data cleaning, materialised views
knowledge discovery in medical databases

Adam WIDERA*, Michał WIDERA*, Daniel FEIGE*,
Krzysztof HOROBA*, Arkadiusz STANKIEWICZ*

DATA CLEANING OF MEDICAL DATA SETS

Each database system evolves during the time. If the primary database schema was designed only to store the limited scope of abstraction classes then the database system improvement process is performed in traditional way (using alter table, update table and create table commands). Anyhow it could be impossible, from the engineering point of view, or too expensive from the economic point of view. Transferring the data from one database schema to another database schema one has to perform an additional step called Data Cleaning. This paper present a basic sketch for the data cleaning theory based on the materialised views idea and corresponding data cleaning environment. The proposed methodology is suitable not only for the data verification but also for the reengineering of the broken references between data fields, recreation of missing rows and data types conversion.

1. INTRODUCTION

Data Cleaning process is one the basic tasks performed during creation of the Data Warehouses (Data Cleaning is a phase of the ETL - Extraction, Transformation, Loading), during modification and integration of database schemas (so called evolution of the database schema) and during the process of Knowledge Discovery in Databases. The managed amount of data undergoes continuous evolution together with the evolution of the computer system. If the primary database schema was designed only to store the limited scope of abstraction classes then the database system improvement process performed in traditional way (using alter table, update table and create table commands) could be impossible from the engineering point of view or too expensive from the economic point of view. The only reasonable solution is to transfer all the information stored in the faulty database system to the new, improved and corrected database schema. We meet a similar scope of problems while evolving The Medical Database System for Management of Patients with Implanted Pacemakers "IMPULS" [5,15].

To solve the problem, the detection of invalid and incoherent values, correction and recreation of missing information in the data repository of patient's diseases was necessary. The missing values in the clinical trail database were reconstructed using partial information

* Institute of Medical Technology and Equipment, 118 Roosevelt St, 41-800 Zabrze, POLAND,
tel./fax: (32) 2716013/2712312, e-mail: adam.widera@itam.zabrze.pl

stored in the evolved database system. This task is commonly known as the problem of Data Cleaning. In the research we created a theory and implemented a suitable environment for designing the data cleaning process coherent with this theory.

2. DATA CLEANING ISSUES

Many scientists are searching for the perfect way to clean the data and make them appropriate for the desired purposes [1,4,6,7,8,10,12,13,14]. Some of the papers do not differentiate between the data cleaning and the Extraction Transformation Loading process performed during the creation of the data warehouse [12]. We think it is wrong since data cleaning is performed not only during the process of ETL but also during the process of Knowledge Discovery on the non-warehouse data sets too. The most popular field where data cleaning methods are used nowadays is the research in genome. Just to mention the BIO-AJAX project [11]. Generally the process of Data Cleaning consists of several tasks. One common classification of those tasks was introduced by Rahm and Do [3] and since now this work is concerned as a survey in the field of Data Cleaning. The presented classification differentiates issues into two classes of problems: found during the process of cleaning of a single data source (Single-source) and several data sources (Multi-source). Both classes of problems are then divided to schema level problems and instance level problems.

According to their classification common single-source schema level problems enumerate: illegal values, violated attribute dependencies, uniqueness violation and referential integrity violation. The single-source instance level problems are: missing values, misspellings, cryptic values and abbreviations, embedded values, misfiled values, violated attribute dependencies, word transpositions (for example a first name present one time before second name some other time after second name in a string), duplicated records, contradiction records, wrong references.

Multi-source schema level problems are: naming conflicts (when the same attribute is present in different data sources under different name or two different attributes are present under the same name), structural conflicts (for example when one abstraction is present in one schema as a set of columns while in other as a set of rows or when different schemas use different data types to represent the same abstraction). The scope of instance-level problems present during integration of two or more data sources is much the same as the scope of instance-level problems presents while data cleaning a single-source. Even though the above classification is valid, to illustrate our own solution in a suitable way we suggest a simplified way of classification presented in Table 1.

Table 1. Simplified classification of data cleaning problems.

