Improving the Completeness and Timeliness by Horizontal Fragmentation of Data Warehouse Tables

Ljiljana Brkić, Mirta Baranović, Igor Mekterović
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, HR-10000 Zagreb
CROATIA
{Ljiljana.Brkic, Mirta.Baranovic, Igor.Mekterovic}@fer.hr

Abstract: - In this paper we discuss the procedure for horizontal fragmentation of data warehouse tables. This procedure is suitable when there are multiple independent organizational units that are producing data of the same structure that is eventually loaded into the consolidated fact table. In such cases, it is important for one organizational unit's errors not to influence the others. We show that horizontal fragmentation improves data warehouse's data quality, namely, completeness and timeliness data quality dimensions. The proposed procedure relies on metadata and is generic and applicable to any data warehouse.

Key-Words: - Data warehouse, Completeness, Timeliness, ETL, Horizontal fragmentation

1. Introduction

It is common to describe and evaluate the quality of the data delivered to users in terms of quality attributes [2]. These attributes are called the data quality dimensions. Each dimension covers a specific aspect of quality. Data quality dimensions may relate to the value of the data or its purpose. Both, the data value quality and data schemes quality affect the quality of business processes. E.g., unnormalized schema of a table in relational data model results in redundancy and anomalies when inserting, updating and deleting data.

Data quality dimensions are usually defined from the qualitative aspect, referring to the general properties of the data values and data schemas without prescribing the way in which the quantitative values are assigned to dimensions. Despite the recognized importance of the quality of data schemes (both conceptual and logical), today, the focus is on the quality of data values [3]. For quantitative evaluation of quality dimensions it is necessary to define metrics and measurement methods. The importance of this issue has been recognized and number of scientific papers (some of which are [2] [4] [5] [6]) deal with methods of measuring and quantifying the dimensions of data quality. There is no common position or accepted standard with regards to data quality dimensions classification or even data quality dimensions' definition. Thus, often for essentially the same data quality definition different terms are used, and vice versa, the same term has a different meaning in works by different authors.

Data quality in a data warehouse is affected not only by the quality of source data, but also by the quality of ETL process. For example, it is possible for an accurate and complete record from a data source to not be transferred to the data warehouse (or not timely transferred) due to faulty internal logic, inappropriate data warehouse refreshment policy or just plain errors in the ETL process. In other words, the data warehouse data quality dimensions depend on the quality of ETL processes.

The process of determining the quality of a data warehouse starts by assessing the quality of an individual attribute. The quality of a tuple is calculated by aggregating the quality of its attribute values. Analogously, the quality of table is determined via its tuples and, ultimately, with that information, the quality of the entire data warehouse is calculated.

In this paper we focus on two data quality dimensions: completeness and timeliness. In order to achieve the improvement of those data quality dimensions, we propose and describe modifications to the data warehouse's ETL process.

2. Horizontal fragmentation of the data warehouse tables

Data warehouses often integrate data of the same structure (schema) from several departments of a company, or a number of companies within a corporation, or a number of state institutions of a country, etc. The quality of the data belonging to a particular department, company or institution should primarily be a reflection of the effort that was put into the process. It is not unusual for one department to be more devoted to proper data administration than the other. Errors in the semantic integrity of the data belonging to one or several organizational units should not affect the others. The ETL process should be modelled and implemented with this in mind.

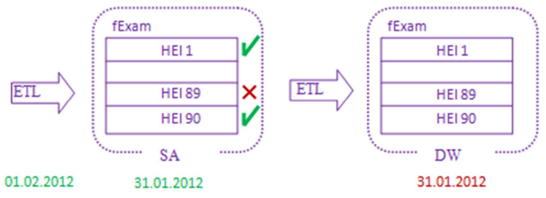

Figure 1. Errors in one fragment stop data of fragments that contain no error data

For instance, Figure 1 depicts update of fExam fact table of the data warehouse integrating data from 90 higher educational institutions (HEI 1 – HEI 90) in the Republic of Croatia [1]. In this project, we had to employ an all-or-nothing refreshment policy; since it was of the upmost importance to have complete data (e.g. data warehouse is used to produce scholarship listings). Suppose that during the ETL process some integrity errors (e.g. foreign key violation) have been detected in records belonging to higher education institution HEI 89. Due to the detected errors, the content of the fact table fExam has not been updated but was kept at the state from the previous update cycle (31.01.2012). Although there were no errors in the other institution's data on 01.02.2012, errors in HEI 89's data caused other institutions' correct data not to be transferred. In other words, an institution is affected by the poor data quality belonging to other institution.

