

Report: Data Warehouse/OLAP System

Project 1

Nitish Goel 50170089

Rohith Doraiswamy Konoor 50169785

Venkata Ramesh Thetakali 50169239

Introduction

The necessity to deal with complex data-sets had led us to evaluate various different schemas (Biostar, Star, Snowflake, Biostar+), which have their own merits, nevertheless, we chose to opt for a more simpler and novel solution keeping in mind our project scope.

Part 1 - Our Schema Implementation:

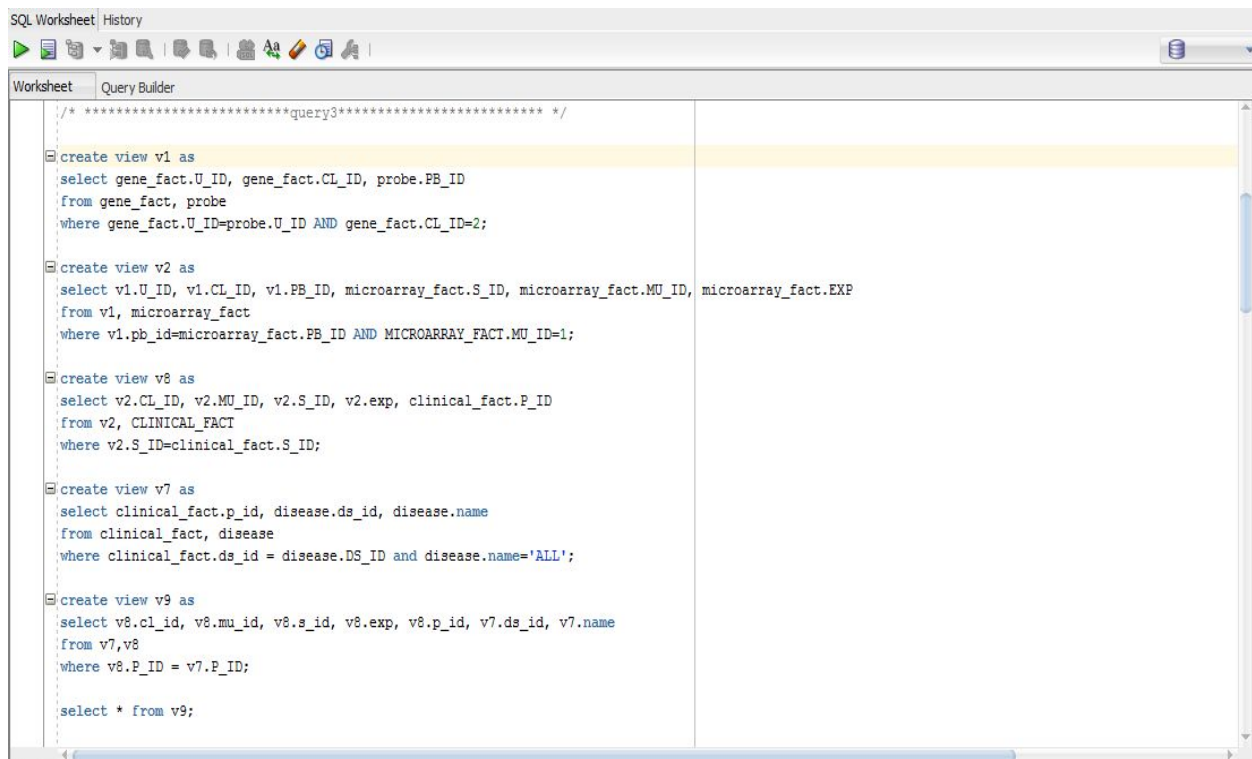
The first thing that we did was that we imported the data-set provided directly into dedicated tables, while ensuring the integrity of the data was not compromised. To ensure every attribute retained the true nature with respect to the data, we modified certain attributes like `ds_date`, `ds_from`, `ds_to` etc to be of DATE type attribute. This ensured that whenever we wanted to query data that involved some time interval range, for example checking disease start and end date, we could use SQL “date” functions directly instead of performing string match. Similarly queries pertaining to integer and double values can be handled easily as we can use arithmetic functions with ease to increase the efficiency of the data warehouse.

