UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

SEARCH MECHANISM ENHANCEMENT OF A GEODATABASE FOR PROMPT DATA RETRIEVAL AND BUSINESS REPORTING

BY

TIMOTHY NJOROGE

P58/61527/2010

SUPERVISOR: DANIEL ORWA

SEPTEMBER, 2012

Project submitted in the fulfillment for the Degree of Master of Science in Computer Science

Declaration

This project is my original work and has not been submitted in support of a degree or any other award in any university

Timothy Njoroge: [illegible] **Date:** 31/10/2012

This project has been submitted with the approval as university supervisor

Mr. Daniel Orwa: [illegible] **Date:** 31/10/12

School of Computing and Informatics, University of Nairobi

Dedication

This work is dedicated to my beloved wife Edith and our precious daughters Gracia and Gloria for giving me the opportunity to do this project

I also appreciate my lecturers for the advice they gave me, particularly my supervisor Mr. Daniel Orwa and Mr. Lawrence Muchemi

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Definitions

GIS	Geographic Information System. GIS is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data.
Geodatabase	Geodatabase provide organizations with a central, scalable data storage and management system. It combines "geo" (spatial data) with "database" (data repository) to create a central data repository for spatial data storage and management.
Metadata	Data about the containers of data.
GE	Geogle Earth. GIS tool used for map presentation.
ArchGIS	Esri's ArcGIS is a geographic information system (GIS) for working with maps and geographic information. It is used for: creating and using maps; compiling geographic data; analyzing mapped information; sharing and discovering geographic information; using maps and geographic information in a range of applications; and managing geographic information in a database.
E&P	Exploration and production; This is the upstream oil sector.
NDC	National Data Center; Exploration and production subsidiaries of the National Oil Corporation of Kenya
Seismic	Seismic exploration uses sound waves to determine the density. It can do this because sound waves will move differently through different material. When seismic exploration is used to explore for gas, the sound waves will move through the earth and the results are used to determine where the best place to drill will be.

Geospatial	Geospatial analysis is an approach to applying statistical analysis and other informational techniques to geographically based data.
Contour lines	Contour lines are elevation. They measure elevation, and never cross because the elevations can't be two different numbers in one spot.
Contour curves	Contour curves, also known as topographic maps, have an enormous number of applications in applied mathematics. Mining and oil exploration, weather maps, contours of temperature (isotherms) and pressure (isobars) are a few examples.
Vector	vector themes (points, lines, and polygons)
Raster	raster or grid (pixels or picture elements)
Grover's algorithm	is a quantum algorithm for searching an unsorted database with N entries in O(N1/2) time and using O(log N) storage space
Oracle hashing	This involves a simple calculation, a jump to a memory location, and then a short in-memory sequential search. When the sequential search is short, the result is an incredibly fast search. A hash function is any algorithm or subroutine that maps large data sets of variable length, called keys, to smaller data sets of a fixed length.
PSC	Production Sharing Contracts
DB	Database
FS	File System
DML	Data Manipulation Language
SQL	Structured Query Language
CTAS	Create Table As Select
BASIN	Exploration Areas, Appraisal Areas, Production Areas
CBO	Oracle Cost-base optimizer
OLTP	Online Transaction Processing refers to a class of systems that

	facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing.
Oracle 9i, 10g, 11g	Oracle database version releases. The latest Oracle database release is Oracle11g version 11.2.0.3
PCTFREE	PCTFREE is a block storage parameter used to specify how much space should be left in a database block for future updates. For example, for PCTFREE=10, Oracle will keep on adding new rows to a block until it is 90% full. This leaves 10% for future updates (row expansion).
Table Cluster	A table cluster a group of tables that share the same data blocks, since they share common columns and are often used together.
Tablespaces	A tablespace is a logical storage unit. It is a container for segments (tables, indexes, etc). A database consists of one or more tablespaces, each made up of one or more data files. Objects in the same schema can be in different tablespaces, and a tablespace can hold objects from different schemas.
Schema	A schema is the set of objects (tables, views, indexes, etc) that belongs to a user account.
B-tree indexes	This is the standard tree index that Oracle has been using
Bitmap indexes	Bitmap indexes are used where an index column has a relatively small number of distinct values (low cardinality). These are super-fast for read-only databases, but are not suitable for systems with frequent updates
Bitmap join indexes	This is an index structure whereby data columns from other tables appear in a multi-column index of a junction table. This is the only create index syntax to employ a SQL-like from clause and where clause.
HC_TYPE	Hydro Carbon type. These are the contents found in oil and gas wells.

Abstract

The purpose of this paper is to present a system design that significantly improves the design and search mechanism of a geodatabase system in the oil and gas, exploration and production sector. The increasing exploration activities in our country require rich collection of related exploration data which need to be stored in a geodatabase in an organized fashion while maintaining integrity of the data with a consistent, accurate database and applying sophisticated rules and relationships to the data.

We conduct a study of the integration of a geodatabase model design with a metadata model design in the database tier. The application tier of the geodatabase contains a metadata search mechanism that enhances retrieval of data when a query is executed. We also compare the time taken to execute a query on a metadata search algorithm to one that has no algorithm implemented, and found that the performance of the metadata search indexing caused the search run four times faster. We give an in-depth modeling of the geodatabase and the inter-relationships between the objects identified in the cause of the research.

Chapter 1: Introduction

1.1 Background

Effective and efficient oil exploration in a country requires careful collection and monitoring of large amount of exploration data. One way to meet this data management challenge is to tie together data sources in the form of text, tables, photographs and maps through computerized geographic information systems (GIS). A GIS is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data.

A GIS system can consists of two subsystems: Gathering subsystem and Query subsystem. The Gathering Sub-System is used to collect spatial GIS file information from remote locations, and to store them in the server. The Gathering subsystem locate and retrieve the necessary GIS vector/raster files information and publish the information in such a way the information can be found by a user and its value recognized. A database is used to store the information gathered from the Gathering Sub-System. The Query Sub-System is used to help the users in their daily work with the GIS; it works as a GIS retrieval (query) tool. The Query Sub-System consists of a User Interface used to retrieve GIS information and filter the retrieved information by identifying the user's necessities.

This research will focus on optimizing the Query subsystem of a GIS system. The objective is to model a geodatabase system and develop a prototype which will demonstrate enhanced geodata retrieval and prompt business reporting. Proper storage and retrieval of the geodata is very essential as this contributes into laying out proper procedures and policies of handling exploration data in a country.

A geodatabase system is a common data storage and management framework for a GIS system. It combines "geo" (spatial data) with "database" (data repository) to create a central data repository for easy access and management. In the oil and gas exploration sector most of the database calls heavily depend on information about wellbores and the seismic data. Employment of metadata search mechanism and advanced database search algorithms will enhance the search mechanism of the geodatabase.

1.2 Problem Definition

The National Data Center (NDC); National Oil Exploration and Production (E&P) department, has been using a document management system known as Case360 that stores and organizes physical files associated with oil and gas exploration activities. However, Case360 is not a fully fledged geodatabase. The proposed geodatabase will go beyond document management; it will be a central repository for exploration spatial and bio data for easy access and management. Rich collection of related exploration data will be stored in the geodatabase in an organized fashion; maintaining integrity of the data with a consistent, accurate database and applying sophisticated rules and relationships to the data. This is essential for exploration activities in a country.

In this regard one of NDC initiative is to implement an enhancement of the current system that will eliminate this information bottleneck. Case360 stores wellbore and seismic physical files in directories. The search criteria are based on file a system, hence causing information retrieval and reporting inefficient and problematic. In the event of a file being put in the wrong directory, that file is considered lost since the current search algorithm would not locate it. This problem will be solved by metadata search and advanced search techniques. The research will explore search algorithms in a Relational Database Management System (RDBMS) and also develop a metadata based search algorithm that will integrate with the geodatabase. Furthermore, high level modulization and dynamization will be employed in the model design, to allow flexible administration of the system and timely production of business reports. Enhanced performance of the system will boost accurate and better business decision making in regards to exploration activities in a country.

To achieve this, both the data and metadata objects will be leveraged on the database. Organizing the metadata (data about the data) on the database will allow the GIS administrator define additional database table columns without contacting a proprietor. This flexibility ensures that all the necessary exploration information is gathered and stored in the database promptly. Such administrative organization and indexing of data will also guarantee timely location and retrieval of the geodatabase data. The geodatabase system will also offer ability to integrate with other GIS softwares such as ArchGIS and Google Earth(GE) used for map presentation and interpretation. The integration with systems that interpret wellbore and seismic data will enlighten both NDC and vendors for possible basins or areas with oil and gas prospects thus boosting the timely discoveries of oil and gas.

Most geodatabase systems do not address the issue of flexibility and dynamism. Exploration activities vary from country to country. When new attributes need to be defined in a geodatabase to fit a country's need, the proprietor of the system in use has to be contacted. Free source GIS systems have to be heavily customized to fit or meet the organization needs. In addition, to enable some GIS system run off the web, certain processes have to be engaged. For instance ArchGIS will need ArchEditor (for uploading), SDE engine (to register the webserver) and ArchView server (to deploy to the webserver).

Because of the vastness of the E&P sector, there is a need to create a geodatabase system that offers the ability to search the geodata promptly and also define new attributes dynamically. This flexibility is made possible by loading the application user interface or front end labels and fields off the metadata. This modeling method will enable the geodatabase users i.e. geophysist, geochemist, geologist and GIS administrators, define new attributes whenever required without having to contact a proprietor.

1.3 Research Objectives

The main objective of the research is to demonstrate performance enhancement of a geodatabase system for fast data retrieval and business reporting in the development of oil and gas exploration opportunities in Kenya. Hence the following are the objectives of this study:

- To identify database search algorithms with a view of enhancing the search mechanism of a geodatabase.
- To propose a geodatabase model that will store exploration data in an organized fashion.
- To propose a Metadata model that will integrate with the geodatabase so as to enhance the search mechanism.
- Develop a prototype and use it to validate the above model.

1.4 Research Questions

The following research questions will arise based on the objectives of the study:

- How can we model a geodatabase?
- What objects will be in the geodatabase to create the centralized data repository for geo data?
- How do we design a metadata search mechanism?
- What are the challenges and benefits of the proposed RDBMS advanced search techniques?
- How can we monitor, evaluate and verify the search algorithms?

1.5 Scope and Limitation

This research problem will focus on opportunities for oil and gas exploration in Kenya. Enhanced performance of the geodatabase system will boost accurate business reporting and provision of better business decision making from the wellbore and seismic data collected during the exploration activities. Integration ability of the enhanced geodatabase system to GIS systems that handle map presentation and interpretation will shade light to NDC and the drilling companies' of possible areas where oil can be discovered and hence provide timely discoveries. The research work will introduce a metadata model which will integrate with the geodatabase model at the database tier so as to enhance the search mechanism of the geodata. The limitation of this research will be to develop a full-fledge geodatabase system with all objects and attributes to make it functional. However, a prototype will be developed to validate the enhancement of the search techniques and management of wellbore and seismic data.

Chapter 2: Literature Review

This literature review is initially done on existing algorithms of query optimization in relational databases. Later on we shall demonstrate a metadata search mechanism that enhances time complexity of retrieving records in an RDBMS. Special attention will be given on how the geodatabase relates to the metadata design which is the basis of our search mechanism. The research also identifies geodatabase objects that apply in the oil and gas exploration sector and their inter-relationship.

2.1 Background to the Relational Model

During the 1950s and 1960s, Johnson (1997) identified bottlenecks caused by programs owning data; unhealthy dependence between data and programs. This led to the development of database systems so that data can be independent and the application programs just access it. With the development of databases, data and application software became independent but interacting components. A Database is a collection of logically related data and a Database Management System (DBMS) is a software product that helps in defining, creating, maintaining and controlling access to a database. DBMSs are grouped according to the model of their development. A database model is an organizing principle that specifies particular mechanisms for data storage and retrieval. There are five database models namely the Hierarchical, Network, Relational, Object and Deductive models; the Relational model being the most popular of all.

The Hierarchical and Network models, which were developed before the Relational model was developed, are referred to as the pre-Relational models while the Object and Deductive models, which were developed after the Relational model was developed, are referred to as the post-Relational models.

2.2 Overview of Query Optimization in Relational Databases

When the relational model was first launched in the late 1970s, one of the major criticisms often cited was inadequate performance of queries Connoly and Begg

(2001). This was because queries used a lot of resources such as processor cycles and memory compared to other models for equal amounts of data. SQL, which is the de facto standard language for data definition and data manipulation in RDBMS Connoly and Begg (2001) offers a variety of ways in which a query can be structured to achieve the same output. The more the complexity of the query the higher the number of ways a query can be represented. On average, the best measure of a relational query complexity is the number of relations a single query joins. In fact, even at the design level, developers introduce redundancy or merge some relations such that joins in frequently invoked queries are minimized. Surajit and Kyuseok (1999), state that the complexity of a query is exponential to the number of joins involved. Complex queries like those in data warehouses that normally join tens of tables are too expensive to process in reasonable time. Since the structural differences in queries depend highly on the way the joins are ordered, the more the tables joined the more the options of writing a query that can bring a single output. These queries, if executed have varying costs and in most cases (if not all) only one is optimal. The probability of writing the most optimal query tends to zero as the query complexity increases and the computer is most likely to waste a lot of resources. It is therefore the complex queries that must be optimized if a computer system is to work efficiently. An extensive query optimization phase must select the most efficient plan among the many available to process the query in an acceptably short time. Without query optimization, RDBMSs would be inefficient and hence unpractical Ramakrishnan and Gehrke (2000). Query optimization is an expensive process because it mostly relies on evaluating the different plans (access paths) and choosing an optimal one among them. The number of alternative access paths grows at least exponentially with the number of relations participating in the query, Kyuseok (1993).

The optimizer therefore, which is nearly 100% sure that the plan sent by a user is not optimal, has to search for an optimal plan and forward it for execution. This has to be done within the time constraint and in a resource-conserving manner. It would not be worthwhile if the difference between the optimized and a pre-optimized query is less than the cost of finding the optimal plan. The user likewise is supposed to get what he requested for, in the same logical presentation as well as in an acceptable time interval. Therefore, as the user specifies what, the optimizer determines the how, but still conserving the what, Elmasri and Navathe (1994). The process of optimization should have no effect whatsoever on the final query output.

The search strategy therefore, on top of conserving the form and content of the query request must be efficient. Optimization is not a matter of transferring the resources that would execute the query to looking for the execution plan. The ability of the optimizer reaching the optimal plan, at the earliest opportunity, with substantial resource savings is therefore of paramount importance. Kroger (2001) summarizes the goal of an optimizer as follows "A plan as cost-effective as possible is looked for as soon as possible". Kroger (2001) further observe that the job of a query optimizer is not necessarily to get the cheapest plan (though the cheapest plan would of course be the best). In fact, if a stage is reached where the cost of further optimizing is higher than the resource savings, it is worthwhile to terminate the search.

The optimizer is supposed to economize the resources spent on looking for a plan as well as putting into consideration the time of processing (time of execution plus time of optimization). Depending on the nature of the problem therefore, a sub-optimal plan may be preferred especially in a real time scenario. Given the large number of possible plans, traversing them, one by one, establishing the cost of each may be the ideal strategy but it is time wasting since the options are too many and it is likely to produce a low cost query but having spent a lot of resources to get it.

2.3 Approaches to Query Optimization

Query Optimization is the process of choosing the efficient execution strategy for executing a query, Connoly and Begg (2001) and it is one of the most important tasks of any RDBMS. Ramakrishman and Gehrke (2000) observe that SQL which is a de facto standard for data definition and data manipulation in RDBMSs has a variety of ways in which a user can express, and therefore a system can evaluate a query. The query optimizer therefore is responsible for finding the best execution strategy so that fewer resources are used to retrieve data. There are three main approaches to query optimization. These are Systematic, Heuristic and Semantic query optimization.

2.3.1 Systematic Query Optimization

In systematic query optimization, the system estimates the cost of every plan and then chooses the best one. The best cost plan is not always universal since it depends on the constraints put on data. For example, joining on a primary key may be done more easily than joining on a foreign key since primary keys are always unique and therefore after getting a joining partner, there is no other key expected. The system therefore breaks out of the loop and hence does not scan the whole table. Though in many cases, it is a time wasting practice and therefore sometimes it can be done away with, Elmasri and Navathe (1994).

