ISSN No. 0976-5697

Volume 1, No. 3, Sept-Oct 2010

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Creation of Gene Database and Implementation of Transaction Processing

Muhammad Rukunuddin* Ghalib and Kauser Ahmed P
School of Computing Sciences and Engineering, VIT University
Vellore, India
ruk.ghalib@vit.ac.in, kauserahmed@vit.ac.in

Jaigam Dilshad
School of Biosciences and Technology, VIT University
Vellore, India
dilshad.sbst@gmail.com

Abstract: A rich set of concepts and techniques has been used in the context of gene database creation and transaction processing along with performance and tuning for the efficient and robust execution of queries. So far, this work has mostly focused on issues related to data-retrieval queries. However, update operations can also exhibit a number of query processing issues, depending on the complexity of the operations and the volume of data to process. Such issues include lookup and matching of values, navigational vs. set-oriented algorithms and trade-offs between plans that do serial or random I/Os .In this paper we present an overview of the basic techniques used to support SQL DML (Data Manipulation Language) in ORACLE database 10g. Our focus is on the collection of gene information and implementation of some concepts of transaction operations and performance and tuning operation into the gene database and cancer database. Although atomicity is a well studied topic in transaction processing and business workflows, such an important capability needs to be revisited in a scientific workflow environment. Atomicity needs to be defined in dataflow-oriented scientific workflow model. The basic principles of all transaction-processing systems are the same. However, the terminology may vary from one transaction-processing system to another.

Keywords: GeneBank, SQL DML, DB Transaction Management, Tuning.

I. INTRODUCTION

During the past decade, the massive growth in genetic and protein databases has created a pressing need for tools to manage, retrieve and analyze the information contained in these libraries. Traditional tools to organize, classify and extract information have often proved inadequate when confronted with the overwhelming size and density of information which includes not only sequence and structural data, but also text that describes the data origin, location, species, tissue sample, journal articles, and so forth. As of this writing, the NCBI (National Center for Biotechnology Information, part of the National Institutes of Health) GenBank library alone consists of nearly 84 billion bytes of data and it is only one of several data banks storing similar information. The scope and size of these databases continues to rapidly grow and will continue to do so for many years to come as will the demand for access.

Currently, retrieval of genomic data is mainly based on well-established programs such as FASTA and BLAST that match candidate nucleotide sequences against massive libraries of sequence acquisitions. There have been few efforts to provide access to genomic data keyed to the extensive text annotations commonly found in these data sets. Among the few systems that deal with keyword based searching are the proprietary SRS system and protein information resource (PIR). These are limited, controlled vocabulary systems whose keys are from manually prepared annotations. To date, there have been no systems reported to directly generate indices from the genomic data sets themselves. The reasons for this are several: the very large size of the underlying data sets, the size of intermediate indexing files, the complexity of the data, and the time required to perform the indexing.

Database consists of an organized collection of data for one or more multiple uses. One way of classifying databases involves the type of content, for example: bibliographic, full-text, numeric, and image. Other classification methods start from examining database architectures [1],[4],[7]. A number of database architectures exist. Many databases use a combination of strategies.

Databases consist of software-based "containers" that are structured to collect and store information so users can retrieve, add, update or remove such information in an automatic fashion. Database programs are designed for users so that they can add or delete any information needed. The structure of a database is tabular, consisting of rows and columns of information.

A database management system (DBMS) consists of software that organizes the storage of data. A DBMS controls the creation, maintenance, and use of the database storage structures of social organizations and of their users. It allows organizations to place control of organization wide database development in the hands of Database Administrators (DBAs) and other specialists. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way.

Database management systems are usually categorized according to the database model that they support, such as the network, relational or object model. The model tends to determine the query languages that are available to access the database. One commonly used query language for the relational database is SQL, although SQL syntax and function can vary from one DBMS to another. A common query language for the object database is OQL; although not all vendors of object databases implement this, majority of them do implement this method. A great deal of the internal engineering of a DBMS is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between the products.

A relational database management system (RDBMS) implements features of the relational model. In this context, Date's "Information Principle" states: "the entire information content of the database is represented in one and only one way, namely as explicit values in column positions (attributes) and rows in relations (tuples). Therefore, there are no explicit pointers between related tables." This contrasts with the object database management system (ODBMS), which does store explicit pointers between related types.

A gene is a unit of heredity in a living organism. It is normally a stretch of DNA that codes for a type of protein or for an RNA chain that has a function in the organism [2]. All proteins and functional RNA chains are specified by genes. All living things depend on genes. Genes hold the information to build and maintain an organism's cells and pass genetic traits to offspring. A modern working definition of a gene is "a locatable region of genomic sequence, corresponding to a unit of inheritance, which is associated with regulatory regions, transcribed regions, and or other functional sequence regions "[3]. Colloquial usage of the term *gene* (e.g. "good genes, "hair color gene") may actually refer to an allele: a *gene* is the basic instruction, a sequence of nucleic acid (DNA or, in the case of certain viruses RNA), while an *allele* is one variant of that instruction.

The human genome is the genome of *Homo sapiens*, which is stored on 23 chromosome pairs. Twenty-two of these are autosomal chromosome pairs, while the remaining pair is sex-determining. The haploid human genome occupies a total of just over 3 billion DNA base pairs. The Human Genome Project (HGP) produced a reference sequence of the euchromatic human genome, which is used worldwide in biomedical sciences.

