10/8/2016

# Data Warehouse/ OLAP System

CSE601 – Project 1

Aditi Asthana – aditiast (50169677)
Aniket Thigale – athigale (50168090)
Shaleen Mathur – shaleenm(50170060)

# DATA WAREHOUSE SCHEMA

## SCHEMA

For the given problem we have created a Bio-Star Schema. *BioStar*, which supports all the requirements of our biomedical data warehousing. First, the *BioStar* model has the property of great extensibility, which is important for some fast-evolving data spaces such as the clinical and gene data spaces. Second, the many-to-many relationships between the central fact entity and dimensions are handled using the *m*-tables. Third, uncertain relationships between the central entity and dimensions may be kept in the *m*-tables. Finally, the *BioStar* model can be used to handle the commonly incomplete data from biomedical studies.

## INDEXING

To speed up the processing, we enabled indexing of tables by defining primary key constraints in every possible table. Oracle automatically creates a B-Tree index for tables which have primary key.

For some tables we do not have a well defined primary key. For those tables we performed Non-Unique indexing.

# USER INTERFACE

For basic queries mentioned in the project description we have used a simple console output.

We have also built an interactive and intuitive UI for the same using Javascript.

# OLAP OPERATIONS

## 1. ROLL- UP

Roll-up performs aggregation on a data cube in any of the following ways:

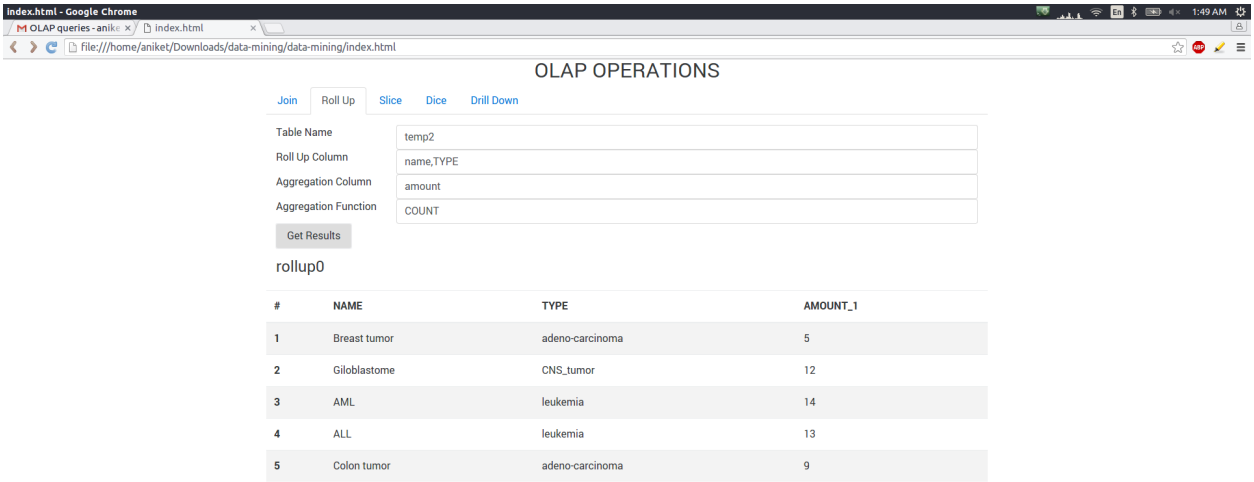
- By climbing up a concept hierarchy for a dimension
- By dimension reduction

### GENERALIZED QUERY

```
SELECT   col1,
         col2,
         col3,
         ...,
         aggregation(aggregation_col)
FROM     table1
GROUP BY col1,
         col2,
         col3,
         ...
```

## SAMPLE QUERIES

```
SELECT name,
       TYPE,
       Count(amount) AS amount_1
FROM   temp2
GROUP  BY name,
          TYPE
```




| # | NAME | TYPE | AMOUNT_1 |
|---|---|---|---|
| 1 | Breast tumor | adeno-carcinoma | 5 |
| 2 | Giloblastome | CNS_tumor | 12 |
| 3 | AML | leukemia | 14 |
| 4 | ALL | leukemia | 13 |
| 5 | Colon tumor | adeno-carcinoma | 9 |

## TIME COMPLEXITY

SQL operation performed in Roll-Up operation is a select query with aggregation. Select takes O(log(n)) and aggregation function takes O(n). So overall time complexity of this operation is :

O(nlog(n))

## 2. DRILL DOWN

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:

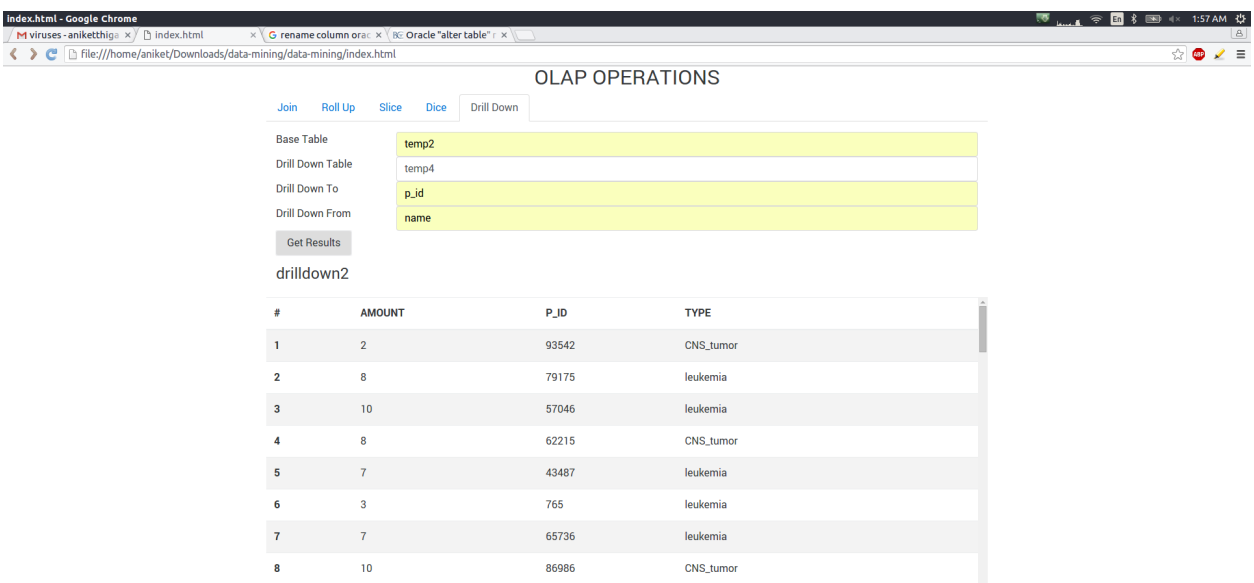
- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.

## GENERALIZED QUERY

```
SELECT col11,
       col12,
       col13,
       .....
from   basetable
where  col1i IN
       (
              SELECT col1i
              FROM   newtable)
AND    col2i IN
       (
              SELECT col2i
              FROM   newtable)
AND    ...
```

## SAMPLE QUERIES

```
SELECT amount,
       p_id,
       TYPE
FROM   temp2
WHERE  name IN (SELECT name
                FROM   temp4)
       AND TYPE IN (SELECT TYPE
                    FROM   temp4)
```




| # | AMOUNT | P_ID | TYPE |
|---|---|---|---|
| 1 | 2 | 93542 | CNS_tumor |
| 2 | 8 | 79175 | leukemia |
| 3 | 10 | 57046 | leukemia |
| 4 | 8 | 62215 | CNS_tumor |
| 5 | 7 | 43487 | leukemia |
| 6 | 3 | 765 | leukemia |
| 7 | 7 | 65736 | leukemia |
| 8 | 10 | 86986 | CNS_tumor |

## TIME COMPLEXITY

Drill down operation canbe seen a a nested select query. Simple select query takes O(log(n)) time and so nested query takes

$O(\log(n))^{k+1}$

where k is number of nested queries.

## 3. SLICE

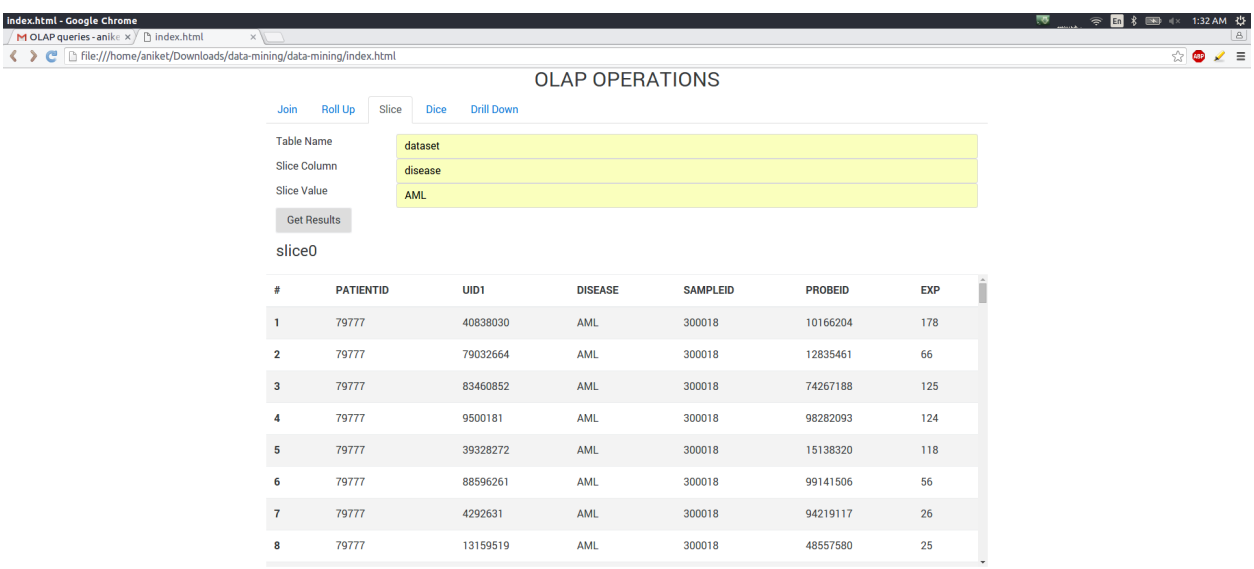
The slice operation selects one particular dimension from a given cube and provides a new sub-cube. Consider the following diagram that shows how slice works.