Problem class	Example
Detection of duplicate values [1, 2]	$name_1$="J.Kowalski", $born_1$=1978 $name_2$="John Kowalski" , $born_2$=1978
Detection of invalid values	$name_1$= "J.Smith" , $hight_1$=178 $name_2$= "J.Kowalski", $high_2$="tall"
Detection of inconsistent values	$born_1$= "1978", age_1=26,id_1=781278xxxxx $born_2$="1960",**age_2=30**,id_2=781260xxxxx
Detection of broken or invalid references	$name_1$="Kowalski",DepartmentID_fk_1=14 $name_2$="Nowak",**DepartmentID_fk_2=16** $name_3$="Smith",DepartmentID_fk_3=15 --- DepartmentID_pk_1=14,$name_1$="Technical" DepartmentID_pk_2=15,$name_2$="Research"
Detection of missing values	$model_1$="Neo 02-Up",serial_no_1=5076096 $model_2$="LCP 201",**serial_no_2=0** $model_3$="TRIOS 02-UP",serial_no_3=50464083
Detection of values out of scope	Id_1=1,$height_1$=178 Id_2=2,**$height_2$=-160**
Separation of values embedded inside others	$Name_1$="Mr. John Smith" $Name_2$=" J. Kowalski PhD" $Forename_1$="John", $surname_1$="Smith", $title_1$="Mr." $Forename_2$="J.", $surname_2$="Kowalski", $title_2$="PhD."

This classification does not differentiate of single-source cleaning problems and problems of multi-source integration. Problems relevant to integration issues are solved by early fusion of all necessary data sources, before the main data cleaning process.

Nowadays, the main goals of data cleaning research are: creating declarative data cleaning query languages (AJAX [6]), developing specialised ETL software applications (ARKTOS [7], Potter's Wheel [8], AJAX [6]) and developing algorithms used during the data cleaning process. [1, 2, 9, 10]. Presented solutions usually concentrate on the data warehouse data cleaning applications. But such applications are usually associated with particular database management system or aim one particular problem, for example verification of business data.

There are two main ways of defining the data cleaning process: using declarative data cleaning query language [4, 6] or using graphical user interface and a set of predefined transformations [8]. The first way is suitable when one has to adapt the data cleaning process to one particular task, even though requires a lot of effort and assumes the user to have programming skills. The second way, usually based on predefined set of transformations [8], makes the process of data cleaning design intuitive and efficient at one hand but the process of adaptation very difficult and time consuming at other hand.

Still important issue in designing the data cleaning process is the problem of determining the origin of invalid values. Using lineaging one could designate the primal reason of the anomaly observed in the cleaned data source. Research papers present two main ways for obtaining such information. The first way assumes that algorithms used

during data cleaning process store some additional information about the source data for performed transformations. The second way uses advanced analysis algorithms to perform backtracking about the origin of the source data based on existing result sets [9].

3. THE DATA CLEANING METHODOLOGY

The term Data Cleaning Methodology is used to describe both the design of the Data Cleaning process and the way data cleaning process works itself. The idea of the proposed methodology could be illustrated using the ordered graph of relation transformations (Fig. 1)

Terms R_i, R_j are relations and the term $P_{i,j}$ is understood as the set containing all transformations between relation R_i and relation R_j. The term $R_{m...k}$ is used to designate all relations directly preceding the relation R_j and $P_{j,j}$ could be especially an empty set. The following expression designates the list of all transformations bounded with the relation R_j (1):

$$P_j := P_{m,j} \cup ... \cup P_{n,j} \cup ... \cup P_{k,j} \cup P_{j,j} \qquad (1)$$