The costs considered in systematic query optimization include access cost to secondary storage, storage cost, computation cost for intermediate relations and communication costs. The importance put on these costs depend on the type of database. For example, for large databases, emphasis is put on minimizing access cost to storage and memory usage. For small databases however, where outputs can be stored in memory, emphasis is put on minimizing computational cost. On the other hand, in distributed databases, where many sites are involved, communication cost is of paramount importance and it has to be minimized since it normally involve costs of channel coding, security coding as well as other network related limitations like bandwidth and noise.

2.3.2 Heuristic Query Optimization

In the heuristic approach, the operator ordering is used in a manner that economizes the resource usage but conserving the form and content of the query output. The principle aim is to:

- Set the size of the intermediate relations to the minimum and increase the rate at which the intermediate relation size tend towards the final relation so as to optimize memory.

- Minimize on the amount of processing that has to be done on the data without affecting the output.

Connolly and Begg (2001) state five main rules which are used in heuristic query optimization:

1. Perform selection operations as early as possible. This reduces the cardinality of the intermediate relation and hence reducing the resources used to process a column as well as the memory occupied per column.
2. Combine the Cartesian product with a subsequent selection operation whose predicate represents a join condition into a join operation. Elmasri and Navathe (1994) observe that this reduces the complexity of the joining algorithm (which is one of the most expensive operations in data retrieval) for example in cases where individual relations are first sorted on the joining fields.
3. Use association of binary operations to rearrange the query so that the most restrictive selection operation is done first. This increases the rate at which the intermediate relation size tends to the final relation size hence minimizing on the memory occupied and the resources required to process a column.
4. Perform projection operations as early as possible. This reduces the order of the intermediate relation. It therefore reduces the memory occupied by the relation together with the amount of resources required to process a row.
5. Compute common expressions once. If a certain expression appears more than once, and it's not too large, it is kept in memory so that when it is required again, it is reused. In case the expression is too big to fit in memory, it can be stored on a disk and later retrieved when wanted so long as the cost of retrieval is not greater than the cost of recomputing it.

2.3.3 Semantic Query Optimization

This is a combination of Heuristic and Systematic optimization. The constraints specified in the database schema can be used to modify the procedures of the heuristic rules making the optimal plan selection highly creative. This leads to heuristic rules that are locally valid though cannot be taken as rules of the thumb. For example, if there is a query such as;

```
        SELECT Employee.lname, Supervisor.lname
        FROM Employee, Supervisor
        WHERE Employee.supervisorNo = Supervisor.No
        AND Employee.salary ≥ Supervisor.salary
```

This is a very unlikely invent and it's likely to be directly or indirectly in the database constraints. The database restriction may be like check Employee.salary between(S1,S2). Check Supervisor.salary between(S3,S4) where S3 >S2. This shows that the Supervisor can never earn less than the Employee therefore the query yields no results.

A Heuristic optimizer would go ahead and parse, optimize and execute the query resulting in no output which is a worst case scenario, Horowitz, Sahni and Rajasekaran (1996). A semantic optimizer would recognize it by use of the constraints and respond "Empty set" and saves the resources.

2.4 Single and Multi-Query Optimization Types

Broadly, computers optimize queries either individually (Single-query optimization) or as batches (Multi-query optimization).

2.4.1 Single Query Optimization

In Single Query Optimization, a query, which is syntactically correct, is broken down, expressed into a relational algebra expression, and a query plan, represented as a tree is created, Ramakrishnan and Gehrke (2000). This is a traditional approach to query optimization and is used in most commercially available optimizers. It is suitable where a database receives a low traffic of simple queries. Depending on the algorithm used, either the different representations of the original query are generated and the best one searched for or the query supplied is adjusted to the optimal one. If the option of choosing the best tree from the different trees available is used, there is a high possibility of logical duplicates (two physically different trees, doing the same thing, the same way). Since the number of options is likely to be high, such options normally overload the memory and require an exhaustive algorithm. There is a likelihood of groups of plans, with the same cost implying that more searching is made but with no practical advantage. Using exhaustive algorithms is a reason for inefficiency of many query optimization research works carried out Bhobe (2001). Single processor optimizers therefore limit the search space by considering only some tree configurations, Kremer and Gryz (1999). This may be left deep, right deep, complex deep and bushy.

Heuristics may as well be used to eliminate some obviously expensive plans before the optimization so as to reduce the search space. The rules may be static or dynamic. Dynamic rules are applied basing on available data like database statistics or system catalog. Failure to reduce the search space may cause effects that lead to the decline in the performance and cost effectiveness of the optimizer. These are:-

- Too many options may overload the memory. In cases where the query is complex, the computer may not have enough memory space to perform other routine activities.
- In the process of conserving the memory, the computer may have to store some of the plans on disk and retrieve them when required. This then brings in costs of writing and reading from disks which increase optimization costs.
- The many options put a bigger load on the processor during cost estimation and comparison.

Restricting the tree types used in optimization therefore eliminating duplicates reduces the number of possible options hence reducing on the search space saving the traversal of many options as well as memory usage.

2.4.2 Multi-Query Optimization (MQO)

In MQO, queries are executed in batches. Some of the MQO techniques act in such a way that in case some queries have a common sub-expression, such a sub expression is executed once and the output shared. In some cases, the sharing does not necessarily take place on individual optimal plans, Roy (2001), but instead sub-optimal plans are used. Decision may as well have to be taken whether the common sub expressions should be pipelined or materialized, Nilesh (2001). Some multi-query optimization techniques, like those described by Kyuseok (1994) basically aim at having parallel optimization of many queries. The queries pass through the different optimization steps together and as an output, which is a set of optimal plans for each query is generated. Roy (2001) criticizes this approach on a basis that further cooperation can be made between the queries that make up the batch. If a certain sub-expression is common, then the computer should execute it once and share out the results.

This is a guiding principle to the Basic Volcano algorithm proposed by Goetz and McKenna (1991), and the Volcano-SH and Volcano RU optimizer algorithms proposed by Roy (2001). Roy further put the sharing of the sub-expressions to a great importance that even if the sharing takes place on a non-optimal plan of the query, so long as the total resource requirement is optimal, it is acceptable.

2.5 Understanding Search Algorithm and their Efficiency

The maximum program efficiency is obtained through a unique search algorithm and data structure, instead of examining the recall ratio and the precision ratio at a higher level, this efficiency is measured in the most basic term of "average number of searches" required for looking up an item. In order to identify an item, at least one search is necessary even if it is found the first time. However, through the use of the hash-address of a key or keyword, in conjunction with an indirect-chaining list-structured table, and a large available space list, the average number of searches required for retrieving a certain item is 1.25 regardless of the size of the file in question. This is to be compared with 15.6 searches for the binary search technique in a 50,000-item file, and 5.8 searches for the letter-table method with no regard to file size.

2.5.1 Linear Search

This is also called sequential search or sequential scan. The linear search of an unordered list or file is the simplest one, but is inefficient because the average number of searches for a given entry in a N-entry file will be **N/2**. For example, if **N = 50,000**, the average number of searches for a given entry
is an enormous **25,000**. It is assumed that the probability of finding a given entry in the file is one. The average number of searches in a linear search is calculated as:
$S = N + 1/2$ or $S = N/2$ if N is a large number.

The linear search has to be performed in a consecutive storage area and this sometimes causes certain inconvenience if the required storage area is very large. The inconvenience can be avoided by using the last computer word (or some bits of it) to index the location of the next section of storage area used and thus form a single chain for searching. This variation of the linear search method is called the single chain method. It differs from the linear search in storage flexibility but is otherwise the same in the efficiency.

2.5.2 Binary Search

Using the binary search method will yield a more satisfactory result. The search starts with the midpoint of the file, and goes to the midpoint of the associated remaining half if a match fails. The comparison of their values will decide which half of the file should be tried next time. This process will be repeated until a notch is found. The average number of searches in the example is calculated through the following formula:
$S=N+1/N \log_2(N+1) - 1$ or $S = \log_2 N$ if N is a large number.
The Hibbard's Double Chain Method and Sussenguth's Distributed Key Method are

compatible to the binary search in search efficiency but have a much better update efficiency because of the list-structured address-chaining mechanism.

The respective calculations of the example are:

S = 1.4 log2N = 21.9 (Hibbard) and **S = 1.24 Iog2N = 19.4** (Sussenguth)

In order to have a gross understanding of various search algorithms, examine and compare Binary and Linear Search in respect to their search efficiencies.

Search method	Average No of searches	Sample S (N=50,000)	Search efficiency	Update efficiency	Storage efficiency
Linear search	N/2	25,000	Poor	Good	Excellent
Binary search	log2N	15.6	Good	Poor	Good

*Table 1: **Comparison of binary and linear search algorithms***

2.5.3 Searching Strings in RDBMS using Functions

When it comes to searching, we often find ourselves using the LIKE (including NOT LIKE) operator in SQL statements. Other well-known tests include equality and comparison (greater or less than), and this includes not only strings, but also numbers. Conceptually, the comparison tests are pretty simple, and indexes can add a tremendous boost to performance when these tests are used. However, what takes place when we are not looking for an exact match, but something, "LIKE" the value of interest? Mathematically, or at least algorithm-wise, constructing a test for equality is pretty simple. How do we suppose RDBMS approaches the problem of determining if the string 'ABCD' appears in the string 'ABCABDABCDAB' and if it does, how many times (as in the use of INSTR)? Further, once we have a result set, what determines the sorted order?

From an algorithm design standpoint, we want the process to be performance. The complexity of the algorithm should be first order, as in O(n) (the big "O" notation) as opposed to $O(n^2)$, nLogO(n) and so on. This implies that the algorithm is efficient, and efficiency can be qualified with goals of not repeating work, skipping over intervals, and be deterministic (we can compute the best case and worst case).

In addition to string searching within SQL, virtually every text editor incorporates a search function. Without knowing the implementation details or underlying code, it is hard to determine exactly how a search editor works. Like most algorithms, there will be cases where some general or even trivial condition will make the algorithm scream along or slow to a crawl. Searching for a word within a string, where the word is longer than the searched string, is an example of a case where adding a simple length check up front will eliminate at least one condition. On the other hand, searching for something not very distinguishable (but distinguishable enough because the pattern does match) can evoke a worst-case performance (search for A in the string BBBBBB...BBBA, no hit until the very end). Overall, the search should be done in no worse than what is called linear time, or O(n).

So, from which end of the target string should the search start? One widely used algorithm, because of its efficiency, is the Boyer-Moore string search algorithm. This algorithm actually starts at the end and works backwards. Once a miss is encountered (a "bad" character, that is, one that is not in the search pattern/string), the algorithm knows to shift the search over by the length of the search. If the bad character falls anywhere within the positions covered by the length, then it is impossible to find a match within that interval as the search string would span the bad character.

Now that we know the general approach of this algorithm, how do we think it applies to Oracle? If the test operator is LIKE, perhaps Oracle uses Boyer-Moore because we, and subsequently Oracle, does not care where the pattern is met or how many times it is met, just that it is met at least once. On the other hand, if using INSTR, we have a choice as to where to start the search (beginning or end), in addition to which occurrence. The fact that this function defaults to the beginning of the target should not imply anything about how Oracle conducts the search. For all we know, Oracle may internally reverse the string for search purposes (which can be quickly done) and leave the default start position as the beginning because this is more intuitively obvious to humans (not implying Oracle is alien, just that in English, we tend to start from left to right).

What may not be obvious at this point is this fact: the longer the search string/pattern, the quicker the search is likely to be performed. If we divide a searched word (or string) of length N by the length M of the search pattern, we have at most N/M chunks to search. The more information, that is, the longer the search pattern is, we can provide up front, the less work the search algorithm has to perform. This is akin to searching for a needle in a haystack. It should be obvious that the longer the needle, the easier it will be to find. The same holds true for string searches.

2.6 Advanced Search Optimization of RDBMS (Case Study for Oracle)

The proposed database for the geodatabase will be ORACLE Database. Oracle has become the world's most flexible database and it stores much more than text and numbers. An Oracle database support video, audio and complex spatial applications. Unlike simpler databases, we can control every aspect of Oracle's behavior. We can control how rows are placed on the data blocks and we can control how Oracle performs hundreds of resource management issues. Oracle is considered to be most powerful and robust database. Most organizations also use Oracle as its primary database due to its capacity to deliver high quality information.

Oracle database has inbuilt query optimization mechanism namely:

- Oracle Indexing
- Oracle Multiple Blocksizes
- Oracle Hashing

- Data Sequencing or Row Re-Sequencing
    - Oracle Sorted Hash Cluster
    - Oracle Index Cluster Tables
- Oracle First_Rows Optimization
- Oracle Index Rebuilding

Each of the Oracle search techniques has advantages and downsides. The Oracle search mechanisms will be implementable in the geodatabase to enhance its query performance.

Search algorithms can use either an authoritarian approach or a more discussion-based approach. For example, when determining a SQL execution plan, the simplest approach is to pick a very authoritarian algorithm. But when the complexity increases, an authoritarian model becomes very inefficient and can produce non-optimal results. When the possible options are known and limited, and the environment stable, an authoritarian model can be very fast and reliable. One example is when Oracle needs to determine if a block is in the buffer cache or when it needs to know if a SQL statement is in the library cache. Oracle could use a discussion-based here but in this situation, the alternative outcomes and scope are limited and well-defined. The buffer is either in the cache or it is not in the cache. So Oracle chose an algorithm called hashing.

2.6.1 Oracle Indexing

The dba_indexes view, which is populated with index statistics when indexes are analyzed. The dba_indexes view contains a great deal of important information for the SQL optimizer. Oracle provides an analyze index validate structure command that provides additional statistics into a temporary tables called index_stats, which, sadly, is overlaid after each command, Burleson 2011

We can use the large (16-32K) blocksize and separate data caches to improve response time under certain conditions. A larger blocksize can result in a reduction in logical I/O.

There is significant response time reduction after a move to a larger blocksize: Giving the response time of the same query from two databases, one having standard block size of eight kilobytes (8k) and other one having sixteen kilobytes (16k)

2.6.2 Oracle Multiple Blocksizes

These are many Oracle Transaction Processing Performance Council (TPC) benchmarks that thoroughly test multiple blocksizes verses one-size fits all. These benchmarks are fully reproducible, so there performance gains can be proven independently. This UNISYS Oracle benchmark used multiple blocksizes to achieve optimal performance.

Considering an online transaction processing (OLTP) database with the following characteristics almost similar to a geodatabase:

- The vast majority of text rows are small, say 80 bytes. A 2k block size would reduce the waste from reading-in a 8k block, only to fetch 80 bytes.
- The database is 100 gigabytes, but there is only 8 gigabytes of available data buffers.
- The database stores images (BLOB, CLOB) in a separate tablespace, requiring a large blocksize to avoid fragmentation.
- The database is heavily indexed, and index access patterns tend to read large sections of the index.
- The data has a typical usage skew, with some popular rows, and some rows that are rarely accessed.

A typical database has popular blocks and unpopular blocks. We know that we need a

16k blocksize to keep our CLOB data from fragmenting, and we do not want the buffer wastage that occurs when we read-in a 32k block just to access an 80 byte row.