## GENERALIZED QUERY

```
SELECT  *
FROM    table1
WHERE   coli = val
```

## SAMPLE QUERIES

```
SELECT  *
FROM    dataset
WHERE   disease = 'AML'
```




| # | PATIENTID | UID1 | DISEASE | SAMPLEID | PROBEID | EXP |
|---|---|---|---|---|---|---|
| 1 | 79777 | 40838030 | AML | 300018 | 10166204 | 178 |
| 2 | 79777 | 79032664 | AML | 300018 | 12835461 | 66 |
| 3 | 79777 | 83460852 | AML | 300018 | 74267188 | 125 |
| 4 | 79777 | 9500181 | AML | 300018 | 98282093 | 124 |
| 5 | 79777 | 39328272 | AML | 300018 | 15138320 | 118 |
| 6 | 79777 | 88596261 | AML | 300018 | 99141506 | 56 |
| 7 | 79777 | 4292631 | AML | 300018 | 94219117 | 26 |
| 8 | 79777 | 13159519 | AML | 300018 | 48557580 | 25 |

### TIME COMPLEXITY

We can see that slice operation cab be generalized as a singe select * query. As we know that a simple SELECT * query takes average time of O(log(n)). So we can say that average time complexity of dice operation is:

O(log(n)

## 4. DICE

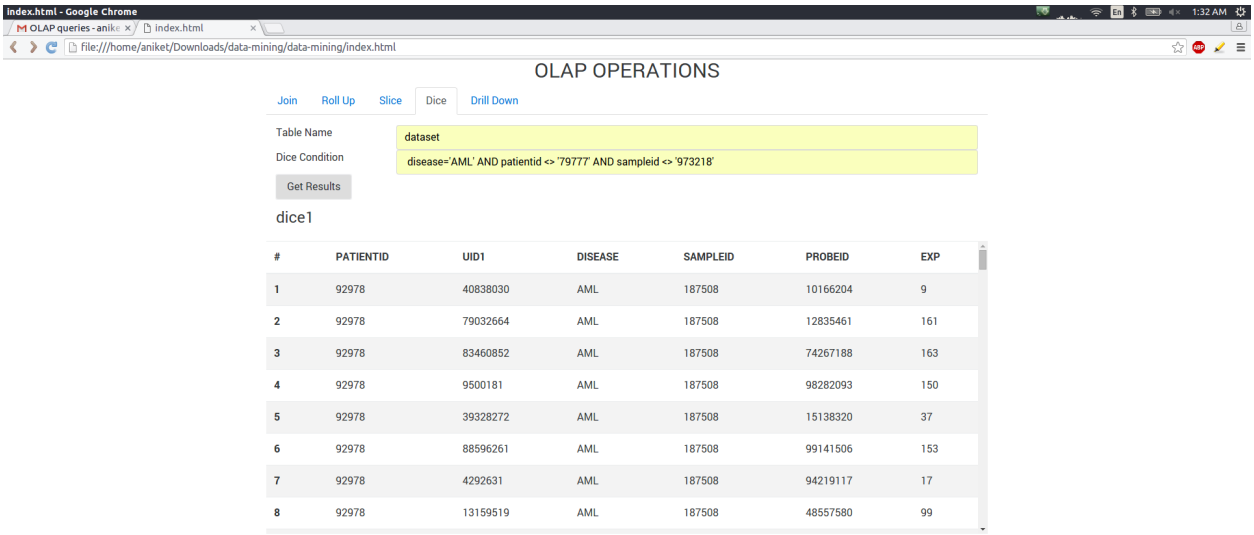
Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.

### GENERALIZED QUERY

```
SELECT *
FROM   table1
WHERE  col1i = val
AND    col2i = val
AND    ...
```

### SAMPLE QUERIES

```
SELECT *
FROM   dataset
WHERE  disease = 'AML'
       AND patientid <> '79777'
       AND sampleid <> '973218'
```




| # | PATIENTID | UID1 | DISEASE | SAMPLEID | PROBEID | EXP |
|---|---|---|---|---|---|---|
| 1 | 92978 | 40838030 | AML | 187508 | 10166204 | 9 |
| 2 | 92978 | 79032664 | AML | 187508 | 12835461 | 161 |
| 3 | 92978 | 83460852 | AML | 187508 | 74267188 | 163 |
| 4 | 92978 | 9500181 | AML | 187508 | 98282093 | 150 |
| 5 | 92978 | 39328272 | AML | 187508 | 15138320 | 37 |
| 6 | 92978 | 88596261 | AML | 187508 | 99141506 | 153 |
| 7 | 92978 | 4292631 | AML | 187508 | 94219117 | 17 |
| 8 | 92978 | 13159519 | AML | 187508 | 48557580 | 99 |

