CSE 4334/5334 – Data Mining

Fall 2014 - Project 1
Due: 11:59pm Central Time, Tuesday, Oct 16, 2014

REQUIREMENTS:

Read the following requirements carefully, and make sure you follow every rule. If you fail to meet any of requirements, marks will be deducted accordingly.

- The project document describes how you tackle the problem
  - No limitation on the number of pages, or the format of the document;
  - The document should clearly describe how you design and implement; **(5 points)**
  - The document should clearly state how to execute your program; **(5 points)**
  - The document should provide execution screenshots of your program. **(3 points)**
- Submit your source code and project document
  - **ONE zip** file contains all source codes, 3rd party libraries and documents.**(5 points)**
  - You have to implement the whole project by yourself;
  - You can use any open source library to handle TSV files;
  - Your source code must pass compilation. Any non-executable submission is not acceptable.
  - Your submission must EXCLUDE the input files; **(2 points)**
  - You can use any language, though I recommend Java, or Python;
    - If you use Java, you may include your eclipse or NetBean project folder. Please remember exclude the input files;
    - If you use python, then you only need to submit your source code.
- Compilation and execution
  - Compile and test your program on omega.uta.edu before you submit

PROBLEM SCENARIO:

BACKGROUND

All programming assignments will use the same dataset, which is collected from a career website. Project 1 is the warm-up project which focuses on data warehousing and data cube. It also gives you a chance to get familiar with data and its format and build foundations for subsequent projects.

INPUT FILE FORMAT

Each file is in .tsv (tab-separated value) format. This means that each line in a .tsv file consists of several fields, which are separated by tabs. To accommodate this file format, fields composed of text use the following ways to represent tabs, newlines, and carriage returns.

1. Tabs are represented by '\t'
2. Newlines are represented by '\n'
3. Carriage returns are represented by '\r'
4. Backslashes are represented by '\\'

For example, the following text is the description of a job (JobID: 1). In the text, the carriage returns and newlines are represented by ‘\r’, and ‘\n’ respectively:

```
<p>Security Clearance Required:&nbsp; Top Secret </p>\r<p>Job Number: TMR-
447</p>\r<p>Location of Job:&nbsp; Washington, DC</p>\r<p>TMR, Inc. is an Equal
Employment Opportunity Company</p>\r<p>For more job opportunities with TMR,
visit our website <a
href="http://www.tmrhq.com/">www.tmrhq.com</a></p>\r<p>Send Resumes to
HR@tmrhq2.com</p>\r<p>&nbsp;</p>\r<p>JOB SUMMARY:</p>\r<p>&nbsp;</p>\r<p>Leads
the customer&rsquo;s overall Cyber Security strategy, formalizes service
offerings consisted with ITIL best practices, and provides design and
architecture support.</p>\r<p>&nbsp;</p>\r<ul>\r    <li>Provide security design
/ architecture support for OJP&rsquo;s IT Security Division (ITSD) </li>\r
<li>Leads the SECOPS team in the day to day OJP Security Operations
support&nbsp; </li>\r    <li>Provides direction when needed in a security
incident or technical issues </li>\r    <li>Works in concert with network
operations on design /integration for best security posture</li>\r
<li>Supports business development functions including Capture Management,
Proposal Development and responses, and other initiatives to include
conferences, trade shows, webinars, developing white papers and the
like.</li>\r    <li>Identifies resources and mentors in-house talent to ensure
TMR remains responsive to growing initiatives and contracts with qualified
personnel.&nbsp;&nbsp; </li>\r</ul>\r<p>&nbsp;</p>\r<p><a
href="https://www.tmrhq.com/jobapplicationstep1.aspx"><span></span></a>&nbsp;</
p>
```

THE FILES

All the files can be downloaded from “Programming Assignment 1” in “Course Material” on Blackboard.

users.tsv contains information about the users. Each row of this file describes a user. The UserID column contains a user's unique id number. The remaining columns contain demographic and professional information about the users. For example:

UserID	City	State	Country	ZipCode	DegreeType	Major	Graduation Date	Work History Count	TotalYears Experience	Currently Employed	Managed Others	Managed HowMany
47	Paramount	CA	US	90723	High School		1999-06-01 00:00:00	3	10	Yes	No	0

user_history.tsv contains information about a user's work history. Each row of this file describes a job that a user held. The UserID column has the same meaning as before. The JobTitle column represents the title of the job, and the Sequence column represents the order in which the user held that job, with smaller numbers indicating more recent jobs. For example:

UserID	Sequence	JobTitle
47	1	National Space Communication Programs-Special Program Supervisor

jobs.tsv contains information about job postings. Each row of this file describes a job post. The JobID column contains the job posting's unique id number. The other columns contain information about the job posting.   For example:

JobID	Title	Description	Requirements	City	State	Country	Zip5	StartDate	EndDate
1	Security Engineer/Technical Lead	… Omitted long text (see an example in INPUT FILE FORMAT)	… Omitted long text (see an example in INPUT FILE FORMAT)	Washington	DC	US	20531	2012-03-07 13:17:01.643	2012-04-06 23:59:59

apps.tsv contains information about applications made by users to jobs. Each row describes an application. The UserID, and JobID columns have the same meanings as above, and the ApplicationDate column indicates the date and time at which UserID applied to JobId. For example:

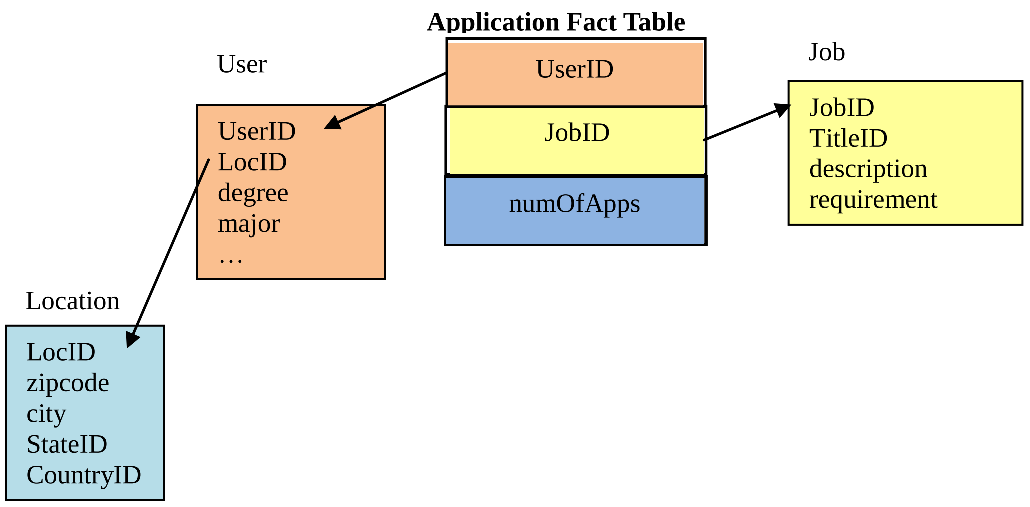
UserID	ApplicationDate	JobID
47	2012-04-04 15:56:23.537	169528

* Note 1: Be aware of NULL values in the file
* Note 2: If you want to take a peek at the content but the file is too large to open, try to use command "more". You may find tons of resources to teach you how to use it:
e.g. http://en.wikipedia.org/wiki/More_(command)

THE PROBLEMS

In this project, we're going to build a small data warehouse, which requires you to read information from one or more of the input files to answer the following questions. No database (MySQL, Oracle, etc.) should be used. In other words, you need to design/implement your data structures and algorithms to complete the tasks, instead of using existing techniques such as SQL. However, you're encouraged to import all data into database so that you can check if your implementation is correct.

Suppose we have a snowflake schema of the data warehouse for the job applications as follows:



The application fact table has one measure, numOfApps, which stores the number of applications. For our convenience, we use job title, state name, country name directly as TitleID, StateId, and CountryID, respectively.

And also we only consider the following concept hierarchies:

- user → state → country → all
- job → title → all

Task 1) **(30 points))** The goal of this task is to answer a query: list top 5 state-wise most popular job positions.

Implement a function in your program. The function
1) materializes cuboid {StateID, JobID}. In other words, it would compute the results of the following SQL query without using SQL and database. (Note the SQL query assumes we have tables *apps*, *users* corresponding to files *apps.tsv*, *users.tsv*, respectively.

```
SELECT U.StateID, A.JobID, count(*) as numOfApps
FROM apps as A, users as U
WHERE A.UserId = U.UserID
GROUP BY U.StateID, A.JobID
```

2) prints out a list of cells in the cuboid which are top 5 cells in terms of the number of applications. The list should be ordered by the number of applications in descending order. You only need to show the StateID, JobID and numOfApps for each cell.

Task 2) **(30 points)** Based on the materialized cuboid of Task 1, answer the query: list top 5 most popular job titles in a given country C (e.g. CA, US etc,.).

Implement a function that takes a parameter as country C. The function then will perform the following OLAP operations:
1) Slice on cuboid {StateID, JobID} with {Country = C}
2) Roll up to cuboid {StateID, TitleID}
3) Roll up to cuboid {CountryID, TitleID}
4) Top-5 operation

After these OLAP operations, the function will take the results of the last operation, sort the resulting cells from the cuboid, and print them out. The results should show the TitleID and numOfApps for each cell, and the results should be ordered by numOfApps in descending order.

EXPECTED OUTPUT

1. Your program can be executed using command line as followed: **(10 points)**

```
java your-main-class-name country /p/t/apps.tsv /p/t/users.tsv /p/t/jobs.tsv /p/t/user_history.tsv

or

python your-script-file.py country /p/t/apps.tsv /p/t/users.tsv /p/t/jobs.tsv /p/t/user_history.tsv
```

* country is used as the parameter for task 2.c. The value can be one of "US", "Canada" etc.
* apps.tsv, users.tsv, jobs.tsv, user_history.tsv are the paths to the input files

2. The output of the program should look like: **(10 points)**

```
Problem 1:
StateID JobID numOfApps
CA      1     2000
TX      2     1745
NY      3     728
WA      4     12
SC      5     5
```

```
Problem 2:
TitleID        numOfApps
Developer      20000
Analyst        18324
Manager        9000
Operator       4293
Secretary      3134
```

* Note: Outputs shown are not necessarily answers