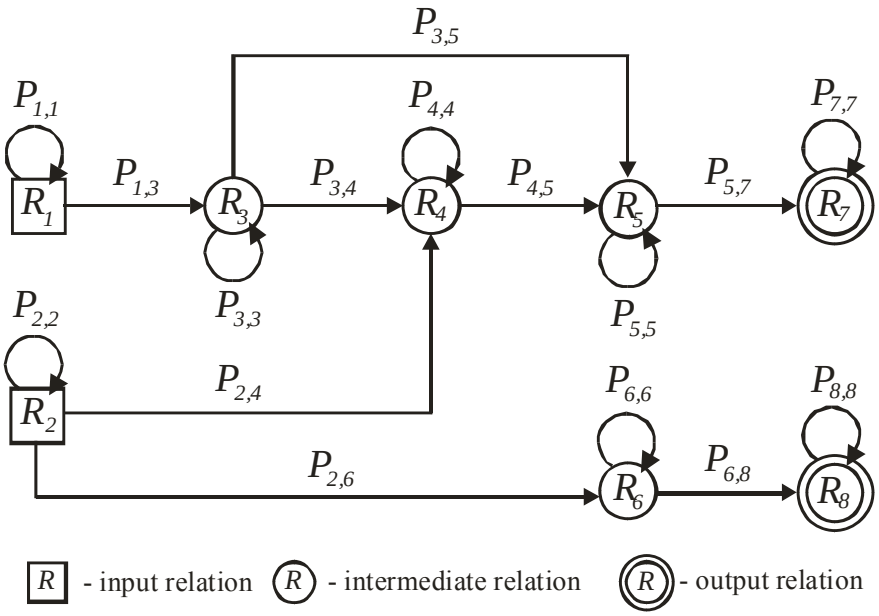

Fig. 1. Example graph of relation transformations

The pair (P_j,R_j) is called a „Data Cleaning Step" and is designated as K_j. Furthermore the Data Cleaning Step during the Data Cleaning Process is physically represented as an object in a programming sense (i.e. a program unit). The classification of transformations based on the localisation of the output is presented in Table 2.

Table 2. Classification of transformations based on the localization of the output.

Symbol	Description
Transformations that generate new attributes in relation R	
P^M	transformations that duplicate attributes from the SQL query result
P^N	transformations that generate new attributes
Transformations that do not generate attributes are classified based on the place in the timeline of the transformation process	
P^B	transformations executed before the first database fetch
P^R	transformations executed after the fetch from the database
P^E	transformations executed after the last fetch from the database

For each data cleaning step K_j the following relationship is valid (2):

$$P_j := P_j^B \cup P_j^M \cup P_j^N \cup P_j^R \cup P_j^E \tag{2}$$

An assumption is made that each transformation that belong to a particular data cleaning step has got a unique order number inside this data cleaning step. When the data cleaning step K_j defines h_j transformations each single transformation defined in this data cleaning step is going to be designated as $P_{j[n]}$, where n is the order number inside the data cleaning step. The following relationship is valid (3),(4):

$$\overline{\overline{P_j}} := h_j \tag{3}$$

Where $\overline{\overline{P_j}}$ represent the cardinality of the set P_j (i.e. the number of elements).

$$P_j := (P_{j[1]}, P_{j[2]}, ..., P_{j[n-1]}, P_{j[n]}, P_{j[n+1]}, ..., P_{j[h_j]} \tag{4}$$

The previously defined (1) set P_j is an ordered set. The transformation $P_{j[n]}$ directly precede the transformation $P_{j[m]}$ inside the data cleaning step K_j, when $n=m-1$ (4).

When x stands for the result of a transformation, and A stands for a set of some input data, (in peculiarity an empty set) the following equation (5):

$$x = P_{j[n]}(A) \tag{5}$$

(5) is read: „The result of the transformation $P_{j[n]}$ on a set A is x", where the datatype of the value x is unrestricted.

Designating the operation of relation creation with symbol „→", the expression (6):

$$\bigcup_m P_{j[m]}(A_m) \rightarrow R_j, where P_{j[m]} \in P_j^M \cup P_j^N \tag{6}$$

We mean: “The set of results generated by transformations that generate attributes creates the relation related with the given data cleaning step”.

The preceding notation is going to be used to illustrate the exemplary data cleaning issue. The algorithm of data cleaning step processing used in the presented methodology (Fig. 2).

4. DESIGNING THE DATA CLEANING PROCESS

To illustrate the process of the data cleaning process design for the purposes of Knowledge Discovery we’ve decided to present a simple example addressing common data cleaning issue. Assume there is a medical trail table containing information about implantation of pacemakers. Assume the form of the data stored in the medical database is dirty, as presented in the Fig. 3.