The haploid human genome contains ca. 23,000 protein-coding genes, far fewer than had been expected before its sequencing. In fact, only about 1.5% of the genome codes for proteins, while the rest consists of non-coding RNA genes, regulatory sequences, introns, and (controversially named) "junk" DNA[3],[4].

The Cancer Gene Census is an ongoing effort to catalogue those genes for which mutations have been causally implicated in cancer. The original census and analysis was published in Nature Reviews Cancer and supplemental analysis information related to the paper is also available [2].

II. RELATED WORK

A. *Indexing genomic sequence libraries*

Most genomic databases include, in addition to nucleotide and protein sequences, a wealth of information in the form of descriptions, keywords, annotations, hyper-links to text articles journals and so forth. In many cases, the text attachments to the data are greater in size than the actual sequence data. Identifying the important keyword terms from this data and assigning an elative weight to these terms is one of the problems addressed in this system. Indexing helps to improve query processing and performance that helps to speed up retrieval of values [1], [3], and [8].

B. Constraint acquisition for Entity-Relationship models

Integrity constraints are conditions that capture the semantics of the application domain under consideration. They restrict the databases to those that are considered meaningful to the application at hand. In practice, the decision of specifying a constraint is very important and extremely challenging.

C. Atomicity and provenance support for pipelined scientific workflows

Atomicity is an important transactional property, which requires that a transaction either runs to completion or has no partial effect (all-or-nothing). In scientific workflows, some tasks might fail during execution due to either the failure of the task itself or inappropriate input to a task. A domain scientist might require the execution of a sub-workflow to be atomic in the sense that either the execution of all the tasks of the sub-workflow runs to completion or none of them has any effect at all[1],[6].

D. *Efficiently supporting secure and reliable collaboration in scientific workflows*

The transactional support is widely used to address the reliability of systems [6], [7]. In traditional database systems and work- flows, the consistency of sharing data and administration among components can be achieved through implementing strict transaction semantics in terms of atomicity, consistency, isolation and durability (ACID). Although extremely reliable, traditional ACID transactions are not suitable for loosely coupled environments such as Web service-based business transactions. This is because fine-grained lock controls and full trustworthiness are not generally applicable in Web services-based transactions.

III METHODOLOGY

- Collection of human gene related information from various database.
- Classifying the collected information into different criteria.
- Create different tables to add this information.
- Implement the concept of performance and tuning and transaction processing.

A. *Steps involved in creating database*

- Installing the Oracle 10*g* database software is a separate process from that of creating a database
- GENE Databases can be created using the Database Configuration Assistant (DBCA tool) or manually using the CREATE DATABASE command
- When creating a GENE Database manually it is best to generate scripts using DBCA first, and then to edit them
- The DBA authentication method determines how Oracle 10*g* validates users logging on with SYSDBA or SYSOPER privileges

- OS authentication relies on the OS's security to validate the user/password, and authorization group.
- The REMOTE_LOGIN_PASSWORDFILE parameter is set to NONE for OS authentication.
- Password file authentication stores user names and passwords and group membership in an encrypted file in the OS
- Set REMOTE_LOGIN_PASSWORDFILE to EXCLUSIVE for password file authentication.
- The ORAPWD utility generates the password file for SYSDBA and SYSOPER and then the database maintains it with changes to passwords.
- Control files can be multiplexed (each subsequent control file is an exact copy of the first control file).
- Multiplexed copies of control files should be located on different physical devices to guard against damage.
- Prevent bottlenecks in data access by placing data on several physical devices (spreads the demand).
  - User-managed file management offers more detailed control over datafiles than Oracle Managed Files, but requires more manual maintenance tasks.
  - DBCA provides an opportunity to customize memory size and initialization parameters.
  - Adjusting of tablespace/datafile sizes and locations depends on the DB type selected using DBCA.
  - After creating GENE database, use Net Manager to set up a Net Service name for the database.
  - Collection of various information regarding GENE from various sources.
  - Adding these information into users schema
  - Implement queries to extract information from GENE DATABASE.
  - Implementation of transaction processing concepts.
  - Implementation of database performance and tuning concept.

B. Creating a Database Using DBCA

DBCA enables us to create a database from predefined templates provided by Oracle or from templates that we or others have created. A template is a description of a database.

C. Selecting the Template

DBCA displays the templates that are available, which includes templates that Oracle ships with the DBCA product. If we or others have created templates, those will be displayed also. We select the appropriate template for the database that we want to create. Clicking the "Show Details..." button displays specific information about the database defined by a template.

IV. IMPLEMENTATION

A. Considerations before Creating a GENE Database

Database creation prepares several operating system files to work together as an Oracle database. We need only create a database once, regardless of how many datafiles it has or how many instances access it. Creating a database can also erase information in an existing database and create a new database with the same name and physical structure.The following topics can help prepare us for database creation.

- Planning for database creation
- Meeting creation prerequisite
- Deciding how to create GENE database

B. Meeting Creation Prerequisites

To create a new database, the following prerequisites must be met:

- The desired Oracle software is installed. This includes setting up various environment variables unique to our operating system and establishing the directory structure for software and database files.
-
- We have the operating system privileges associated with a fully operational database administrator. We must be specially authenticated by our operating system or through a password file, allowing we to start up and shut down an instance before the database is created or opened.
- There is sufficient memory available to start the Oracle instance.
- There is sufficient disk storage space for the planned database on the computer that executes Oracle.

