Carnegie Mellon University
Department of Computer Science
15-826 Multimedia Databases and Data Mining
C. Faloutsos, Spring 2016

Homework 1 - Solutions
Due: hard copy, in class, at 3:00pm, on 1/25/2016

VERY IMPORTANT:
• This homework is on prerequisite, elementary material. People who score 80 or lower, will be encouraged to drop the course or switch to audit.
• Deposit hard copy of your answers, in class. For ease of grading, please
  1. Separate your answers, on different page(s) for each question (staple additional pages, if needed).
  2. Type the full info on each page: your name, Andrew ID, course#, Homework#, Question# on each of the pages.

Reminders:
• Plagiarism: Homework is to be completed individually.
• Typeset all of your answers whenever possible. Illegible handwriting may get zero points, at the discretion of the graders.
• Late homeworks: in that case, please email it
  – to all TAs
  – with the subject line exactly 15-826 Homework Submission (HW 1)
  – and the count of slip-days you are using.

For your information:
• Graded out of 100 points; 3 questions total
• Rough time estimate: 2-6 hours

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-trees</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Linear Hashing</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Revision: 2016/02/02 18:15
Question 1: B-trees....................[15 points]

On separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your-name]’

Grading info: graded by Varshaa Naganathan

Consider the B-trees of order \(d=2\) \((2d+1 = 5 = \text{maximum fanout})\), as shown in Figure 1.
We want to delete key ‘31’.

![B-tree example](image)

(a) original
(b) your answer

Figure 1: B-tree example

(a) \([3 \text{ points}]\) Before the deletion, the B-tree has 3 nodes - how many will it have after the deletion of key ‘31’?

(a) \(3\)

(b) \([4 \text{ points}]\) Which keys (if any) will be in the same node as key ‘11’?

(b) key ‘18’

Grading info: 2/4 if 11,18 are mentioned

(c) \([8 \text{ points}]\) Draw the tree, after the deletion - use Figure [1](b); cross out (‘×’) any nodes that you don’t need.

Grading info: 5/8 if correct tree with multiple guesses(?)
Grading info: 2/8 if explanation attempted
Grading info: 6/8 if one key is off

Homework 1 continues...
**Question 2: Linear Hashing**

Consider a hash table that operates under linear hashing. When it started, the initial hashing function was \( h(x) = x \mod 4 \); the hash table had \( B = 4 \) buckets \((0,1,2,3)\), and the split pointer was \( s = 0 \). After some other splits, the table has \( N = 64 \) buckets (numbered \( 0,1,\ldots, 63 \))

(a) [3 points] Where is the split pointer \( s \)? Give the bucket number it points to (integer in the range \( 0 \ldots 63 \))

Solution: \( h(x) = x \mod 64 \)

Grading info: if partially correct \((h(x) = \text{mod} 64, h(x) = h(x) \mod 64 \text{ etc.})\), -1 pt

(b) [2 points] How many hashing functions are active?

Solution: \( 1 \)

Grading info: if they give one wrong cell, -4 pt

Grading info: if they give the correct pair of cells, but not which one will be the destination, then -2 pt

---

Homework 1 continues...
Question 3: SQL .................................................. [70 points]

On separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your-name]’

Grading info: graded by Wan-Shue Lai

For this part, we will use sqlite3 (version 3.7.17), which is available on the andrew unix machines (ssh unix.andrew.cmu.edu).

Set up

1. Download the SQL database from
   http://www.cs.cmu.edu/~christos/courses/826-resources/DATA-SETS-HOMWORKS/
   oscars-data/oscars.db
2. and operate on it with the command
   sqlite3 oscars.db
   which should bring you the sqlite> prompt.

Data description: The oscars.db database has 3 tables, with information about movie academy awards ('Oscar'), including both nominations as well as wins. The tables and the meaning of their attributes are as follows:

- **people** (personID, firstName, lastName). The personID is a unique identifier for each actor/actress; the rest are self-explanatory.
- **nominations** ( nominationId, year, catId, personId, title, characterName, won) Each row corresponds to a nomination of an actor/actress (personID), for a specific movie (title), for a given year.
  - nominationId is a unique identifier (1, 2, ...).
  - won is a boolean ‘t’/’f’, depending on whether the nominated person won the award or not.
  - characterName is the name of the character in the movie.
- **categories** ( catId, category) . This table has only 4 rows: ’1’ for ‘Leading Actor’, ’2’ for ‘Supporting Actor’, etc.

