CSE 601: Data Mining and Bioinformatics

Project 1: Data Warehouse/OLAP System

TEAM

Name	**UB Person #**
Samved Divekar	50135204
Hrishikesh Sathe	50134055
Ankit Kapur	50133149

Part I : Implementing Data Warehouse Schema and populating data

We were provided with 5 different data spaces as follows.

1. Clinical data space

Entities: patient, disease, drug, test and sample

Fact table: clinical_fact

2. Sample data space

Entities: sample, marker, assay, term

Fact table: sample_fact

3. Microarray and proteomic data space

Entities: probe, measureUnit

Fact table: microarray_fact

4. Gene data space

Entities: gene, go, cluster, domain, promoter

Fact table: gene_fact

5. Experiment data space

Entities: experiment, project, platform, norm, person, protocal, publication

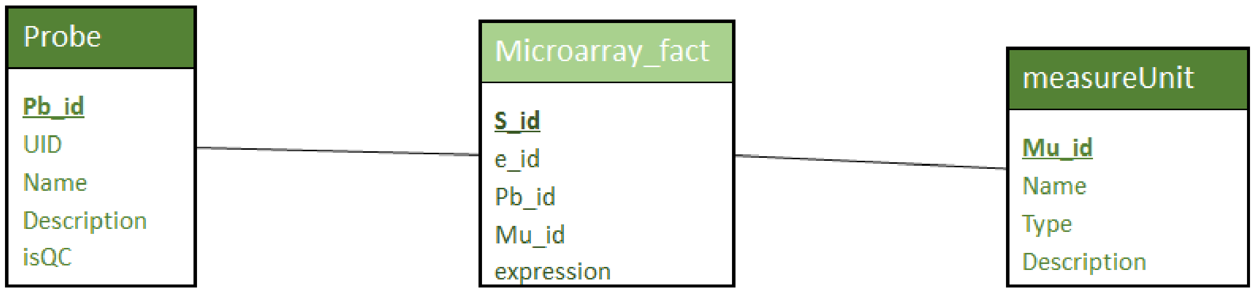
Fact table: experiment_fact

We created these tables and connected them to form star schema. After forming star schema, we needed additional tables in order to model it into Bio Star schema. Hence we created few more measurement tables. Viz.

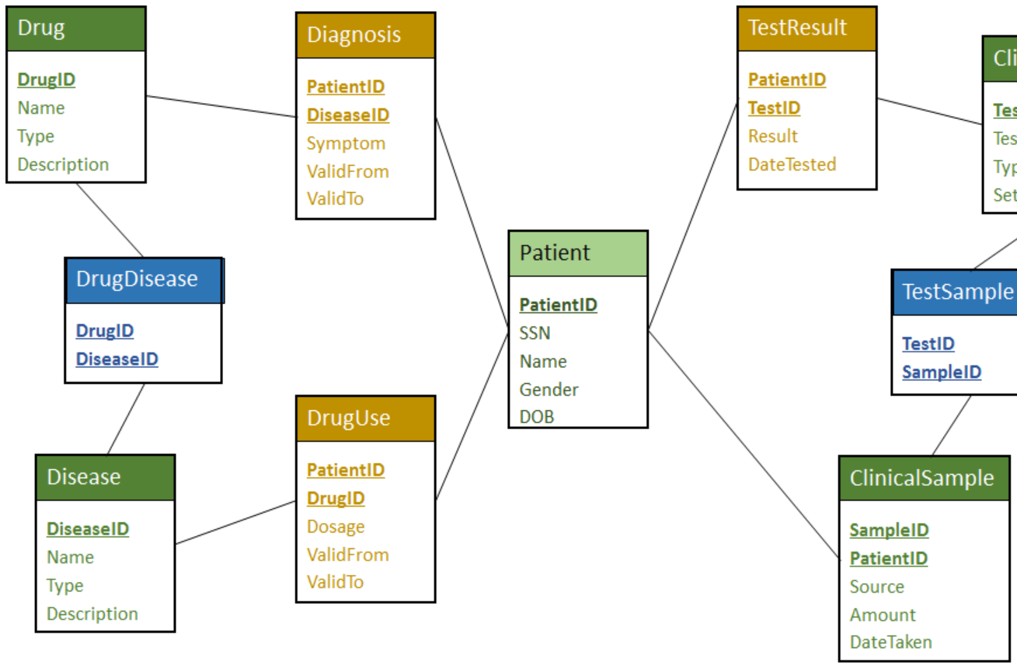
1. Diagnosis - Connects patient with disease
2. Drug_use - Connects drug with patient
3. GO_annotation - Connects go with gene
4. clinical_sample - Connects sample with patient

We have implemented Not Null, Primary Key and Foreign key constraints whenever necessary. Also, we have implemented few tables to add efficiency to querying operation for query 3.2. It will be explained later.