C. Specifying Mode, Initialization Parameters, and Datafiles

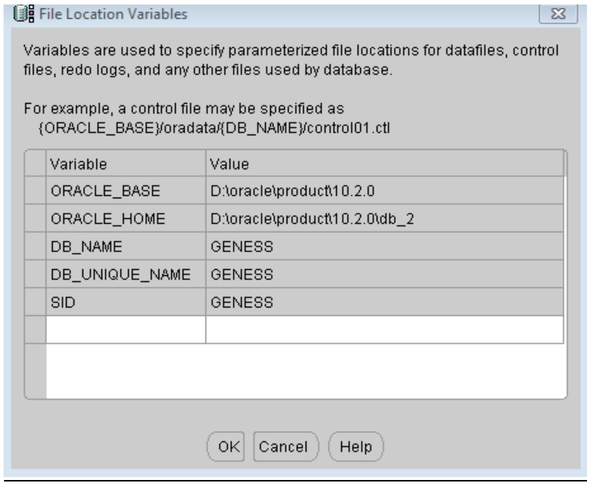
The next pages enable us to further define our database. We specify mode (dedicated server of shared server), set initialization parameters, and specify datafile locations. Oracle can determine specific values for we based upon our description of the database we are trying to create. For example, Oracle can choose appropriate settings for SGA memory sizing parameters depending upon whether we select a typical or custom database.

D. Completing Database Creation

After we have completed the specification of the parameters that define our database we can:

- Create the database now
- Save the description as a database template
- Generate database creation scripts

If we choose to generate scripts, we can use them to create the database later without using DBCA, or we can use them as a checklist.



Variables are used to specify parameterized file locations for datafiles, control files, redo logs, and any other files used by database.

For example, a control file may be specified as
{ORACLE_BASE}/oradata/{DB_NAME}/control01.ctl

Variable	Value
ORACLE_BASE	D:\oracle\product\10.2.0
ORACLE_HOME	D:\oracle\product\10.2.0\db_2
DB_NAME	GENESS
DB_UNIQUE_NAME	GENESS
SID	GENESS

Figure 1: Parameterized locations for datafills

E. Implementing Transactions and Performance Tuning Concepts

A transaction is a logical unit of work that contains one or more SQL statements. A transaction is an atomic unit. The effects of all the SQL statements in a transaction can be either all committed (applied to the database) or all rolled back (undone from the database).A transaction begins with the first executable SQL statement. A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued [5],[6],[7],[8]. Tuning graphs are shown in figures 8, 9, 10, 11.

F. Rollback of Transactions

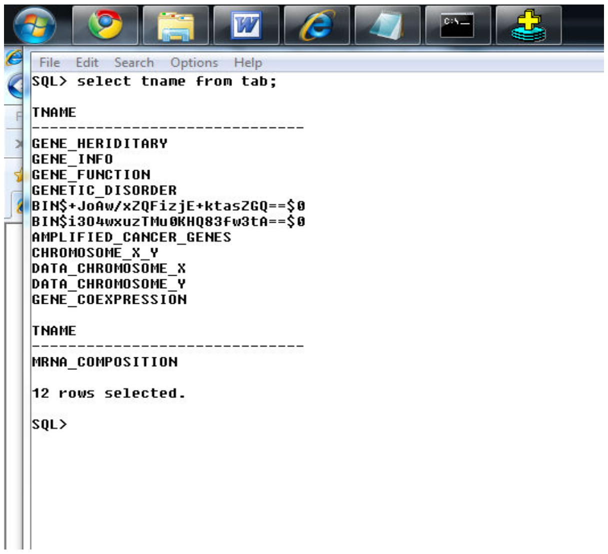
Rolling back means undoing any changes to data that have been performed by SQL statements within an uncommitted transaction. Oracle uses undo tablespaces (or rollback segments) to store old values. The redo log contains a record of changes.

Oracle lets us roll back an entire uncommitted transaction. Alternatively, we can roll back the trailing portion of an uncommitted transaction to a marker called a savepoint.
All types of rollbacks use the same procedures:

- Statement-level rollback (due to statement or deadlock execution error)
- Rollback to a savepoint
- Rollback of a transaction due to user request
- Rollback of a transaction due to abnormal process termination
- Rollback of all outstanding transactions when an instance terminates abnormally
- Rollback of incomplete transactions during recovery

