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

Students' Group

Prof. Dr. Michael Böhlen
Professor
Phone +41 44 635 43 33
Fax +41 44 635 68 09
boehlen@ifi.uzh.ch

Zürich, June 27, 2012

Master Project in Informatik
Datenbanktechnologie

Topic: Implementation of a 1NN-Join technique

The Swiss Feed Data Warehouse stores lab-analysis about animal feeds coming from various places in Switzerland. Each lab-analysis is identified by a sample number and consists of a set of measurements. We will refer to each of those measurements as a 'fact'. Nearest neighbor join is used in the Feed Data Warehouse to compute *derived* facts. A derived fact is a fact generated from existing measures using mathematical operations, data transformation, data aggregations, etc. Examples include averages, sums, percentages, and generally any mathematical expression. We are looking for a method to efficiently computing derived facts.

Consider, as a derived fact, the computation of an energy content called BE. Each BE row represents a derived fact. This is calculated by the formula $BE = TS\ (OS + RP)$ where TS is the dried matter of a given sample, OS is its organic matter and RP its raw proteins content. As you can see in Figure 1(d), if we would like to calculate the value of BE, for a given sample not all the needed information are available; i.e. for sample 1111 a value of OS is missing. Furthermore, even considering all the samples belonging to a given timestamp, an entry for TS, for OS and for RP may still be missing. For example, as you can see in the picture, no value of TS has been collected on May 2nd. This is the reason why Nearest Neighbor Joins is used, in order to provide a temporal estimation of a derived fact substituting missing data with their temporal closest information available.

id	g	t	k	m
0	Hay	2011-01-12	TS	0.9

(a) TS

id	g	t	k	m
1	Hay	2011-01-12	RP	3.2
8	Hay	2011-05-02	RP	3.8
9	Stroh	2011-06-12	RP	6.8

(b) RP

id	g	t	k	m
3	Stroh	2011-02-15	OS	0.5
4	Hay	2011-04-25	OS	1.3
7	Hay	2011-05-02	OS	0.7

(c) OS

sample	g	t	k	m
1111	Hay	2011-01-12	TS	8.70
1111	Hay	2011-01-12	RP	1.20
	Hay	*2011-01-12*	*BE*	*90.4*
2222	Stroh	2011-02-15	OS	9.20
3333	Hay	2011-04-25	OS	9.00
	Hay	*2011-04-25*	*BE*	*87.9*
4444	Corn	2011-04-27	OS	4.13
5555	Hay	2011-05-02	OS	8.90
6666	Hay	2011-05-02	RP	1.10
	Hay	*2011-05-02*	*BE*	*87*
7777	Stroh	2011-06-12	RP	0.50
7777	Stroh	2011-06-12	Na	0.08

(d) $BE = TS\ (OS + RP)$

Figure 1: Energy content calculation process

Derived facts must be computed on the fly because their value changes at each query. Depending on user selections, the same derived fact will be computed on a different partition of the dataset, and a different result value will be returned. Because of this, and due to the large size of the data, a fast and scalable solution to compute Nearest Neighbor Joins must be developed.

The algorithm should:

1. Take as input parameters a table R and a formula k'
   - In general, R is the result of a query selecting entries for just the nutrients to which k' refers and specifying a set θ of predicates. Each of those predicates may refer to any of the attributes of the schema of the Feed Data Warehouse.

     $R \leftarrow \sigma_{\theta \wedge k \in \{k_i,k_j,...,k_z\}}(FT \bowtie D_1 \bowtie ... \bowtie D_n)$

     $k' = k_i \otimes k_j ... \otimes k_z$, with $\otimes$ being any arithmetical operator.