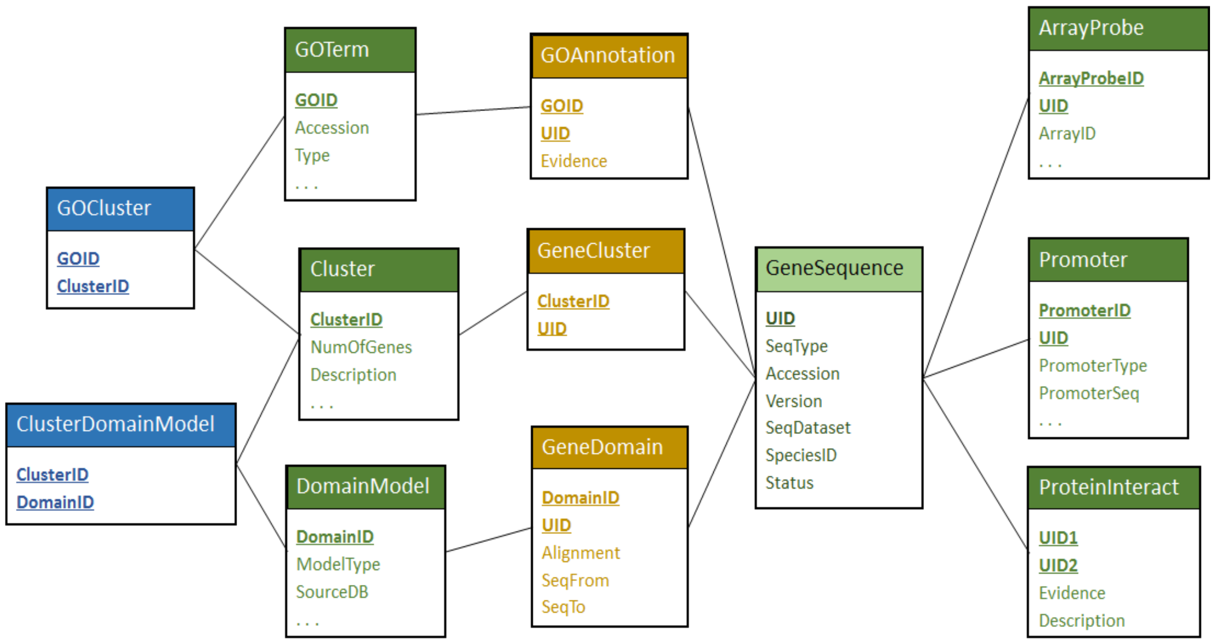
Our proposed design:



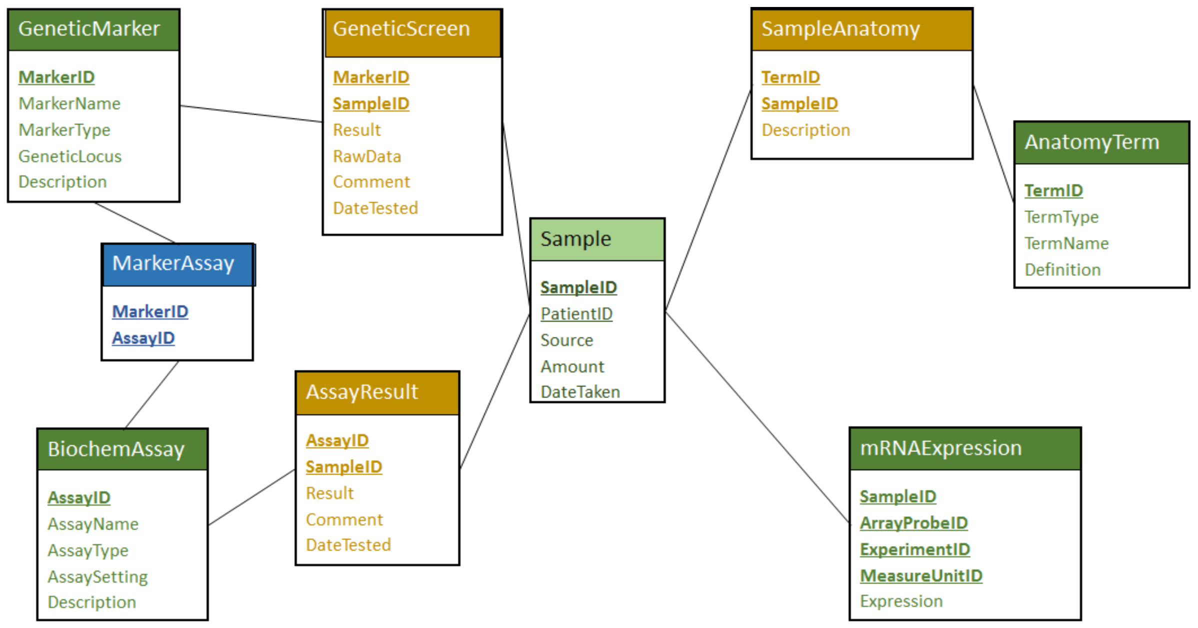
Microarray Data space



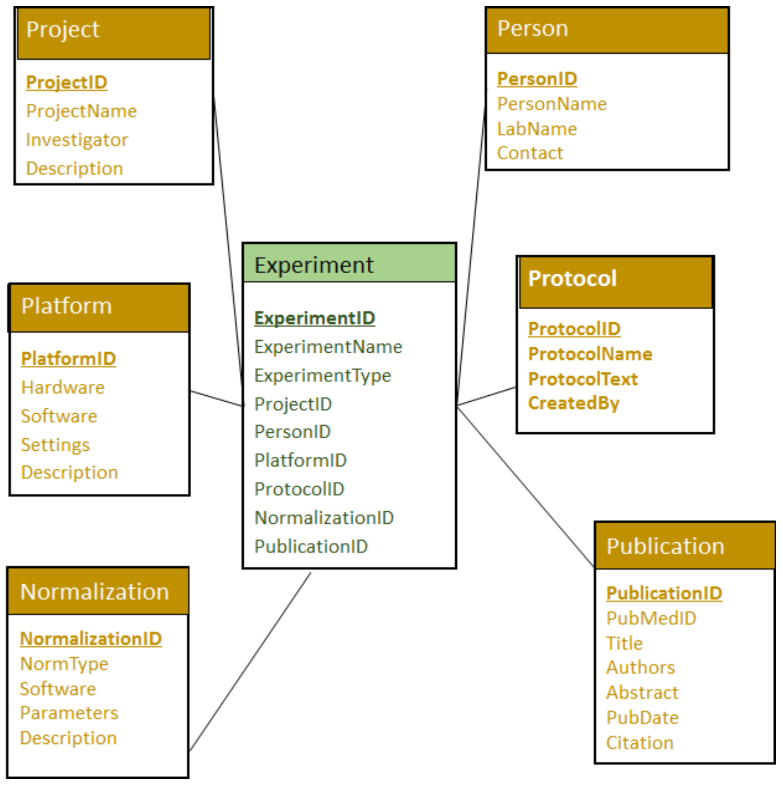
Clinical Data Space



Gene data space



Clinical Sample Data space



Experiment Data space

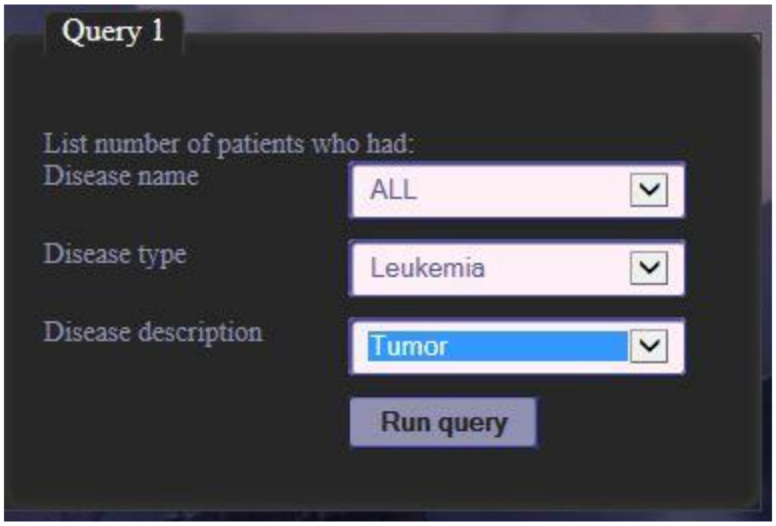
We had thought of implementing additional connector tables (shown in blue) as an enhancement to Bio-star schema. However, after analysis, we found out that they are of not much use for the current scope of the project. Hence we decided to go with Bio Star.

Part II: Regular and statistical OLAP operations

1. **List the number of patients who had “tumor” (disease description), “leukemia” (disease type) and “ALL” (disease name), separately.**

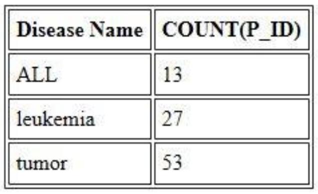
We have created dropdowns where disease description, disease type and disease name can be varied and the query is generated dynamically with given information. This query is then executed against the Data Warehouse and results are displayed on the webpage.

Time complexity = O(mn) where m and n are tuples in diagnosis and disease resp.
SQL Query:
select d.name,count(dg.p_id) from diagnosis dg inner join disease d on (dg.ds_id = d.ds_id) where d.name = 'ALL' group by d.name
Union
select d.type,count(dg.p_id) from diagnosis dg inner join disease d on (dg.ds_id = d.ds_id) where d.type = 'leukemia' group by d.type
Union
select d.description ,count(dg.p_id) from diagnosis dg inner join disease d on (dg.ds_id = d.ds_id) where d.description = 'tumor' group by d.description;


Query 1

List number of patients who had:
Disease name: ALL
Disease type: Leukemia
Disease description: Tumor

Query Results:



Disease Name	COUNT(P_ID)
ALL	13
leukemia	27
tumor	53

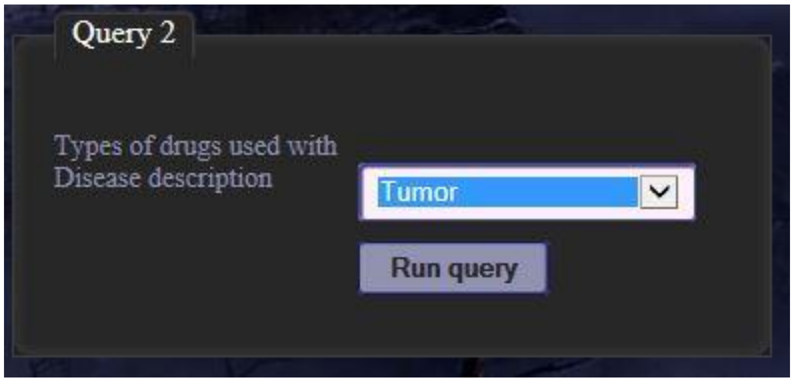
2. List the types of drugs which have been applied to patients with "tumor".

In this query, we have provided an dropdown where one can choose the disease description and we can get the list of types of drugs which have been applied to the patients who belong to that specific disease description.