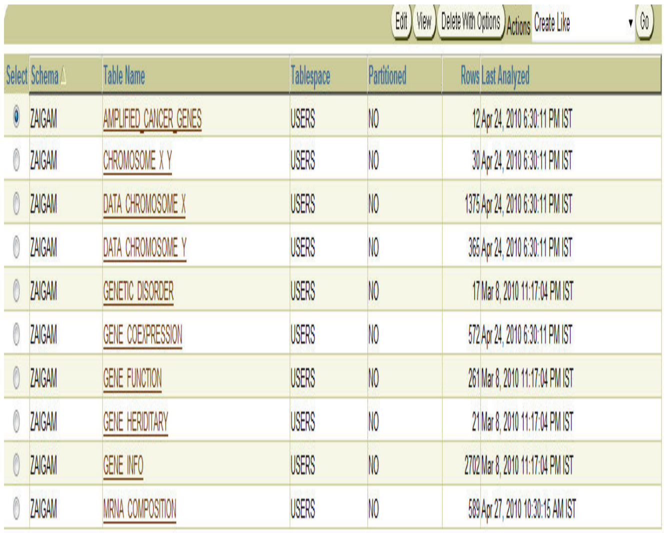
V. RESULTS AND DISCUSSIONS

Organizing collected information in different table to create GENE and Cancerous gene database After collecting all the information related to genes and cancerous gene my target is to arrange all these collected information into ORACLE DATABASE to create two separate database . GENE database listed all the information about genes, CANCEROUS gene database list only the gene those are responsible for cancer. In order to create gene database in a much organized way we create several tables using SQL in ORACLE DATABASE 10g as given in figure 2.



```
SQL> select tname from tab;

TNAME
------------------------------
GENE_HERIDITARY
GENE_INFO
GENE_FUNCTION
GENETIC_DISORDER
BIN$+JoAw/xZQFizjE+ktasZGQ==$0
BIN$i3O4wxuzTMu0KHQ83fw3tA==$0
AMPLIFIED_CANCER_GENES
CHROMOSOME_X_Y
DATA_CHROMOSOME_X
DATA_CHROMOSOME_Y
GENE_COEXPRESSION

TNAME
------------------------------
MRNA_COMPOSITION

12 rows selected.

SQL>
```

Figure 2:creating gene tables using oracle databse 10G



Select	Schema	Table Name	Tablespace	Partitioned	Rows	Last Analyzed
●	ZAIGAM	AMPLIFIED CANCER GENES	USERS	NO	12	Apr 24, 2010 6:30:11 PM IST
○	ZAIGAM	CHROMOSOME X Y	USERS	NO	30	Apr 24, 2010 6:30:11 PM IST
○	ZAIGAM	DATA CHROMOSOME X	USERS	NO	1375	Apr 24, 2010 6:30:11 PM IST
○	ZAIGAM	DATA CHROMOSOME Y	USERS	NO	365	Apr 24, 2010 6:30:11 PM IST
○	ZAIGAM	GENETIC DISORDER	USERS	NO	17	Mar 8, 2010 11:17:04 PM IST
○	ZAIGAM	GENE COEXPRESSION	USERS	NO	572	Apr 24, 2010 6:30:11 PM IST
○	ZAIGAM	GENE FUNCTION	USERS	NO	261	Mar 8, 2010 11:17:04 PM IST
○	ZAIGAM	GENE HERIDITARY	USERS	NO	21	Mar 8, 2010 11:17:04 PM IST
○	ZAIGAM	GENE INFO	USERS	NO	2702	Mar 8, 2010 11:17:04 PM IST
○	ZAIGAM	MRNA COMPOSITION	USERS	NO	589	Apr 27, 2010 10:30:15 AM IST

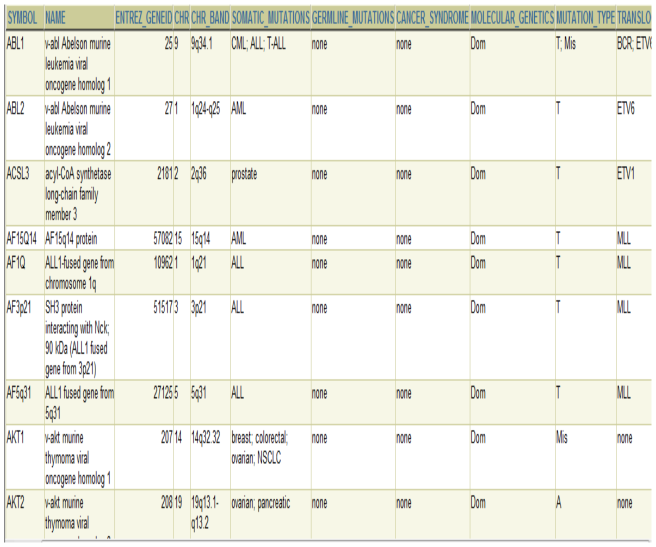
Figure 3: Table of Cancer Genes

A. Query of the gene and cancer database to retrieve the values

In order to retrieve the values from these databases (figure 3) we have to perform query which list the set of information related to these databases. Cancer Database instance is shown in figure 12.

Query 1

SELECT "SYMBOL", "NAME", "ENTREZ_GENEID", "CHR", "CHR_BAND", "SOMATIC_MUTATIONS", "GERMLINE_MUTATIONS", "CANCER_SYNDROME", "MOLECULAR_GENETICS", "MUTATION_TYPE", "TRANSLOCATION_PARTNER" FROM "AMAN"."CANCEROUS_GENES"