PatientID	Pacemaker	Implantation_date	Control_date	Death_date
1	“MEDTRINIC SC100 #2000413”	08-12-1992	“Dec 24 1993 “	“10 dec 2001”
2	“ST JOHN MED #03214A23”	10-09-1999	“Dec 24 2000 “	None
3	“SG-200 Bit. SN3254132”	12-03-1997	“09 24 1998”	“15 September 2004”

Fig. 3. Example of a dirty data set[1]

Assume there is a strong need to transform the data to a form understandable by some kind of a Knowledge Discovery algorithm. For convenience of the knowledge discovery engineer each case has to receive unique number. Both the case unique number and the patient number have to be preceded with the two-digit id of the implantation centre. Moreover the Pacemaker field has to be separated into fields containing the name of the producer, model and serial number. The date of the first pacemaker control and the date of death have to be converted into number of days after implantation. As a result of the data cleaning step the algorithm should return the number of processed rows. Due to such conditions the cleaned data have to be transformed into the data set presented in the Fig. 4.

1 Legal notice: Both the name of the pacemaker producer and the presented periods of time are fictional.

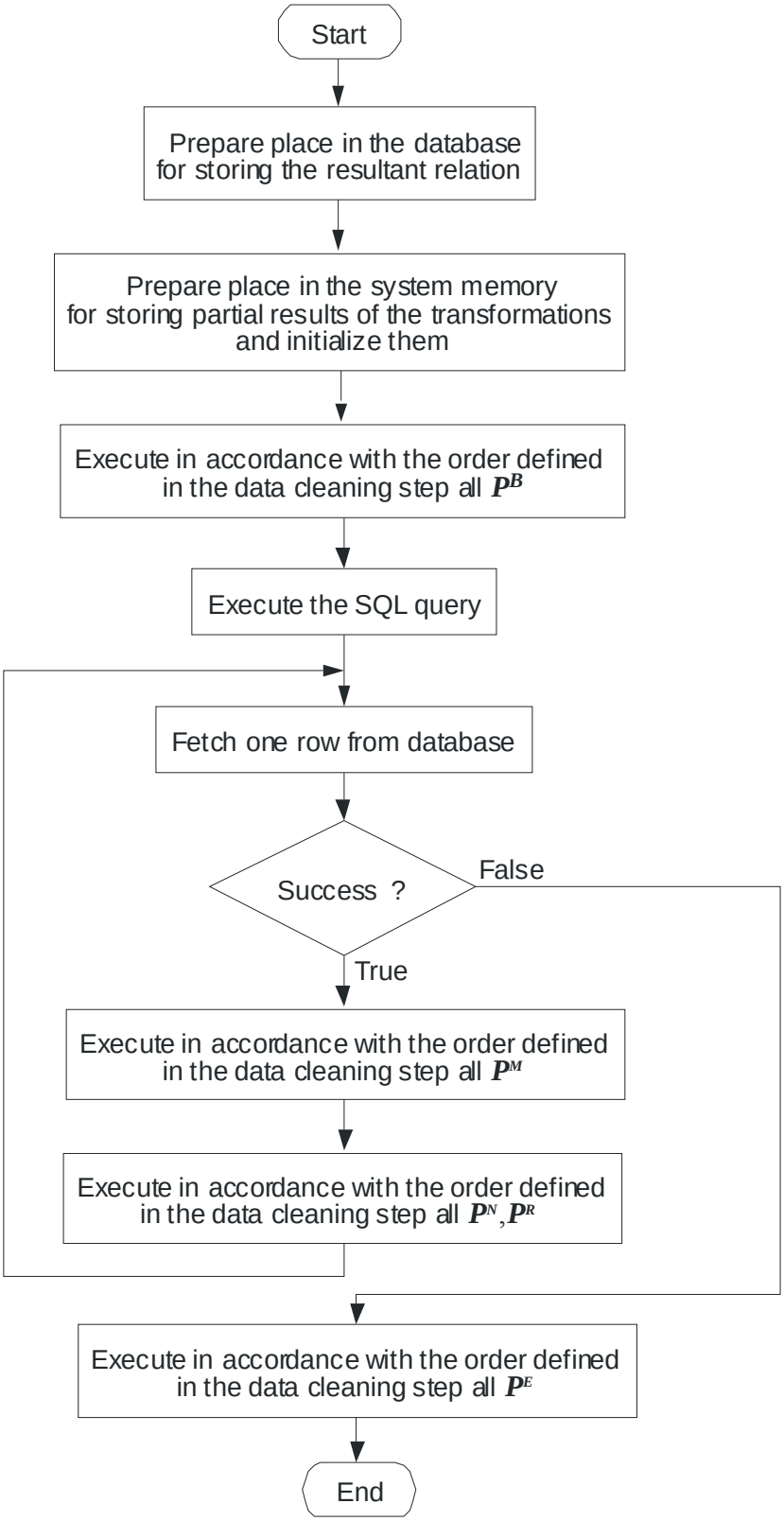

Fig. 4. The algorithm of data cleaning step processing.

CaseID	ID	Implantation	Producer	Model	Serial	Control	Death
22000001	22000003	12-03-1997	BITTRONIC	SG-200	3254132	561	2744
22000002	22000001	08-12-1992	MEDTRINIC	SC100	2000413	381	3289
22000003	22000002	10-09-1999	ST JOHN MED	None	0314A23	471	None

Fig. 5. Example of a clean data set[1]

For such terms lied down we define the following transformations (in the form of functions):

$P_{1[1]}$ = transcribe the given value (value)
$P_{1[2]}$ = transcribe the column from the retrieved row(column_id)
$P_{1[3]}$ = transcribe the column from the retrieved row(column_id)
$P_{1[4]}$ = transcribe the column from the retrieved row(column_id)
$P_{1[5]}$ = transcribe the column from the retrieved row(column_id)
$P_{1[6]}$ = transcribe the column from the retrieved row(column_id)
$P_{1[7]}$ = add 1 to the given value (value)
$P_{1[8]}$ = generate integer identifier preceded with the given prefix (id, prefix)