## TIME COMPLEXITY

We can see that Dice operation cab be generalized as a singe select * query. As we know that a simple SELECT * query takes average time of O(log(n)). So we can say that average time complexity of dice operation is:

O(log(n))

## 5. JOIN (NOT OLAP- BASE OF EVERY OPERATION)

This operation is used to join 2 tables based on a common attribute and different constraints. We have considered following operations while joining 2 tables:

- Join condition
- Group-by condition
- Aggregation
- Where clause

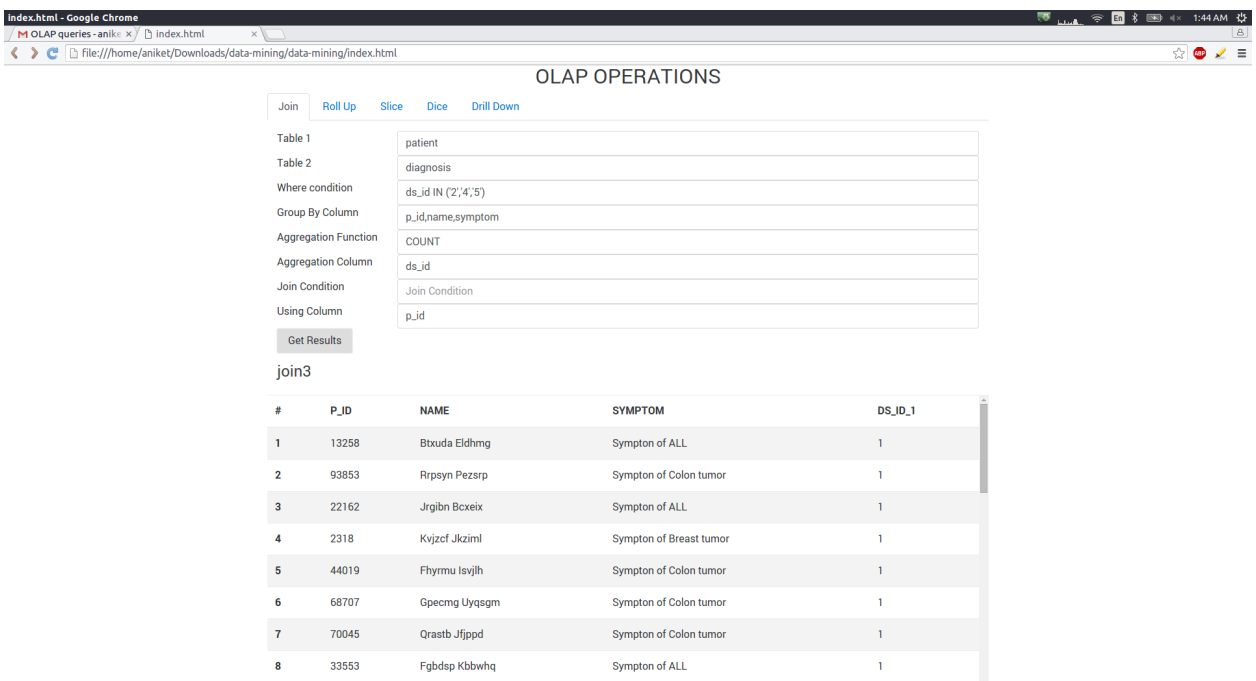
## GENERALIZED QUERY

```
SELECT          col1,
                col2,
                col3,
                ... ,
                Aggregation(aggr_col)
FROM            table1
FULL INNER JOIN table2
```

```
USING           (col_i)
WHERE           condition
GROUP BY        col1,
                col2,
                col3...
```

## SAMPLE QUERIES

```
SELECT          p_id AS patient_id,
                name,
                sympton,
                Count(ds_id) AS count_disease
FROM            patient
full inner join diagnosis
USING           (p_id)
WHERE           ds_id IN ('2',
                          '4',
                          '5')
GROUP BY        p_id, name, sympton
```




## TIME COMPLEXITY

As the table is indexed as well as Primary Key constraint is applied to the dataset, the time complexity will be :

O(min(n,m)*log(max(n,m))

## PART II QUERIES

### 1. NUMBER OF PATIENTS FOR DISEASE

List the number of patients who had " tumor" (disease description)

```
SELECT Count(p_id)
FROM   diagnosis
WHERE  ds_id IN (SELECT ds_id
                 FROM   disease
                 WHERE  description = 'tumor')
```

```
Select the query to execute:
      1. Number of patients for disease
      2. List type of drugs for disease
      3. List of mRNA values of probes for disease
      4. T-Test of expression values for patients with different disease
      5. F statistics of expression values for patients with different disease
      6. Average Correlation of expression values for patients with different disease
      7. Find Informative genes
      8. Classify patients based on informative genes
      9. Exit
Query Number : 1

Select the field you want to enter
      1. Disease description
      2. Disease type
      3. Disease name
Choice : 1
Disease Name : 'tumor'
Count of Patients with description = tumor is 53
```