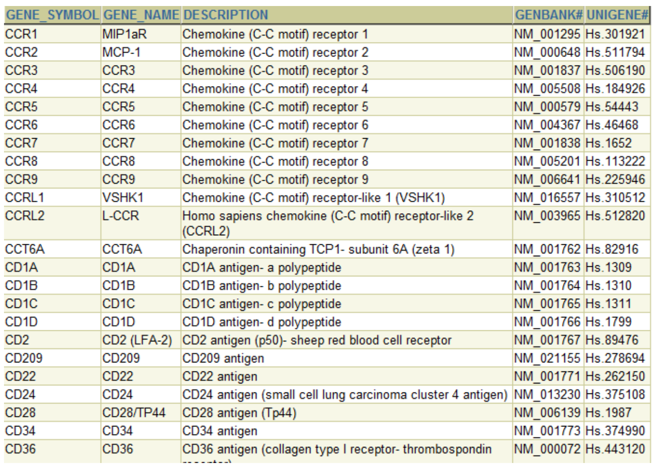


SYMBOL	NAME	ENTREZ_GENEID	CHR	CHR_BAND	SOMATIC_MUTATIONS	GERMLINE_MUTATIONS	CANCER_SYNDROME	MOLECULAR_GENETICS	MUTATION_TYPE	TRANSLO
ABL1	v-abl Abelson murine leukemia viral oncogene homolog 1	25	9	9q34.1	CML; ALL; T-ALL	none	none	Dom	T; Mis	BCR; ETV
ABL2	v-abl Abelson murine leukemia viral oncogene homolog 2	27	1	1q24-q25	AML	none	none	Dom	T	ETV6
ACSL3	acyl-CoA synthetase long-chain family member 3	2181	2	2q36	prostate	none	none	Dom	T	ETV1
AF15Q14	AF15q14 protein	57082	15	15q14	AML	none	none	Dom	T	MLL
AF1Q	ALL1-fused gene from chromosome 1q	10962	1	1q21	ALL	none	none	Dom	T	MLL
AF3p21	SH3 protein interacting with Nck; 90 kDa (ALL1 fused gene from 3p21)	51517	3	3p21	ALL	none	none	Dom	T	MLL
AF5q31	ALL1 fused gene from 5q31	27125	5	5q31	ALL	none	none	Dom	T	MLL
AKT1	v-akt murine thymoma viral oncogene homolog 1	207	14	14q32.32	breast; colorectal; ovarian; NSCLC	none	none	Dom	Mis	none
AKT2	v-akt murine thymoma viral	208	19	19q13.1-q13.2	ovarian; pancreatic	none	none	Dom	A	none

Figure 4: Output of Query I

Query 2

SELECT "GENE_SYMBOL", "GENE_NAME", "DESCRIPTION", "GENBANK#", "UNIGENE#" FROM "AMAN"."GENE_INFO"

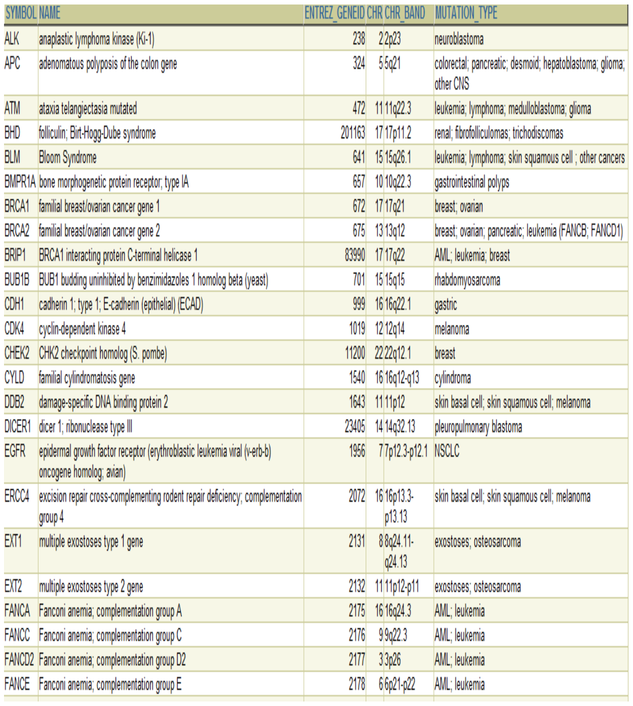


GENE_SYMBOL	GENE_NAME	DESCRIPTION	GENBANK#	UNIGENE#
CCR1	MIP1aR	Chemokine (C-C motif) receptor 1	NM_001295	Hs.301921
CCR2	MCP-1	Chemokine (C-C motif) receptor 2	NM_000648	Hs.511794
CCR3	CCR3	Chemokine (C-C motif) receptor 3	NM_001837	Hs.506190
CCR4	CCR4	Chemokine (C-C motif) receptor 4	NM_005508	Hs.184926
CCR5	CCR5	Chemokine (C-C motif) receptor 5	NM_000579	Hs.54443
CCR6	CCR6	Chemokine (C-C motif) receptor 6	NM_004367	Hs.46468
CCR7	CCR7	Chemokine (C-C motif) receptor 7	NM_001838	Hs.1652
CCR8	CCR8	Chemokine (C-C motif) receptor 8	NM_005201	Hs.113222
CCR9	CCR9	Chemokine (C-C motif) receptor 9	NM_006641	Hs.225946
CCRL1	VSHK1	Chemokine (C-C motif) receptor-like 1 (VSHK1)	NM_016557	Hs.310512
CCRL2	L-CCR	Homo sapiens chemokine (C-C motif) receptor-like 2 (CCRL2)	NM_003965	Hs.512820
CCT6A	CCT6A	Chaperonin containing TCP1- subunit 6A (zeta 1)	NM_001762	Hs.82916
CD1A	CD1A	CD1A antigen- a polypeptide	NM_001763	Hs.1309
CD1B	CD1B	CD1B antigen- b polypeptide	NM_001764	Hs.1310
CD1C	CD1C	CD1C antigen- c polypeptide	NM_001765	Hs.1311
CD1D	CD1D	CD1D antigen- d polypeptide	NM_001766	Hs.1799
CD2	CD2 (LFA-2)	CD2 antigen (p50)- sheep red blood cell receptor	NM_001767	Hs.89476
CD209	CD209	CD209 antigen	NM_021155	Hs.278694
CD22	CD22	CD22 antigen	NM_001771	Hs.262150
CD24	CD24	CD24 antigen (small cell lung carcinoma cluster 4 antigen)	NM_013230	Hs.375108
CD28	CD28/TP44	CD28 antigen (Tp44)	NM_006139	Hs.1987
CD34	CD34	CD34 antigen	NM_001773	Hs.374990
CD36	CD36	CD36 antigen (collagen type I receptor- thrombospondin	NM_000072	Hs.443120