$P_{1[9]}$ = generate integer identifier preceded with the given prefix (id, prefix)
$P_{1[10]}$ = transcribe the given value (value)
$P_{1[11]}$ = generate three element array with producer, model and serial number substracted from the given string (value)
$P_{1[12]}$ = transcribe the given value (value)
$P_{1[13]}$ = transcribe the given value (value)
$P_{1[14]}$ = transcribe the given value (value)
$P_{1[15]}$ = transform the given string into valid date format
$P_{1[16]}$ = count the number of days between the start_date and end_date (start_date, end_date)
$P_{1[17]}$ = transform the given string into valid date format
$P_{1[18]}$ = count the number of days between the start_date and end_date (start_date, end_date)
$P_{1[19]}$ = transcribe the given value (value)

Where:

$P_{1[1]} \in P^B$
$P_{1[2]}, P_{1[3]}, P_{1[4]}, P_{1[5]}, P_{1[6]} \in P^M$
$P_{1[7]}, P_{1[11]}, P_{1[15]}, P_{1[17]} \in P^R$
$P_{1[8]}, P_{1[9]}, P_{1[10]}, P_{1[12]}, P_{1[13]}, P_{1[14]}, P_{1[16]}, P_{1[18]} \in P^N$
$P_{1[19]} \in P^E$

In such case the solution of the problem resolves to plan the following transformations:

ImplantationCentreID=$P_{1[1]}(22)$
PatientID=$P_{1[2]}(0)$

Pacemaker=$P_{1[3]}$(1)
ImplantationDate=$P_{1[4]}$(2)
ControlDate=$P_{1[5]}$(3)
DeathDate=$P_{1[6]}$(4)
Counter=$P_{1[7]}$(Counter)
CaseID=$P_{1[8]}$(Counter, ImplantationCentreID)
ID=$P_{1[9]}$(PatientID, ImplantationCentreID)
Implantation=$P_{1[10]}$(ImplantationDate)
ProducerModelSerial=$P_{1[11]}$(Pacemaker)
Producer=$P_{1[12]}$(ProducerModelSerial[0])
Model=$P_{1[13]}$(ProducerModelSerial[1])
Serial=$P_{1[14]}$(ProducerModelSerial[2])
Date1=$P_{1[15]}$(ControlDate)
Control=$P_{1[16]}$(Implantation, Date1)
Date2=$P_{1[17]}$(DeathDate)
Death=$P_{1[18]}$(Implantatio, Date2)
RowsProcessed=$P_{1[19]}$(Counter)

Where the initial value of the partial results are:
Counter=0
ProducerModelSerial=[None, None, None]
Date1=None
Date2=None

And the database types of the resulting relation are:

CaseID=Integer
ID=Integer
Implantation=Date
Producer=Char(20)
Model=Char(20)
Serial=Char(20)
Control=Integer
Death=Integer

The table notation of transformations is presented in the Table 3.