Time complexity = O(n^3) where n is number of tuples

SQL Query:

select distinct type from drug where DR_ID in
(select du.dr_id from drug_use du where du.P_ID in (select distinct dg.p_id from diagnosis dg, disease d where dg.ds_ID = d.DS_ID and d.DESCRIPTION='tumor'));



Query Results: Number of drug types = 20

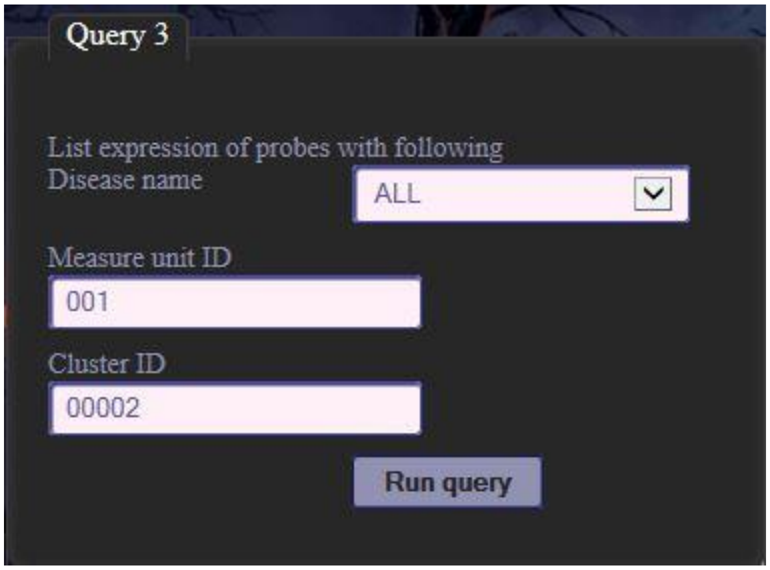
TYPE
Drug Type 002
Drug Type 018
Drug Type 013
Drug Type 001
Drug Type 008
Drug Type 009
Drug Type 017
Drug Type 005
Drug Type 012
Drug Type 004
Drug Type 016
Drug Type 014
Drug Type 006
Drug Type 007
Drug Type 003
Drug Type 015
Drug Type 019
Drug Type 010
Drug Type 011
Drug Type 020

3. For each sample of patients with "ALL", list the mRNA values (expression) of probes in cluster id "00002" for each experiment with measure unit id = "001". (Note: measure unit id corresponds to mu_id in microarray_fact.txt, cluster id corresponds to cl_id in gene_fact.txt, mRNA expression value corresponds to exp in microarray_fact.txt, UID in probe.txt is a foreign key referring to gene_fact.txt)

We have provided dropdowns for selecting disease name, cluster id and measure unit id. Based on selected values, we list the MRNA values and display them on our DW user interface.

SQL Query:

select exp from microarray_fact where s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'ALL')) and s_id is not null) and pb_id in (select pb_id from probe where U_ID in (select u_id from gene_cluster where cl_id = '00002')) and mu_id = '001';



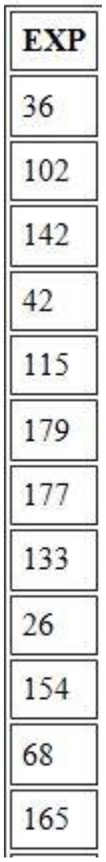
List expression of probes with following

Disease name: ALL

Measure unit ID: 001

Cluster ID: 00002

Query Results: Total number of expressions = 325

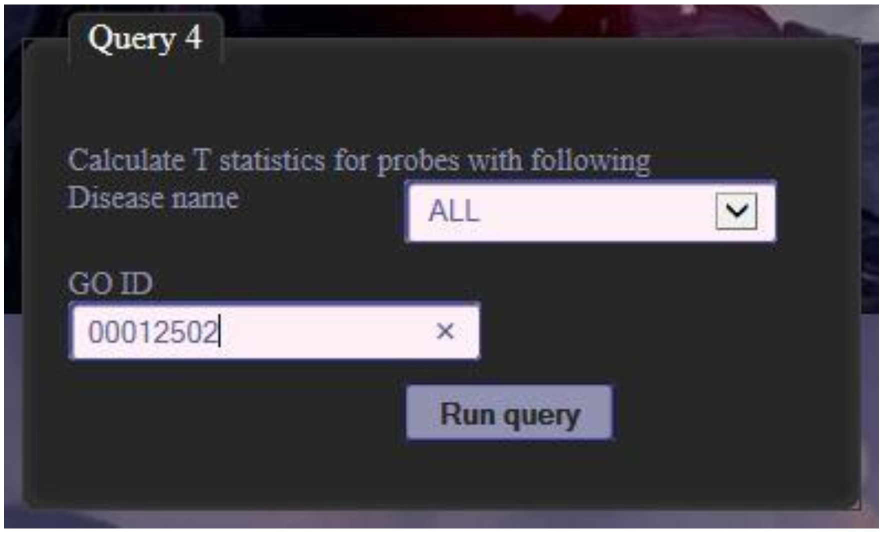


EXP
36
102
142
42
115
179
177
133
26
154
68
165

4. For probes belonging to GO with id = "0012502", calculate the t statistics of the expression values between patients with "ALL" and patients without "ALL". (Note: Assume the expression values of patients in both groups have equal variance, use the t test for unequal sample size, equal variance)
In this query, GO_id can be entered and also the disease for which we want to calculate t-stats can be selected from a dropdown. Once selected, we use Apache Commons Math3 library functions to calculate t-stats. (Assuming equal variance.)