List number of patients with " leukemia" (disease type)

```
SELECT Count(p_id)
FROM   diagnosis
WHERE  ds_id IN (SELECT ds_id
                 FROM   disease
                 WHERE  TYPE = 'leukemia');
```

```
Select the query to execute:
        1. Number of patients for disease
        2. List type of drugs for disease
        3. List of mRNA values of probes for disease
        4. T-Test of expression values for patients with different disease
        5. F statistics of expression values for patients with different disease
        6. Average Correlation of expression values for patients with different disease
        7. Find Informative genes
        8. Classify patients based on informative genes
        9. Exit
Query Number : 1

Select the field you want to enter
        1. Disease description
        2. Disease type
        3. Disease name
Choice : 2
Disease Name : 'leukemia'
Count of Patients with type = leukemia is 27
```

List number of patients with " ALL" (disease name)

```
SELECT Count(p_id)
FROM   diagnosis
WHERE  ds_id IN (SELECT ds_id
                 FROM   disease
                 WHERE  name = 'ALL');
```

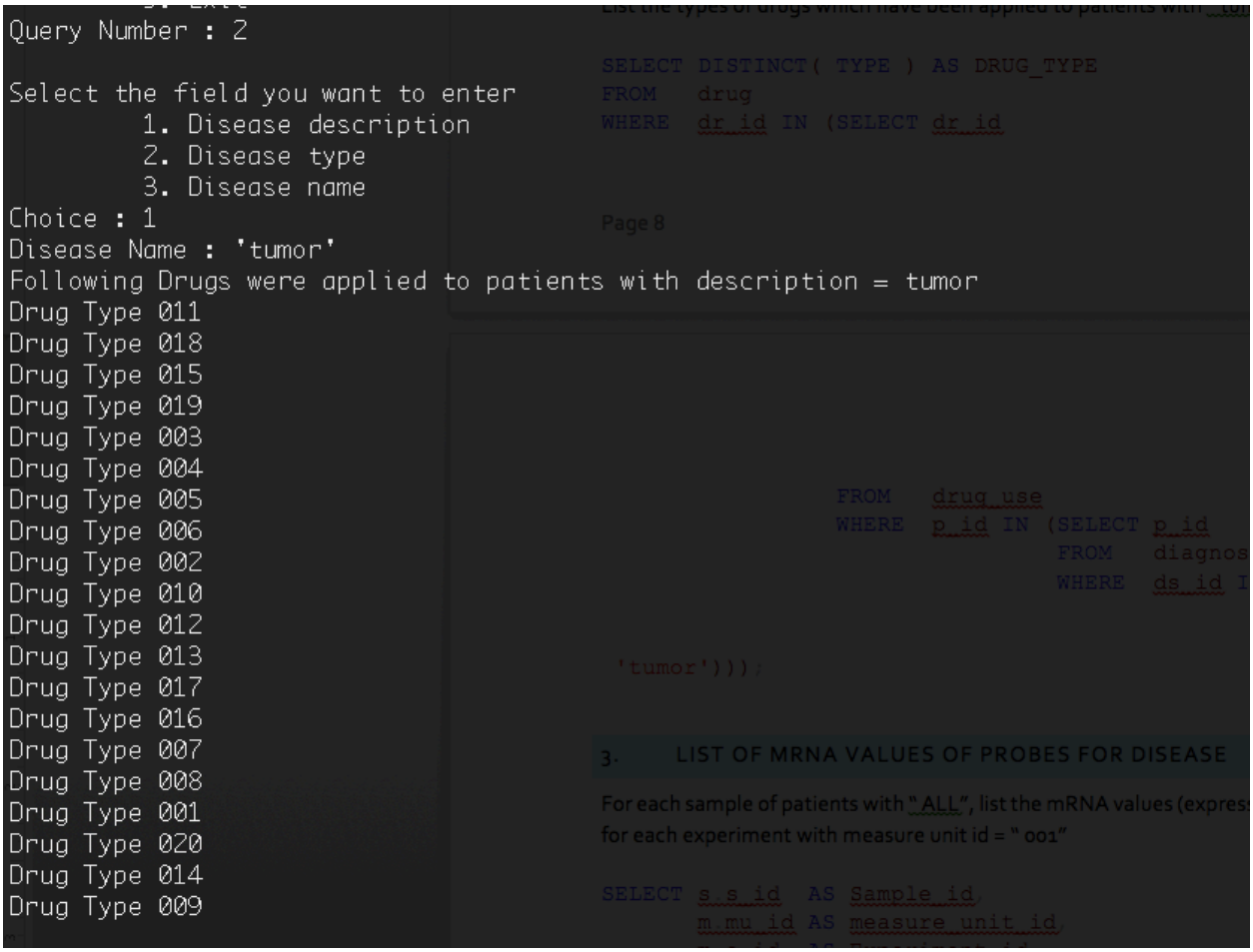
```
Select the query to execute:
        1. Number of patients for disease
        2. List type of drugs for disease
        3. List of mRNA values of probes for disease
        4. T-Test of expression values for patients with different disease
        5. F statistics of expression values for patients with different disease
        6. Average Correlation of expression values for patients with different disease
        7. Find Informative genes
        8. Classify patients based on informative genes
        9. Exit
Query Number : 1

Select the field you want to enter
        1. Disease description
        2. Disease type
        3. Disease name
Choice : 3
Disease Name : 'ALL'
Count of Patients with name = ALL is 13
```