Figure 5: Output of Query II

Query 3

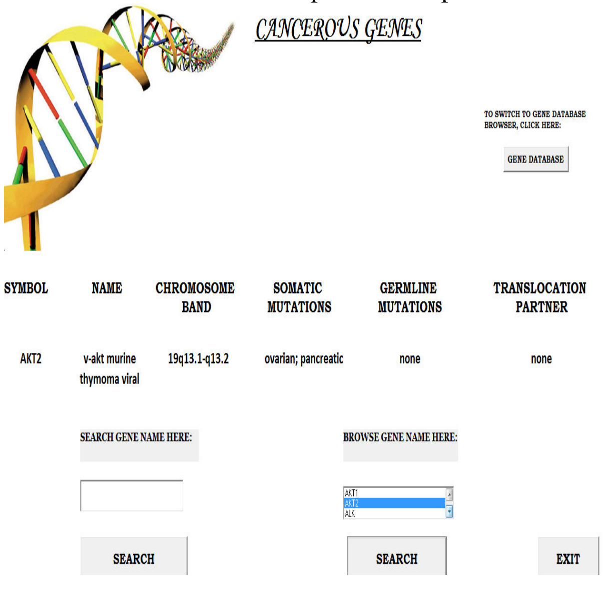
SELECT "SYMBOL", "NAME", "ENTREZ_GENEID", "CHR", "CHR_BAND", "MUTATION_TYPE" FROM "AMAN"."CANCEROUS_GENE_CHROMOSOME"



SYMBOL	NAME	ENTREZ_GENEID	CHR	CHR_BAND	MUTATION_TYPE
ALK	anaplastic lymphoma kinase (Ki-1)	238	2	2p23	neuroblastoma
APC	adenomatous polyposis of the colon gene	324	5	5q21	colorectal; pancreatic; desmoid; hepatoblastoma; glioma; other CNS
ATM	ataxia telangiectasia mutated	472	11	11q22.3	leukemia; lymphoma; medulloblastoma; glioma
BHD	folliculin; Birt-Hogg-Dube syndrome	201163	17	17p11.2	renal; fibrofolliculomas; trichodiscomas
BLM	Bloom Syndrome	641	15	15q26.1	leukemia; lymphoma; skin squamous cell ; other cancers
BMPR1A	bone morphogenetic protein receptor; type IA	657	10	10q22.3	gastrointestinal polyps
BRCA1	familial breast/ovarian cancer gene 1	672	17	17q21	breast; ovarian
BRCA2	familial breast/ovarian cancer gene 2	675	13	13q12	breast; ovarian; pancreatic; leukemia (FANCB; FANCD1)
BRIP1	BRCA1 interacting protein C-terminal helicase 1	83990	17	17q22	AML; leukemia; breast
BUB1B	BUB1 budding uninhibited by benzimidazoles 1 homolog beta (yeast)	701	15	15q15	rhabdomyosarcoma
CDH1	cadherin 1; type 1; E-cadherin (epithelial) (ECAD)	999	16	16q22.1	gastric
CDK4	cyclin-dependent kinase 4	1019	12	12q14	melanoma
CHEK2	CHK2 checkpoint homolog (S. pombe)	11200	22	22q12.1	breast
CYLD	familial cylindromatosis gene	1540	16	16q12-q13	cylindroma
DDB2	damage-specific DNA binding protein 2	1643	11	11p12	skin basal cell; skin squamous cell; melanoma
DICER1	dicer 1; ribonuclease type III	23405	14	14q32.13	pleuropulmonary blastoma
EGFR	epidermal growth factor receptor (erythroblastic leukemia viral (v-erb-b) oncogene homolog; avian)	1956	7	7p12.3-p12.1	NSCLC
ERCC4	excision repair cross-complementing rodent repair deficiency; complementation group 4	2072	16	16p13.3-p13.13	skin basal cell; skin squamous cell; melanoma
EXT1	multiple exostoses type 1 gene	2131	8	8q24.11-q24.13	exostoses; osteosarcoma
EXT2	multiple exostoses type 2 gene	2132	11	11p12-p11	exostoses; osteosarcoma
FANCA	Fanconi anemia; complementation group A	2175	16	16q24.3	AML; leukemia
FANCC	Fanconi anemia; complementation group C	2176	9	9q22.3	AML; leukemia
FANCD2	Fanconi anemia; complementation group D2	2177	3	3p26	AML; leukemia
FANCE	Fanconi anemia; complementation group E	2178	6	6p21-p22	AML; leukemia

Figure 6: Output of Query III

To access the information from database in user-friendly way we create an application using visual basic. This application run in graphical user mode (figure 7) and accepts the search term from user to print the output.



SYMBOL	NAME	CHROMOSOME BAND	SOMATIC MUTATIONS	GERMLINE MUTATIONS	TRANSLOCATION PARTNER
AKT2	v-akt murine thymoma viral	19q13.1-q13.2	ovarian; pancreatic	none	none

Figure 7: Application in graphical user mode

B. Savepoints in Transactions

We can declare intermediate markers called savepoints within the context of a transaction. Savepoints divide a long transaction into smaller parts.

Using savepoints, we can arbitrarily mark our work at any point within a long transaction. We then have the option later of rolling back work performed before the current point in the transaction but after a declared savepoint within the transaction. For example, we can use savepoints throughout a long complex series of updates, so if we make an error, we do not need to resubmit every statement.