We also performed data-cleaning within our schema, wherein we changed all blank values from all the tables that were stored as a blank string (`""`), to NULL type. This further refined our queries as it removed string matching overhead, and allowed us to validate data by checking whether the data attribute in a tuple was NULL or not.

Using Intermediate Views:

After performing this data sanitization, we moved towards performing our OLAP queries. As the data was distributed across various tables, we noticed that executing queries would require us to perform multiple join operations to obtain desired results. Therefore, we decided to create temporary tables (by virtue of creating views) for storing intermediate results. Creating these views allowed us to visualize the data clearly (moreover, we felt it was more intuitive to us as opposed to performing many joins in a single query) and using these views we derived the final result for some of the queries asked.

One Example Case (Part II - 3rd Query):



```
/* *****query3***** */  
  
create view v1 as  
select gene_fact.U_ID, gene_fact.CL_ID, probe.PB_ID  
from gene_fact, probe  
where gene_fact.U_ID=probe.U_ID AND gene_fact.CL_ID=2;  
  
create view v2 as  
select v1.U_ID, v1.CL_ID, v1.PB_ID, microarray_fact.S_ID, microarray_fact.MU_ID, microarray_fact.EXP  
from v1, microarray_fact  
where v1.pb_id=microarray_fact.PB_ID AND MICROARRAY_FACT.MU_ID=1;  
  
create view v8 as  
select v2.CL_ID, v2.MU_ID, v2.S_ID, v2.exp, clinical_fact.P_ID  
from v2, CLINICAL_FACT  
where v2.S_ID=clinical_fact.S_ID;  
  
create view v7 as  
select clinical_fact.p_id, disease.ds_id, disease.name  
from clinical_fact, disease  
where clinical_fact.ds_id = disease.DS_ID and disease.name='ALL';  
  
create view v9 as  
select v8.cl_id, v8.mu_id, v8.s_id, v8.exp, v8.p_id, v7.ds_id, v7.name  
from v7,v8  
where v8.P_ID = v7.P_ID;  
  
select * from v9;
```

As we can see from the example figure above, another reason for using intermediate views was that by using them we introduce some scope for handling dynamism at almost all levels of the query which involves any variables or conditions. This gave a lot of flexibility in terms of handling modified queries and new queries as compared to queries with a lot of joins which usually require multiple conditions to handle.

However, we noticed that the number of views we were creating pertaining to each query were actually turning out to be difficult to manage. Furthermore, we noticed that creating multiple views through JAVA for different queries sometimes resulted in us getting a SQL Timed-Out Exception. Even though the queries ran perfectly on the SQL end, we figured that we needed a better approach if the queries were to be invoked through JAVA.

Creating Dedicated Auxiliary Tables:

In pursuit of a better approach, we settled for creating dedicated auxiliary tables that contained data which we thought would be most frequently queried. One reason for using auxiliary tables was that we felt that the data would now be more organised and this would also help us in reducing the number of join operations. Considering the same aforementioned query, our queries were now modified as:

```
/** optimized query3 */  
select cl_id, mu_id, exp, p_id, ds_name  
from gene_fact, probe, microarray_fact, PATIENTDISEASESAMPLE  
where clusters.cl_id = gene_fact.cl_id  
and gene_fact.U_ID = probe.U_ID  
and gene_fact.cl_id = 2  
and probe.pb_id = microarray_fact.pb_id  
and microarray_fact.mu_id = 1  
and microarray_fact.s_id = PATIENTDISEASESAMPLE.S_ID  
and PATIENTDISEASESAMPLE.DS_NAME = 'ALL';
```

The above query shows an optimized version by using an intermediate table called patientdiseasesample (the table has been shown in the next figure). This shows how we can avoid multiple views and create a single query that can handle the sample query requirements. This however limits our scope to add dynamism as the query construction is java needs to handle multiple cases in creating the query string.