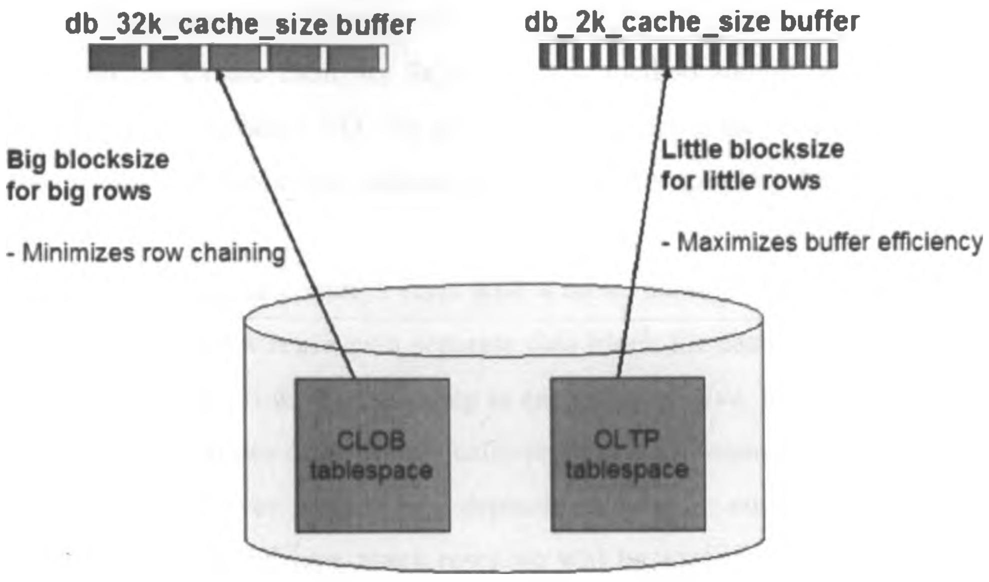


*Figure 1: **Multiple blocksizes for mixed I/O signatures.***

2.6.3 Oracle Hashing

Hash functions are primarily used in hash tables, to quickly locate a data record given its search key. Specifically, the hash function is used to map the search key to the hash. The index gives the place where the corresponding record should be stored. Oracle hashing and hashing algorithm converts a symbolic key or ROWID to a disk or RAM address in a heap. It is very fast; it takes 50 microseconds to returns a ROWID or RAM address.

Hashing is lightening fast and requires only a small amount of memory as opposed to a large amount of disk space or CPU time. There is basically a simple calculation, a jump to a memory location, and then a short in-memory sequential search. When the sequential search is short, the result is an incredibly fast search; much faster than using any kind of index or a cost-based approach.

2.6.4 Data Sequencing or Row Re-Sequencing

If response times are lagging in our high-transaction system, reducing disk I/O is the best way to bring about quick improvement. When we access tables in a transaction system exclusively through range scans in primary-key indexes, reorganizing the tables with the Create Table As Select (CTAS) method should be one of the first strategies we use to reduce I/O. By physically sequencing the rows in the same order as the primary-key index, this method can considerably speed up data retrieval.

For queries that access common rows with a table, unordered tables can experience huge I/O as the index retrieves a separate data block for each row requested. Like disk load balancing, row re-sequencing is easy, inexpensive, and relatively quick. It shortens response times-often dramatically-in high-I/O systems. The degree to which re-sequencing improves performance depends on how far out of sequence the rows are when we begin and how many rows we will be accessing in sequence. We can find out how well a table's rows match the index's sequence key by looking at the dba_indexes and dba_tables views in the data dictionary.

The benefits of row resequencing cannot be underestimated. In large active tables with a large number of index scans, row resequencing can triple the performance of queries. Data sequencing is an extremely powerful SQL performance tactic for sequential heavy queries.

2.6.5 Oracle First_Rows Optimization

The first_rows_n mode allows us to tell the Cost-based Optimizer (CBO) how many rows we plan to use, thereby allowing the optimizer to make an intelligent execution plan. Since Oracle9i most systems will have many frequently-referenced tables cached in the KEEP pool, the first_rows_n parameter may only be helpful in reducing logical I/O, and not necessarily the more expensive disk I/O. The first_rows_n optimization method improves SQL execution plans for OLTP systems that only need to deliver the first part of a larger solution set. Oracle Corporation states that with first_rows_n optimization, Oracle queries give the best possible response time for the first rows of a result set.

2.6.6 Oracle Index Rebuilding

We cannot generalize to say that index rebuilding for performance is common or rare, it depends on many factors, most importantly the characteristics of the application.

- In scientific applications (GIS Systems, clinical, laboratory) where large datasets are added and removed, the need to rebuild indexes is "common".
- Conversely, in systems that never update or delete rows, index rebuilding rarely improves performance.
- In systems that do batch DML jobs, index rebuilding "often" improves SQL performance.

There are many compelling reasons to manage indexes within Oracle. In an OLTP system, index space is often greater than the space allocated for tables, and fast row data access is critical for sub-second response time. Oracle offers a wealth of index structures:

- B-tree indexes - This is the standard tree index that Oracle has been using since the earliest releases.
- Bitmap indexes - Bitmap indexes are used where an index column has a relatively small number of distinct values (low cardinality). These are super-fast for read-only databases, but are not suitable for systems with frequent updates
- Bitmap join indexes - This is an index structure whereby data columns from other tables appear in a multi-column index of a junction table. This is the only create index syntax to employ a SQL-like from clause and where clause.

The following are recommendations of Oracle on Index rebuilding:

- Using bigger blocks means more data transfer per I/O call; this is an advantage since the cost of I/O setup dominates the cost of an I/O.

- Using bigger blocks means more space for key storage in the branch nodes of B*-tree indexes, which reduces index height, which improves the performance of indexed queries.
- Using a block size that is k times bigger than our current one will save us (k-1)f/(kb-f) bytes of space for large segments, where f is the size of a block's fixed block header (61 bytes for tables, 57+4n for n-table clusters, 113 for indexes).
- When using large block there are less probability of chained and migrated rows, which in turn reduced the number of reads required to get the information.

2.5 Performance Analysis

2.5.1 Measuring the Performance

Test bed performance measurement analysis of a system can provide exact answers regarding the performance of the system. The system in question is observed directly - no details are abstracted away, and no simplifying assumptions need to be made regarding the behaviour of the system. However, measurement is only an option if the system in question already exists. The measurements that are taken may or may not be accurate depending on the current state of the system. For example, if the utilization of a network is measured during an off-peak period, then no conclusions can be drawn about either the average utilization of the network or the utilization of the network during peak usage periods.

2.5.2 Analytical Models

Analytical models (e.g Markovian models), can provide exact results regarding the performance of a system. The results are exact, in that they are not estimates of the performance of the system. However, the results provided by analytical models may or may not be accurate, depending on the assumptions that have been made in order to create the model. In many cases it is difficult to accurately model industrial-sized systems with analytical models. In fact, it has been observed that when analysing computer systems "analytical modeling requires so many simplifications and assumptions that if the results turn out to be accurate, even the analysts are surprised." (Jain, 2001)

2.5.3 Simulation-Based

Simulation-based performance analysis can be used as an alternative to analytical techniques. Simulation can rarely provide exact answers, but it is possible to calculate how precise the estimates are. Furthermore, larger and more complex models can generally be created and analysed without making restrictive assumptions about the system. There are two main drawbacks to using simulation: it may be time consuming to execute the necessary simulations, and it may be difficult to achieve results that are precise enough. Simulation-based performance analysis of a model involves a statistical investigation of output data, the exploration of large data sets, the appropriate visualisation, and the verification and validation of simulation experiments.

2.7 Proposed Model for Metadata Search Mechanism

At a conceptual level, the geodatabase consists of a multitier architecture that implements advanced logic and behavior in the application tier on top of a data storage tier. The database tier can be further subdivided into two schemas – the Metadata and the Geodatabase. The geodatabase combines geo (spatial data) with database (specifically a relational database management system or RDBMS).

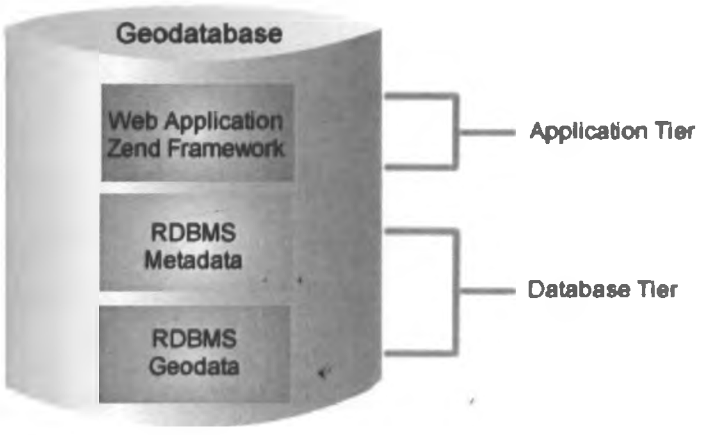


*Figure 2: **Figure showing geodatabase tier***

On the database tier, the RDBMS provides a simple, formal data model for storing and managing information in tables. The metadata model in the RDBMS will manage data about the containers of data. Names of all tables in the geodata RDBMS will be stored in a metadata table. Names of all columns in the geodata RDBMS will be stored in the metadata columns table. Names of all primary keys and foreign keys will be stored in their respective tables in the metadata schema.

The application tier accesses the geodatabase through the metadata. The logic of the metadata search mechanism will seat in the application. Users will execute queries in the application tier. The application will have the search algorithm which forms the basis of our proposed metadata search mechanism. The search function will intelligently relate the search string and the metadata and then down to the required row in the geodatabase.

The search mechanism will allow users search based on simple sequel language (SQL) queries by providing the WHERE part of the SQL query. The search will also advanced search interface. The model of the proposed metadata search mechanism is elaborated in section 3.4. A pseudocode is also provided in section 3.4.1.

Chapter 3: Methodology

In this chapter, we focus on the details of how we plan to solve the research problem earlier identified in chapter 1, which is summarized as: To model a geodatabase system at NDC for prompt data retrieval and business reporting, using the solution that was proposed in chapter 2; which is a proposed metadata search based geodatabase system developed using Oracle database and Zend Framework tools, enhanced using Oracle search techniques. To determine if we can achieve the objectives earlier set chapter 1, in line with this research problem, we discuss the following important things:

- Conceptual system model
- Conceptual model of the geodatabase
- Logical models of the geodatabase
- Logical models of the Metadata
- Proposed search mechanism
- Pseudocode of the metadata search algorithm
- Detailed flow chart of the metadata search mechanism
- Measuring the performance
- Our research design method
- The system implementation tools and data acquisition
- The limitations of the chosen methodology

3.1 Conceptual System Model

From the current file management system at NDC presented in 2.3.5, we derive the focus of this paper.

The current system is document managed based and lacks the concept of a geodatabase. A geodatabase combines geo (spatial data) with database. The RDBMS provides a straightforward formal structure for storing and managing the exploration data in tables; Data storage and retrieval are implemented with simple tables. Certain characteristics of geodata management, such as definition of attribute types, query processing, and multiuser transaction processing, are delegated to the RDBMS.

The following conceptual model design, illustrate the processes involved from how users in various exploration departments will provide high level queries to retrieve data from the geodatabase to accurate business reporting for better business decision making. It also shows that the geodatabase will integrate with other GIS systems for map presentation and interpretation.

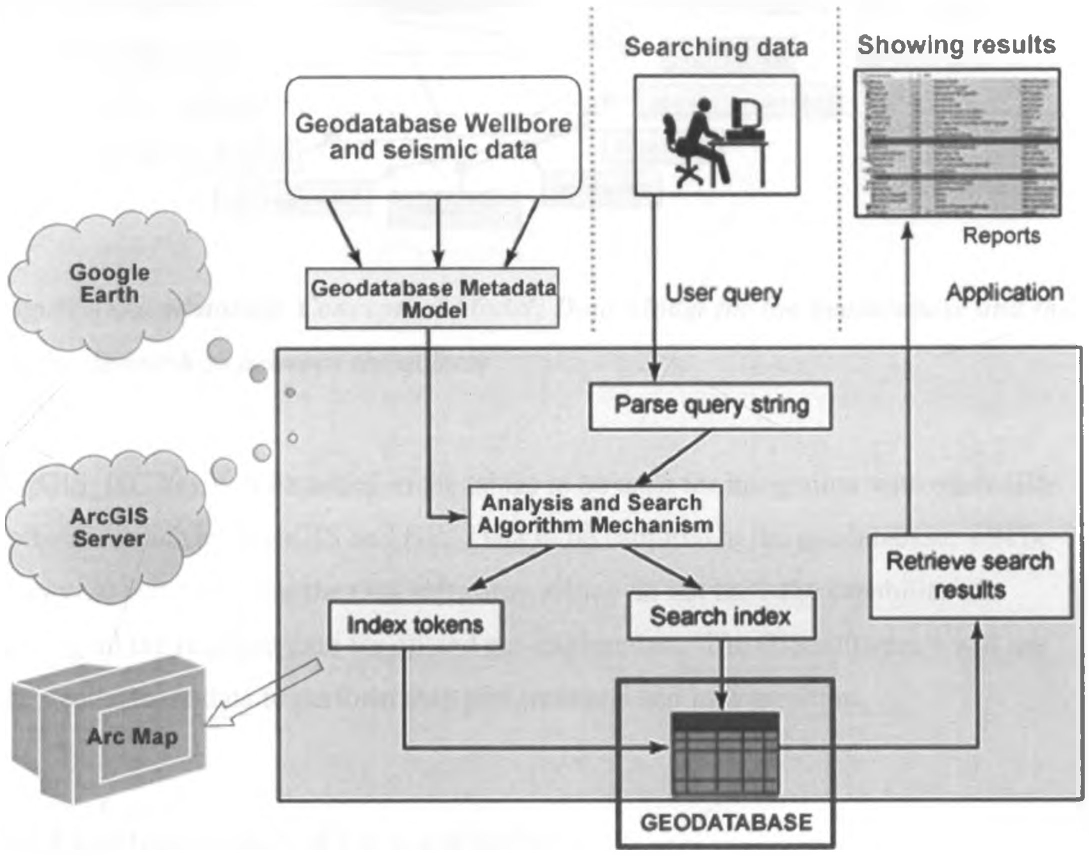


*Figure 3: **Conceptual System Model***

3.2.1 Conceptual Model of The Geodatabase

In the research work, we propose a high level conceptual model showing the different entities in our data and how they relate to one another. Figure 27, show the data models identified to be included in the geodatabase.

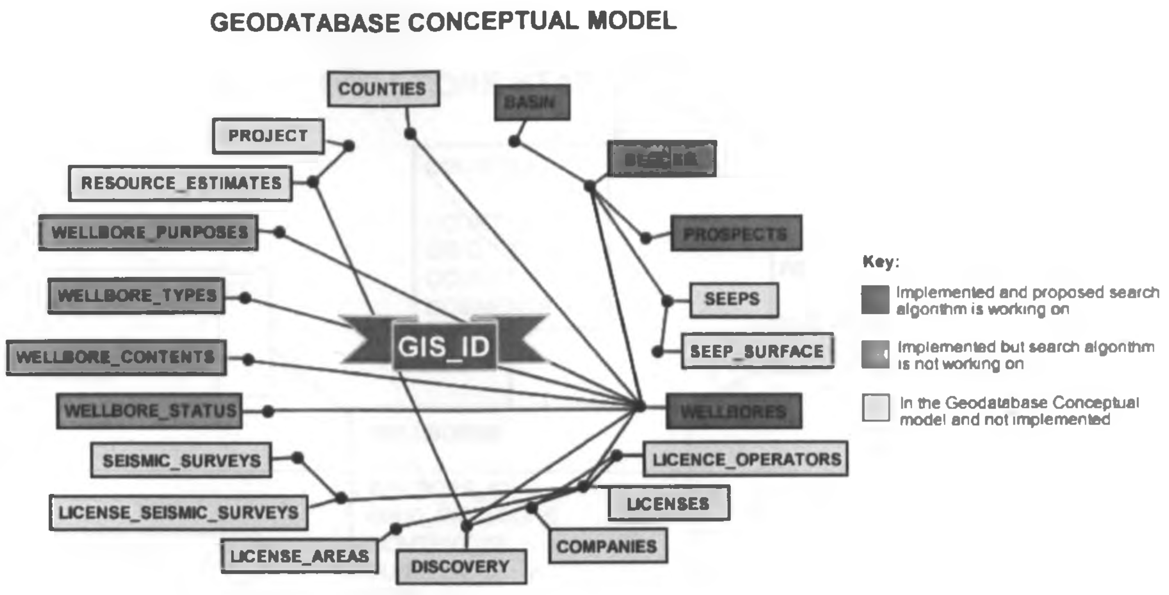


*Figure 4: **Geodatabase Conceptual Model; Data Model for the geodatabase and the inter-relationships between the objects***

A "GIS_ID" key will be added in the tables to be used for integration with other GIS software's such as ArchGIS and GE. Data to be captured in the geodatabase, will be very vital and useful for the GIS softwares, which do not have the capability of storing all the required data for oil and gas exploration. The GIS software's will use the geodatabase data to perform map presentations and interpretation.

3.2.2 Logical Models of the Geodatabase

From the conceptual model a logical data model is proposed so that we understand the details of our data. This will be the basis of forming a physical data model, from which we will know exactly how to implement our data model in the database of choice.