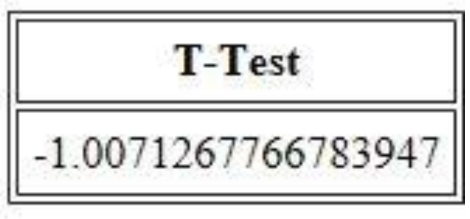
SQL Query With ALL:
select exp from microarray_fact where pb_id in (select pb_id from probe where u_id in (select u_id from go_annotation where go_id = '00012502')) and s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'ALL')) and s_id is not null);

SQL Query Without ALL
select exp from microarray_fact where pb_id in (select pb_id from probe where u_id in (select u_id from go_annotation where go_id = '00012502')) and s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name != 'ALL')) and s_id is not null);



Calculate T statistics for probes with following
Disease name: ALL
GO ID: 00012502

Query Results:



T-Test
-1.0071267766783947

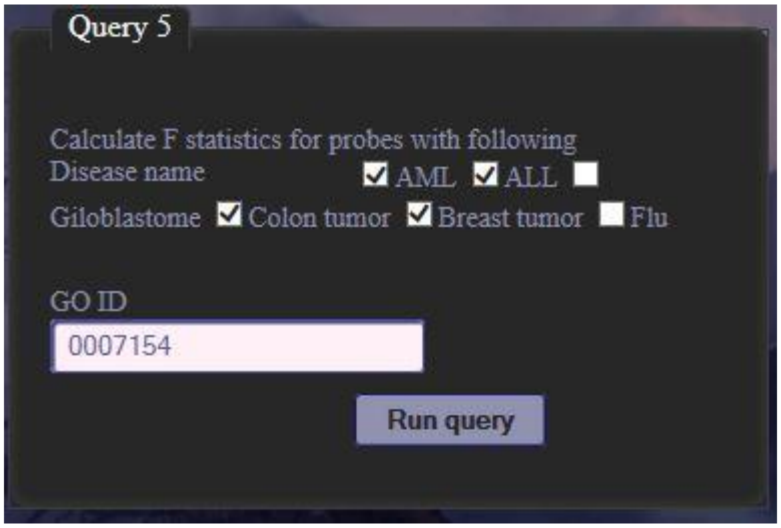
5. For probes belonging to GO with id="0007154", calculate the F statistics of the expression values among patients with "ALL", "AML", "colon tumor" and "breast tumor". (Note: Assume the variances of expression values of all four patient groups are equal.)
In this query, GO id can be entered in a textbox and also checkboxes where one or more disease names can be selected.

SQL Query:
select exp from microarray_fact where pb_id in (select pb_id from probe where u_id in (select u_id from go_annotation where go_id = '0007154')) and s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'AML')) and s_id is not null);

select exp from microarray_fact where pb_id in (select pb_id from probe where u_id in (select u_id from go_annotation where go_id = '0007154')) and s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'Colon tumor')) and s_id is not null);

select exp from microarray_fact where pb_id in (select pb_id from probe where u_id in (select u_id from go_annotation where go_id = '0007154')) and s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'Breast tumor')) and s_id is not null);

select exp from microarray_fact where pb_id in (select pb_id from probe where u_id in (select u_id from go_annotation where go_id = '0007154')) and s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'ALL')) and s_id is not null);

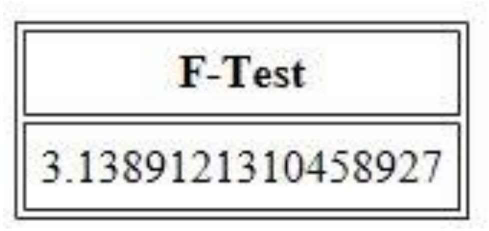


Calculate F statistics for probes with following

Disease name ✓ AML ✓ ALL ☐ Giloblastome ✓ Colon tumor ✓ Breast tumor ☐ Flu

GO ID: 0007154

Query Results:



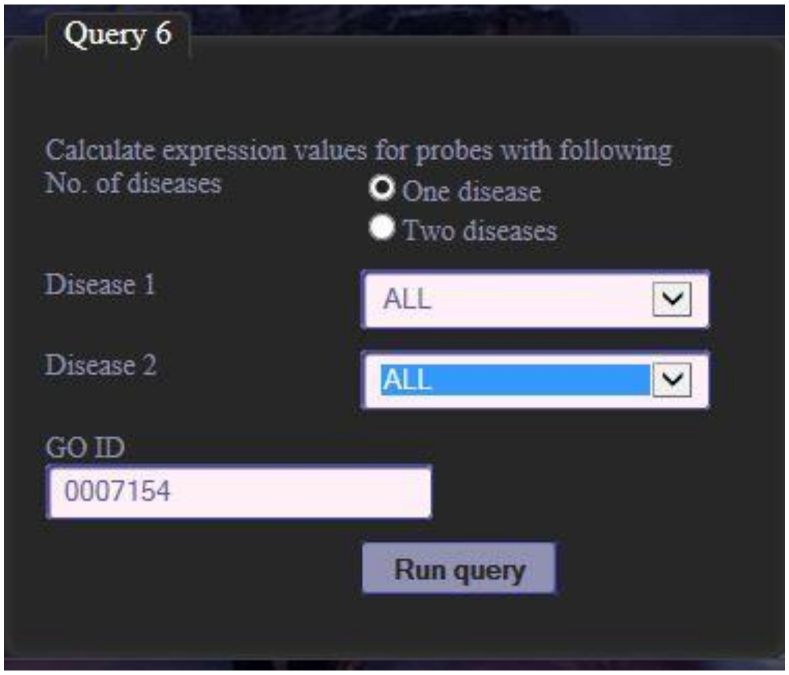
F-Test
3.1389121310458927

6. For probes belonging to GO with id="0007154", calculate the average correlation of the expression values between two patients with "ALL", and calculate the average correlation of the expression values between one "ALL" patient and one "AML" patient.

