CONTINUOUS CONCEPTUAL SCHEMA QUALITY CHECKING

Christian Kop
Institute of Applied Informatics, Alpen-Adria-Universitaet Klagenfurt, Universitätsstrasse 65-67, 9020 Klagenfurt, Austria

Keywords: Requirements Engineering, Conceptual Modelling, Data Schema Quality, Controlled Language Queries.

Abstract: Since a conceptual database schema is the backbone of data intensive software, the schema must be continuously checked during the requirements engineering step in order to get the proper quality. One possibility is to check if the conceptual schema of the database provides the necessary data for retrieval needs. Talking about the retrieval needs would help but SQL is too technical for that purpose. To solve this problem, this paper presents an approach which uses controlled language queries for checking a conceptual schema.

1 INTRODUCTION

Since a database schema is the backbone of data intensive software, effort must be spent already during the early design of a conceptual schema (i.e. during requirements engineering). A conceptual database schema described in an UML class diagram or an ORM diagram must be of high quality. Therefore such a schema must be continuously checked in order to get the proper quality. Particularly, the database must be able to manage all kind of structured information which the software will need to make retrievals from the data base. Therefore, a conceptual database schema has a good quality, if all necessary concepts and their relations are modelled to offer all retrieval possibilities which the future software must provide to the end users. The earlier the designer knows which data will be retrieved in which way, the better he will be able to decide if the conceptual schema has already a good quality. It is also necessary that end users are strongly involved in this quality checking step. But how can end users be involved? Some of them are not willing to read conceptual schemes. Some of them will try to read the schema but will not be able to understand the schema. Many others will try to check the quality of the schema, but without any additional support, they will not know what to check and if they do their job well according to the purpose of quality checking. Thus it might also happen, that end users ignore those diagrams at all and just rely that the designers have done their job well. To overcome this problem one strategy is to generate a fast prototype or at least user interfaces. However even here, some assumptions about the data retrieval must already be made. Another strategy is to offer the end user a possibility to talk about the data he needs in his reports and future user interfaces. Then it can be checked if these information given by the end users are already fulfilled by the schema. This is an idea which is worth to look for. However the problem remains, how can the retrieval and manipulation of data in a database be expressed? Using SQL here is not very successful since in SQL many technical terms are used. Furthermore SQL operates in a stage where the database is already created and stable.

Therefore, the query language must be more user centred than SQL and more related to the conceptual modelling notions. There are many approaches which visualize and create SQL by navigating through the conceptual schema (see section 5). However, also all these strategies need an already final and stable schema. In contrary to other approaches this paper focuses on the idea:

- *That the user must not be influenced by the schema when he expresses his retrieval needs.*
- *Schema checking must be done as early as possible and*
- *A query cannot be applied on a stable schema since the aim and purpose of the query is the improvement of the schema.*

To fulfil the above stated needs, an approach which uses controlled language is proposed. A controlled language is a subset of natural language with a restricted grammar. Such kind of natural languages

Kop C. (2009).
CONTINUOUS CONCEPTUAL SCHEMA QUALITY CHECKING.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 186-193

are in use in approaches that try to extract tuples from a database or which try to generate SQL. However they are not used for schema checking.

Therefore, the paper is structured as follows. Section 2 explains the quality checking process. In section 3, information is given about the schema checking with a controlled language. Section 4 continues with an overview of a prototype that was built for a technical proof of concepts of this approach. Section 5 describes related work. Finally a summary and outlook is made in section 6.

2 THE QUALITY CHECKING PROCESS

In this section, the idea proposed will be embedded into a defined procedure of continuous quality checking where all stakeholders can participate. Continuous quality checking means, that checking using queries starts as early as possible. This is visualized in figure 1.

According to initial requirements on the database **(1)**, a first draft schema is generated **(2)**. This is done by a designer. These initial requirements already cover some query needs. However, it cannot be expected, that these initial requirements already completely fulfil all the retrieval needs for the database.

Therefore, together with the other stakeholders, the additional retrieval needs are generated **(3)**. It is not necessary that the end users see the conceptual schema of the database when they generate the queries. Designers and end users are only talking what kind of retrievals they need. These needs are then expressed in controlled language queries. Each of the queries is then automatically executed on the schema using a query parser and interpreter **(4)**. Based on the reports of the parser **(5)**, a discussion of the schema is made **(6)**. In the reports for each query, the end users can see which parts of the schema are affected. If defects are found in the schema, new requirements can be derived from them **(7a)** and the schema is refined **(2).** The iterative process is finished if no more new requirements can be found and the schema becomes stable **(7b)**.

It is also a good practice that at least the designer checks which concepts in the schema were not affected by any of the query tests. Thus checking the quality of the schema together with the end user becomes an integrated part of the schema modelling process.