	P_ID	DS_ID	DS_NAME	S_ID
1	47880	2	ALL	973218
2	90893	5	Breast tumor	792525
3	13243	5	Breast tumor	940071
4	14087	5	Breast tumor	132730
5	79352	4	Colon tumor	309825
6	62215	1	Giloblastome	449398
7	70863	2	ALL	290514
8	86986	1	Giloblastome	878232
9	70045	4	Colon tumor	113849
10	77689	2	ALL	550524
11	22162	2	ALL	632702
12	68707	4	Colon tumor	277464
13	304	3	AML	516776
14	58484	2	ALL	419875
15	68309	3	AML	524329
16	13258	2	ALL	580723
17	93542	1	Giloblastome	784165
18	21772	4	Colon tumor	726933

The table above is an intermediate table that connects all the IDs of patients to their diseases and the sample IDs to which they are connected. This table lets us handle quite a few sample queries that relate to patients and their genes and expressions as the sample ID is the connecting factor for most of them.

While the use of dedicated tables allowed better organisation of our data, it also reduced the time taken to obtain our results, thereby eliminating the timed-out exceptions that we were getting earlier. As this database is architected for performing bioinformatic data OLAP operations and assisting in knowledge discovery, we're naming our database BAK Database (Bioinformatics Assisting Knowledge-Discovery Database).

Time complexity:

As we can see from the UI, we are calculating the time taken in milliseconds for each query to retrieve the information from the database. The queries are fast for almost all of the sample queries as we have intermediate tables and respective views that not only concentrate on fast retrieval but also ensure step by step viewing of the query working in the database. We also have optimised queries using views with unique values for informative genes, which avoids the unnecessary checking of tuples with duplicate entries.

Part 2 - Results of Sample Queries in Part II:

- Query 1 Result:

Part 2-OLAP Operations Part 3-Knowledge Discovery

Query: Query 1 Query Desc: Patients with a particular disease

Disease: ALL Type: leukemia Description: tumor

Reset Execute

Query Results Query took: 27 ...

PATIENT_ID	DISEASE_NAME	DISEASE_TYPE	DISEASE_DESC
765	ALL	leukemia	tumor
2378	ALL	leukemia	tumor
6060	ALL	leukemia	tumor
13258	ALL	leukemia	tumor
22162	ALL	leukemia	tumor
33553	ALL	leukemia	tumor
47360	ALL	leukemia	tumor
47880	ALL	leukemia	tumor
58484	ALL	leukemia	tumor
65736	ALL	leukemia	tumor
70863	ALL	leukemia	tumor
77689	ALL	leukemia	tumor
79175	ALL	leukemia	tumor

Result Count: 13

Query executed successfully...

- Query 2 Result:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Query

Query 2

Query Desc: Drugs applied to patients with a particular disease

Disease

Select Everything

Type

Select Everyt...

Description

tumor

Reset

Execute

Query Results

Query took: 48 ...

DRUG_ID	DRUG_NAME	DRUG_TYPE	DISEASE_NAME	DISEASE_TYPE	DISEASE_DESCRIPTION
49359	Shgteilguc	Drug Type 002	Colon tumor	adeno-carcinoma	tumor
18133	Ftjsxugise	Drug Type 002	Colon tumor	adeno-carcinoma	tumor
33874	Bycgghmria	Drug Type 002	AML	leukemia	tumor
20430	Xmatplzxd	Drug Type 003	ALL	leukemia	tumor
27013	Rswfbsytf	Drug Type 004	ALL	leukemia	tumor
22137	Jitbpaalfc	Drug Type 004	AML	leukemia	tumor
33524	Rxshnupwib	Drug Type 004	Breast tumor	adeno-carcinoma	tumor
96360	Xuyhlnmlc	Drug Type 005	ALL	leukemia	tumor
3344	Ywpggtcmpr	Drug Type 005	Giloblastome	CNS_tumor	tumor
77628	Mfnzyacsvy	Drug Type 005	Giloblastome	CNS_tumor	tumor
67983	Zsqmbuylsi	Drug Type 006	Giloblastome	CNS_tumor	tumor
69199	Thedbwqfkj	Drug Type 006	Breast tumor	adeno-carcinoma	tumor
92418	Lixyoxwhpx	Drug Type 007	ALL	leukemia	tumor

Result Count: 53

Query executed successfully...

