Carnegie Mellon University
15-826 – Multimedia Databases and Data Mining
Fall 2014, C. Faloutsos
Homework 0
Due Date: Sep 23rd, at classroom 3:00pm
Prepared by: Jay-Yoon Lee

Reminders

- All homeworks are to be done **INDIVIDUALLY**.
- For code submission to blackboard submit a single file ([andrew-id]-hw1.tar.gz, e.g. jaylee-hw1.tar.gz) that contains all the codes for the questions wihtout the data. Write the code assuming that the data folder and src folder are in the same folder. (e.g. HW1/data/, HW1/src)
- 10% of total homeworks weight = 1% of total course grade.
- Expected effort for this homework (approximate times):
  - 2-6 hours
    - 1-4 hours to write and debug all sql queries
    - 1-2 hours to run them queries and record the answers.

Grading

- Q2,3: graded by Jay-Yoon Lee.
- Q4,5: graded by Peixin Zheng
- Q6,7: graded by Chenying Hou

Grading info:

- *Overall :*

  No SQL code relevant to answer: -5 pts
  No answer relevant to question: -5 pts
  Makefile error: -10 pts
- *Instructions for specific questions will be provided at the end of corresponding solution.*

Re-grade requests

In case you believe you deserve more points, please

- contact the corresponding TA, *only*
- in **writing/email**,
- within 1 week from the return of the homework
- specifying your name and andrew-id, and
  1. The question you are disputing (e.g., 'HW0-Q6')
  2. The reason (e.g., 'I only forgot to exclude the self-pairs')

3. The extra points you are asking (e.g., '5 more points out of 20')

In the remote case that there is still a disagreement, please contact the instructor.

DBMS and SQL [100pts]

Problem Description: For each question in this part, provide both the SQL statement(s) and the resulting answer(s), unless specified otherwise. You'll be working with *Marvel dataset* which lists the appearances of Marvel characters in comic books. [1] [2] [3]
Hint: Please use SQLite3; version 3.6.20 is already on the andrew cluster machines. You may use your own machine and your own sqlite3 installation, as long as your submitted code runs correctly on the andrew cluster machines.

Implementation Details: Write sql code for the following queries.

1. **[0 pt]** *Data preparation*

   - Download the Marvel-data.tar.gz[2] from `https://www.andrew.cmu.edu/~jaylee/Marvel-data.tar.gz` which consists of
     - `marvel.txt:` characterID (1st column) and comcID (2nd column).
     - `marvelCharacters.txt:` charcterID and the name of the character.
     - `marvelComicBooks.txt:` comicID and the name of the comic book.
   - Also download the makefile folder at `http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW0/makefile-hw0.tar.gz`.
     Implement .sql files inside the folder so that when you hit make, the answers can be printed.

[1] Marvel data reference: `http://bioinfo.uib.es/~joemiro/marvel.html`.
[2] You can download the data (`Marvel-data.tar.gz`) available at `https://www.andrew.cmu.edu/~jaylee/Marvel-data.tar.gz`
[3] make file: `http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW0/makefile-hw0.tar.gz`

Solution:

- Preprocess input files into 'tab' separated. 'Tab' shows as 4 blanks, in this solution write-up.

```
fnames="marvel.txt marvelCharacters.txt marvelComicBooks.txt"
for i in $fnames
do
    echo "working on $i ...."
    sed 's/ /    /' < $i > ${i%.txt}-new.txt
    mv $i ${i%.txt}-old.txt
    mv ${i%.txt}-new.txt $i
done
```

- Import input files

```
DROP TABLE IF EXISTS marvel;
CREATE TABLE marvel(
charId int,
comicId int
);

.separator '    '
.import marvel.txt marvel

DROP TABLE IF EXISTS marvelchar;
CREATE TABLE marvelchar(
charId int,
charName varchar
);

.separator '    '
.import marvelCharacters.txt marvelchar

DROP TABLE IF EXISTS marvelcomics;
CREATE TABLE marvelcomics(
comicId int,
comicName varchar
);

.separator '    '
.import marvelComicBooks.txt marvelcomics
```

Grading info: zero weight - no rules for partial credit

Print answers for question 2, 3 on one sheet of paper with '[course-id] [hw#] [question#] [andrew-id] [your name]'

2. **[10 pt]** 1 *Create Database and Count*
   Create and load the following table using the extracted text file.
   - `marvel(charId INTEGER, comicId INTEGER)`
   - `marvelchar(charId INTEGER, charName VARCHAR)`
   - `marvelcomics(comicId INTEGER, comicName VARCHAR )`

   Report the number of rows for each table using hw0.2.sql in the order of marvel, marvelchar, marvelcomics. Check if the number of characters and comics match the number of lines the corresponding text files.
   (Hint1: check `.help` and `.import` in SQLite3)
   (Hint2: To check the number of line in text file, use $\mathsf{wc} - \mathsf{l}$ code.)