5. IMPLEMENTATION

The prototype application has been implemented using Python programming language [11]. The visualisation of the data cleaning process as well as the graphics user interface of the data cleaning development environment has been implemented using the platform independent QT library [12] and its port for the python language – PyQT [13]. The data

cleaning project files as well as the result files are stored in XML documents. To connect to the database we've created our own Python ODBC port based on the c-types library [14]. We have also created ports for other Python database modules (mxODBC, win32all.odbc, DynWin.odbc).

Table 3. The table notation of transformations. (Design of the programming unit)

ID	@	SET	NAME	INIT	EQUAL
			INC		add 1 to the given value (value)
			GENERATE_ID		generate integer identifier preceded with the given prefix (id,prefix)
			SUBSTRACT		generate three element array with producer, model and serial number substracted from the given string (value)
			GET_CLEAN_DATE		transform the given string into valid date format
			GET_DIFFERENCE_DAYS		count the number of days between the start_date and end_date (start_date,end_date)
$P_{1[1]}$	No	P^B	ImplantationCentreID		22
$P_{1[2]}$	No	P^M	PatientID	Char(20)	Query[0]
$P_{1[3]}$	No	P^M	Pacemaker	Char(40)	Query[1]
$P_{1[4]}$	No	P^M	ImplantationDate	Date	Query[2]
$P_{1[5]}$	No	P^M	ControlDate	Char(20)	Query[3]
$P_{1[6]}$	No	P^M	DeathDate	Char(20)	Query[4]
$P_{1[7]}$	No	P^R	Counter	0	self.INC(self.Counter)
$P_{1[8]}$	Yes	P^N	CaseID	Integer	self.GENERATE_ID(self.Counter,self.ImplantationCentreID)
$P_{1[9]}$	Yes	P^N	ID	Integer	self.GENERATE_ID(self.PatientID,self.ImplantationCentreID)
$P_{1[10]}$	Yes	P^N	Implantation	Date	self.ImplantationDate
$P_{1[11]}$	No	P^R	ProducerModelSerial	[None,None,None]	self.SUBSTRACT(self.Pacemaker)
$P_{1[12]}$	Yes	P^N	Producer	Char(20)	ProducerModelSerial[0]
$P_{1[13]}$	Yes	P^N	Model	Char(20)	ProducerModelSerial[1]
$P_{1[14]}$	Yes	P^N	Serial	Char(20)	ProducerModelSerial[2]
$P_{1[15]}$	No	P^R	Date1	None	self.GET_CLEAN_DATE(self.ControlDate)
$P_{1[16]}$	Yes	P^N	Control	Integer	self.GET_DIFFERENCE_DAYS(self.Implantation,Date1)
$P_{1[17]}$	No	P^R	Date2	None	self.GET_CLEAN_DATE(self.DeathDate)
$P_{1[18]}$	Yes	P^N	Death	Integer	self.GET_DIFFERENCE_DAYS(self.Implantation,Date2)
$P_{1[19]}$	No	P^E	RowsProcessed		self.Counter

6. SUMMARY, CONCLUSIONS AND FUTURE WORK

The above solution presents high flexibility and potential, making the prosecuted research in this matter well purposeful. The use of a methodology based on the idea of materialised views as well as the use of the Python scripting language seems to be a good decision. The empirical evidence of such thesis is the successful evolution of The Medical Database System for Management of Patients with Implanted Pacemakers “IMPULS” System Database. Using the presented methodology we have verified the medical trail database, repaired the broken references between data fields, recreated missing rows and performed data type conversion. Thanks to the methodology based on the idea of materialised views the detection of faults in the implemented data cleaning algorithms became as simple task as debugging typical applications in a high level language. Therefore the time necessary to implement a data evolution solution has been highly reduced. During the research process we've developed basic sets of most commonly used transformations. We have perceived a great potential in this methodology as a basement for a Knowledge

Discovery environment. We plan to develop appropriate algorithms and transformations for the presented methodology. We also plan to extend the presented methodology to parallelise the process of data cleaning along the local area network.