## 2. LIST TYPE OF DRUGS FOR DISEASE

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

```
SELECT DISTINCT( TYPE ) AS DRUG_TYPE
FROM   drug
WHERE  dr_id IN (SELECT dr_id
```

```
                    FROM    drug_use
                    WHERE   p_id IN (SELECT p_id
                                     FROM    diagnosis
                                     WHERE   ds_id IN (SELECT ds_id
                                                       FROM    disease
                                                       WHERE   description =
 'tumor')));
```




## 3. LIST OF MRNA VALUES OF PROBES FOR DISEASE

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"

```
SELECT s.s_id  AS Sample_id,
       m.mu_id AS measure_unit_id,
       m.e_id  AS Experiment_id,
       m.exp   AS EXP
FROM   microarray_fact m,
       clinical_sample s,
       diagnosis d,
```

```
        disease di
WHERE   m.mu_id = '1'
        AND m.s_id = s.s_id
        AND s.p_id = d.p_id
        AND d.ds_id = di.ds_id
        AND m.pb_id IN (SELECT DISTINCT( pb_id )
                        FROM   probe
                        WHERE  uid1 IN (SELECT DISTINCT( uid1 )
                                        FROM   gene_cluster
                                        WHERE  cl_id = '2'))
        AND di.name = 'ALL';
```

```
  9. Exit
Query Number : 3
Enter the cluster id : '2'
Enter the measure unit id : '1'
Disease Name : 'ALL'
List of mRNA Values
Number of records returned : 325
('Sample_id', 'measure_unit_id', 'Experiment_id', 'EXP')
('973218', '1', '71', '36')
('973218', '1', '71', '102')
('973218', '1', '71', '142')
('973218', '1', '71', '42')
('973218', '1', '71', '115')
('973218', '1', '71', '179')
('973218', '1', '71', '177')
('973218', '1', '71', '133')
('973218', '1', '71', '26')
('973218', '1', '71', '154')
('973218', '1', '71', '68')
('973218', '1', '71', '165')
('973218', '1', '71', '144')
('973218', '1', '71', '138')
('973218', '1', '71', '197')
('973218', '1', '71', '89')
('973218', '1', '71', '146')
('973218', '1', '71', '185')
('973218', '1', '71', '121')
('973218', '1', '71', '13')
('973218', '1', '71', '51')
('973218', '1', '71', '84')
```

## 4. T-TEST OF EXPRESSION VALUES FOR PATIENTS WITH DIFFERENT DISEASE

For probes belonging to GO with id = " 0012502", calculate the t statistics of the expression values between patients with " ALL" and patients without " ALL".

```
CREATE OR replace VIEW temp_t_test
AS
```

```
  SELECT di.name AS disease,
         m.exp   AS exp
  FROM   microarray_fact m,
         clinical_sample s,
         diagnosis d,
         disease di
  WHERE  m.s_id = s.s_id
         AND s.p_id = d.p_id
         AND d.ds_id = di.ds_id
         AND m.pb_id IN (SELECT DISTINCT( pb_id )
                         FROM   probe
                         WHERE  uid1 IN (SELECT DISTINCT( uid1 )
                                         FROM   go_annotation
                                         WHERE  go_id = '12502'));

CREATE OR replace VIEW temp_t_test1
AS
  SELECT 'GroupA' AS grp,
         exp
  FROM   temp_t_test
  WHERE  disease = 'ALL'
  UNION ALL
  SELECT 'GroupB' AS grp,
         exp
  FROM   temp_t_test
  WHERE  disease <> 'ALL';

SELECT Stats_t_test_indep(grp, exp, 'STATISTIC', 'GroupA') T_OBSERVED,

       Stats_t_test_indep(grp, exp)                        P_VALUE
FROM   temp_t_test1;
```

```
Select the query to execute:
      1. Number of patients for disease
      2. List type of drugs for disease
      3. List of mRNA values of probes for disease
      4. T-Test of expression values for patients with different disease
      5. F statistics of expression values for patients with different disease
      6. Average Correlation of expression values for patients with different disease
      7. Find Informative genes
      8. Classify patients based on informative genes
      9. Exit
Query Number : 4
Enter the go id : '12502'
Enter the disease name : 'ALL'
T_OBSERVED value is -1.00712677668
P_VALUE is 0.314065698727
```