Below we show the star schemas of the conceptual model of the geodatabase. These logical models will map to the physical models of the geodatabase.

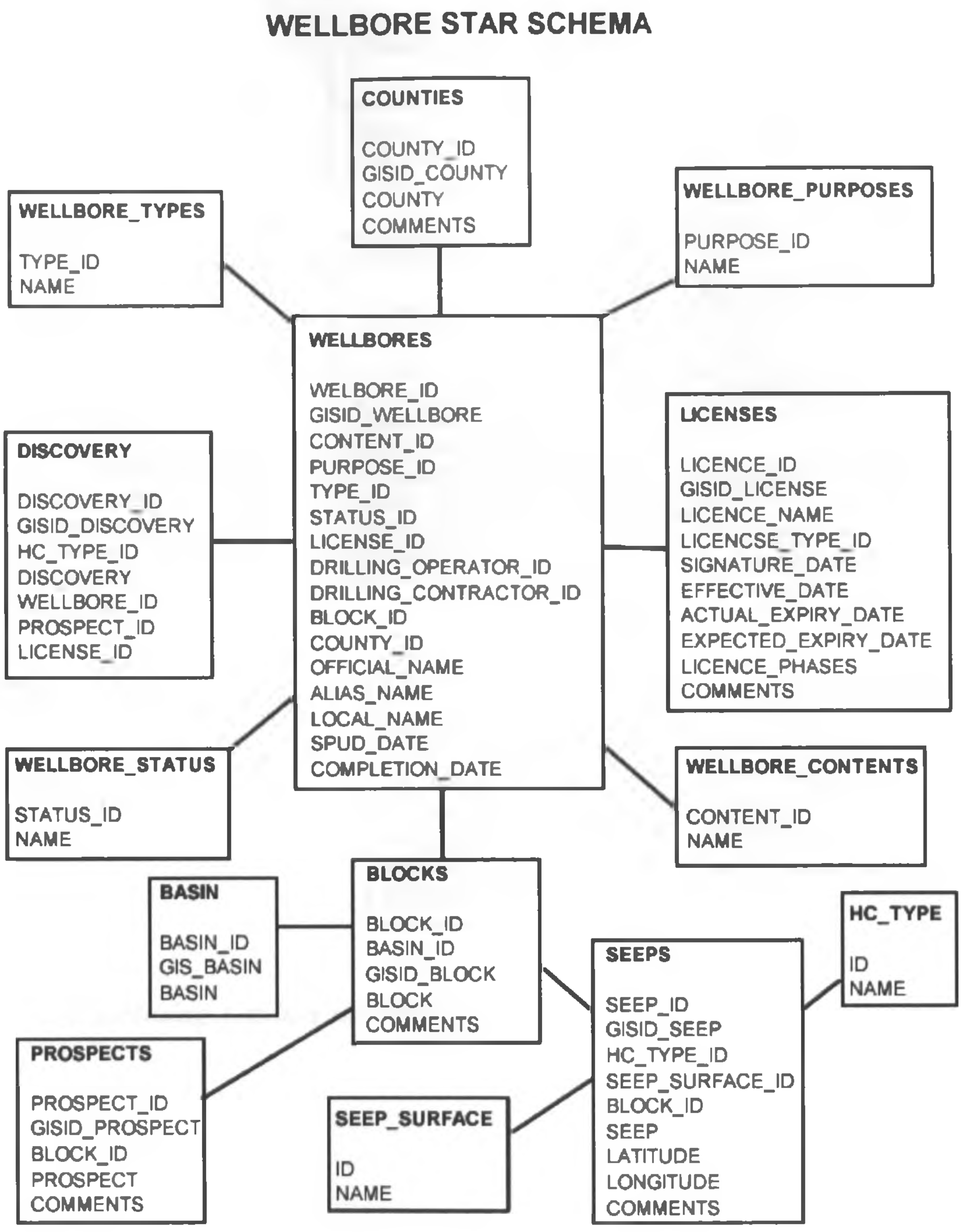


*Figure 5: **Wellbore Star Schema***

LICENCES STAR SCHEMA

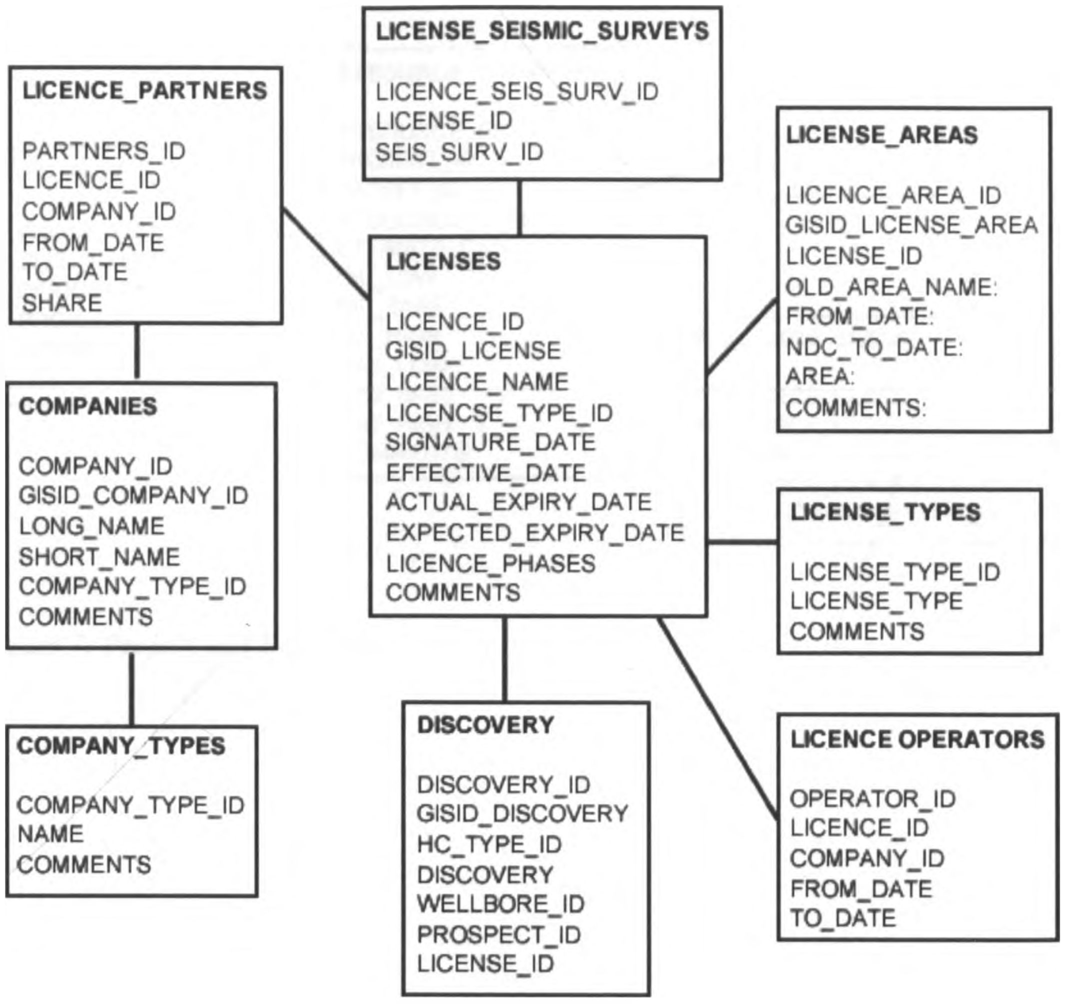


Figure 6: ***Licensing Activities star schema***

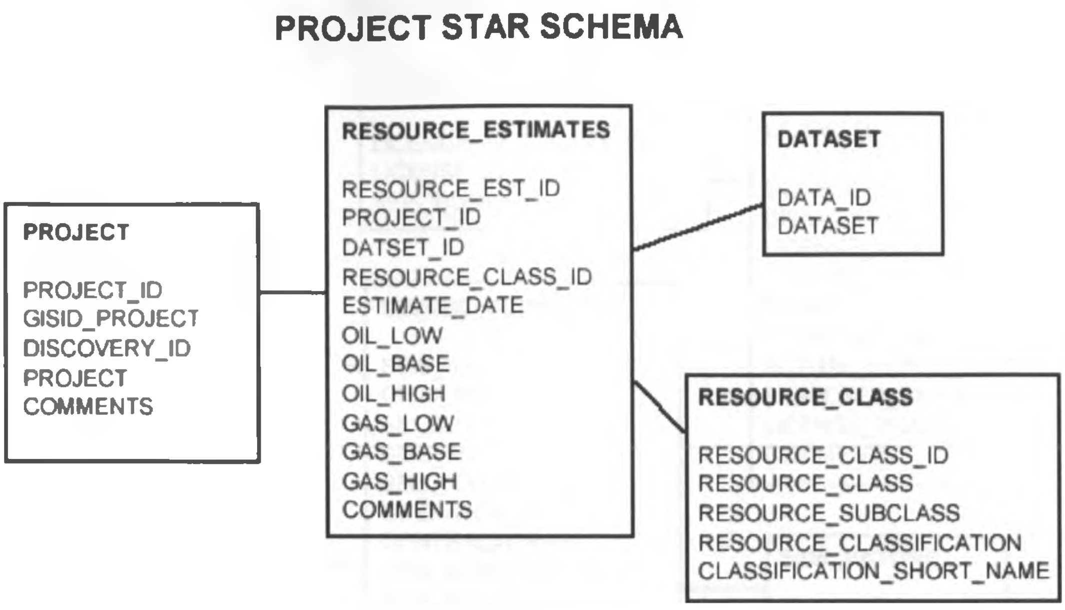


*Figure 7**: Project and Resource Estimates star schema***

SEISMIC STAR SCHEMA

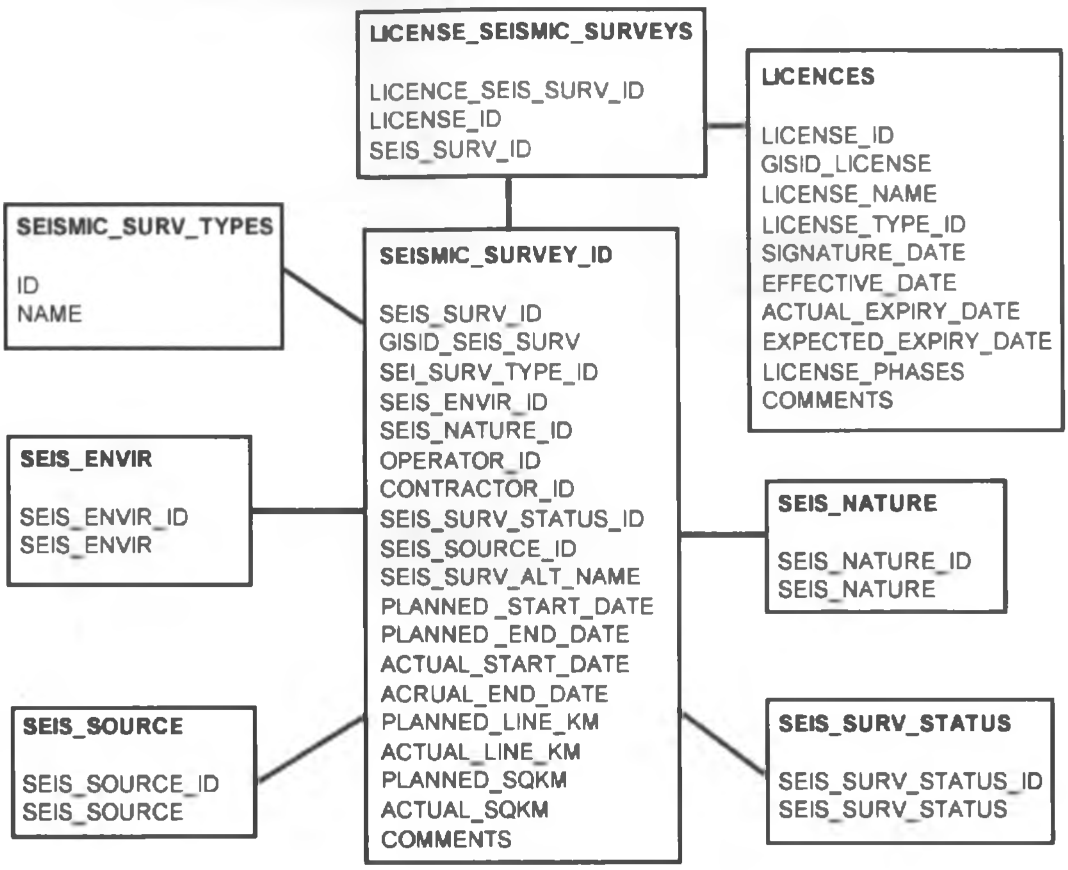


*Figure 8: **Seismic Survey star schema***

3.3 Logical Model Design of the Metadata

The metadata schema will integrate and seat on the same database as the geodatabase schema

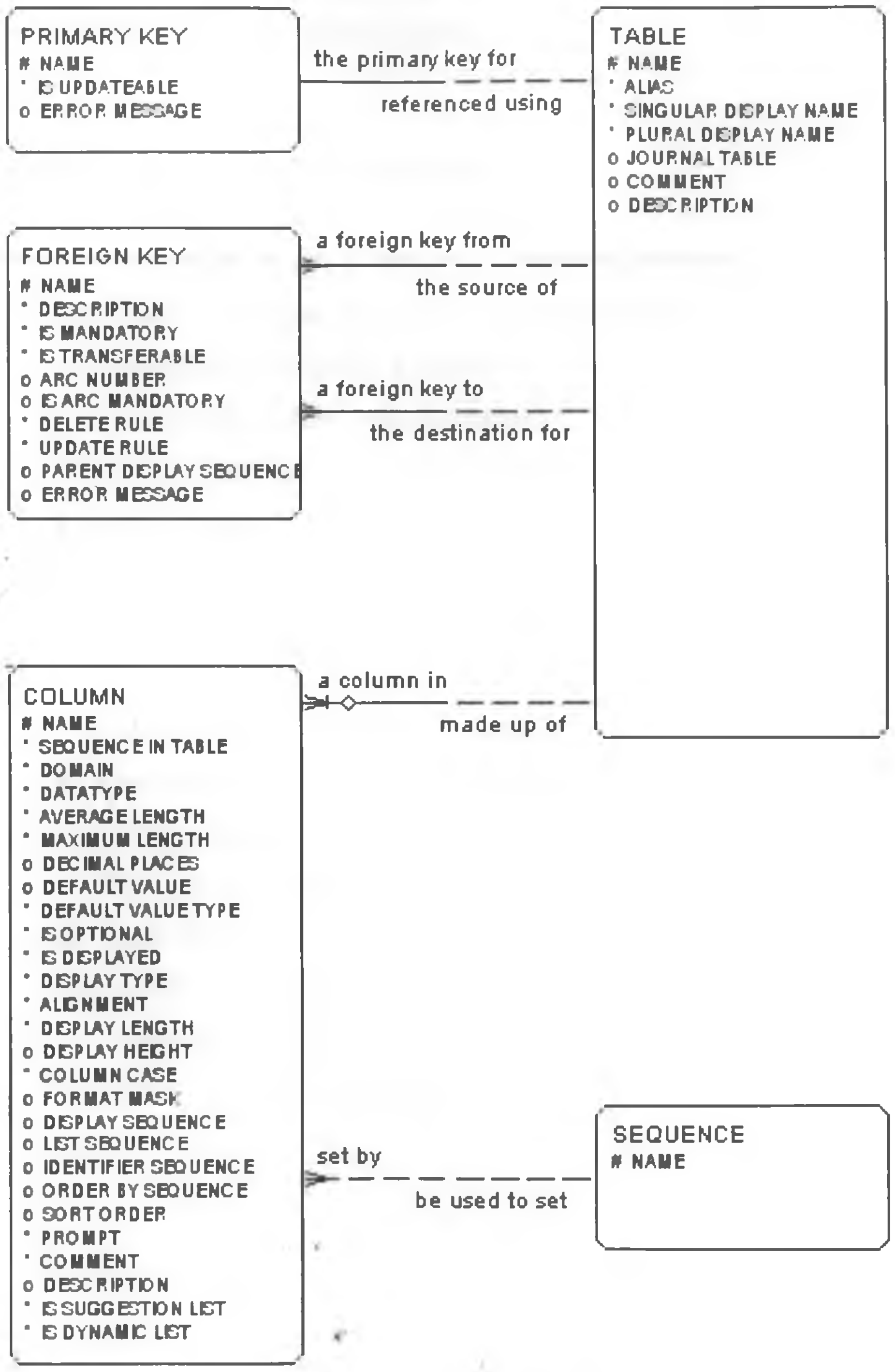


*Figure 9: **Logical Model Design of the Metadata Model***

The metadata model design will form the basis of our enhanced search mechanism. This is the database that stores the data about the geodatabase. It has two main tables namely MTD_TABLES and MTD_COLUMNS which store information about the geodatabase tables and columns respectively.

