University of Zurich[UZH]

Department of Informatics

University of Zürich
Department of Informatics
Binzmühlestr. 14
CH-8050 Zürich
Phone. +41 44 635 43 11
Fax +41 44 635 68 09
www.ifi.uzh.ch/dbtg

UZH, Dept. of Informatics, Binzmühlestr. 14, CH-8050 Zürich

Francesco Luminati
Samuele Zoppi

Prof. Dr. Michael Böhlen
Professor

Phone +41 44 635 43 33
Fax +41 44 635 68 09
boehlen@ifi.uzh.ch

Zürich, July 4, 2012

Master Project in Informatik
Datenbanktechnologie

Topic: Implementation of a 1NN-Join technique

Overview:

The Swiss Feed Data Warehouse stores lab-analysis of animal feeds coming from various places in Switzerland. Each lab-analysis is identified by a sample number and consists of a set of measurements. A measurement is represented by a fact in the fact table of the data warehouse.

A *derived fact* is a fact that is computed from other facts. An example of a derived fact is the computation of the energy content BE. The BE content is calculated by the formula $BE = TS\ (OS + RP)$ where TS is the dried matter of a sample, OS is its organic matter and RP its raw proteins content. The goal of this Master Project is to design, define, implement, and evaluate an efficient technique to compute derived facts.

Since the data is sparse, a nearest neighbor join must be used to compute derived facts. For example, if we want to calculate the value of BE for a given sample usually not all required nutrients are available; i.e. for sample 11 the value of OS is missing. Even when considering all samples of a given timestamp, an entry for TS, for OS and for RP often is missing. For example, no value of TS is available for May 2nd. This is the reason why a nearest neighbor join must be used to substitute missing data with the temporal closest information.

Detailed description:

Assume a data warehouse $DW = (F, D_1, ..., D_n)$ with fact table F and dimension tables $D_1, ..., D_n$. An *adaptive nearest neighbor join* $nnj(q, X, k)$ computes the derived facts for $q(F')$. q defines the tuples from $F' = \pi_{s,g,t,k,m}(F \bowtie D_1 \bowtie ... \bowtie D_n)$ that must be considered

for the nearest neighbor join, X is the derived fact that shall be added, and k' is the expression defining X:

$R^{+X} = nnj(R, X, k')$ where $R = q(F')$ and $F' = \pi_{s,g,t,k,m}(F \bowtie D_1 \bowtie ... \bowtie D_n)$

F'

s	g	t	k	m
11	Hay	2011-01-12	TS	0.9
11	Hay	2011-01-12	RP	3.2
22	Stroh	2011-01-15	OS	3.3
33	Hay	2011-04-25	OS	3.8
44	Corn	2011-04-27	OS	4.13
55	Hay	2011-05-02	OS	6.8
66	Hay	2011-05-02	RP	3.6
77	Stroh	2011-06-12	RP	0.5
77	Stroh	2011-06-12	Na	0.08

R

s	g	t	k	m
11	Hay	2011-01-12	TS	0.9
11	Hay	2011-01-12	RP	3.2
22	Stroh	2011-01-15	OS	3.3
33	Hay	2011-04-25	OS	3.8
55	Hay	2011-05-02	OS	6.8
66	Hay	2011-05-02	RP	3.6
77	Stroh	2011-06-12	RP	0.5

R^{+BE}

t	k	m
2011-01-12	BE	5.85
2011-01-15	BE	5.85
2011-04-25	BE	6.66
2011-05-02	BE	9.36
2011-06-12	BE	5.84

The nnj algorithm works as follows:

1. Input:
   $R = \sigma_{g \in \{'Hay','Stroh'\} \wedge k \in \{'TS','OS','RP'\}}(F')$
   $X = BE$
   $k' = TS * (OS + RP)$

   Output:
   $R^{+X} = nnj(R, X, k')$
2. For each nutrient k_i in k', a cursor c_i should be declared that ranges over $\sigma_{k=k_i}(R)$:

   $\sigma_{k='TS'}(R)$

s	g	t	k	m
11	Hay	2011-01-12	TS	0.9

   $\sigma_{k='RP'}(R)$

s	g	t	k	m
11	Hay	2011-01-12	RP	3.2
66	Hay	2011-05-02	RP	3.6
77	Stroh	2011-06-12	RP	0.5

   $\sigma_{k='OS'}(R)$

s	g	t	k	m
22	Stroh	2011-01-15	OS	3.3
33	Hay	2011-04-25	OS	3.8
55	Hay	2011-05-02	OS	6.8

3. Declare a cursor c_0 that scans the records of R in chronological order.
4. All the other cursors positions will be incremented, if needed, until all the c_i will point to the entry which is temporal closest to c_0.
5. An estimation of the formula must be calculated before incrementing the position of c_0. This is represented in Figure 1 by the rows of R^{+BE}.

The Algorithm will be implemented using procedural SQL and should be called as follows:

```
SELECT *
FROM nnj(R̄, BE, TS * (OS + RP));
```

where $\bar{R}$ is the SQL query defining R.

Derived facts cannot be precomputed since their value changes along with q. Here follows an example of R' showing that $R \neq R' \iff nnj(R, X, k) = nnj(R', X, k)$:

$$R' = \sigma_{g \in \{'Hay'\} \wedge k \in \{'TS','OS','RP'\}}(F')$$

As you can see, different values of BE are computed for dates 2011-01-12 and 2011-05-02.

F'

s	g	t	k	m
11	Hay	2011-01-12	TS	0.9
11	Hay	2011-01-12	RP	3.2
22	Stroh	2011-01-15	OS	3.3
33	Hay	2011-04-25	OS	3.8
44	Corn	2011-04-27	OS	4.13
55	Hay	2011-05-02	OS	6.8
66	Hay	2011-05-02	RP	3.6
77	Stroh	2011-06-12	RP	0.5
77	Stroh	2011-06-12	Na	0.08

R

s	g	t	k	m
11	Hay	2011-01-12	TS	0.9
11	Hay	2011-01-12	RP	3.2
33	Hay	2011-04-25	OS	3.8
55	Hay	2011-05-02	OS	6.8
66	Hay	2011-05-02	RP	3.6

R^{+BE}

t	k	m
2011-01-12	BE	6.30
2011-04-25	BE	6.66
2011-05-02	BE	9.36

Tasks:

1. Implement the nnj algorithm. Precisely describe your algorithm using pseudo code.
2. Incorporate your solution into the online application of the Feed Data Warehouse. Users shall have the possibility to seamlessly use derived nutrients along with stored nutrients. Derived nutrients shall be displayed in temporal charts, the temporal result table, and statistical overviews.
3. Prove the correctness of your solution analytically and empirically.
4. Evaluate the efficiency of your approach analytically and empirically. Design query optimization techniques to optimize the evaluation of multiple derived nutrients.
5. Compare your approach with the standard sql implementation of NNJ [2] and the optimal sql implementation of NNJ [1]. How are performances of the three approaches related to the dataset size? How are performances of the three approaches related to the number of tables to be joined?
6. In your report define the problem and your solution precisely and design a representative running example to illustrate your approach.
7. Present progress and plans once every two weeks to your supervisor.

Optional tasks:

1. Extend the above algorithm for computing NNJ basing on dimensional attributes *time, altitude, canton*.
2. Minimize the fact table data to be considered in computing a derived fact. If there is a threshold ε of error we are disposed to pay, can we reduce the amount of data to be loaded? If we want to compute a daily/weekly/monthly derived fact, how much and which data can we ignore guaranteeing an error smaller or equal to ε?

Literature:

[1] K Nearest Neighbor Queries and KNN-Joins in Large Relational Databases (Almost) for Free. Bin Yao et al. ICDE 2010

[2] The k-Nearest Neighbor Join: Turbo Charging the KDD Process. Christian Böhm, et al. Journal Knowledge and Information Systems, 2003

Supervisor: Francesco Cafagna

Starting date: 2nd July 2012

Ending date: 31th October 2012

Department of Informatics, University of Zurich

Prof. Dr. Michael Böhlen