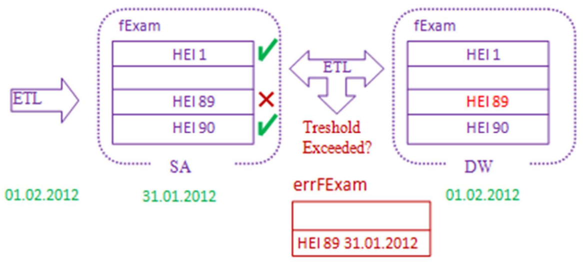

Figure 2. Errors in one fragment don't stop data of fragments that contain no errors

Figure 2 illustrates a better solution employing the following improvements:

- *fExam* table is *logically* divided to horizontal fragments, each fragment belonging to a particular higher education institution
- ETL process is modified as to be capable to find error free fragments in the staging area, and accordingly refresh corresponding *fExam* fragments. Fragments that exceed the error threshold (maximal allowed number of incorrect tuples) are not updated, i.e. remain as they were after on the last successful refreshment cycle.

This is much more robust and customizable approach. Each higher education institution is allowed to define the number of errors that it is willing to tolerate. If the defined threshold for a particular institution is exceeded, the associated fragment of the fact table will not be updated. That fragment retains the state in which the number of errors was within the predefined threshold. Such strategy can lead to the state in which the different fragments of fact and dimension tables are updated in different update cycles. Tables become unions of approved (above threshold) fragments, with each fragment having its own timestamp. Diligent institutions (departments, etc.) are no more affected by others and are more likely to have up-to-date data in the data warehouse. On the other hand, if one wants to ignore errors in favour of timeliness, one should only set high enough threshold. Note that by setting the threshold to infinity, errors are simply ignored and DW tables are always updated with newly arrived data.

3. Metadata for the horizontal fragmentation process

In order to be generic and easy to implement in the existing systems, horizontal fragmentation procedure is designed and implemented with the help of the metadata. Metadata is stored in the extended data dictionary and used by generic ETL procedures. This way, to implement horizontal fragmentation, it is only required to populate metadata tables and to insert ETL procedure calls at the appropriate point in the existing ETL process.

Figure 3 shows the relational model of the metadata. Certain information must be entered manually (e.g., whether a table is a fact table or a dimension table - metaTableType and metaTable.tableTypeID).

However, most of the information is already available in the DBMS data dictionary: tables, attributes, primary keys, foreign keys and indexes. This information can be copied automatically (different scripts have to be written for various DBMSs) to our metadata tables (*metaTable*, *metaColumn*, *metaPKColumns*, *metaFK*, *metaFKColumns*, *metaIDX*, *metaIDXColumns*). This information is used by the procedures (that generate and execute dynamic SQL) to drop and recreate integrity rules (foreign and primary keys) and indexes before and after data loading, and thus speed things up.