- Query 3 Result:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Query

Query 3

Query Desc: Expression values of patients with a particular disease

Disease

ALL

Cluster Id

2

MesureUnit ID

1

Reset

Execute

Query Results

Query took: 229...

CL_ID	MU_ID	EXP	P_ID	DS_NAME
2	1	36	47880	ALL
2	1	102	47880	ALL
2	1	142	47880	ALL
2	1	42	47880	ALL
2	1	115	47880	ALL
2	1	179	47880	ALL
2	1	177	47880	ALL
2	1	133	47880	ALL
2	1	26	47880	ALL
2	1	154	47880	ALL
2	1	68	47880	ALL
2	1	165	47880	ALL
2	1	144	47880	ALL

Result Count: 325

Query executed successfully...

- Query 4 Result:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Query

Query 4

Query Desc: T-Statistics of expression values between two disease groups

Disease

ALL

Gold

12502

Statistics

T Statistics

Reset

Execute

Query Results

Query took: 169...

T_OBSERVED	TWO_SIDED_P_VALUE
-1.0071267766783914876490350094840463543E00	.31406569872666135

Result Count: 1

Query executed successfully...

- Query 5 Result:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Query

Query 5

Query Desc: F-Statistics of expression values between two or more disease groups

Gold

7154

Statistics

F Statistics

Reset

Execute

F Statistics Between

☒ ALL

☒ AML

☒ Breast tumor

☒ Colon tumor

☐ Flu

☐ Glioblastoma

Query Results

Query took: 135...

F_RATIO	P_VALUE
3.139	0.025

Result Count: 1

Query executed successfully...

- Query 6 Result:

Correlation between ALL and AML Patients:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Query

Query 6

Query Desc: Avg. Correlation of expression values between two disease groups

Gold

7154

Avg. Correlation Between

ALL

A...

AML

Reset

Execute

Query Results

Query took: 421...

DISEASE_GROUP_1	DISEASE_GROUP_2	AVG_CORRELATION
ALL	AML	-0.0034756008319304595

Result Count: 1

Query executed successfully...

Correlation between ALL and ALL Patients:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Query

Query 6

Query Desc: Avg. Correlation of expression values between two disease groups

Gold

7154

Avg. Correlation Between

ALL

A...

ALL

Reset

Execute

Query Results

Query took: 231...

DISEASE_GROUP_1	DISEASE_GROUP_2	AVG_CORRELATION
ALL	ALL	0.14354434750160214

Result Count: 1

Query executed successfully...

Part 3 - Results of Sample Queries in Part III:

All Informative Genes:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Find What

Informative Genes

Disease

ALL

Threshold P-Value

0.01

Reset

Execute

Knowledge discovery

Query Results

Query took: 396 ms

INFORMATIVE_GENES	T_OBSERVED	TWO_SIDED_P_VALUE
1433276	6.76958663060671364228886318653516704851...	.000000012585127168517841
4826120	-6.5975161268525255895571880580309165309...	.000000023530152879987504
11333636	3.06822402925192148151095405068581823057...	.0034437159134287302
13947282	-2.7614806514457240554607386431021364237...	.0079786108168006242
15295292	-5.410069930248898655346540944697896864...	.0000016939097033310018
16073088	5.79088097056040298980267355802775306157...	.00000043530610551616304
18493181	-5.158010547635533062003353749924828826...	.0000041190598527929013
21633757	-4.9779037858638531674654603624925738672...	.0000077214540778103148
24984526	-2.7049715965680519616765817824314075242...	.0092629955885054748
28863379	5.25082224934096498478074559753425787116...	.0000029730666516717367
31308500	-4.8818975271590912544441756988439195898...	.000010766321379602457
31997186	-3.4190946962819865551318044056273102832...	.0012435749259310294
37998407	2.73531975918301439829218051292374043207...	.0085513282307072402

Result Count: 38

Classification Result:

- Patient 1:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Find What

Classify Patient

Disease

ALL

Threshold P-Value

.01

Patient Id

P1

Reset

Execute

Knowledge discovery

Patient P1 P-Value: 0.000000000000000000000003

Patient P1 is classified as having ALL

Query Results

Query took: 873 ms

- Patient 2:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Find What

Classify Patient

Disease

ALL

Threshold P-Value

.01

Patient Id

P2

Reset

Execute

Knowledge discovery

Patient P2 P-Value: 0.00000003254748466905394

Patient P2 is classified as having ALL

- Patient 3:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Find What

Classify Patient

Disease

ALL

Threshold P-Value

.01

Patient Id

P3

Reset

Execute

Knowledge discovery

Patient P3 P-Value: 0.7735705184719895

Patient P3 is classified as having not ALL

- Patient 4:

Part 2-OLAP Operations

Part 3-Knowledge Discovery

Find What

Classify Patient

Disease

ALL

Threshold P-Value

.01

Patient Id

P4

Reset

Execute

Knowledge discovery

Patient P4 P-Value: 0.000000000000000000000006

Patient P4 is classified as having ALL

- Patient 5:

Part 2-OLAP Operations Part 3-Knowledge Discovery

Find What: Classify Patient

Disease: ALL

Threshold P-Value: .01

Patient Id: P5

Reset Execute

Knowledge discovery

Patient P5 P-Value: 0.003823812450926733

Patient P5 is classified as having ALL

Part 4 - Additional UI features:

The User Interface (UI) implemented for our project highlights a lot of features that streamlines our querying experience. The UI takes advantage of the variables in the queries to provide a dynamic approach where the user can choose to view data that is not limited to just the sample queries by entering a new set of variable values.

The basic UI provides a result box that displays the results returned by each of the sample queries and their modified versions. We also have count of the result set to exactly show the number of entries being displayed. We have a timer showing the exact time in milliseconds taken by the query to retrieve the necessary results. We also have a description for each of the queries. The knowledge discovery part of the project has a UI space that can display any information that we have discovered while also showing the result set retrieved from the database that was used to come to the conclusion we did. Some of features, but not limited to, supported by our system are mentioned below for each of the queries.

Query 1:

We have provided dropdowns for each of the mentioned details for query one, name disease name, disease type and disease description where the user can choose his own combination of values to retrieve patients pertaining to those particular combination of values. This gives the user the freedom to find patients as per his requirement.

Query 2:

In the second query we have not only implemented dropdowns for the dynamic entry of the given variable but have also included extra variable options that can be used to filter out the results even more. The user can not only choose the disease description but can also select a disease name and disease type as well.

Query 3:

The third query has values that are used to filter the results. We have provided the user with enough flexibility to enter any value of his choice to return the exact information required by him. We have ensured the validation of the fields so that the user cannot leave a space blank and expect all the values to be shown. We have also added the option of letting him see all the disease patients rather than just looking one of them.

Query 4:

This query is to show the results of the statistics retrieved from the database. We have provided the user with dropdowns for selecting one particular disease against all the other diseases. The go id can be entered as the user wishes but makes sure that the user does enter a value for which he needs to see the results for. An additional concept that we have added is the running of an F test as well. The user can choose to run an F

test on the data as well as a T test by selecting the appropriate option from the dropdown given.

Query 5:

The fifth query includes a dedicated checkbox section, using which the user can select multiple diseases for which he/she wishes to calculate the F-statistics expression values. Also a text box for go id has been provided so that the user can enter a particular id for obtaining the desired result.

Query 6:

The average correlation of patients for a given disease and also the for one disease against another is the sample query. The user can choose a go id for which he wishes to check the average correlation for. We have provided the user with the flexibility to expand on the sample query by letting him find the average correlation for patients between any two given diseases or between patients of any single disease.

Query 7 (Informative Gene Query):

We are required to find the genes that act as informative genes for knowledge discovery in the future part for a given disease. The user is not limited by this sample query as he can find the informative genes related to any disease. The user can also choose the value that defines these genes as informative or not as it gives him the scope to further work with the query. The result set also provides the user with the values for reference.

Query 8 (New Patient Classification):

The provision of expressions for genes of new patients is the input for which we perform knowledge discovery tasks and provide results that can be used to classify the new entries. The user can provide his own set of values for which he can find the

informative genes for a given disease and classify if the new patient has that particular disease or not. The user has the liberty to tweak the value that distinguishes the genes as informative or not and has the luxury to run the classification test for any disease in the database.