2, Edinburgh, Scotland, ACM 1-59593-323-9/06/0005.

[26] Sriram Mohan, Arijit Sengupta, Yuqing Wu, "**Access Control for XML- A Dynamic Query Rewriting Approach**" 2005, CIKM'05, ACM 1-59593-140-6/05/0010.

[27] Yu Jiao and Ali R. Hurson, "**Mobile Agents in Mobile Data Access Systems**", 2002, Computer Science and Engineering Department Pennsylvania State University

[28] Yabin Meng, "**SQL Query Disassembler. An Approach to Managing the Execution of Large SQL Queries**" 2007, Queen’s University Kingston, Ontario, Canada.

المنارة للاستشارات

www.manaraa.com

[29] William M. Farmer, Joshua D. Guttman, and Vipin Swarup, "**Security for Mobile Agents: Authentication and State Appraisal**" 2002, The MITRE Corporation.

[30] URL "**http://en.wikipedia.org/wiki/SQL**", web site accessed on 09-Mar-2009.

[31] URL "**http://en.wikipedia.org/wiki/Linear_search**", web site accessed on 15-Mar-2009.

[32] URL "**http://en.wikipedia.org/wiki/Binary_search**", web site accessed on 15-Mar-2009.

المنارة للاستشارات
www.manaraa.com

جامعة عمان العربية للدراسات العليا

تنغيم أوامر لغة الاستفسار المهيكلة باستحداث طبقة وسطية شفافة للوكيل المتنقل

اعداد

عماد حسن صالح

اشراف

الاستاذ الدكتور علاء حسين الحمامي

ايار، 2009

عمان الاردن

المنارة للاستشارات
www.manaraa.com