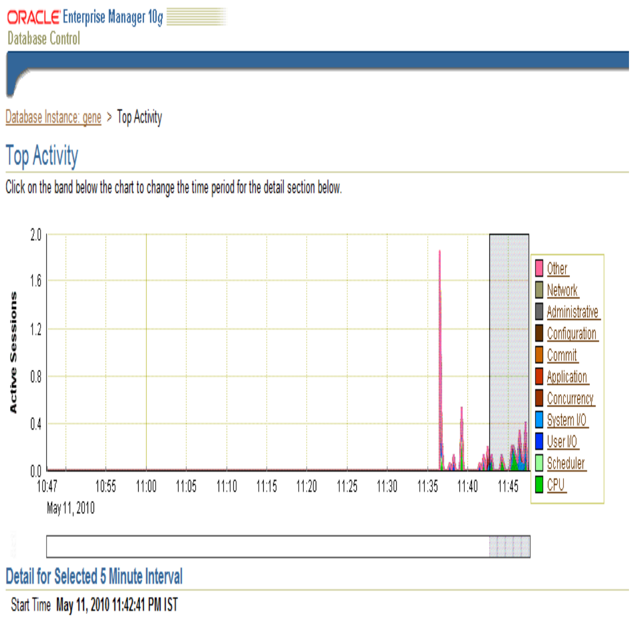
Savepoints are similarly useful in application programs. If a procedure contains several functions, then we can create a savepoint before each function begins. Then, if a function fails, it is easy to return the data to its state before the function began and re-run the function with revised parameters or perform a recovery action.

After a rollback to a savepoint, Oracle releases the data locks obtained by rolled back statements. Other transactions that were waiting for the previously locked resources can proceed. Other transactions that want to update previously locked rows can do so.

When a transaction is rolled back to a savepoint, the following occurs:

1. Oracle rolls back only the statements run after the savepoint.
2. Oracle preserves the specified savepoint, but all savepoints that were established after the specified one are lost.
3. Oracle releases all table and row locks acquired since that savepoint but retains all data locks acquired previous to the savepoint.
4. Record the transaction in the transaction journal

Oracle must allow for two situations. If all three SQL statements can be performed to maintain the accounts in proper balance, the effects of the transaction can be applied to the database. However, if a problem such as insufficient funds, invalid account number, or a hardware failure prevents one or two of the statements in the transaction from completing, the entire transaction must be rolled back so that the balance of all accounts is correct.



Click on the band below the chart to change the time period for the detail section below.

Start Time **May 11, 2010 11:42:41 PM IST**

Figure 8: Performance tuning graph of the transaction processes

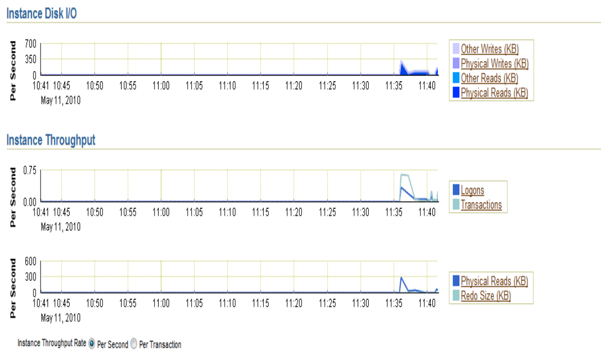
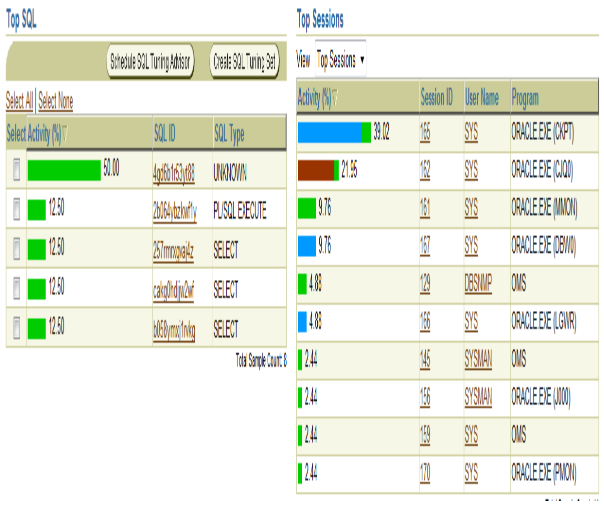


Figure 9: Instance Disk I/O and Instance throughput

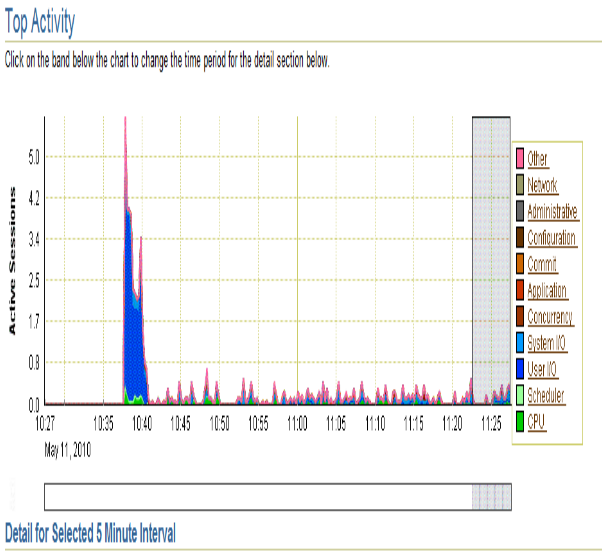


Select	Activity (%)	SQL ID	SQL Type
☐	50.00	4gd6b1r53yt88	UNKNOWN
☐	12.50	2b064ybzkwf1y	PL/SQL EXECUTE
☐	12.50	257rmrxgvaj4z	SELECT
☐	12.50	cakg0hdjjw2wf	SELECT
☐	12.50	b058ymxj1nkg	SELECT