BIBLIOGRAPHY

[1] MONGE A. E.: Matching Algorithms within a Duplicate Detection System. IEEE Bulletin of the Technical Committee on Data Engineering, 2000, t. 23, nr 4, s. 14-20.
[2] NAVARRO G., BAEZA-YATES R., SUTINEN E., TARHIO J.: Indexing Methods for Approximate String Matching. IEEE Bulletin of the Technical Committee on Data Engineering, 2001, t. 24 nr 4, s. 19-27
[3] RAHM E., DO H.: Data Cleaning: Problems and Current Approaches. IEEE Bulletin of the Technical Committee on Data Engineering, 2000, t. 23 nr 4, s. 3-13.
[4] GALHARDAS H., FLORESCU D., SHASHA D.: Declarative Data Cleaning: Language, Model, and Algorithms. INRIA Technical Report RR-4149, 2001.
[5] GAŁECKA, J., JAROCKI B., ŻMUDZIŃSKI J., KOSIŃSKI W., BADURA G., FIGA D.: Measurement station with database IMPULS supporting the management of patients with implanted pacemaker. Prace Naukowe Instytutu Górnictwa Politechniki Wrocławskiej Seria: Konferencje. 1999, Nr 24, s. 133-137.
[6] GALHARDAS H., FLORESCU D., SHASHA D., SIMON E.: AJAX: An Extensible Data Cleaning Tool. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, 2000, s. 590.
[7] VASSILIADIS P., VAGENA Z., SKIADOPOULOS S., KARAYANNIDIS N., SELLIS T.: ARKTOS: A Tool For Data Cleaning and Transformation in Data Warehouse Environments. IEEE Bulletin of the Technical Committee on Data Engineering, 2000, t. 23 nr 4, s. 42- 47.
[8] RAMAN V., HELLERSTEIN J. M.: Potter's Wheel: An Interactive Data Cleaning System. Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy. 2001, s. 381-390.
[9] CUI Y., WIDOM J.: Lineage tracing for general data warehouse transformations. The VLDB Journal — The International Journal on Very Large Data Bases, 2003, t. 12, z. 1, s. 41 – 58.
[10] SUNG S. Y., LI Z., SUN P.: A Fast Filtering Scheme for Large Database Cleansing. Proceedings of the eleventh international conference on Information and knowledge management 2002, McLean, Virginia, USA , November 04-09 2002, 2002, s.76 – 83.
[11] HERBERT K. G., GEHANI N. H., PIEL W.H., WANG J. T. L., WU C. H.: BIO-AJAX: An Extensible Framework for Biological Data Cleaning. SIGMOD Record , 2004, t.33, nr 2.
[12] VASSILIADIS P., SIMITSIS A., SKIADOPOULOS S.: Conceptual Modeling for ETL Processes. Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP 2002, McLean, Virginia, USA, November 08008 2002, 2002, s. 14-21
[13] MALETIC J. I., MARCUS A.: Data Cleansing: Beyond Integrity Analysis. Proceedings of the 5th conference on Information Quality IQ2000, Boston, MA, USA, October 20-23, 2000, s.200-209
[14] HERNANDEZ M. A., STOLFO S. J.: Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. Data Mining and Knowledge Discovery, 1998, t. 2 nr 1, s. 2-37.
[15] WIDERA A., WIDERA M., OWCZAREK A., MOMOT. M., SIKORA M., Metoda oczyszczania danych w oparciu o mechanism perspektyw materializowanych. Współczesne problemy sieci komputerowych – Nowe technologie. red. S. Węgrzyn, B.Pochopień, T.Czachórski, Wydawnictwa Naukowo-Techniczne, s. 401, Warszawa, 2004

This study is financed by the State Committee for Scientific Research resources in 2003-2005 years as a research project KBN Grant No. 4 T11F 002 25.