3.4 Proposed Search Mechanism

3.4.1 Pseudocode of the Metadata Search Algorithm

The following is the pseudocode of the metadata search:

```
1. Check if the request has a search
2. Get the search value from the request
3. Redirect null searches to the display page
4. Get the model or table for database transaction
5. Get the column metadata for model in step 4
6. Check for column metadata whose search sequence has a value
7. Check for a SQL WHERE clause, e.g. = < > LIKE BETWEEN IN and Ignore
   UNION to avoid SQL injection attacks
8. Check that this is a valid WHERE clause
9. Use the SQL as is
10. If we get an exception then perform the default i.e. non-SQL search
11. If we didn't find a SQL WHERE clause perform a default search
12. Use search items (if defined) or NAME columns
13. If search columns have been defined then use them
14. Otherwise search any NAME columns
```

3.4.1 Detailed Flow Chart of the Metadata Search Mechanism

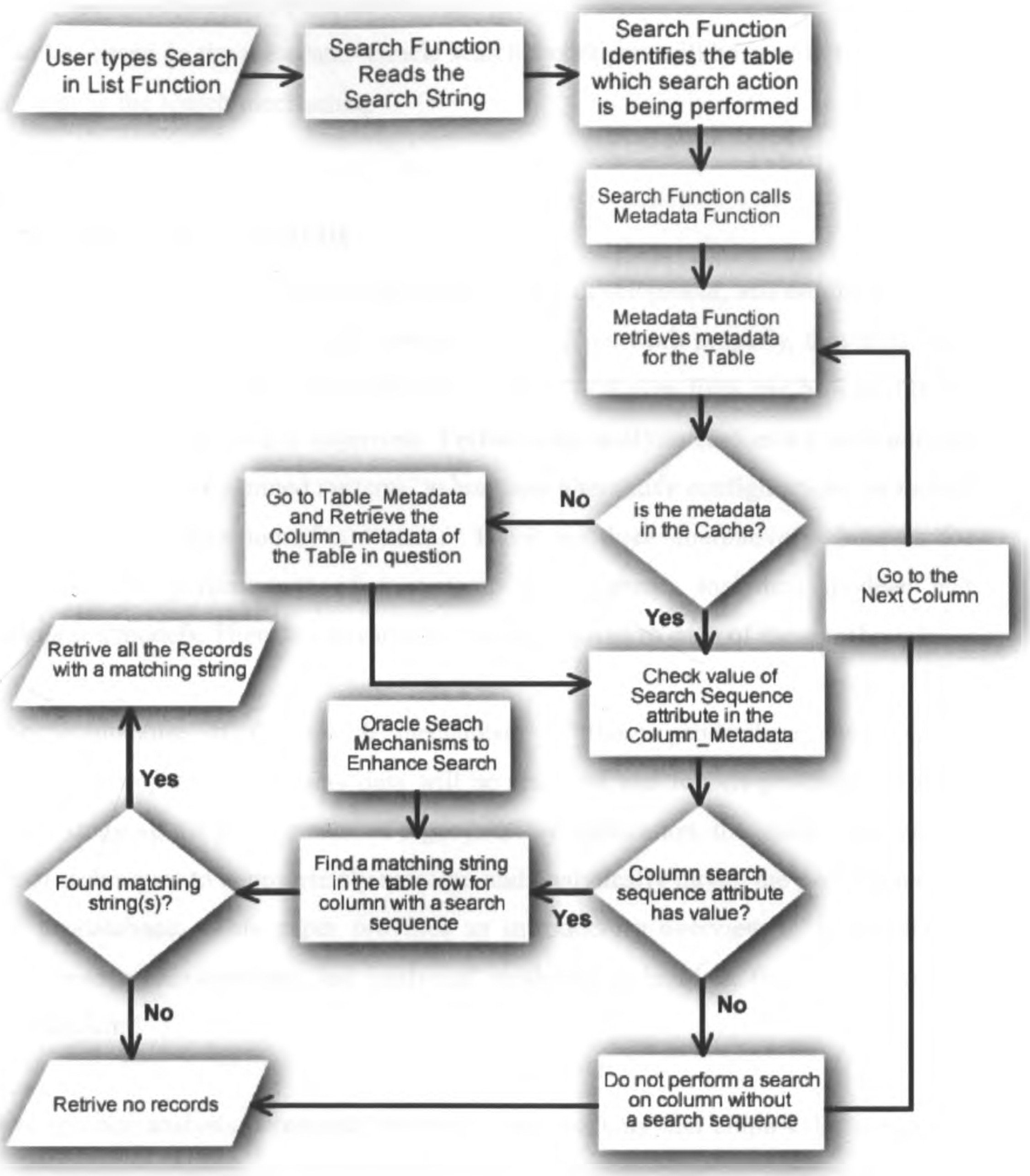

User types Search in List Function
Search Function Reads the Search String
Search Function Identifies the table which search action is being performed
Search Function calls Metadata Function
Metadata Function retrieves metadata for the Table
is the metadata in the Cache?
No
Go to Table_Metadata and Retrieve the Column_metadata of the Table in question
Yes
Go to the Next Column
Retrive all the Records with a matching string
Check value of Search Sequence attribute in the Column_Metadata
Oracle Seach Mechanisms to Enhance Search
Yes
Found matching string(s)?
Find a matching string in the table row for column with a search sequence
Yes
Column search sequence attribute has value?
No
No
Retrive no records
Do not perform a search on column without a search sequence

Figure 10: ***System Model Flow Chart for the Metadata Search Mechanism***

From the conceptual model we move on to the logical data model illustrated by the flow chart above, so that we understand the details of how the enhanced search mechanism will function. Object oriented programming is the choice of the system coding using Zend Framework technology. A search function is developed to perform searches based on the metadata. Oracle search techniques will be used in the backend to enhance the search mechanism.

3.5 Performance Analysis

Performance is often a central issue in the design, development, and configuration of systems. It is not always enough to know that systems work properly, they must also work effectively. Studies show that time, money, and even lives can be saved if the performance of a system is improved. Performance analysis studies are conducted to evaluate existing or planned systems, to compare alternative configurations, or to find an optimal configuration of a system. There are three alternative techniques for analyzing the performance of a system: measurement, analytical models, and simulation models. There are advantages and drawbacks to each of these techniques.

The architecture of the search mechanism of these geodatabase systems has significant impacts on the way data will be retrieved and reports produced on time. When studying the performance of a geodatabase application, the analyst will need a set of techniques to appropriately measure and analytically model the performance of the geodatabase. This paper provides an introductory overview to geodatabases, performance measurement, and analytical modeling techniques, focusing on search mechanism.

Performance analysis commonly involves benchmarking and empirically measuring performance to validate or assist with creating analytical models of performance

3.5.1 Analytical Models

Analytical models (e.g Markovian models), can provide exact results regarding the performance of a system. The results are exact, in that they are not estimates of the performance of the system. However, the results provided by analytical models may or may not be accurate, depending on the assumptions that have been made in order to create the model. In many cases it is difficult to accurately model industrial-sized systems with analytical models. In fact, it has been observed that when analysing computer systems "analytical modeling requires so many simplifications and assumptions that if the results turn out to be accurate, even the analysts are surprised." (Jain, 2001)

3.5.2 Simulation-Based

Simulation-based performance analysis can be used as an alternative to analytical techniques. Simulation can rarely provide exact answers, but it is possible to calculate how precise the estimates are. Furthermore, larger and more complex models can generally be created and analysed without making restrictive assumptions about the system. There are two main drawbacks to using simulation: it may be time consuming to execute the necessary simulations, and it may be difficult to achieve results that are precise enough. Simulation-based performance analysis of a model involves a statistical investigation of output data, the exploration of large data sets, the appropriate visualisation, and the verification and validation of simulation experiments.

3.5.3 Measuring the Performance

Performance analysis is both an art and a science. One of the arts of performance analysis knows which of these three analysis technique to use in which situation. Measurement can obviously not be used if the system in question does not exist. Simulation should probably not be used if the system consists of a few servers and queues, in this case queuing networks would be a more appropriate method. Simulation and analytic models are often complementary. Analytic models are excellent for smaller systems that fulfill certain requirements; such as exponentially distributed inter arrival periods and processing times. Simulation models are more appropriate for large and complex systems with characteristics that render them intractable for analytic models. Performance analysts need to be familiar with a variety of different techniques, models, formalisms and tools. Creating models that contain an appropriate level of detail is also an art. It is important to include enough information to be able to make a reasonable representation of the system; however, it is equally important to be able to determine which details are irrelevant and unnecessary.

Test bed performance measurement analysis is the proposed approach to model the data retrieval and reporting enhancement of a geodatabase system. It involves conducting real world field experiments using actual data test beds. Real world performance measurements are the most ideal methods of obtaining realistic geodatabase performance characteristics. It is a good means of validating analytical and simulation models. This will therefore involve developing geodatabase system prototype and use test bed performance measurement to test and validate its performance.

3.5.4 Measuring Tools

We have familiarized with tools that can be used to monitor systems, analyze data, and present results. The tools used for development of the prototype and analysis include the following:

- Zend Framework function Zend_Log and PHP function microtime() to log the CPU time used to perform a search in the geodatabase frontend.
- Oracle Database; Oracle "set timing on" command used to capture the time taken to perform a an SQL query command in Oracle.

Performance measurement tools will be used to gain insight into the search performance. One of the most basic measures of performance is how much elapsed time an application takes from the user's submitted request to the system's completed response, or response time. In an Oracle environment, the "set timing on" command provides a straightforward way to measure elapsed time of any command (e.g. an

SQL SELECT command).

This information may be considered "coarse grained" because it only provides the total time spent running the command, rather than data about the time spent in individual function calls. Unlike several other tools discussed in this section, the "set timing on" command does not require recompiling the source code to support gathering data.

The Zend Framework Zend_Log and PHP microtime() functions will be used to measure the user CPU time to process a search performed by a user in the geodatabase application. Other PHP function will also provide memory usage.

We will use the results to plot charts showing the improved search time against the original time taken to perform a search in the geodatabase.

3.5.5 Measuring Parameters

The measurement parameters will be the number of record and time taken to execute a certain query within a given sample size.

Chapter 4: Results and Evaluation

In chapter 3, a detailed logical model of the geodatabase was presented. A key deliverable of this chapter was a working prototype of an Oracle geodatabase backend and a frontend application developed using Zend Framework technology. In this chapter, we conduct actual analysis experiments of the prototype in order to analyze performance of its search mechanism.

4.1 Objectives of Evaluation

- Demonstrate improved search performance using adaptive sampling
- Demonstrate how the metadata architecture improve the search mechanism using adaptive sampling
- Report the results and findings

4.2 A Distributed Adaptive Sampling

Simulations were carried out using the Zend Framework geodatabase application prototype and the Oracle database, to demonstrate the improved CPU time for data retrieved from the Oracle database when the Oracle search techniques are implemented.

4.2.1 Adaptive Sampling Parameter Configurations

In the frontend application configuration file a log render time to record the CPU time is set as shown below:

```
registry.logRenderTime = 0.01
```

The application prototype continuously generates random samples. The values generated are stored in a log file called "execution_time.log". The outputs of these random samples represent the CPU time a search took to process a query.

4.2.2 Measuring Geodatabase Record Query Performance

In an Oracle environment, the "set timing on" command provides a straightforward way to measure elapsed time of an SQL search command. Values generated from this sampling will be used to plot charts showing the improved performance of the geodatabase. The charts will plot the execution time against the number of records.

4.3 Results of Analysis

Implementing the Oracle Search mechanisms reduced the CPU time spent in performing a query in the geodatabase. The tables and graphs below show the distribution of elapse time of SQL searches performed against the number of records of a table in the geodatabase.

4.3.1 Comparing Execution time for 8k and 16k Tablespaces with Metadata Indexing

We demonstrate improved performance by setting a higher tablespace size on tables and also by applying the metadata search technique. In the metadata search algorithm, every column in a table can be indexed and thus retrieving a row in a split of a second. A simple query is executed as shown below and we achieve the following results.

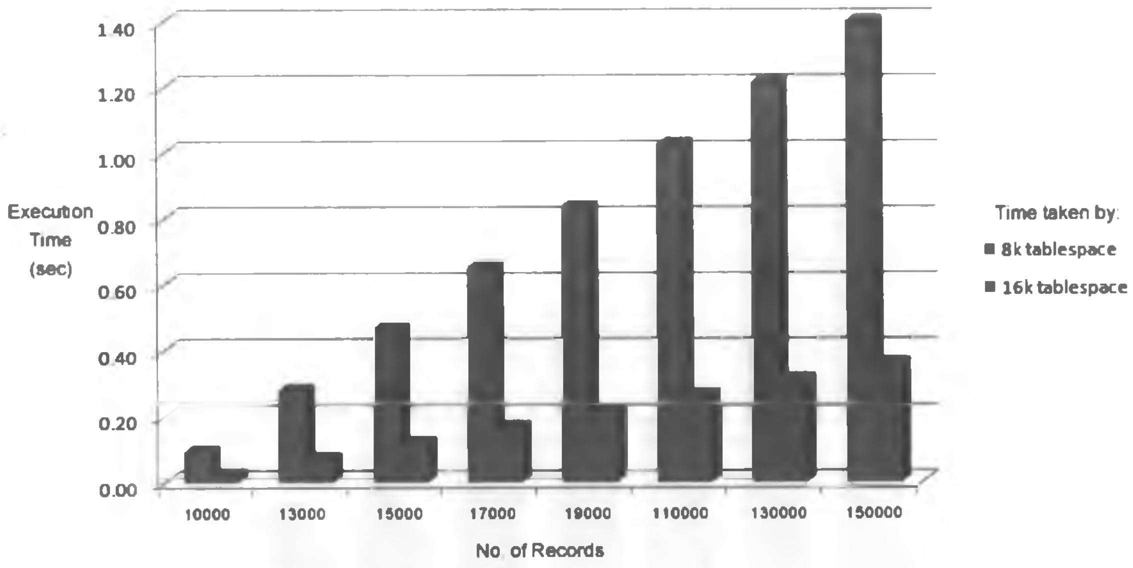
```
SELECT count(well_log_name) FROM  well_logs WHERE
log_date>to_date('27/01/2010','dd/mm/y
and log_date <to_date('06/03/2012','dd/mm/yyyy')
```

No. of Records (Sample space)	Time taken by non Metadata Indexing 8k Table Space	Time taken by Metadata Indexing 16k Table Space
10000	0.09	0.02
30000	0.28	0.07

50000	0.47	0.12
70000	0.65	0.17
90000	0.84	0.22
110000	1.03	0.26
130000	1.21	0.31
150000	1.40	0.36

*Table 2: **Improved time by setting Large Blocksize on Tablespace with metadata indexing***

The graph below plots samples of number of records against the time taken to perform a search on various sample spaces.