3 SCHEMA CHECKING WITH CONTROLLED LANGUAGE

A controlled language is used to prevent the ambiguities of pure natural language. It is a well established technique in many disciplines (e.g. ontology representation (Fuchs et.al, 2005) and is also used in query languages (Owei et.al, 1997) (Stratica et.al, 2005). Following the strategy of using a controlled language has also the advantage that no additional huge lexicons for words must be used or even generated. Instead, ***the conceptual schema itself is the lexicon***.

3.1 Provided Schema Information

The conceptual schema, which is checked by the queries, is based on a lean model. This model describes a graph of **concepts**. Classes and attributes are all concept nodes in the graph. Hence "*course*" as well as "*course name*" will become concepts in this model (see section 4, figure 2 for a schema). This idea was adopted from NIAM/ORM diagrams (Nijssen et.al, 1989). Especially during the early phase of modelling where changes often occur, such models have the advantage that schema changes can be made easily. Since the checking procedure is applied to classes and attributes, a mapping from concepts to classes and attributes and vice versa exists.

UML class diagrams are transformed in the following way to the model which is used here:

- Classes as well as attributes become concepts in the graph.
- The information if this concept is a class or attribute is kept since it will be used during the checking procedure.
- If an attribute belongs to a class, then the graph has an edge between them.
- If two classes are related to each other by a binary association, then this is represented as an edge between them in the graph.
- If classes are related by a n-ary association or an association class, then an artificial concept (node) is introduced in the graph. This artificial node represents either the n-ary association or the association class. Edges connect the artificial concept with the other involved concepts.
- If the attribute has already a related value type (e.g. "Integer", "String", "Boolean") this

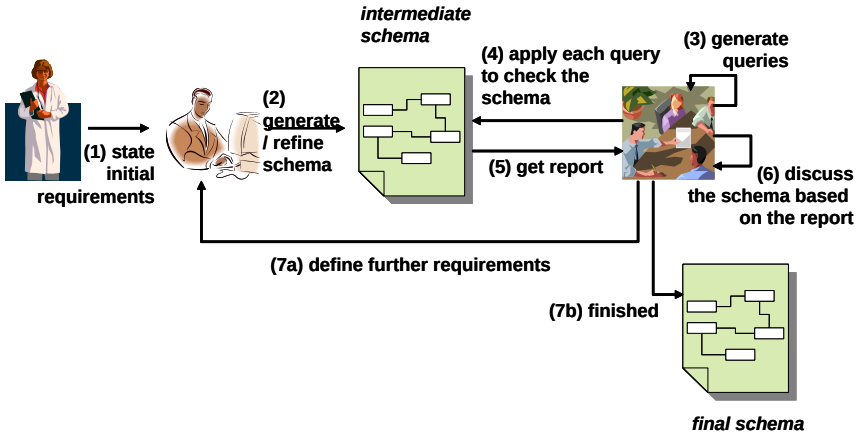

Figure 1: Continuous Conceptual Schema Quality Checking.

information is stored in the respective concept node for the attribute.
Additionally, the approach allows collecting and storing synonyms and examples of concepts in additional dictionaries.

3.2 The Requested Result

The idea here is, to use the queries mainly to detect defects and not to produce an SQL statement. Looking in detail on the restrictions between the concepts (classes, attributes) mentioned in a query and the concepts defined in the schema several types of defects can be found.

The following typical restrictions must hold between a query and the schema which is queried:

- If a concept (class, attribute) appears in the query then the concept itself or at least a similar concept must also appear in the schema.
- If a concept (especially a class) is found in a schema then it can be found only once in the schema.
- If two or more schema classes are involved in a query, then a path between these classes must exist.
- If there is a restriction clause in the query (i.e. WHERE attribute ….) of course the concept mentioned in the query must also be an attribute.
- If there is a restriction clause in the query (i.e. imagine the SQL WHERE Clause: WHERE attribute >= 100) then the value to which the attribute is restricted must be of the same type as it's value type in the schema.
- If the aggregate functions *sum*, *average*, *minimum*, *maximum* are defined on a concept, then the concept must be an attribute.
- If the aggregate functions *sum*, *average*, *minimum*, *maximum* are defined on an attribute, then the attribute must have a numeric value (numeric value type in the schema definition).

From that restrictions possible schema defects can be derived, using a controlled language query:

- Concept in the query cannot be found in the schema
- Query concept is found more then once in the schema
- Concepts are not related.
- Mismatch of concept type. A class is used where an attribute was assumed.
- Mismatch of the value type of an attribute (E.g. a string was defined in the schema but a number is expected according to the query).

It is recommended that the designer discusses the query as well as the schema with the end user if a schema defect is detected.