   Solution:
   - marvel: 96662
   - marvelchar: 6486
   - marvelcomics: 12942
   - SQL code

     ```sql
     SELECT COUNT(*) FROM marvel;
     SELECT COUNT(*) FROM marvelchar;
     SELECT COUNT(*) FROM marvelcomics;
     ```

   - Counting lines of each file
     wc -l ./marvel.txt → 96662
     wc -l ./marvelCharacters.txt → 6486
     wc -l ./marvelComicBooks.txt → 12942

3. **[10 pt]** *Distinct elements in a Table* using `hw0.3.sql`
   (a) Get the distinct number of characters form 'marvel' databse and check with your previous answer. **[5 pt]**
   (b) Get the distinct number of comic books form 'marvel' databse and check with your previous answer.**[5 pt]**

(Hint: DISTINCT() command)

Solution:

- Distinct number of characters: 6486
- Distinct number of comic books: 12942

```
--num character
SELECT COUNT(DISTINCT(charId)) FROM marvel;
--num comic books
SELECT COUNT(DISTINCT(comicId)) FROM marvel;
```

Grading info: -1 pt if distinct() *is on name rather than ID.*

Print answers for question 4, 5 on one sheet of paper with '[course-id] [hw#] [question#] [andrew-id] [your name]'

4. **[15 pt]** *The most popular character A*
Find character A who appeared most frequently over all comic books?

Solution: 5306|"SPIDER-MAN/PETER PAR"

```
DROP TABLE IF EXISTS Temp;
DROP VIEW IF EXISTS popularity;
CREATE VIEW popularity AS
SELECT charId, COUNT(*) ct
FROM marvel
GROUP BY charId;

SELECT M.charId, M.charName FROM marvelchar AS M
WHERE M.charId IN (SELECT T.charId FROM popularity T
WHERE T.ct = (SELECT MAX(T2.ct) FROM popularity T2));
```

5. **[20 pt]** *Popularity distribution of characters* Our objective is to plot the popularity distribution of comic characters appearance in comic books. For that we define few terminologies:
   - p, `the popularity of characters:` the number of comic books that each characters appeared in.
   - f, `the frequency of popularity` p: count of comic characters that has the popularity p.

   Attach the plot the f vs. p in log-log scale (y-axis: f and x-axis p should both be in log scale). `Example:` If 'superman' appeared on 1000 comics, p for superman should be 1000. And if 'superman', 'spiderman', 'wolverine' are only characters that has $p = 1000$, then $p = 1000$ has $f = 3$.

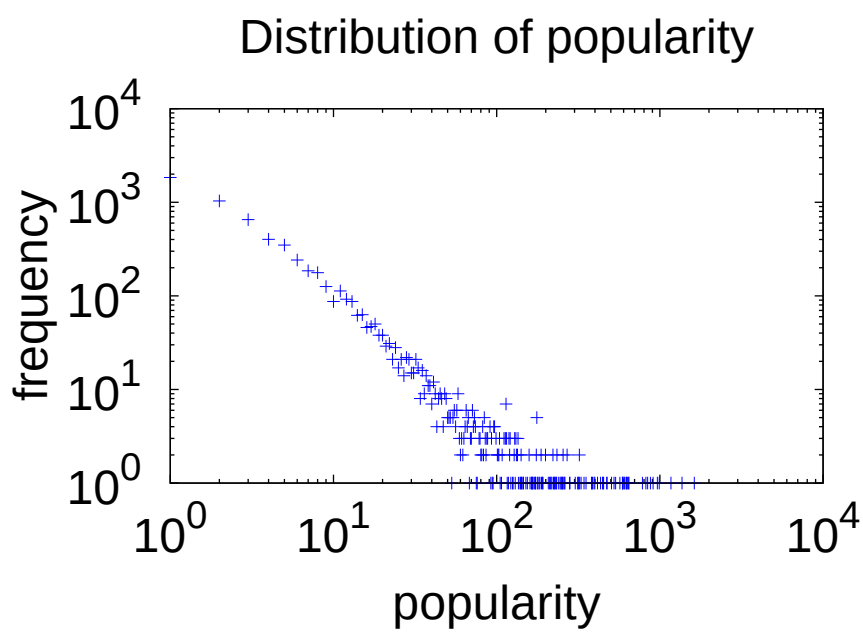
Solution:

- **SQL code**

```
DROP VIEW IF EXISTS frequency;
CREATE VIEW frequency AS
SELECT p.ct, COUNT(*) f
FROM popularity p
GROUP BY p.ct;

.mode tabs
.output pdist.txt
SELECT * FROM frequency;
```

- **Distribution plot**



Grading info: -5 pt if the plot does not look correct or is not in log-log scale.