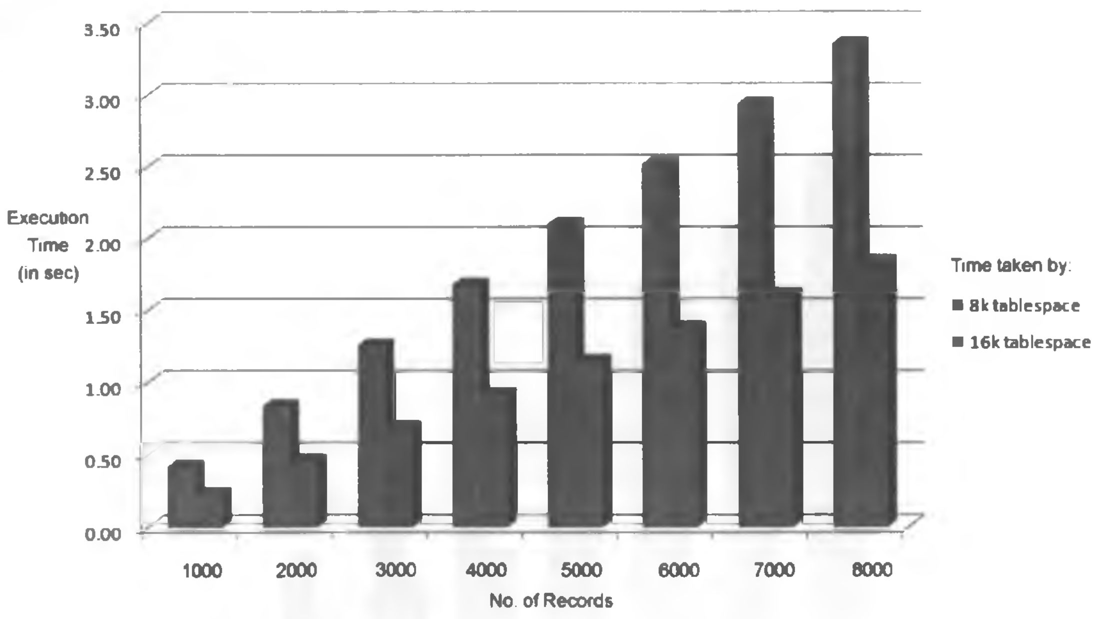
Graph 1: Showing Improved time by Setting Large Blocksize on Tablespace with metadata indexing

4.3.2 Comparing Execution time for 8k and 16k Tablespaces

The table below demonstrates improved performance by setting a higher tablespace size on data tables. The results were obtained by executing a complex query that involved more than one table.

No. of Records	Time taken by 8k Tablespace	Time taken by 16k Tablespace
1000	0.42	0.23
2000	0.84	0.46
3000	1.26	0.69
4000	1.68	0.92
5000	2.10	1.15
6000	2.52	1.38
7000	2.94	1.61
8000	3.36	1.84

*Table 3: **Improved time by Setting Large Blocksize on Tablespace***



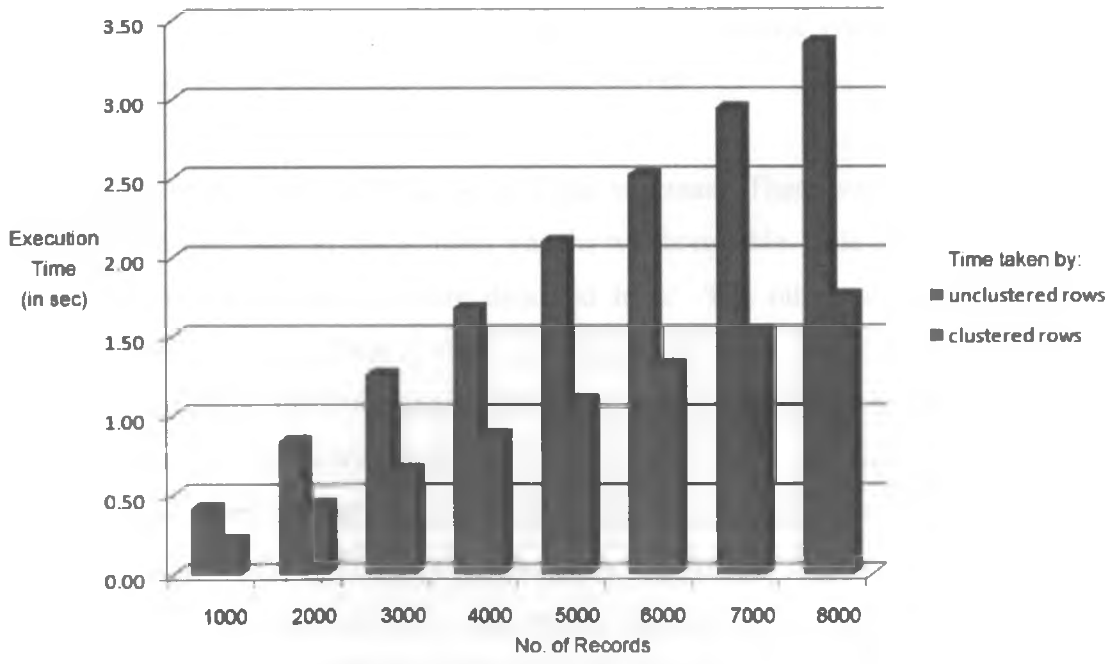
Graph 2: Showing Improved time by Setting Large Blocksize on Tablespace

4.3.3 Comparing Execution time for a table that is Row Resequenced and one that is not

The table below demonstrates improved performance of time taken by a query on a resequenced table.

No. of Records	Non Resequenced Table	Row Resequenced Table
1000	0.42	0.22
2000	0.84	0.44
3000	1.26	0.66
4000	1.68	0.88
5000	2.10	1.10
6000	2.52	1.32
7000	2.94	1.54
8000	3.36	1.76

*Table 4: **Showing Improved time by Row Resequencing***



Graph 3: Showing Improved time by Row Resequencing

Chapter 5: Discussion and Conclusion

5.1 Achievement of Objectives

The objective of this study was to model a geodatabase for National Data Centre that will provide a structured data storage and retrieval mechanism of the exploration data. The current file system used at NDC does not offer this ability.

The geodatabase had two parts in the database tier. This included the geodata schema and the metadata schema. The geodata schema contained tables that hold wellbore and seismic data. It also hold all other related data collected during the exploration activities. A questionnaire was conducted among expert in the exploration field who included geophysist, geochemist and geologist, to identify objects that were required to model the geodatabase. A conceptual model of the geodatabase was presented in Chapter 3 section 3.2.1. A few of these objects were implemented in the prototype to be used to perform test bed performance measurement analysis. Some objects were implemented in the geodatabase but not linked to the metadata, while other objects were in the geodatabase conceptual model but were not implemented in the geodatabase. Implementation of a fully functional geodatabase system to be used in the oil and gas sector was considered to be future work.

The geodata objects identified had several star schemas. There was a wellbore star schema. The core table in this schema was the wellbore table. It is considered as an information carrier of the wellbore data and logs. The other tables referencing wellbore formed a relationship in this star schema. Objects in this schema hold data about the identified exploration basins and prospects of exploration opportunities. It also contained data about wells and seeps. Reference tables in this schema contained types of hydrocarbon and types of wellbores. The second star schema identified was the license star schema. The schema identified objects that needed a license to be operational. These included wells, discoveries, seismic surveys, companies, license areas and operators. The reference tables in this schema included license types. The third star schema was the project star schema. This contained data of the ongoing projects and the resource estimates in the projects. This give more information about the contents discovered in the wells. The fourth was the seismic star schema. It explored data of the seismic surveys conducted. It related with the licenses objects because seismic surveys are subject to licenses. Reference tables in this schema were seismic source, seismic environment, seismic types, seismic survey status and seismic nature.

In general, a well modeled geodatabase has plenty of advantages. It will offer the ability to do the following:

- Store a rich collection of spatial data in a centralized location.
- Apply sophisticated rules and relationships to the data.
- Define advanced geospatial relational models (e.g., topologies, networks).
- Maintain integrity of spatial data with a consistent, accurate database.
- Work within a multiuser access and editing environment.
- Integrate spatial data with other IT databases.
- Support custom features and behavior.
- Leverage spatial data to its full potential.

The metadata schema had tables that contained data about the containers of data. The following were the main tables in the metadata schema:

- MTD_TABLES which contained the names of all the tables in the geodata schema.
- MTD_COLUMNS which contained the names of the columns in the geodata schema
- MTD_PRIMARY_KEYS which contained the names of the primary keys in the geodata schema
- MTD_FOREIGN_KEYS which contained the names of the foreign keys in the geodata schema

The metadata formed the basis of the retrieval techniques of the data in the geodata schema. The integration of the two schemas offered the ability of an enhanced search mechanism.

The metadata search algorithm concept for the frontend was coded using Zend Framework technology. Zend framework is a fully fledged object oriented programming tool. The search functions and metadata algorithms were developed that also integrated with the libraries provided by Zend Framework to enhance the search mechanism. Zend_Log a Zend library was also used to log the CPU time taken to perform a search query. The time recorded in the log file to perform a search was plotted against the sample space, to give an analysis of the performance of the search algorithm.

5.2 Achievement of Research Questions

5.2 Discussion of Results in Light of Literature Review

The measuring performance methodology presented in section 4.3.1 showed measurement of time taken to execute a query against a sample size. The measurement compared two scenarios: Measuring the time taken to execute a query using the metadata search mechanism and measuring time taken by the same query without employing the search mechanism. The results showed that the former was faster.

The metadata search algorithm used the metadata for each column in a given table to locate the search row in a query. The performance of this indexing caused the search run four times faster.

5.3 Limitations

Zend Framework is highly modulized and hence we needed more time to know how its internal works. We only implemented the necessary functions to develop a prototype to validate the research work. For example, Zend Framework has a search function known as Zend Lucene Search. We needed more time to implement this function which will definitely be used by the geodatabase system for document management and archiving purposes.

Oracle database extensive. Implementing an Oracle technique requires thorough tuning to ascertain that the mechanism applies for that particular database. Each database behaves differently and through the process of optimization we can establish what would apply best. The techniques that didn't well apply for the project such as cluster groups were not implemented.

5.4 Recommendation

Developing a geodatabase based on metadata model proves a superior method of database design. The metadata design also goes a long way of improving the search mechanism. This research demonstrate performance enhancement of a geodatabase system for prompt data retrieval and accurate business reporting for provision of better business decision making in the oil and gas exploration field. A geodatabase with excellent data management having information presented in a unified and open way will also help the oil companies operating inside our country explore the opportunities available in the Exploration and Production sector. This will also attract new companies and investors.

A clearer conceptual framework of how the search mechanism will be enhanced was arrived at. We recommend finishing the remaining work and writing a paper. We propose the development of a functional geodatabase and implementation of the same at NDC.

5.5 Future Work

We propose to develop a fully fledged functional geodatabase system that will be used at the National Data Center; the Exploration and Production department of National Oil Corporation of Kenya. All objects presented in the geodatabase

Conceptual Model in section 3.2.1 and in section 3.2.2, will be implemented in the proposed geodatabase system.

Further research work should look into GIS internet mapping application and integration with other GIS systems that do map presentation and interpretation so as to have a complete GIS suite for the Exploration and Production sector.

References

Armstrong, The Quarks of Object-Oriented Development, 1998

Baker, Michael, "GIS DATABASE DESIGN: Geodatabase Model" Jul 2000

Burleson, Donald, "Oracle Clustering Factor", Oct 2011

Burleson, Donald, "Oracle sorted hash cluster", Sep 2011

Burleson, Donald, "Reduce I/O with Oracle cluster tables", Jun 2011

Burleson, Donald, "The latest consensus on index rebuilding", Nov 2007

C. J. Date, Hugh Darwen. Foundation for Future Database Systems: The Third Manifesto (2nd Edition)

C. J. Date, Introduction to Database Systems, 6th-ed., Page 650

Chaudhuri, S. and Kyuseok, S. (1999). Optimization of queries with user defined predicates. ACM Transactions on Database Systems, vol.24 No.2, pages 177-228.

Connoly, T. and Begg, C. (2001). Database Systems: A practical Approach to design, Implimentation and Management, Third Edition. Edison and Wesley.

Cosar, A. Lim, E. and Jaideep, S. (2001). Multiple query Optimization with depth-first branch and bond and dynamic queryordering. International Conference on Information and Knowledge Management.

Dan Hotka. Oracle8i GIS (Geeky Internal Stuff): Index Internals. OracleProfessional, November, 2000.

Eduardo Cobian, Easy PHP websites with the Zend Framework, 2010

Elmasri, R. and Navathe, B.S.(1994). Fundamentals of database systems, Second Edition. Benjamin Cummings.

Fegaras, L. (1998) A new Heuristic for Optimising Large Queries Research Paper, Department of Computer Scienceand Engineering . The University of Texas at Arlington.

Graefe, G. and DeWitt, D.J. (1987). The EXODUS OptimizerGenerator ACM SIGMOD Records, Volume 16, Issue 3 pages 160-172

Graefe, G. and McKenna,W.J. (1991).Extensibility and Search efficiency in the Volcano Optimiser generator. TechnicalreportCU-CS-91-563. University of Colorado

Gupta, A.Sudarshan, S.Viswanathan, S.(2001) QueryScheduling in mutiquery optimization Research paper, Indian Institute of Technology - Bombay.

Gutmans, Andi, "Zend Framework". Andi on Web & IT, Feb 2009
http://www.suffolkva.us/gis/docs/database-design.pdf

Horowitz, E. Sahni, S. and Rajasekaran, S. (1996). Computer Algolithms C++ , Computer Science Press.

Hurigeri, S. Seshadri, S. and Sudarshan, S. (2001). Memory Cognizant Query Optimization. Research paper, Indian Instituteof Technology - Bombay.

IBM Oracle Technical "Oracle Architecture and Tuning on AIX" Nov 2006

Jan van Leeuwen (Ed.): Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, 1990

Jan van Leeuwen (Ed.): Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, 1990

John C. Mitchell, Concepts in programming languages, Cambridge University Press, p.278, 2003

John Wang (Ed.): Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes), 2009

Johnson, J.L.(1997). Database: Models, Languages, Design, Oxford University Press.

Kernel, Sean Michael.,"Google Data Joins PHP Zend Framework", Dec 2006

Kremer, M. and Gryz, J. (1999). A Survey of Query Optimization in Parallel Database. Technical Report CS-1999-04, Department of Computer Science,York University.

Kroger, J.Stefan, P.and Heuer,A.(2001).Queryoptimisation.On theordering of Rules.Research paper, cost - and rule based optimisation of object - oriented queries(CROQUE) Project.Universityof Rostock and University of Hamburg, Germany.

Krill, Paul, "Microsoft, Zend boost PHP for Windows". infoworld.com. Oct 2006

Kyuseok, S. Sellis, T.K and Nau, D. (1994). Improvements on a Heuristic algorithm for Multiple-queryOptimization. Technicalreport, University of Maryland, Department of Computer Science.

Kyuseok, S. (1993). Advanced query optimization techniques for relational database systems. PhD dissertation, Universityof Maryland.

LaMonica, Martin, "IBM backs open-source Web software", Feb 2009

Ling Liu, M. Tamer Özsu (Eds.): Encyclopedia of Database Systems. Springer US 2009

Mark Gurry, O'Reilly book Oracle SQL Tuning Pocket Reference, Page 66, Jan 2000

Michael Lee Scott, Programming language pragmatics, 2006

Mohan, C. An Efficient Method for Performing Record Deletions and Updates Using Index Scans

Neward, Ted, "The Vietnam of Computer Science". Interoperability on Large Databases, August 2002

Nilesh, N.V. Sumit, K.S. Roy,P. and Sudarshan, S. (2001). Pipelining in multi-query optimisation. Research paper, Indian Institute of Technology. Bombay.

Park,P.andSegar,A.(1988).Using common sub-expressions to optimise multiplequeries. Proceedings of the IEEE International Conferenceon Data Engineering.

Pierce, Benjamin. "What is Object-Oriented Programming?", 2002

Poll, Erik. "Subtyping and Inheritance for Categorical Datatypes", 2011.

Potter, Mike, "Adobe Contributing AMF Support to Zend Framework", Feb 2009

Ramakrishnan, R. and Gehrke,J. (2000). Database Management Systems Third Edition. McGraw Hill. Rao, J. and Ross, R.K. (2000) Power Pipelining for Enhanced Query Performance . Technical Report CUCS-007-00, Columbia University.

Roy,P.Seshadri,S.Sudarshan, S.and Bhobe,S.(2001).Efficient and extensible algorithms for Multiquery optimisation. Researh Paper,SIGMOD International Conference on management of data.