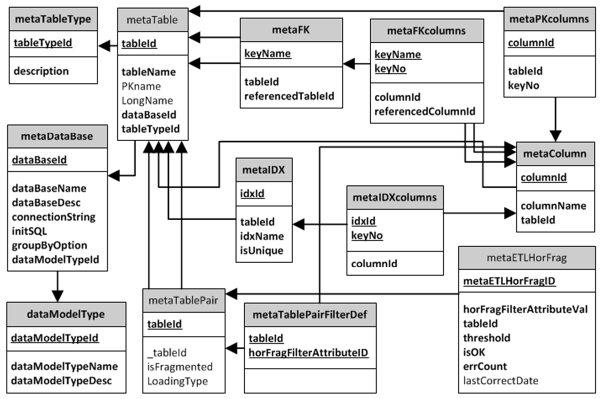

Figure 3 Metadata for the horizontal fragmentation process

The *metaDataBase* table is used for storing the data describing sub-systems of data warehousing system (source databases, staging area and data warehouse). For each fact and dimension table we introduce a matching “prime table” [1]. Prime tables form a parallel set of tables to the production tables having the same structure and analogous constraints; e.g. if a fact table F(a, b, c) references dimension table D(b, d, e) via attribute b, then F'(a, b, c) references D'(b, d, e) via attribute b. In short, prime tables are used to prepare and test data, and when ready, prime table data is quickly copied to the production tables. Data describing relations between each data warehouse table (*metaTablePair.tableId*) with its prime table (*metaTablePair._tableId*) is stored in the *metaTablePair* table. For such pair, fragmentation attribute (or multiple attributes) is defined (*metaTablePairFilterDef*). *HorFragFilterAttribute* is usually the id of the department, institution, etc. The *metaETLHorFrag* table holds data about allowable number of incorrect tuples (threshold) for different fragments. We designed *horFragFilterAttributeVal* attribute as a char field so that any data type values (or multiple values) could be stored in there. To reduce the number of tuples (thresholds) that user has to define, we adopt a convention that the threshold for the missing values (e.g. department) is default (in our case, zero). The *metaETLHorFrag* table is, on procedure completion, updated with the outcome attributes (*isOK*, *errCount*, *lastCorrectDate*).

4. Improving the completeness and timeliness

In the context of horizontal fragmentation of the data warehouse tables, we do not consider the problems of data quality between the real world and the source data, but the data quality problems that arise from the extraction, transformation and loading of data from a relational source into the data warehouse environment. Accordingly, the relational data source is considered complete and timely and incompleteness and timeliness appear in the data warehouse when it does not represent the exact image of the source data.

4.1 Completeness

The data warehouse is complete as much as the data which should be extracted from the data source, is, in fact, extracted, transformed and loaded into the data warehouse.

Let r be the source table comprising of M tuples with schema R comprising of N attributes. Let s be the data warehouse DW table having r as a source table. Let $ref(r)$ be the referent table for r consisting of all semantic correct tuples of the r, $card\big(ref(r)\big) \le card(r)$. The completeness of the table s is defined with the following expression:

$$Q_{Compl}(s) = \frac{card(s)}{card(ref(r))}$$

Completeness of the table defined with above expression takes value from the interval [0, 1].

Completeness of the data warehouse DW comprising of K tables, according to [7] is defined by the completeness of the consisting tables:

$$Q_{Compl}(DW) = \frac{\sum_{k=1}^{K} Q_{Compl.}(r_k) * g_k}{\sum_{k=1}^{K} g_k}$$

$g_k \in [0,1]$, in the above expression, represents the relative importance of the table r_k in the data warehouse DW, and provides the opportunity that the completeness of table, that is estimated more important, affects the completeness of the entire data warehouse more.

An example from the Figure 4 and Figure 5 shows how the horizontal fragmentation process improves the completeness of the data warehouse table and consequently the completeness of the entire data warehouse. Consider an isolated segment of a data warehouse that integrates data from more higher education institutions. The observed segment consists of one dimension (*dStudent*) and one fact table (*fExam*). In the source tables (*student* and *exam*) data from number of higher education institutions are stored as well. The numbers in cells are number of tuples for particular fragment and table. The data from three higher education institutions (HEI 1, HEI 2 and HEI 3) are taken into account in the example. Two consecutive cycles of data warehouse refreshment c_i and c_{i-1} are observed. In the c_i cycle there was no erroneous tuples in the source and the warehouse is updated with all the source data. Upon completion of the c_i cycle completeness of the data warehouse is maximal and equals 1. In the c_{i+1} cycle, for the higher education institution HEI 3, one erroneous tuple appears in the *student* table and two erroneous tuples in the *fExam* table. The ETL process without horizontal fragmentation implemented, due to erroneous tuples, will not refresh the content of *dStudent* nor the content of *fExam* table. The content of *dStudent* and *fExam* after the c_{i+1} cycle will be the same as after the c_i cycle completion.

	cycle c_i						cycle c_{i+1}					
	source		DW				source			DW		
	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3
student/dStudent	4000	2000	1000	4000	2000	1000	4400	2200	**1100**	4000	2000	1000
exam/fExam	80000	40000	20000	80000	40000	20000	84000	42000	**21000**	80000	40000	20000