Total Sample Count: 8

Activity (%)	Session ID	User Name	Program
39.02	165	SYS	ORACLE.EXE (CKPT)
21.95	162	SYS	ORACLE.EXE (CJQ0)
9.76	161	SYS	ORACLE.EXE (MMON)
9.76	167	SYS	ORACLE.EXE (DBW0)
4.88	129	DBSNMP	OMS
4.88	166	SYS	ORACLE.EXE (LGWR)
2.44	145	SYSMAN	OMS
2.44	156	SYSMAN	ORACLE.EXE (J000)
2.44	159	SYS	OMS
2.44	170	SYS	ORACLE.EXE (PMON)

Figure 10: Sessions Tuning Activity



Click on the band below the chart to change the time period for the detail section below.

Figure 11:Active sessions tuning for 5 minutes interval

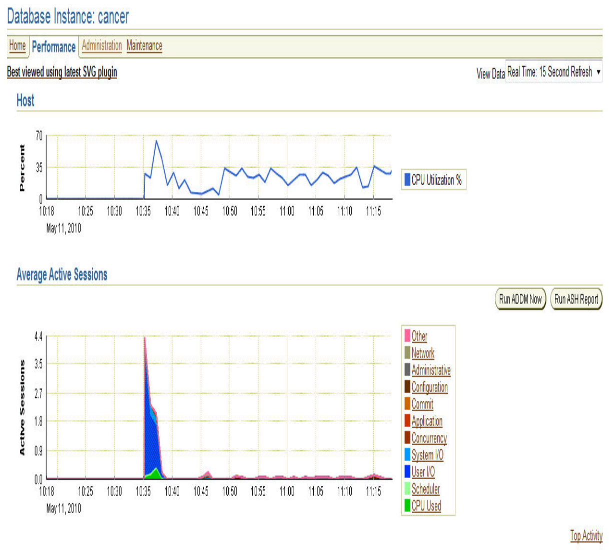


Figure 12: Database instance for Cancer Genes

VI. CONCLUSION AND FUTURE ASPECTS

Our database consists of an organized collection of data for one or more multiple uses. One way of classifying databases involves the type of content, for example: bibliographic, full-text, numeric, and image. Other classification methods start from examining database architectures. A number of database architectures exist. Many databases use a combination of strategies. Until our database is perfectly tuned it is not efficient.

Oracle includes numerous data structures to improve the speed of Oracle SQL queries. Taking advantage of the low cost of disk storage, Oracle includes many new indexing algorithms that dramatically increase the speed with which Oracle queries are serviced. Voluminous amount of information collected from different databases are stored in Gene Database and Cancer Database in an organized manner. Implementation of concept of performance and tuning helps to smooth database performance. Indexing the table provides a best solution to minimize the query execution plan.

In traditional database systems and work-flows, the consistency of sharing data and administration among components can be achieved through implementing strict transaction semantics in terms of atomicity, consistency, isolation and durability (ACID). Although extremely reliable, traditional ACID transactions are not suitable for loosely coupled environments such as Web service-based business transactions. This is because fine-grained lock controls and full trustworthiness are not generally applicable in Web services-based transactions. Although a number of proposals are presented to address this issue, currently, the existing Web service frameworks still lack effective models and approaches for the reliable (fault-tolerant, transactional) execution of a group of Web services is our future focus.

This project comprises of voluminous information about Human Gene and Human Cancerous gene which will greatly help researcher to retrieve all the information regarding their research without browsing so many database. Accessing these information in graphical modes from any platform (Linux, windows, solaris) through the browser convenient to researcher.

VII. REFERENCES

[1] Melanie R. Nelson, Stephanie J. Reisinger, Stephen G. Henry, “Designing databases to store biological information”, Biosilico, 2003, Volume 1, Issue 4 Pages 134-142

[2] Patricia G Baker, Andy Brass, “Recent developments in biological sequence databases”, Current Opinion in Biotechnology, 1998, Volume 9, Issue 1, Pages 54-58

[3] Edmond J. Breen, Keith L. Williams, “Hash function performance on different biological databases”, Computational Methods Programs, Biomed, 1989, Volume 28, Issue 2, Pages 87-91.

[4] Graham Chen, “Distributed transaction processing standards and their applications”, Computer Standards & Interfaces archive, 1995, Volume 17, Issue 4, Pages: 363 - 373

[5] Ranjit Bose, Stephen D. Burd,”Control and coordination of heterogeneous transaction processing systems”, Information and Software Technology, 1997, Volume 39, Issue 3, Pages 171-184.

[6] Özgür Ulusoy, “Transaction processing in distributed active real-time database systems”, Journal of Systems and Software, 1998, Volume 42, Issue 3, Pages 247-262

[7] Philip A. Bernstein, Eric Newcomer, “Transaction Processing Application Architecture”, Principles of Transaction Processing (Second Edition), ISBN: 978-1-55860-623-4.

[8] Arthur J. Bernstein, David S. Gerstl, Philip M. Lewis, “Concurrency control for step-decomposed transactions”, Information Systems, 1999, Volume 24, Issue 8, Pages 673-698