Databases and Data Mining — Assignment 5

Rick van der Zwet

January 31, 2010

“1a. *Propose an algorithm, in pseudo-code, for the following: The automatic generation of a concept hierarchy for numerical data based on the equal-width partitioning rule.*”

```
input = [] # Input array of all input numbers
num_intervals = %User input of number of intervals needed%
max = maximum(input)
min = minimum(input)
interval_width = (max - min) / num_intervals

output = [] # Output array, where the values of interval k
            # is stored in value output[k]

for value in input:
  interval = value / interval_width # Find it's correct bin
  output[interval].append(value)    # Put the value inside the bin
endfor
```

“1b. *Propose an algorithm, in pseudo-code, for the following: The automatic generation of a concept hierarchy for numerical data based on the equal-frequency partitioning rule.*”

```
input = [] # Input array of all input numbers
num_intervals = %User input of number of intervals needed%
input_length = length(input) # Number of items in list
interval_width = input_length / num_intervals

sorted_input = sorted(input) # Sort items on value from small to large

output = [] # Output array, where the values of interval k
            # is stored in value output[k]

interval = 0
counter = 0
for value in sorted_input:
  output[interval].append(value)    # Put the value inside the bin

  # If the width has been 'filled' continue with the next one
  counter++
  if counter > interval_width:
    interval++
    counter = 0
  endif
endfor
```

“2. *Suppose that a data warehouse consists of four dimensions, date, spectator, location, and game, and two measures, count, and charge,*

> *where charge is the fare that a spectator pays when watching a game on a given date. Spectators may be students, adults, or seniors, with each category having its own charge rate.*"

> "2a. *Draw a star schema diagram for the data warehouse.*"

See figure 1.

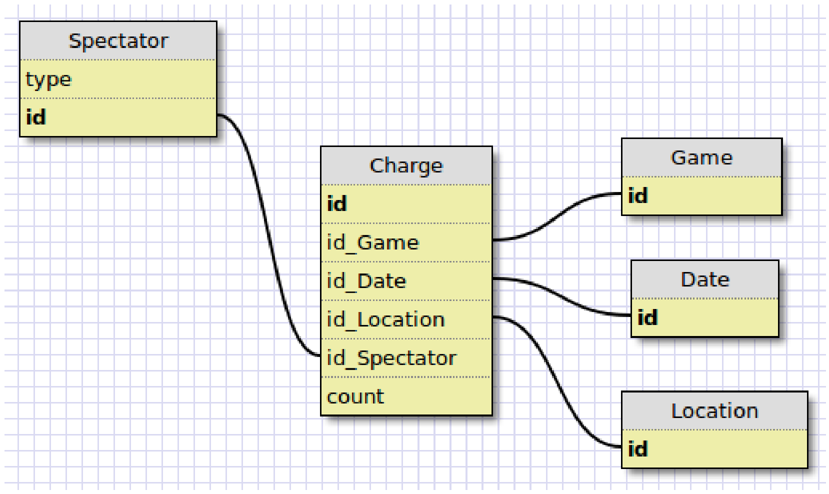

Figure 1: Star schema diagram 2a data warehose

> "2b. *Starting with the base cuboid [date, spectator, location, game], what specific OLAP operations should one perform in order to list the total charge paid by student spectators at GM_Place 2004?*"

You first will need to *slice* on the condition `game == 'GM_Place'`. Secondly you will need to slice on `date.year == '2004'`. This will give you all the charges for GM Place in 2004. Next we slice to `spectator.type == 'student'`. Lastly we sum all the charges in the display phase (`pivot`).

> "2c. *Bitmap indexing is useful in data warehousing. Taking this cube as an example, briefly discuss advantages and problems of using a bitmap index structure.*"

Bitmap indexing in this case is useful to have a compact representation of for example the spectator type. As this can only be 4 options a four bit representation fits the need to store all possible combinations. This required binary operators to do searches in this set are quick, but their results will need to be processed before beeing able to represent it. One other advantage is the fact that the bitmap indexing compresses really well as patterns are reoccuring.

> “3. *A data cube, C, has n dimensions, and each dimension has exactly p distinct values in the base cuboid. Assume that there are no concept hierarchies associated with the dimensions. What is the maximum number of cells possible (including both base cells and aggregate cells) in the data cube, C?*”

Take for example 2 dimensions with each 4 distinct values in the base cuboid, this will a total of $4^2 = 16$ possibilities. Taking this more general will this be p^n.

> “4. *The Apriori algorithm uses prior knowledge of subset support properties.*”

> “4a. *Prove that all nonempty subsets of a frequent itemset must also be frequent.*”

If an itemset I is frequent it means it's occurence classifies a minimal support level, hence if you take a subset of I (I_{sub}). The count will remain the same -or larger if the subset is also part of an other itemset J, which has distint matches from I- as subset I. So I_{sub} will also be frequent if I is frequent.