1 erroneous tuple in student
2 erroneous tuples in exam

Figure 4 Completeness – data warehouse loading without horizontal fragmentation

Completeness of the *dStudent* and *fExam* table is calculated with the following expressions:

$$Q_{Compl}(dStudent) = \frac{card(dStudent)}{card\big(ref(student)\big)}$$
$$= \frac{4000+2000+1000}{4400+2200+(1100-1)} = \frac{7000}{7699} = 0{,}9092$$
$$Q_{Compl}(fExam) = \frac{card(fExam)}{card\big(ref(exam)\big)}$$
$$= \frac{80000+40000+20000}{84000+42000+(21000-2)} = \frac{140000}{146998} = 0{,}9523$$

Completeness of the data warehouse segment is calculated based on the completeness of the tables contained in observed segment. Assuming that each table affects data warehouse completeness with an equal weight, weighting factor g_k equals to 1 for both tables.

$$Q_{Compl}(DW) = \frac{Q_{Compl}(dStudent) * g_1 + Q_{Compl.}(fExam) * g_2}{1+1}$$
$$= \frac{0{,}9092 * 1 \; + 0{,}9523 * 1}{2} = 0{,}9307$$

Figure 5 illustrates refreshment of the data warehouse segment using an ETL process with implemented horizontal fragmentation. A natural criterion for the fragmentation in this example is the identifier of the higher education institutions. Assuming that the tolerated number of erroneous records (*metaETLHorFrag.treshold*) is set to 0 for each institution and each table, the associated fragment of the *dStudent* and *fExam* for the HEI 3 in the c_{i+1} cycle will not be refreshed, while the fragments associated to the remaining higher education institutions having no errors will be refreshed. The completeness of *dStudent* and *fExam* tables and the entire data warehouse, with horizontal fragmentation implemented, amounts to:

$$Q_{Compl}(dStudent) = \frac{4400+2200+1000}{4400+2200+(1100-1)} = \frac{7600}{7699} = 0{,}9871$$
$$Q_{Compl}(fExam) = \frac{84000+42000+20000}{84000+42000+(21000-2)} = \frac{146000}{146998} = 0{,}9932$$
$$Q_{Compl}(DW) = \frac{0{,}9871 * 1 \; + 0{,}9923 * 1}{2} = 0{,}9901$$

	cycle c_i						cycle c_{i+1}					
	source			DW			source			DW		
	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3
student/dStudent	4000	2000	1000	4000	2000	1000	4400	2200	**1100**	4400	2200	1000
exam/fExam	80000	40000	20000	80000	40000	20000	84000	42000	**21000**	84000	42000	20000

Figure 5 Completeness – data warehouse loading with horizontal fragmentation

1 erroneous tuple in student
2 erroneous tuples in exam

4.2 Timeliness

The data arrive into the data warehouse timely if it is updated (copied) within the first possible data warehouse refresh cycle.

Expression proposed in [8] is used to define timeliness on the attribute value level:

$$Q_{Time}\left(v_O(t_i.A_j)\right) \;=\; exp(-decline(A) * \; age\;(v_O(t_i.A_j)))$$

$Q_{Time}\left(v_O(t_i.A_j)\right)$ denotes the probability that the attribute value is still valid. $v_O(t_i.A_j)$ is the observed and recorded value of the attribute A_j of the tuple t_i. The *decline* (A) is the decline rate indicating how many values of the attribute considered become out of date on average within one period of time The *age* $(v_O(t_i.A_j))$ denotes the age of the attribute value $v_O(t_i.A_j)$ which is computed by means of two factors: the instant when attribute value timeliness is quantified and the instant of attribute value acquisition.

Assuming that the data warehouse is being refreshed once a day, decline and age of an attribute value are calculated with a day as the time unit. The example illustrated with Figure 6 and Figure 7 shows the improvement of the timeliness of attribute values, tuple, table, and the entire data warehouse accomplished with the horizontal fragmentation.