For example the 3rd entry in nominations is shown in Table 1

<table>
<thead>
<tr>
<th>nominationId</th>
<th>year</th>
<th>catId</th>
<th>personId</th>
<th>title</th>
<th>characterName</th>
<th>won</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2010</td>
<td>1</td>
<td>3</td>
<td>The Social Network</td>
<td>Mark Zuckerberg</td>
<td>f</td>
</tr>
</tbody>
</table>

Table 1: 3rd entry of the nominations table

It is the result of the query select * from nominations where nominationId=3, and it corresponds to the 2010 nomination of actor Jesse Eisenberg (with personID =3), for the movie 'The Social Network', for leading actor (catId=1); he played the character 'Mark Zuckerberg', and he did not win ('won'='f').

Question 3 continues...
Queries, and what to hand in: For all the queries below, hand-in both the SQL code of your answer, as well as the output of your code. Use \texttt{headers on} for prettier output.

(a) [20 points] Super-stars: List all the people that have been nominated 9 or more times. Specifically, give the \texttt{personId}, first and last name, and count of nominations. Order most-nominations-first, and break ties by last name (ascending), and then by first name (also ascending).

\textit{(FYI - Relationship to data mining: Grouping, sorting, and spotting of 'heavy hitters' are powerful, for data mining tasks like information summarization, and anomaly detection.)}

\textbf{Solution:}

\begin{verbatim}
select P.personId, P.firstName, P.lastName, 
      count(*) as numNominations
from nominations as N, people as P
where N.personId = P.personId
group by P.personId
having numNominations > 8
order by numNominations desc, P.lastName, P.firstName;
\end{verbatim}

\begin{tabular}{lcll}
  personId & firstName & lastName  & numNominations \\
  -------- & --------- & ---------- & --------------
  450      & Meryl     & Streep     & 16             \\
  534      & Katharine & Hepburn    & 12             \\
  37       & Jack      & Nicholson  & 12             \\
  578      & Bette     & Davis      & 11             \\
  62       & Paul      & Newman     & 9              \\
  104      & Laurence  & Olivier    & 9              \\
  129      & Spencer   & Tracy      & 9              \\
\end{tabular}

(b) [25 points] Competitors of 'Jesse Eisenberg': Find the people who competed against him, that is, they got nominated in the same year(s) and the same categorie(s) that he got nominated. For each such actor, print (\texttt{lastName}, \texttt{firstName}, \texttt{title}, \texttt{year}, \texttt{catId}), and sort by \texttt{lastName} (ascending), and then by \texttt{firstName}.

\textit{(FYI - Relationship to data mining: Such queries are useful in link prediction and product recommendation, like, say Amazon: 'people who bought product-X, tend to also buy product-Y'.)}

\textbf{Solution:}

\begin{verbatim}
select P2.lastName, P2.firstName ,
      N2.title, N2.year, N2.catId
from nominations as N1, people as P1, nominations as N2, people as P2
where N1.personId = P1.personId
  and N2.personId = P2.personId
  and N1.year = N2.year
  and N1.catId = N2.catId
  and P1.firstName = 'Jesse'
  and P2.firstName != 'Jesse'
  and N1.catId = 'category'
  and N2.catId = 'category'
order by N1.lastName, N1.firstName;
\end{verbatim}

Question 3 continues...
from nominations N1, nominations N2, people P1, people P2
where N1.catId = N2.catId
   and N1.year = N2.year
   and N1.personId = P1.personId
   and N2.personId = P2.personId
   and P1.firstName = 'Jesse'
   and P1.lastName = 'Eisenberg'
   and P1.personId <> P2.personId
order by P2.lastName ;

Solution:

<table>
<thead>
<tr>
<th>lastName</th>
<th>firstName</th>
<th>title</th>
<th>year</th>
<th>catId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bardem</td>
<td>Javier</td>
<td>Biutiful</td>
<td>2010</td>
<td>1</td>
</tr>
<tr>
<td>Bridges</td>
<td>Jeff</td>
<td>True Grit</td>
<td>2010</td>
<td>1</td>
</tr>
<tr>
<td>Firth</td>
<td>Colin</td>
<td>The King’s</td>
<td>2010</td>
<td>1</td>
</tr>
<tr>
<td>Franco</td>
<td>James</td>
<td>127 Hours</td>
<td>2010</td>
<td>1</td>
</tr>
</tbody>
</table>

(c) **[25 points]** Duplicate detection: Usually, there is only one winner for each year, and each category. Are there violations to that conjecture? List the exceptions, if any - for each exception, print year, catId and the count of winners (which should be > 1).

*(FYI - Relationship to data mining: Spotting exceptions and rule-violations are typical tasks of data cleaning, which is usually the most time consuming step of data mining.)*

Solution:

```sql
select N.year, N.catId, count(*) as winnerCount
from nominations N
where N.won = 't'
group by N.year, N.catId
having winnerCount >1;
```

Solution:

<table>
<thead>
<tr>
<th>year</th>
<th>catId</th>
<th>winnerCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1931</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1968</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

End of Homework 1