Sushil Kumar, "Oracle Database 10g: The Self-Managing Database", Feb 2003

Walid H. Shayya, "An Introduction to ArcView GIS", Jun 2011

Weier O'Phinney, Matthew. "Zend Framework 1.11.12 Released!", Jun 2012

Weier O'Phinney, Matthew. "Zend Framework 2.0.0beta5 Released!" Jul 2012

Zend Technologies Inc, Programmer's Reference Guide - Zend Framework, 2005

Zend, "History of PHP and related projects". The PHP Group, Feb 2009

Zend, "Introduction to Zend Framework". ZF Programmer's Reference Guide, Feb 2009

[illegible] Research [illegible]

[illegible]

Appendix I: Research Questionnaire

1. What objects can be modeled in a geodatabase for oil and gas exploration activities?
2. What information can be stored in an E&P geodatabase?
3. What are the information carrier objects of an E&P geodatabase?
4. How would you want the geodatabase integrate with ArchGIS?
5. What is the purpose of ArchGIS?
6. Who would be the users of the Geodatabase?
7. Will each department in E&P have separate privileges to different modules in the geodatabase?
8. What types of reports do you expect from the geodatabase?
9. Which appraisal and production areas are dynamic and which are static?
10. Any other important information you wish to state or clarify?

Appendix II: Source Code for Search Mechanism

The following are the metadata search and the search action source codes written in PHP using Zend Framework that were developed to demonstrate the effectiveness and flexibility of a metadata search mechanism.

```
/**
 * Metadata Search: Returns a where clause to perform a general search of the
table
 */
public function search($search = null) {
    $where = null;
        if ((strstr($search, ' = ') || strstr($search, '>')
          || strstr($search, '<') || strstr($search, ' LIKE ')
          || strstr($search, ' IS NULL') || strstr($search, ' IS NOT NULL')
          || strstr($search, ' BETWEEN ') || strstr($search, ' IN(') || strstr($search, ' IN
(')
          ) && (strpos($search, ' UNION ') == false)) {

          $select = $this->_db->select();
          $select->from($this->_name, $this->_cols, $this->_schema);
          $select->where($search);
          try {
             $stmt = $this->_db->query($select);

             $where = $search;
          } catch (Exception $e) {

          }
        }
        if ($where === null) {

           if (count($this->_searchItems)) {
              foreach ($this->_searchItems as $column) {
                 if ($where !== null) {
                    $where = $where . ' OR ';
                 }
                 $where = $where . '(' . $this->getAdapter()->quoteInto('UPPER(' .
$column . ') LIKE UPPER(?)', '%' . $search .'%') . ')';
              }

           } else {
              foreach ($this->_cols as $column) {
                 if (preg_match('/NAME/', $column)) {
                    if ($where !== null) {
                       $where = $where . ' OR ';
                    }
                    $where = $where . '(' . $this->getAdapter()->quoteInto('UPPER(' .
$column . ') LIKE UPPER(?)', '%' . $search .'%') . ')';
                 }
              }
           }
        }
     }
     return $where;
  }
```

The following code is the Search Action that calls the Metadata Search Function:

```
/**
 * **Search Action: Displays a list of a subset of the records for a model matching
certain criteria**
 */
function searchAction()
   {
      Lib_Log::functionStart(get_class($this),__FUNCTION__);

      if ($this->_request->has('search')) {
         $search= urlencode(urlencode($this->_request->getParam('search')));
         $this->_helper->Redirector-
>gotoRouteAndExit(array('searchstring'=>$search));
      }

      $search = $this->_request->getParam('searchstring');
      $search = urldecode($search);

      if ( $search == null || $search == 'SEARCH' ) {
         $this->_helper->Redirector->gotoRouteAndExit(array(), $resource . 'List');
      }

      $this->_helper->viewRenderer->setScriptAction('list');

      $table = $this->_helper->Model->getModel();
```

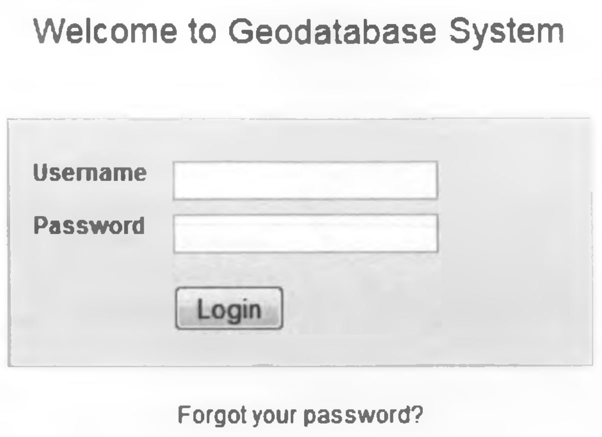
```
    $this->view->pageTitle = $table->getSingularDisplayName() . ' Search Results
for ' . $search;
    $this->view->headTitle($this->view->pageTitle, 'PREPEND');
    $this->view->modelClass = substr(strrchr(get_class($table), '_'), 1);

    $pageIcon = $this->_helper->Model->getIcon();
    if ($pageIcon !== null) {
        $this->view->pageIcon = $pageIcon;
    }

    $page = $this->_request->getParam('page');
    $sort = $this->_request->getParam('sort');
    $direction = $this->_request->getParam('direction');

    if ( strlen($sort) > 0 ) {
        $select->order(strtoupper($sort . ' ' . $direction));
    }

    $this->_preSearch($select);

    $paginator = Zend_Paginator::factory($select, 'DbTableSelect');
    $paginator->setCurrentPageNumber($page)
            ->setItemCountPerPage($table->getItemCountPerPage());
    $this->view->paginator = $paginator;
    $this->view->table = $table;
    $this->view->search = $search;

    $this->view->Breadcrumb()->add ( $this->_request->getRequestUri(), 'Search ' .
$table->getPluralDisplayName() . ' (' . $search . ')', false, 2);
```

```
    Lib_Log::functionEnd(get_class($this),__FUNCTION__);
}
```

Appendix III: User Manual

A. The Geodatabase Authentication

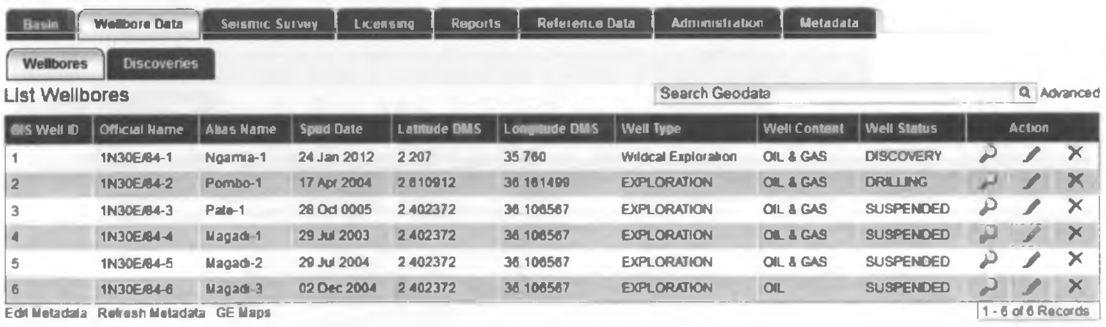
Below is a screen shot of the login screen of the geodatabase



When a user login to the geodatabase system, the system uses an Access Control Limit (ACL) that determines which objects the user has privileges to.

B. The Geodatabase List Module

Upon Login the system prompts the user to a default list of geodata records particularly the Wellbore data as shown below. It is from this list model that the Search mechanism has been implemented.

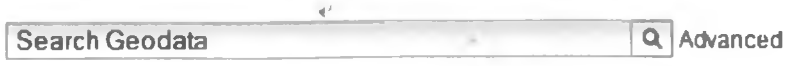


List Wellbores

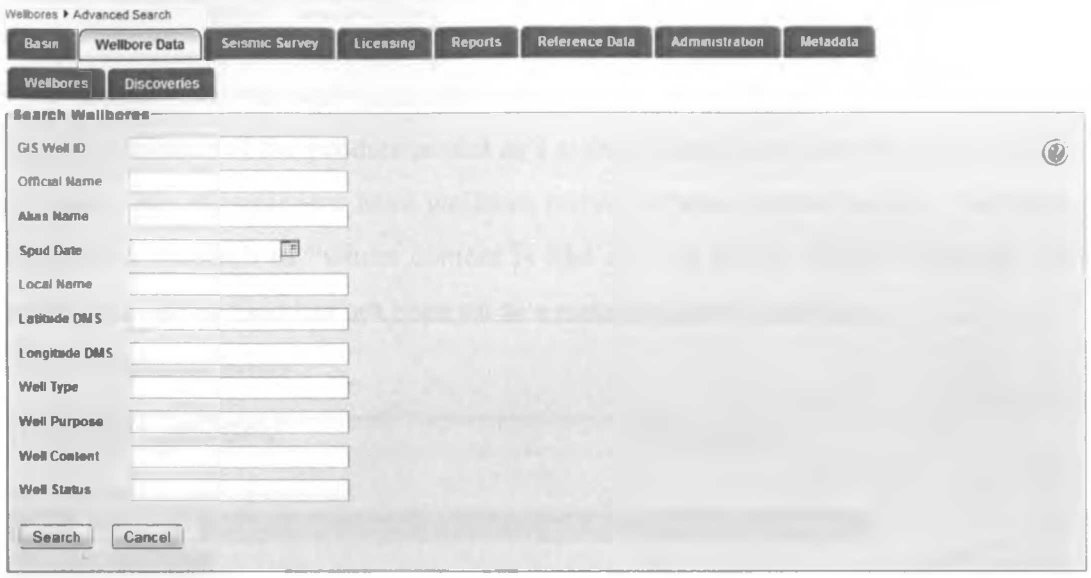
GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action
1	1N30E/84-1	Ngamia-1	24 Jan 2012	2 207	35 760	Wildcat Exploration	OIL & GAS	DISCOVERY	
2	1N30E/84-2	Pombo-1	17 Apr 2004	2 610912	36 181499	EXPLORATION	OIL & GAS	DRILLING	
3	1N30E/84-3	Pate-1	28 Oct 0005	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	
4	1N30E/84-4	Magadi-1	29 Jul 2003	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	
5	1N30E/84-5	Magadi-2	29 Jul 2004	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	
6	1N30E/84-6	Magadi-3	02 Dec 2004	2 402372	36 106567	EXPLORATION	OIL	SUSPENDED	

1 - 6 of 6 Records

There are two interfaces provided to perform the metadata based search i.e through the search text field and the advanced search screen as shown below.



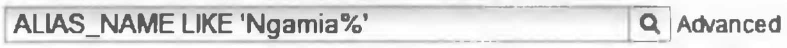
Upon clicking advanced search, it opens a form similar to the edit or add form of the geodata and a search can be performed based on any of the form items.



GIS Well ID	
Official Name	
Alias Name	
Spud Date	
Local Name	
Latitude DMS	
Longitude DMS	
Well Type	
Well Purpose	
Well Content	
Well Status	

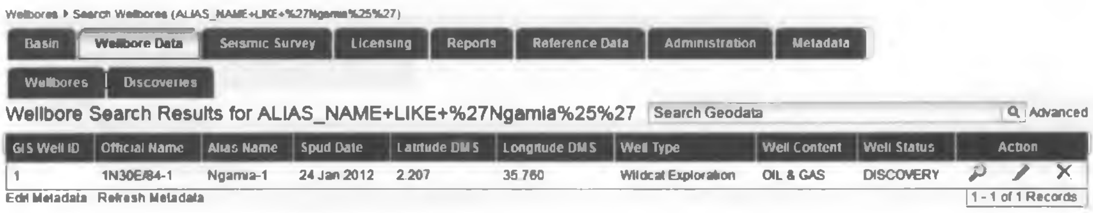
C. The Geodatabase Metadata Search

Search can be done by providing the WHERE CLAUSE of the model as shown below.



ALIAS_NAME LIKE 'Ngamia%'

Below is a search result of the above search.

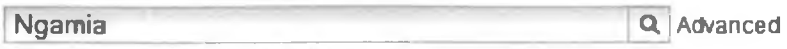


Wellbore Search Results for ALIAS_NAME+LIKE+%27Ngamia%25%27

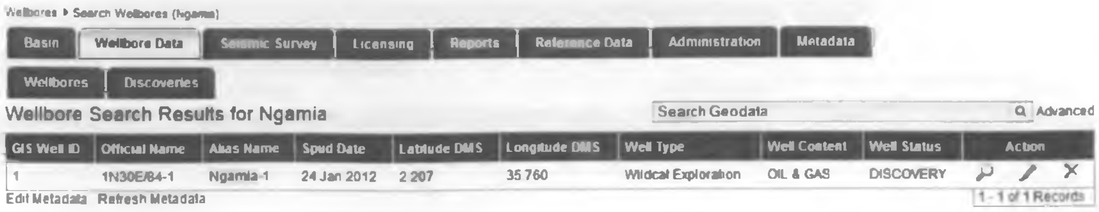
GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action
1	1N30E/84-1	Ngamia-1	24 Jan 2012	2.207	35.760	Wildcat Exploration	OIL & GAS	DISCOVERY	

1 - 1 of 1 Records

Search can also be done by giving a search text and obtained results as shown below.



Ngamia


Wellbores ▸ Search Wellbores (Ngamia)

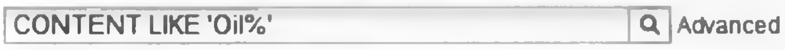
Wellbore Search Results for Ngamia

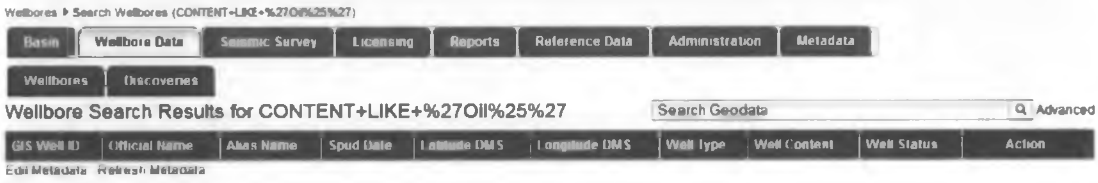
GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action
1	1N30E/84-1	Ngamia-1	24 Jan 2012	2 207	35 760	Wildcat Exploration	OIL & GAS	DISCOVERY	

Edit Metadata Refresh Metadata 1 - 1 of 1 Records

The metadata searches above are based on how the metadata has been setup. Failure to setup a column of the geodata model as a search criterion, no search results will be obtained. For example, we have wellbore records whose content are oil. Yet when we perform a search of "where content is like oil" we get no results. Because the content column or field has not been set as a metadata search criterion.

This is illustrated below:


CONTENT LIKE 'Oil%'


Wellbores ▸ Search Wellbores (CONTENT+LIKE+%27Oil%25%27)

Wellbore Search Results for CONTENT+LIKE+%27Oil%25%27

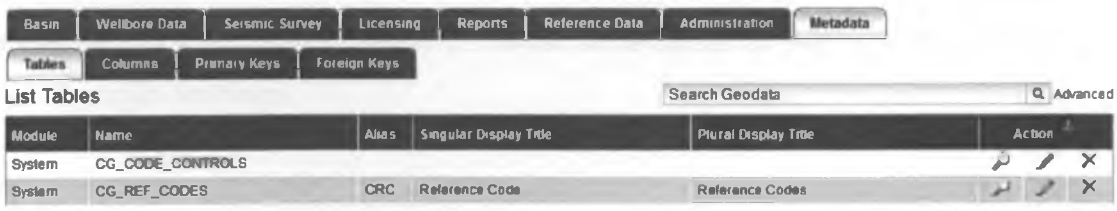
GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action

Edit Metadata Refresh Metadata

The search above yields no result because metadata search indexing is not set.