## 5. F STATISTICS OF EXPRESSION VALUES FOR PATIENTS WITH DIFFERENT DISEASE

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".

```
CREATE OR replace VIEW temp_f_test
AS
  SELECT di.name AS disease,
         m.exp   AS exp
  FROM   microarray_fact m,
         clinical_sample s,
         diagnosis d,
         disease di
  WHERE  m.s_id = s.s_id
         AND s.p_id = d.p_id
         AND d.ds_id = di.ds_id
         AND m.pb_id IN (SELECT DISTINCT( pb_id )
                         FROM   probe
                         WHERE  uid1 IN (SELECT DISTINCT( uid1 )
                                         FROM   go_annotation
                                         WHERE  go_id = '7154'));

CREATE OR replace VIEW temp_f_test1
AS
  SELECT disease,
         exp
  FROM   temp_f_test
  WHERE  disease IN ( 'ALL', 'AML', 'Breast tumor', 'Colon tumor' );

SELECT Stats_one_way_anova(disease, exp, 'F_RATIO') F_OBSERVED
FROM   temp_f_test1;
```

```
Select the query to execute:
      1. Number of patients for disease
      2. List type of drugs for disease
      3. List of mRNA values of probes for disease
      4. T-Test of expression values for patients with different disease
      5. F statistics of expression values for patients with different disease
      6. Average Correlation of expression values for patients with different disease
      7. Find Informative genes
      8. Classify patients based on informative genes
      9. Exit
Query Number : 5
Enter the go id : '7154'
Enter the disease names seperated by comma : 'ALL','AML','Colon tumor','Breast tumor'
F_OBSERVED value is 3.13891213105
```

## 6. AVERAGE CORRELATION OF EXPRESSION VALUES FOR PATIENTS WITH DIFFERENT DISEASE

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

```
CREATE OR replace VIEW dataset
AS
  SELECT s.p_id  AS patientid,
         p.uid1  AS UID1,
         di.name AS disease,
         s.s_id  AS sampleID,
         p.pb_id AS probeid,
         m.exp   AS exp
  FROM   microarray_fact m,
         clinical_sample s,
         diagnosis d,
         disease di,
         probe p,
         go_annotation g
  WHERE  m.s_id = s.s_id
         AND s.p_id = d.p_id
         AND d.ds_id = di.ds_id
         AND m.pb_id = p.pb_id
         AND p.uid1 = g.uid1
         AND g.go_id = '7154'
         AND di.name IN ( 'ALL', 'AML' );

CREATE OR replace VIEW correlation_temp3
AS
  SELECT d1.patientid                                    AS pid1,
         d1.exp                                          AS exp1,
         d2.patientid                                    AS pid2,
         d2.exp                                          AS exp2,
         Corr(d1.exp, d2.exp)
           over (
             PARTITION BY d1.patientid, d2.patientid) AS CORR
  FROM   dataset d1,
         dataset d2
  WHERE  d1.disease = 'ALL'
         AND d2.disease = 'ALL'
         AND d1.patientid <> d2.patientid
         AND d1.probeid = d2.probeid
  ORDER  BY d1.patientid,
            d2.patientid;
```

```
SELECT Avg(corr) AS CORR_ALL_ALL
FROM   correlation_temp3;
```

```
Select the query to execute:
        1. Number of patients for disease
        2. List type of drugs for disease
        3. List of mRNA values of probes for disease
        4. T-Test of expression values for patients with different disease
        5. F statistics of expression values for patients with different disease
        6. Average Correlation of expression values for patients with different disease
        7. Find Informative genes
        8. Classify patients based on informative genes
        9. Exit
Query Number : 6
Enter the go id : '7154'
Enter the disease1 name : 'ALL'
Enter the disease2 name : 'ALL'
Average Correlation between patients of ALL and ALL is 0.143544347502
```

Calculate the average correlation of the expression values between one " ALL" patient and one " AML" patient

```
CREATE OR replace VIEW dataset
AS
  SELECT s.p_id  AS patientid,
         p.uid1  AS UID1,
         di.name AS disease,
         s.s_id  AS sampleID,
         p.pb_id AS probeid,
         m.exp   AS exp
  FROM   microarray_fact m,
         clinical_sample s,
         diagnosis d,
         disease di,
         probe p,
         go_annotation g
  WHERE  m.s_id = s.s_id
         AND s.p_id = d.p_id
         AND d.ds_id = di.ds_id
         AND m.pb_id = p.pb_id
         AND p.uid1 = g.uid1
         AND g.go_id = '7154'
         AND di.name IN ( 'ALL', 'AML' );


CREATE OR replace VIEW correlation_temp3
AS
  SELECT d1.patientid                                    AS pid1,
         d1.probeid                                      AS pbid1,
         d1.exp                                          AS exp1,
```

```
        d2.patientid                                    AS pid2,
        d2.probeid                                      AS pbid2,
        d2.exp                                          AS exp2,
        Corr(d1.exp, d2.exp)
          over (
            PARTITION BY d1.patientid, d2.patientid) AS CORR
  FROM  dataset  d1,
        dataset  d2
  WHERE d1.disease = 'ALL'
        AND d2.disease = 'AML'
        AND d1.patientid <> d2.patientid
        AND d1.probeid = d2.probeid
  ORDER BY d1.patientid,
           d2.patientid;

SELECT Avg(corr) AS CORR_ALL_AML
FROM   correlation_temp3;
```

```
Select the query to execute:
        1. Number of patients for disease
        2. List type of drugs for disease
        3. List of mRNA values of probes for disease
        4. T-Test of expression values for patients with different disease
        5. F statistics of expression values for patients with different disease
        6. Average Correlation of expression values for patients with different disease
        7. Find Informative genes
        8. Classify patients based on informative genes
        9. Exit
Query Number : 6
Enter the go id : '7154'
Enter the disease1 name : 'ALL'
Enter the disease2 name : 'AML'
Average Correlation between patients of ALL and AML is -0.00347560083193
```