3.3 The Query Language

Sentences patterns are used to support the parsing of the query. Following the typical structure of a simple SQL query with it's SELECT clause, FROM clause, WHERE clause, ORDER BY clause, sentence patterns must exist for specifying the SELECT, WHERE and ORDER BY clause. Furthermore aggregation queries must be allowed. The FROM clause is not necessary. This information can be derived by finding a path between the concepts of the graph. Hence, the following kinds of sentence patterns must be considered:

- Sentences that reflect the SELECT clause of SQL (e.g. *List the first name, last name and address, show me …, give me …*).
- Sentences that reflect the WHERE clause (e.g. *The age must be greater than 32*).

- Aggregation queries: "*Count the number of students*" or "*Show the minimal price*"
- Sentences that reflect the ORDER BY clause (*"The result must be ordered by first name"*)
- Interrogatives like "*which*", "*how many*", "*How much*" are also part of the controlled language (e.g. *which student visits the course "xyz"* or *which students visit which course*).
- Because of the special usage of the queries also queries must be formulated which have the additional aspect to query the structure of the schema. Examples for such queries are: *Does a customer exist? Is the customer a person? Does a person have a car? Is a student connected with a course?*

These building blocks can be combined as different sentences in a query text.

3.4 Parsing and Schema Checking

The query parsing and schema checking procedure consists of five steps: ***query parsing***, ***concept matching***, ***concept type and value type checking***, ***path finding*** and ***report generation***.

During **query parsing,** the query text is divided into sentences and the sentences into tokens. Each sentence then is firstly scanned by the parser to classify the sentence pattern to which a query sentence belongs. After that, the parser parses that sentence according to the underlying syntax of the certain sentence pattern.

After sentence parsing, schema concept candidates are collected from the sentence pattern by the parser. For the tasks mentioned before, a special parser was implemented.

In the **concept matching** step, the selected concepts are matched with the schema. The best case is a match for identical names. That means, a concept in the query (e.g. „*first name*") also appears in the schema. If this cannot be achieved, the query interpreter tries to find a match on similarity. Similar concepts are either:

- Concepts found in the query which contain a schema concept (e.g. *graduate student* was found in the query and *student* appears in the schema) or
- Concepts found in the query which are only part of a schema concept (e.g. *member* was found in the query but the schema contains *university member*).
- Synonyms and similar words with a special meaning.

If the query concept is more specialized as described in the first case (e.g. *graduate student* vs. *student*), then collected examples are searched in a example dictionary to decide if the query concept is a valid schema concept. It is a usual design task, that concept examples are also collected during a conceptual schema design. This strategy is based on the idea, that a modifier like "*graduate*" can also either be an example of a concept itself or an example of the attributes of that concept (if the concept is a class).

If the query concept is more general like in the second case, then the number of schema concepts that would fit with that query concept is counted. Here the match is successful if only one schema concept was found. Otherwise the query was not well defined since an ambiguous concept was taken for the query.

If the query concept cannot be found in the schema using the strategies above, then the list of synonyms is scanned for the query concept. If the query notion is a synonym for a schema concept, then the synonym is replaced by the schema concept. A synonym check is done by searching in a synonym dictionary. It is also a usual conceptual design task, that synonyms of concepts found during design are collected.

The **concept type and value type check** provides additional possibilities to check the schema. Here the category of a schema concept (class vs. attribute) is compared with the location of the concept in the query. For example, if the query concept was found in an aggregation function (e.g. "*List the minimal price"*) or in a restriction sentence (e.g. "*The price must be ... "* ←→ "*WHERE price = ...* ") then this schema concept must be an attribute. For the aggregate functions *sum*, *minimal*, *maximal*, *average* the value of the attribute must be numeric. It is a further defect if it is not! An exception to this rule is the aggregation function which only counts something (e.g. *the number of cities*, *number of customer names* etc.)

Furthermore, if a value restricts an attribute in a restriction sentence ("*The age must be greater than 20*"), then the value must be compatible with the value type specified in the schema. A defect is detected if the value is not compatible. In the given example the value type specified in the schema for *age* must be a numeric type (e.g. *Number*, *Integer*…).

If attributes from different classes were mentioned in the query, then in the **path finding step** the interpreter tries to find a path between these classes. Therefore for each of the attributes stated in the query the corresponding class is determined. Afterwards the interpreter searches for a path

between these classes. This step is necessary since the user has expressed a query where a connection (join) between the classes is needed. Thus it has to be checked if the schema can provide this connection.

Finally, a report about the "success" of the query is generated and presented to the user (**report generation**). If at least one of the above checks were not successful, then the designer together with the end users must analyse the schema as well as the query. They must decide if something is missing in the schema.