SQL Query:

SELECT diag.P_ID, mrna.EXP FROM MICROARRAY_FACT mrna, SAMPLE samp, CLINICAL_SAMPLE clinsamp, DIAGNOSIS diag WHERE mrna.S_ID = samp.S_ID AND samp.S_ID = clinsamp.S_ID AND clinsamp.P_ID = diag.P_ID AND diag.DS_ID IN (SELECT DS_ID FROM DISEASE WHERE NAME = 'ALL') AND mrna.PB_ID IN (SELECT prob.PB_ID FROM PROBE prob, GENE_FACT genefact WHERE genefact.GO_ID = '007154' AND genefact.U_ID = prob.U_ID);

SELECT diag.P_ID, mrna.EXP FROM MICROARRAY_FACT mrna, SAMPLE samp, CLINICAL_SAMPLE clinsamp, DIAGNOSIS diag WHERE mrna.S_ID = samp.S_ID AND samp.S_ID = clinsamp.S_ID AND clinsamp.P_ID = diag.P_ID AND diag.DS_ID IN (SELECT DS_ID FROM DISEASE WHERE NAME = 'AML') AND mrna.PB_ID IN (SELECT prob.PB_ID FROM PROBE prob, GENE_FACT genefact WHERE genefact.GO_ID = '007154' AND genefact.U_ID = prob.U_ID);


Query 6

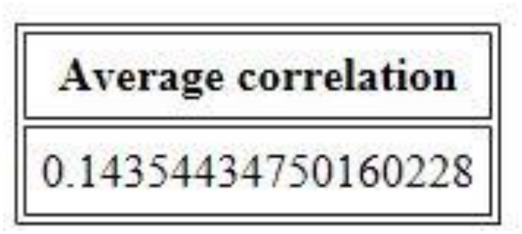
Calculate expression values for probes with following
No. of diseases [✓ One disease] Two diseases

Disease 1: ALL

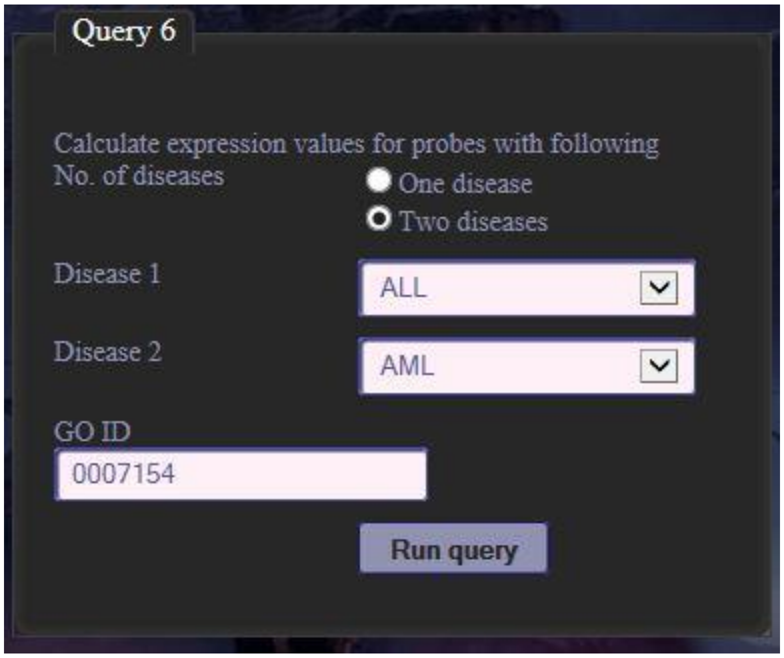
Disease 2: ALL

GO ID: 0007154

Query Results:



Average correlation
0.14354434750160228



Query 6

Calculate expression values for probes with following

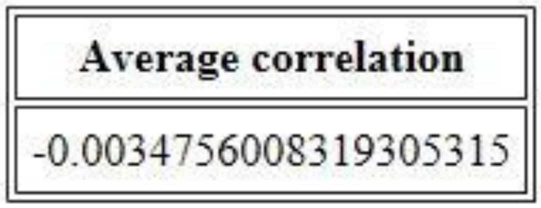
No. of diseases: ( ) One disease [✓ Two diseases]

Disease 1: ALL

Disease 2: AML

GO ID: 0007154

Query Results:



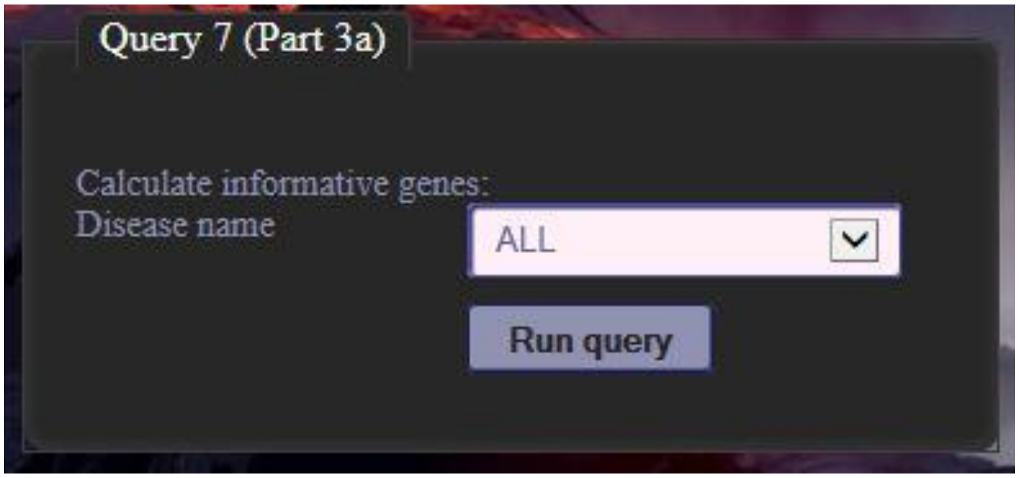
Average correlation
-0.0034756008319305315

Part III : Using data warehouse and OLAP operations to support knowledge discovery

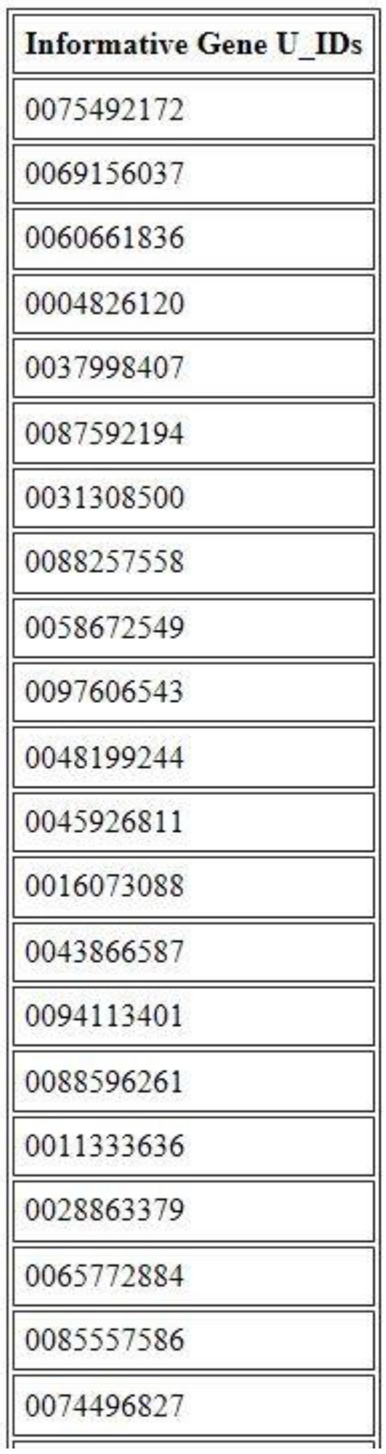
1. Given a specific disease, find the informative genes.
We allow the user to input a disease name through the dropdown and on execution we get the list of informative genes.
Here is a partial list of t-stats calculated intermediately.

SQL Query:
select g.u_id, mf.exp from microarray_fact mf inner join probe p on mf.pb_id = p.pb_id inner join gene g on p.u_id = g.u_id where s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name = 'ALL')) and s_id is not null) order by g.u_id;

select g.u_id, mf.exp from microarray_fact mf inner join probe p on mf.pb_id = p.pb_id inner join gene g on p.u_id = g.u_id where s_id in (select s_id from clinical_sample where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from disease where name != 'ALL')) and s_id is not null) order by g.u_id;



Query Results : Total informative genes found = 38



Informative Gene U_IDs
0075492172
0069156037
0060661836
0004826120
0037998407
0087592194
0031308500
0088257558
0058672549
0097606543
0048199244
0045926811
0016073088
0043866587
0094113401
0088596261
0011333636
0028863379
0065772884
0085557586
0074496827

2. Use informative genes to classify a new patient

Sample_test table contains mRNA values of 5 new patients that we have to classify. We have provided drop-down menu to select the disease for which we have to check the new patient. We take each patient from this table, get informative genes for that patient and calculate their correlation to mRNA values of each of the patients with the given disease. We also perform the same step for patients without the given disease. Based on this information, we can classify the new patient as with disease or without disease.