## PART III QUERIES

### 1. FIND INFORMATIVE GENES

Find the informative genes for the cancer "ALL"

```
CREATE OR replace VIEW patient_gene_dataset
AS
  (SELECT c.p_id  AS patientID,
          g.uid1  AS UID1,
          ds.name AS disease,
          m.exp   AS EXP
   FROM   gene g,
          probe p,
          microarray_fact m,
          clinical_sample c,
```

```
          diagnosis d,
          disease ds
   WHERE  g.uid1 = p.uid1
          AND p.pb_id = m.pb_id
          AND m.s_id = c.s_id
          AND c.p_id = d.p_id
          AND d.ds_id = ds.ds_id)


CREATE OR replace VIEW patient_gene_dataset1
AS
  SELECT 'GroupA' AS grp,
         exp,
         uid1
  FROM   patient_gene_dataset
  WHERE  disease = 'ALL'
  UNION ALL
  SELECT 'GroupB' AS grp,
         exp,
         uid1
  FROM   patient_gene_dataset
  WHERE  disease <> 'ALL'

CREATE OR replace VIEW patient_gene_dataset2
AS
  (SELECT uid1,
          Stats_t_test_indep(grp,exp,'STATISTIC', 'GroupA') T_observed
,
          Stats_t_test_indep(grp, exp) two_sided_p_value
   FROM   patient_gene_dataset1
   GROUP  BY uid1)

SELECT uid1
FROM   patient_gene_dataset2
WHERE  two_sided_p_value < 0.01
```

```
Query Number : 7
Enter the disease name : 'ALL'
Informative Genes for ALL are Following
11333636
13947282
1433276
15295292
16073088
18493181
21633757
24984526
28863379
31308500
31997186
37998407
40567338
41333415
41464216
43866587
45926811
47276861
48199244
4826120
52948490
53478188
58672549
58792011
60661836
65772884
69156037
74496827
75434512
75492172
83398521
85557586
87592194
88257558
88596261
92443312
94113401
97606543
```

## 2. CLASSIFY PATIENTS BASED ON INFORMATIVE GENES

Use informative genes to classify a new patient.
Five test cases in test_samples.txt are given in the data which we imported into the schema in form of a table named TEST_SAMPLES.

```
CREATE OR replace VIEW test_group_a
AS
  SELECT p.patientid,
         Corr(p.exp, t.exp) AS CORR
  FROM   persongene p,
         (SELECT test1 AS UID1,
                 test2 AS exp
          FROM   test_samples) t
  WHERE  p.uid1 = t.uid1
         AND p.uid1 IN (SELECT uid1
                        FROM   info_gene_all)
         AND p.disease = 'ALL'
  GROUP  BY p.patientid;

CREATE OR replace VIEW test_group_b
AS
  SELECT p.patientid,
         Corr(p.exp, t.exp) AS CORR
  FROM   persongene p,
         (SELECT test1 AS UID1,
                 test2 AS exp
          FROM   test_samples) t
  WHERE  p.uid1 = t.uid1
         AND p.uid1 IN (SELECT uid1
                        FROM   info_gene_all)
         AND p.disease <> 'ALL'
  GROUP  BY p.patientid;

SELECT Stats_t_test_indep(grp, corr) p_value
FROM   (SELECT 'GroupA' AS grp,
               corr
        FROM   test_group_a
        UNION
        SELECT 'GroupB' AS grp,
               corr
        FROM   test_group_b);
```

```
        9. Exit
Query Number : 8
Enter the disease name : 'ALL'
Enter the patient name : 'test5'
Prediction : test5 have ALL
```

```
        9. Exit
Query Number : 8
Enter the disease name : 'ALL'
Enter the patient name : 'test4'
Prediction : test4 have ALL
```

```
        9. Exit
Query Number : 8
Enter the disease name : 'ALL'
Enter the patient name : 'test3'
Prediction : test3 do not have ALL
```

```
        9. Exit
Query Number : 8
Enter the disease name : 'ALL'
Enter the patient name : 'test2'
Prediction : test2 have ALL
```

```
        9. Exit
Query Number : 8
Enter the disease name : 'ALL'
Enter the patient name : 'test1'
Prediction : test1 have ALL
```