Analysing changes in source tables for the HEIS [1] for regarded warehouse segment, we found that, on average, 2 ‰ tuples change in one day in the student table and 0.6 ‰ in the exam table. For a precise calculation of the timeliness dimension the decline of each relevant attribute should be assessed. However, that is not in the focus of this paper and so we consider here that the variability of all attributes is the same.

	cycle c_i						cycle c_{i+1}					
	source			DW			source			DW		
	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3
student/dStudent	4000	2000	1000	4000	2000	1000	4000	2000	1000	4000	2000	1000
#of upd.-tuples							200	100	50	200	100	50
exam/fExam	80000	40000	20000	80000	40000	20000	80000	40000	20000	80000	40000	20000
#of upd.-tuples							400	200	100	400	200	100

Figure 6 Timeliness- data warehouse loading without horizontal fragmentation

1 erroneous touple in student
2 erroneous touples in exam

The timeliness of an arbitrary attribute of the student table with age of one and two days amounts to:

$Q_{Time}(v_O(t_i.A_j)) = exp(-0{,}002 * 1) = 0{,}9977$

$Q_{Time}(v_O(t_i.A_j)) = exp(-0{,}002 * 2) = 0{,}9959$

Similarly, the timeliness of an arbitrary attribute of the exam table amounts to:

$Q_{Time}(v_O(t_i.A_j)) = exp(-0{,}0006 * 1) = 0{,}9993$

$Q_{Time}\left(v_O\left(t_i.A_j\right)\right) = exp(-0{,}0006 * 2) = 0{,}9987$

If an attribute value fails to update (due to an error) at the very first cycle its age increases and timeliness decreases. The timeliness of the tuple is determined based on the timeliness of its attribute values. With the simplification that all attributes are equally important and have an equal decline rate and age, timeliness of tuple will be equal to the timeliness of the single attribute value. The timeliness of the r table comprising of the $card(r)$ tuples is determined by the timeliness of the its tuples:

$$Q_{Time}(r) = \frac{\sum_{j=1}^{card(r)} Q_{Time}(t_j)}{card(r)}$$

The timeliness of the data warehouse DW is determined by the timeliness of its tables:

$$Q_{Time}(DW) = \frac{\sum_{k=1}^{K} Q_{Time}(r_k) * g_k}{\sum_{k=1}^{K} g_k}$$

In the Figure 6 , the row titled “#of upd tuples” shows the number of tuples updated between two consecutive cycles c_i and c_{i+1}.

If the changes that had happened in the *dStudent* and *fExam* tables are successfully conducted in the c_{i+1} cycle, the timeliness of the tables and the data warehouse will be maximal. Suppose that the age of all tuples in the tables *dStudent* and *fExam* after the c_i cycle is one day. After the c_{i+1} cycle the age of altered (200 +100 +50 in *dStudent* and 400 +200 +100 in *fExam*) tuples is equal to one day and of all other tuples it is two days. However, in the presence of erroneous tuples, those changes will not reflect on dStudent and fExam in the c_{i+1} cycle if the strategy for the data warehouse refreshment without horizontal fragmentation is employed.

The timeliness of *dStudent* and *fExam* after the c_{i+1} cycle is equal to:

$$Q_{Time}(dStudent) = \frac{\sum_{j=1}^{card(dStudent)} Q_{Time}(t_j)}{card(dStudent)} =$$

$$\frac{(4000+2000+1000)*0.9959}{4000+2000+1000} = 0{,}9959$$

$$Q_{Time}(fExam) = \frac{(80000+40000+20000)*0.9987}{80000+40000+20000} = 0{,}9987$$

The data warehouse timeliness is equal to:

$$Q_{Time}(DW) = \frac{Q_{Time}(dStudent)*g_1 + Q_{Time}(fExam)*g_2}{1+1}$$

$$= \frac{0{,}9959*1+0.9987*1}{1+1} = 0{,}9973$$

With implemented horizontal fragmentation changes (Figure 7) in the tuples belonging to the horizontal segments of HEI 1 and HEI 2 will be carried out successfully. After c_{i+1} cycle the age of modified tuples will be one, and of all other tuples two days.