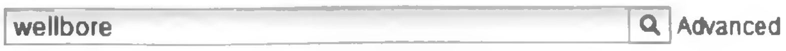
D. Setting up the Metadata Search Index

To set a metadata search index for wellbore content for the wellbore table go to its metadata listing as shown below:


List Tables

Module	Name	Alias	Singular Display Title	Plural Display Title	Action
System	CG_CODE_CONTROLS				
System	CG_REF_CODES	CRC	Reference Code	Reference Codes	

Search for WELLBORE metadata


wellbore

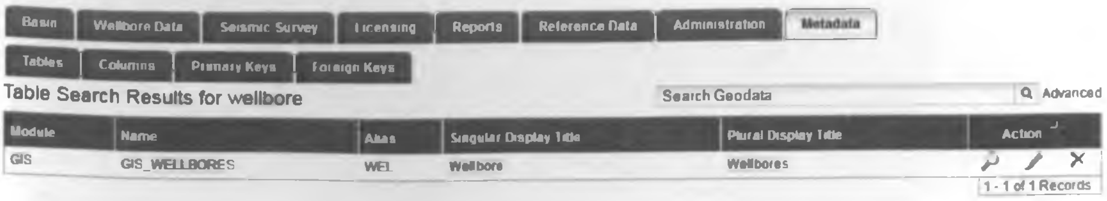
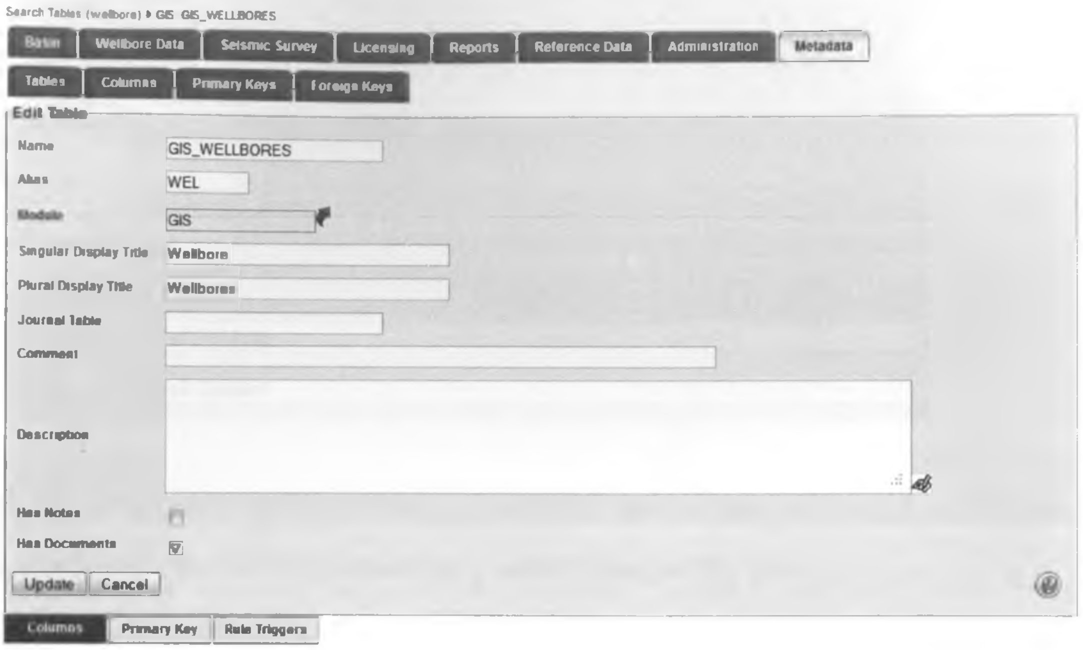


Table Search Results for wellbore

Module	Name	Alias	Singular Display Title	Plural Display Title	Action
GIS	GIS_WELLBORES	WEL	Wellbore	Wellbores	

1 - 1 of 1 Records

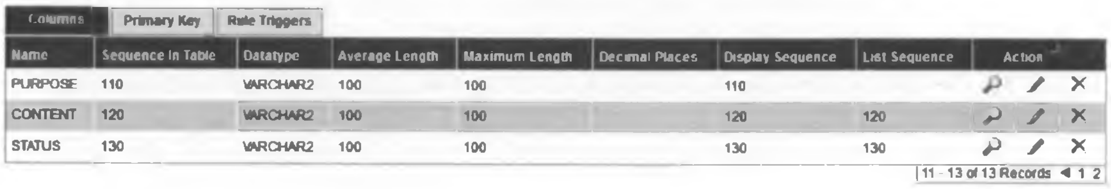
Click the edit icon. This opens a screen as one shown below:



Edit Table

Name	GIS_WELLBORES
Alias	WEL
Module	GIS
Singular Display Title	Wellbore
Plural Display Title	Wellbores
Journal Table	
Comment	
Description	
Has Notes	☐
Has Documents	☑

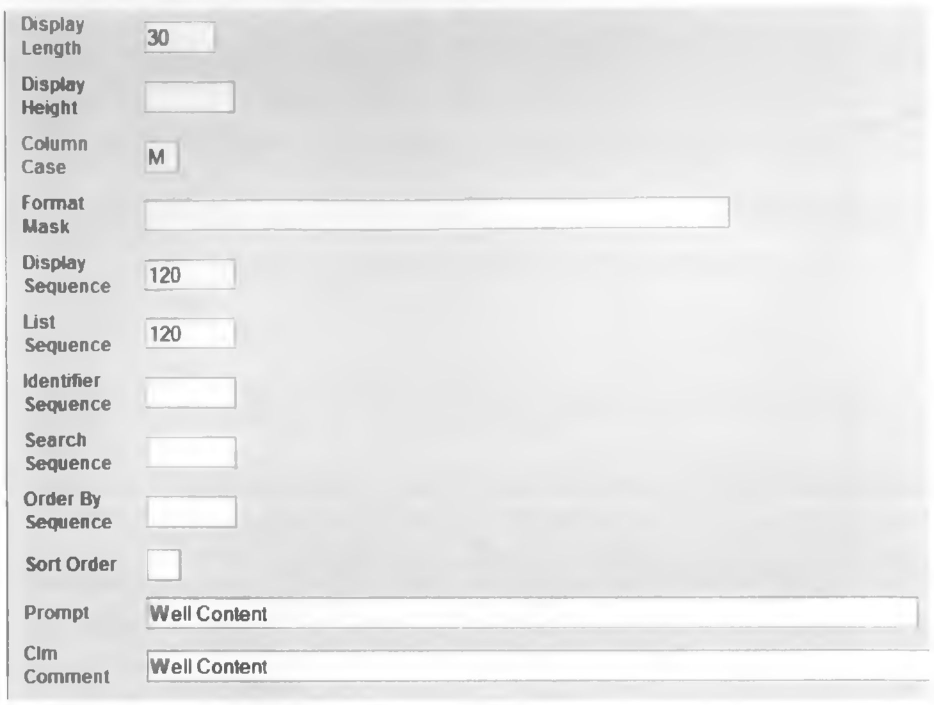
Go to the Columns submenu of the WELLBORE metadata as shown below:



Name	Sequence In Table	Datatype	Average Length	Maximum Length	Decimal Places	Display Sequence	List Sequence	Action
PURPOSE	110	VARCHAR2	100	100		110		
CONTENT	120	VARCHAR2	100	100		120	120	
STATUS	130	VARCHAR2	100	100		130	130	

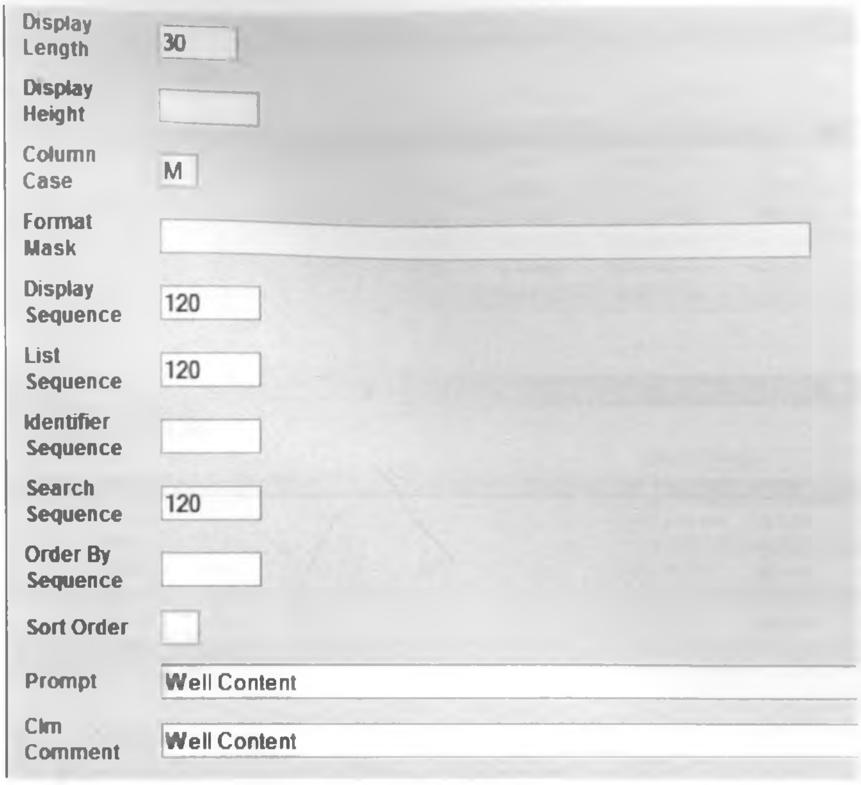
11 - 13 of 13 Records ◄ 1 2

Click the edit icon on the CONTENT column metadata of the WELLBORE table and get screen shown below:



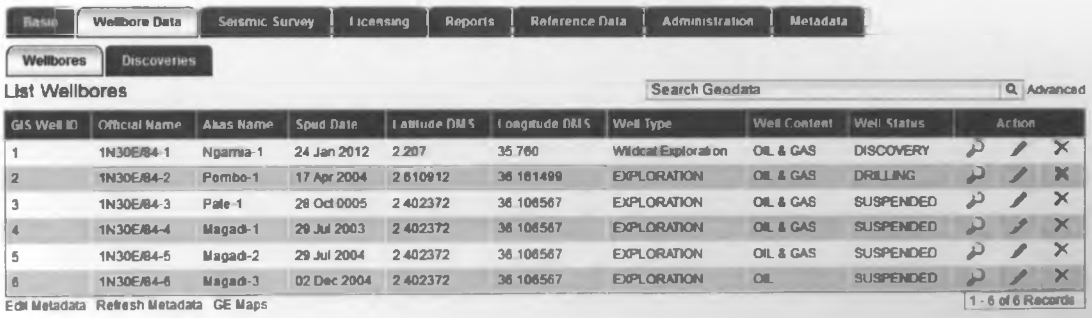
Display Length	30
Display Height	
Column Case	M
Format Mask	
Display Sequence	120
List Sequence	120
Identifier Sequence	
Search Sequence	
Order By Sequence	
Sort Order	
Prompt	Well Content
Clm Comment	Well Content

Set some value for the Search Sequence field so that the metadata search mechanism can consider this field while querying a search string in this table.



Display Length	30
Display Height	
Column Case	M
Format Mask	
Display Sequence	120
List Sequence	120
Identifier Sequence	
Search Sequence	120
Order By Sequence	
Sort Order	
Prompt	Well Content
Clm Comment	Well Content

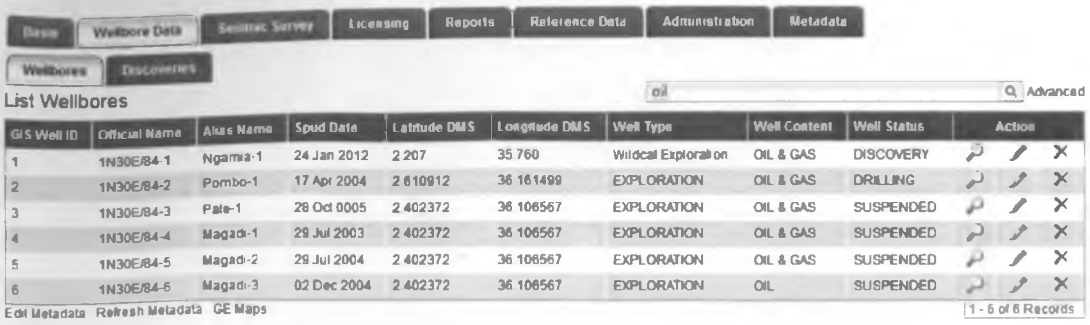
Going back to the WELLBORE table list, we refresh the metadata; click Refresh Metadata link



List Wellbores

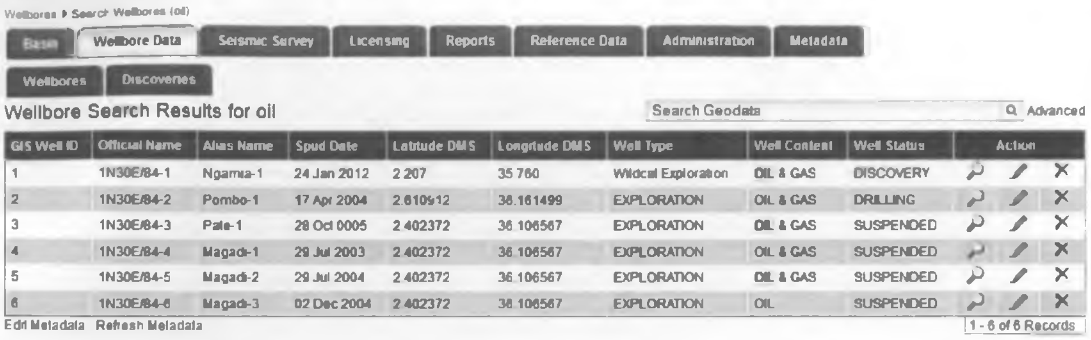
GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action
1	1N30E/84-1	Ngamia-1	24 Jan 2012	2.207	35.760	Wildcat Exploration	OIL & GAS	DISCOVERY	
2	1N30E/84-2	Pombo-1	17 Apr 2004	2.610912	36.161499	EXPLORATION	OIL & GAS	DRILLING	
3	1N30E/84-3	Pale-1	28 Oct 0005	2.402372	36.106567	EXPLORATION	OIL & GAS	SUSPENDED	
4	1N30E/84-4	Magadi-1	29 Jul 2003	2.402372	36.106567	EXPLORATION	OIL & GAS	SUSPENDED	
5	1N30E/84-5	Magadi-2	29 Jul 2004	2.402372	36.106567	EXPLORATION	OIL & GAS	SUSPENDED	
6	1N30E/84-6	Magadi-3	02 Dec 2004	2.402372	36.106567	EXPLORATION	OIL	SUSPENDED	

Now perform search for Well Content and observe the results as shown below:



List Wellbores

GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action
1	1N30E/84-1	Ngamia-1	24 Jan 2012	2 207	35 760	Wildcat Exploration	OIL & GAS	DISCOVERY	🔍 ✏ ✕
2	1N30E/84-2	Pombo-1	17 Apr 2004	2 610912	36 161499	EXPLORATION	OIL & GAS	DRILLING	🔍 ✏ ✕
3	1N30E/84-3	Pate-1	28 Oct 0005	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	🔍 ✏ ✕
4	1N30E/84-4	Magadi-1	29 Jul 2003	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	🔍 ✏ ✕
5	1N30E/84-5	Magadi-2	29 Jul 2004	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	🔍 ✏ ✕
6	1N30E/84-6	Magadi-3	02 Dec 2004	2 402372	36 106567	EXPLORATION	OIL	SUSPENDED	🔍 ✏ ✕



Wellbore Search Results for oil

GIS Well ID	Official Name	Alias Name	Spud Date	Latitude DMS	Longitude DMS	Well Type	Well Content	Well Status	Action
1	1N30E/84-1	Ngamia-1	24 Jan 2012	2 207	35 760	Wildcat Exploration	OIL & GAS	DISCOVERY	🔍 ✏ ✕
2	1N30E/84-2	Pombo-1	17 Apr 2004	2.610912	36.161499	EXPLORATION	OIL & GAS	DRILLING	🔍 ✏ ✕
3	1N30E/84-3	Pate-1	28 Oct 0005	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	🔍 ✏ ✕
4	1N30E/84-4	Magadi-1	29 Jul 2003	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	🔍 ✏ ✕
5	1N30E/84-5	Magadi-2	29 Jul 2004	2 402372	36 106567	EXPLORATION	OIL & GAS	SUSPENDED	🔍 ✏ ✕
6	1N30E/84-6	Magadi-3	02 Dec 2004	2 402372	36 106567	EXPLORATION	OIL	SUSPENDED	🔍 ✏ ✕

It is this metadata search indexing feature that makes this search mechanism speed up data retrieval considerably. The indexing, sequence the columns in an organized order and this bring about quick improvement in the data retrieval response time.