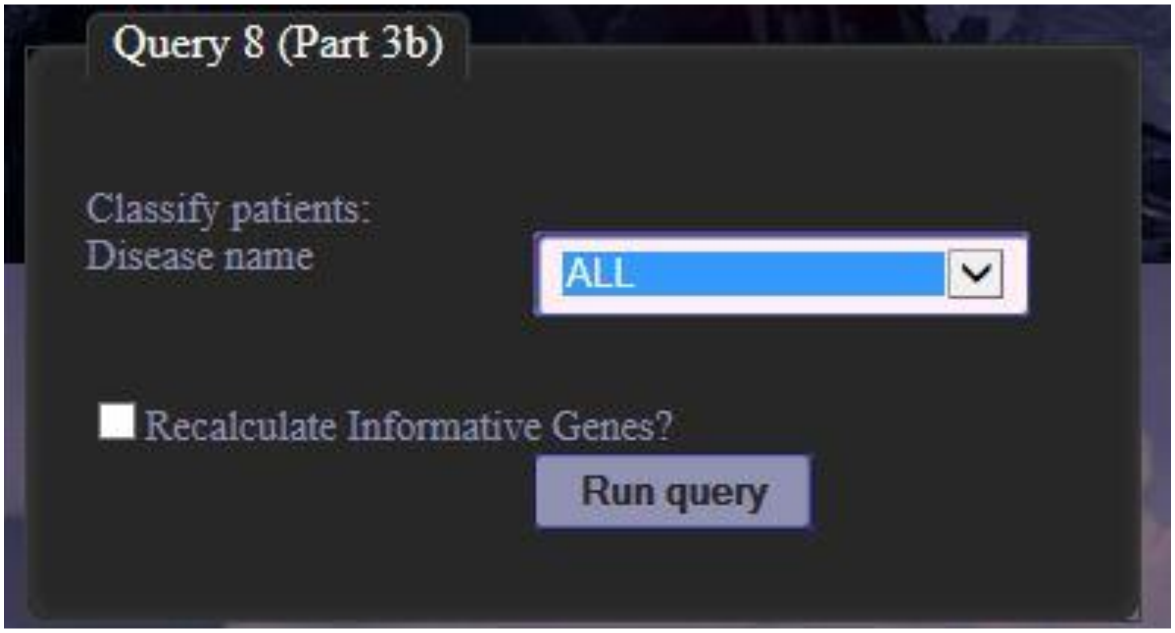
In addition to this, we have decided to precompute and store the informative genes for all diseases in tables. This will help tremendously when the number of entries of patients and their gene information will grow very large. Since we pre-compute the informative genes for each disease, each new patient's classification can be done very efficiently. This is particularly useful since in a DW, entries are not very frequently inserted/modified/deleted. However, we want a fast querying performance.

SQL QUERY:

```
select cs.p_id,g.u_id,mf.exp from microarray_fact mf inner join probe p on mf.pb_id =
p.pb_id inner join  gene g on p.u_id = g.u_id inner join (select * from clinical_sample
where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from
disease where name = 'ALL')) and s_id is not null) cs on mf.s_id=cs.s_id  where g.u_id in
(select u_id from all_ig) order by cs.p_id, g.u_id;
```

```
select cs.p_id,g.u_id,mf.exp from microarray_fact mf inner join probe p on mf.pb_id =
p.pb_id inner join  gene g on p.u_id = g.u_id inner join (select * from clinical_sample
where p_id in (select distinct p_id from diagnosis where ds_id in (select ds_id from
disease where name != 'ALL')) and s_id is not null) cs on mf.s_id=cs.s_id  where g.u_id in
(select u_id from all_ig) order by cs.p_id, g.u_id;
```

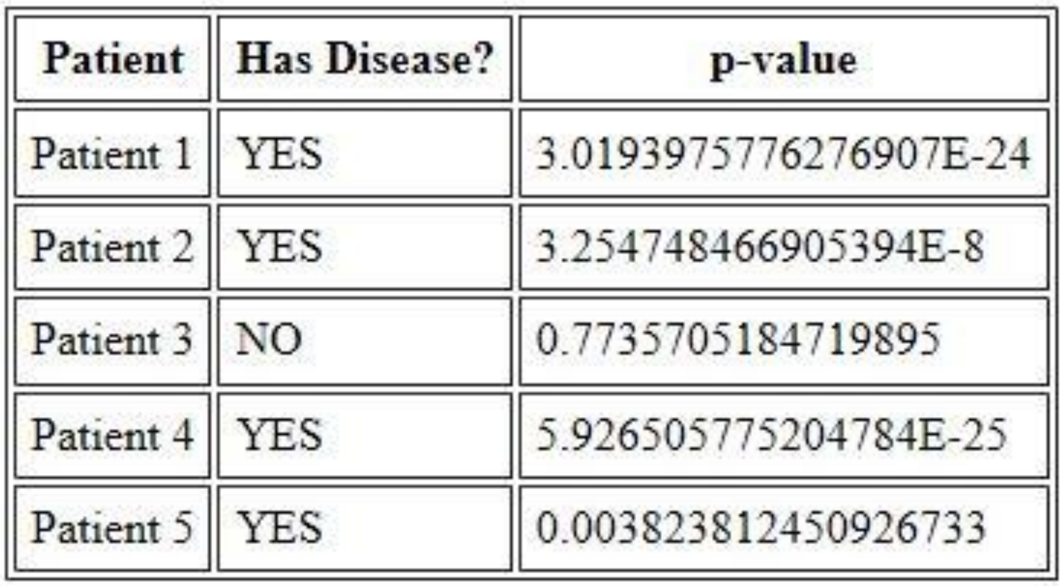
```
select * from test_samples where u_id in (select u_id from all_ig) order by u_id;
```


Query 8 (Part 3b)

Classify patients:
Disease name: ALL

☐ Recalculate Informative Genes?

Query Results:



Patient	Has Disease?	p-value
Patient 1	YES	3.0193975776276907E-24
Patient 2	YES	3.254748466905394E-8
Patient 3	NO	0.7735705184719895
Patient 4	YES	5.926505775204784E-25
Patient 5	YES	0.003823812450926733

Conclusion:

We have successfully designed and implemented Data warehouse, with support for regular and statistical OLAP operations. ALso we have used these operations for knowledge discovery. In addition to that, we have used a concept similar to materialized view to significantly reduce querying time for larger data sets (For Part 3).