The timeliness of *dStudent* and *fExam* and of the observed data warehouse segment amounts to:

$$Q_{Time}(dStudent) = \frac{(4000-200+2000-100+1000)*0.9959+(200+100)*0.9977}{4000+2000+1000} = 0{,}99597$$

$$Q_{Time}(fExam) = = \frac{(80000-4000+40000-2000+20000)*0.9987+(4000+2000)*0.9993}{80000+40000+20000} = 0{,}99873$$

$$Q_{Time}(DW) = \frac{0{,}99597*1+0{,}99873*1}{1+1} = 0{,}99735$$

For determining timeliness exponential function (with negative exponent) with extremely slowly decreasing value is used. Hence, it is understandable that the differences in the timeliness of one day are not drastic – they are expressed in per mils. Never the less, the fact is that the process of horizontal fragmentation improves the timeliness of data in a given time interval.

	cycle c_i						cycle c_{i+1}					
	source			DW			source			DW		
	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3	HEI 1	HEI 2	HEI 3
student/dStudent	4000	2000	1000	4000	2000	1000	4000	2000	1000	4000	2000	1000
#of upd.-tuples							200	100	50	200	100	50
exam/fExam	80000	40000	20000	80000	40000	20000	80000	40000	20000	80000	40000	20000
#of upd.-tuples							4000	2000	1000	4000	2000	1000

Figure 7 Timeliness- data warehouse loading with horizontal fragmentation

1 erroneous touple in student
2 erroneous touples in exam

In conclusion, the timeliness of the data warehouse refreshed employing ETL process without horizontal fragmentation will be the same as the timeliness of the same data warehouse refreshed employing ETL process with horizontal fragmentation when all the correct data are successfully updated.The benefit from horizontal fragmentation is in improving the timeliness in the occasions when, due to errors, correct data isn't updated.

5. Conclusion

In this paper we've discussed a generic procedure for horizontal fragmentation of the data warehouse. Proposed procedure is suitable when there are multiple somewhat independent departments (institutions, ...) that are generating data of the same structure (i.e. belonging to the same business process). For instance, different higher education institutions are generating students' exam records. In order to be generic, the procedure relies on metadata, which is presented and discussed. The criterion for the fragmentation of the table is arbitrary (e.g. institutionID). Using metadata, horizontal fragmentation can be customized to be more or less forgiving to errors. For instance, with high enough threshold values, algorithm recedes to a special case when all errors are ignored and data warehouse always gets updated with the new valid tuples (which is also a legit and popular approach).

We show that employing horizontal fragmentation improves data warehouse quality, namely, the completeness and timeliness data quality dimensions. Improvement is achieved in the cases when the ETL process without horizontal fragmentation would not update correct data due to errors. The applicability of the proposed procedure has been tested in the real world project: data warehousing system integrating data from 90 higher educational institutions in the Republic of Croatia. Erroneous data detected in the process is logged and ultimately shown relevant users of the data warehousing system. Having ability to examine errors on-line, users are motivated to eliminate them and consequently raise the quality of data in the data warehouse, which presents a nice side benefit.

References:

[1] I. Mekterović; Lj. Brkić, M. Baranović: Improving the ETL process and maintenance of Higher Education Information System Data Warehouse. *WSEAS transactions on computers.* 8 (2009), 10; 1681-1690.

[2] R. Y. Wang, M.P. Reddy, H. B. Kon: Toward quality data: An attribute-based approach, *Decision Support Systems 13* (1995), 349-372

[3] C. Batini, M. Scannapieca: Data Quality Concepts, Methodologies and Techniques, *Springer-Verlag Berlin Heidelberg*, (2006)

[4] L. L. Pipino, Y. W. Lee, R. Y. Wang: Data Quality Assessment, *Communications of the ACM 45* (2002), no. 4.

[5] B. Heinrich: A Novel Data Quality Metric for Timeliness Considering Supplemental Data, *17th European Conf. on Information Systems (ECIS)*, (2009)

[6] M. Kaiser: A Conceptional Approach to Unify Completeness, Consistency and Accuracy as Quality Dimensions of Data Values, *European and Mediterranean Conf. on Information Systems*, (2010)

[7] B. Heinrich, M. Kaiser, M. Klier: Does the EU Insurance Mediation Directive help to improve Data Quality? - A metric-based analysis. *Proc. of the 16th European Conference on Information Systems (ECIS)*, (2008)

[8] M. Kaiser, M. Klier, B. Heinrich: How to Measure Data Quality?-A Metric-Based Approach, *Intern. Conf. on Information Systems (ICIS)*, (2007)