     $D_1, ..., D_n$ are the Dimension Tables of the Swiss Feed Data Warehouse. They contain geographical, temporal and information about feeds, nutrients and samples. The Fact Table FT will store the measurements and, for each dimension, the foreign key pointing to the dimensional data related to the measurement itself.
   - Referring to Figure 1(d):

     $R \leftarrow \pi_{sample,g,t,k,m}(\sigma_{g \in \{'Hay','Stroh'\} \wedge k \in \{'TS','OS','RP'\}}(FT \bowtie D_1 \bowtie ... \bowtie D_n))$

     $k' = TS * (OS + RP)$

     Rows in gray will not be part of R because they do not satisfy the selection predicates.
2. $\forall k_i \in \Delta(k')$ a cursor c_i should be declared over $\sigma_{k=k_i}(R)$
   - Referring respectively to Figures 1(a), 1(b), and 1(c), the cursor c_1 is declared over the entries of $\sigma_{k='TS'}(R)$, c_2 to $\sigma_{k='RP'}(R)$, and c_3 to $\sigma_{k='OS'}(R)$
3. A cursor c_0 will also be declared and will fetch one by one the records of R, Figure 1(d).
4. R will be read sequentially incrementing the position of c_0 by 1 at each step of the algorithm.

5. All the other cursors positions will be incremented, if needed, until when all the c_i will point to the entry which is temporal closest to c_0 and will hold, at the same time, the condition $c_i.g = c_0.g$.
6. An estimation of the formula must be calculated before incrementing the position of c_0. This is represented in Figure 1(d) by the rows in italic.

The Algorithm will be implemented using procedural SQL and should be called from the outside by the query:

`SELECT * FROM your_function(`$\bar{R}$, k'`);`

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

Tasks:

1. The students' group will implement during the first month the above illustrated algorithm. The algorithm design should be precisely and in details described, using flowcharts or pseudocode, and will be inserted in the report of point 3.
2. The implemented solution should be able to show, referring to Figure 1(d):
   - only rows in italic (derived facts)
   - all rows (facts + derived facts)

   The second option has to be used to test the correctness of the result, i.e. to check that no closer neighbor could be used in a derived fact computation.
3. The students' group will incorporate their solution inside an Online Application that access the Feed Data Warehouse. A formulas table will be provided to assess the quality of the solution; a good solution should work for all the formulas with up to 10 components.
   Those formulas will be included in the online application:
   - a new tab, allowing the user to select derived nutrients, should appear
   - the temporal charts should display also derived nutrients' values
   - the result data table will include a row for each timestamp of the temporal estimation of a derived nutrient
   - the statistics table should contain entries also for derived nutrients

   The work should be ready at the end of the third month.
4. A report should be incrementally produced during each of the previous phases, be completed at end of the forth month and, in case, integrated with the solution to one of the later proposed optional task, during the fifth month.
   - Given two relations $R(\mathcal{R})$ and $S(\mathcal{S})$ and an attribute t with $t \subseteq \mathcal{R}$ and $t \subseteq \mathcal{S}$, formally define:
     - derived facts, as functions on n fact table entries
     - the distance $d(r, s)$ between tuples r and s basing on t
     - the nearest neighbor join result $R\ {}_{\mathbf{G}}\ltimes_d S$ between R and S with d as distance function and **G** as grouping attribute set.

- the extending process of the online application should be described
- the runtime complexity and space requirements of the solution should be calculated and justified
- best and worst cases of the algorithm should be described
- the report should also contain a running example that illustrate functionalities, details and peculiarities of the implemented approach.
- an evaluation of the algorithm should be accurately written. How does the approach compare with the standard sql implementation of NNJ [2]? How does the approach compare with the optimal sql implementation of NNJ [1]?

5. A meeting every 10 days will be held with the supervisor

Optional tasks:

1. Extend the above algorithm for computing NNJ basing on dimensional attributes *time, altitude, canton*.
   Given two relations $R(\mathcal{R})$ and $S(\mathcal{S})$and a set of attributes $\mathcal{A}$ with $\mathcal{A} \subseteq \mathcal{R}$ and $\mathcal{A} \subseteq \mathcal{S}$, define:
   - The distance $d_{\mathcal{A}}(r, s)$ between two tuples $r \in R$ and $s \in S$ basing on $\mathcal{A}$

   Does the definition of $R \,{}_{\mathbf{G}}\ltimes_{d}\, S$ change? If yes, formally define its result.
   Can we find a generalized solution for computing NNJ without knowing the size of $\mathcal{A}$? How related performances are to the number n of dimensions considered?
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