Print answers for question 6, 7 on one sheet of paper with '[course-id] [hw#] [question#] [andrew-id] [your name]'

6. **[20 pt]** *Count of pairs*
Count the coappearing pairs in comic books excluding self-pairs, and mirror-pairs. (*Hint:* 1: Use condition `node1>node2`, a usual trick to exclude self-pairs and mirror pairs; *Hint:* 2: Join is better than nested select; *Hint:* 3: Avoid creating an intermediate table of pairs - it may be too slow).
`Example:` Excluding the mirror pairs and self pairs, Table 1 has 4 count of pairs total.

char1	char2	comicBook	pair type
Spiderman	Superman	sh1	
Spiderman	Superman	sh2	
Superman	Wolverine	sh3	
Captain America	Thor	sh4	
Wolverine	Superman	sh3	mirror pair with row 3
Spiderman	Spiderman	sh1	self pair

Table 1: Example of pairs, self-pairs and mirror-pairs.

Solution:

```
SELECT COUNT(*) FROM
marvel m1
INNER JOIN
marvel m2
ON m1.comicId = m2.comicID AND m1.charId>m2.charId;
JOined table.
```

Grading info: -5 pts if they use wrong join operator, or include self-pairs

7. **[25 pt]** *Query Optimization*
   (a) **[4 pt]** *Report* the wall-clock running time of your query for the question 6, using, e.g., the `time` Linux/Unix command.

   Solution: 22m37.902s *Grading info: -2 pts if runtime is wrong*

   (b) **[8 pt]** *Index impact: Report* the wall-clock time of your query for the question 6, again with indices on the column `charID`.

Solution: 46m29.701s

```
SELECT "started building index";

CREATE INDEX IF NOT EXISTS comic1
ON marvel (charId);

SELECT "done with building index";


--Optional, if you want the explain query (START)
SELECT "   here is the query plan------";
EXPLAIN SELECT COUNT(*) FROM
 marvel m1
INNER JOIN
 marvel m2
ON m1.comicId = m2.comicId AND m1.charId>m2.charId;
SELECT "   end of query plan - starting query -------";
--Optional, if you want the explain query (END)

SELECT COUNT(*) FROM
 marvel m1 INNER JOIN marvel m2
ON m1.comicId = m2.comicId
WHERE m1.charId>m2.charId;

SELECT "done with query ---- ";
```

Grading info: -2 pts if runtime is wrong

(c) **[8 pt]** *Index impact: Report* the wall-clock time of your query for the question 6 again with indices on the column `comicID` .
(follow the syntax shown here).

Solution: 0m1.357s

```
SELECT "Drop index on charID";
DROP INDEX IF EXISTS comic1;

SELECT "started building new index";

CREATE INDEX IF NOT EXISTS comic2
ON marvel (comicId);

SELECT "done with building index";

--Optional, if you want the explain query (START)
SELECT "   here is the query plan------";
EXPLAIN SELECT COUNT(*) FROM
 marvel m1
INNER JOIN
 marvel m2
ON m1.comicId = m2.comicId AND m1.charId>m2.charId;
SELECT "   end of query plan - starting query ------";
--Optional, if you want the explain query (END)

SELECT COUNT(*) FROM
 marvel m1
INNER JOIN
 marvel m2
ON m1.comicId = m2.comicId
WHERE m1.charId>m2.charId;
```

Grading info: -2 pts if runtime is wrong

(d) **[5 pt]** *Query optimization:* Use the `explain select` for the last three sub-questions, i.e., with, and without indices. Based on the `explain select` and time you measured, select fastest query for question 6.

i. No index
ii. Join with index on charId
iii. Join with index on comicId

Hint: See `http://www.sqlite.org/draft/eqp.html` for a (rough) description of the output of `explain`.

Solution: iii. Join with index on comicId
Grading info: -5 pts if they do not answer which method is fastest, -1 or -2 pts if they are partially correct

What to turn in:

- **<u>Code:</u>**
  Submit a tar file to blackboard in a file name `[andrew-id]-hw0.tar.gz` (without the data such as marvel.db or Marve-data/), with all the SQL queries and commands you need to answer the questions above. Make sure it runs: we will grade it using
  ```
  make all
  ```
  (organize your code files following the provided makefile-hw0 folder.)
- **<u>Answers:</u>**
  Submit hardcopy at classroom that contains **1) question number**, **2) answers**, and **3) SQL queries**. After finishing up, `make all` will provide you all answers except plot for question 5. For grading purpose, please group answers in pairs of (2,3), (4,5), and (6,7) respectively. `Make sure you put '[course-id] [hw#] [question#] [andrew-id] [your name]' on top of every page you submit.`