> “4b. *Given frequent itemset l and subset s of l, prove that the confidence of the rule $s => (l - s)$ cannot be more than the confidence of the rule $s => (l - s)$ where s is a subset of s.*”

If s' has an higher confidence than s, than a smaller itemset have a higher conditional propability of also containing a larger itemset. If the repeat this assumption, you will end up on the left side with a empty set and on the right the full itemset l, which then would have the highest confidence. While this would not be valid. As rules like 'Customer buying nothing', will also buy 'beer' can not be existing.

> “5. *An optimization in frequent item set mining is mining closed patterns, or mining max patterns instead. Describe the main differences of mining closed patterns and mining max patterns.*”

By definition the itemset X is closed means X is frequent and there exists no super-pattern X/inY wth the same support as X. While X is max if X is frequent and there exists no *frequent* super-pattern X/inY. Maximum frequent itemsets has the downside that you don't not know the actual support of those sub-itemsets. The closed frequent itemsets helps purging a lot of itemsets that are not needed.

> “6. *The price of each item in a store is nonnegative. For each of the following cases, identify the kinds of constraint they represent (e.g. anti-monotonic, monotonic, succinct) and briefly discuss how to mine such association rules efficiently:*”

> “6a. *Containing one free item and other items the sum of whose prices is at least $190.*”

This is *monotonic*, as adding items will never lower the sum of prices. Mining this will be bestly done by first finding the free items and next order the other items based on price, starting with the most expensive ones first. The soon you get to $190 can al it's supersets to matching as well, next you remove the last item and try to match up again. Continue this till you removed the most expensive item and tried to match up again. Next *JOIN* with all the free items and the list is complete.

> “6b. *Where the average price of all the items is between $120 and $520.*”

This is convertible *anti-monotonic* and convertible *monotonic* if you look at one contrain. *anti-monotonic* If itemset S violates the sub-constraint $avg(S.Price) \leq$ $520. so does every itemset with S in prefix item value descending order. *monotonic* If itemset S constraint $avg(S.Price) \geq$ $120 so does every itemset having S as prefix item values descending order.

Satifing both conditions how-ever does make it neither *anti-monotonic* nor *monotonic*. A fast way to generate this set is to use the algoritm used in 6a but modify the number constraint on-the-fly. Like average of 3 items between a centrain range $(120 - 520)$ It the same of checking whether the sum of 3 items is between $(3 * 120 - 3 * 520)$.

> “7. *Suppose a city has installed hundreds of surveillance cameras at strategic locations in busy streets. All the captured video (each stream being 640 pixels x 480 pixels, 24- bits (RGB) per pixel, 25fps) is streamed to a central observation post in real time. Describe an efficient system that would allow real time data mining and continuously querying these video streams for abnormal events, such as explosions, people in panic, etc.. Also discuss the computational costs and the memory requirements of your system.*”

Every camera generates $640 * 480 * 24bits * 25fps = 180.000kbit/sec \approx 21.9MByte/sec$. I would first use a compression algoritm to send-the-data over the wire by sending the changes only and not the full frame every time. Secondly applying abnomality detections using heuristics. Keeping the detected abnormalities available un-compressed and uncut and save all other data in a much lower resolution (1fps), with a compression ratio of 4:1 this would make storage stream of approx 19GB/camera/day. Processing power required depends on the algoritms used, but a 500Mhz CPU/camera should fit the bill when simple algoritms are used.

> “8. *A flight data warehouse for a travel agent consists of six dimensions: traveler, departure (city), departure_time, arrival (city), arrival_time, and flight; and two measures count, and avg_fare, where*

avg_fare stores the concrete fare at the lowest level but the average fare at the other levels. Suppose the cube is fully materialized. Starting with the base cuboid [traveler, departure, departure_time, arrival, arrival_time, flight], what specific OLAP operations (e.g roll-up flight to airline, etc.) should one perform in order to list the average fare per month for each business traveler who flies American Airlines (AA) from Los Angeles (LA) in the year 2007?"

Note bit tricky as we use the departure_time to determine the year and arrival_time to determine the month of the flight. When these values are not set to the same day we might run into inconsistencies.

```
* Roll-up departure\_time to 'Year'
* Roll-up flight to 'Airline'
* Dice departure = 'LA' and departure_time = '2007' and flight = 'American Airlines'
* Roll-up traveler to 'Consumer/Business'
* Roll-up arrival_time to 'Month'
* Slice traveler = 'Business'
```

"9. *In graph mining what would be the advantage of the described apriori-based approach over the pattern growth based approach (see lecture slides) and vice versa.*"

The advantage comes in the simplicity of the algotitm when it get to computations needed and it's setup, but this does not make it effient.

On the other hand apriori uses the costly process of candidate generation and testing, making it more 'heavy' then pattern grow based. and also avoid costly database scans.