4 THE PROTOTYPE

For a technical proof of concepts a prototype was extended. The prototype consists of two parts, an editor which displays the graph of concepts (see: figure 2) and the new feature of a query editor (see: figure 3). In the upper part of the query editor the query text can be either inserted or loaded from a file. In the text area at the bottom, the report is returned, after the query is executed. Following the process described in section 3 this could look as follows with the tool. Figure 2 shows a schema which might be derived from initial requirements in an university domain.

The discussion group consisting of end users and designer can now start to find queries that check if the built schema fulfils their needs. After the queries in the controlled language are defined by the discussion group, each of them is applied on the schema. Figure 3 shows the first query that might be generated by such a discussion group.

If all the query concepts as well as the path between them are found in the schema, then the involved classes derived from the query are marked in red colour together with the path between these concepts (see figure 4)[1]. The current version of the tool only visualizes the classes which are affected and the connection between these classes, since they represent the tables which will be joined in a future relational database schema. The attributes are not considered in this visualization. In the first query shown in figure 3 instead of "*type*", "*category*" was specified. If *category* is a defined synonym for *course type* then the synonym is replaced by the original schema concept (see figures 2 and 3).

[1] For a better readability in the print out, red coloured matches are additionally marked manually with a dotted line in the figure.

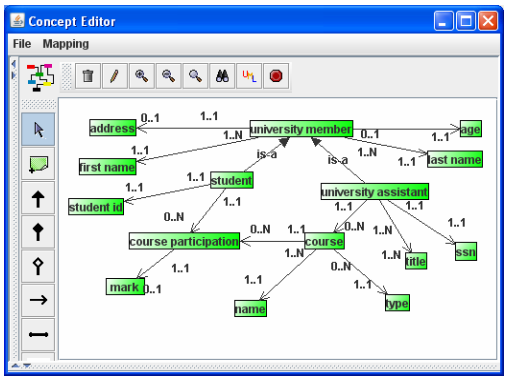

Figure 2: A simple university schema.

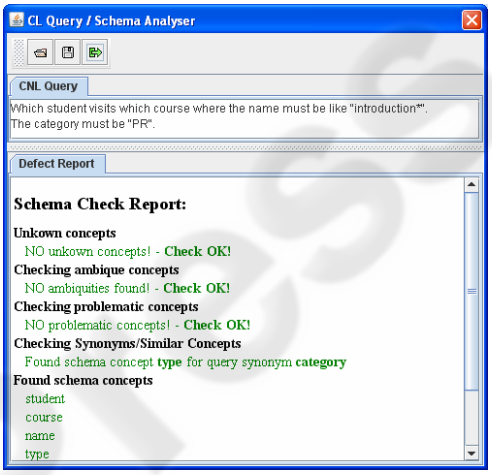

Figure 3: The query applied on the schema with the query editor/schema analyzer.

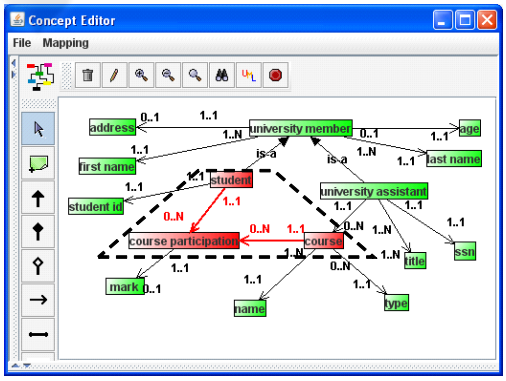

Figure 4: Result - found classes and the path – marked in red color.

However if one of the queries fails (see figure 5) then the defect report shows where it failed (see report message: "*Found concept course but course description cannot be resolved*"). Based on the results of the discussions, new requirements are

derived which are returned to the designer for a next schema refinement iteration

5 RELATED WORK

According to (Lindland et.al, 1994) three dimensions have to be considered for conceptual modelling quality, namely: syntax, semantics and pragmatics. If the schema follows the rules and the grammar defined in its corresponding model, then the schema has a syntactic quality. Semantic quality is given, if the schema only contains true statements of the domain and is complete (no important concepts or statements are missing). Lastly, pragmatic quality relates the schema to the interpretation of the user. A pragmatic quality of a schema is given, if it is understandable to the user. There is much research on this topic, either how to check the quality or how to improve the quality. However, no extensive research was found about (controlled) natural language queries and quality checking.

In order to improve the quality, in (Assenova et.al, 1996) it is proposed to use well defined schema transformations. Also a proposal to raise the comprehensibility of a conceptual schema by introducing icons and pictures instead of simple graphical primitives (rectangles, ellipses etc.) are introduced in (Moody, 1996). An exploratory study showed that view points are also an important technique to improve the quality of the schema (Easterbrook et.al., 2005).

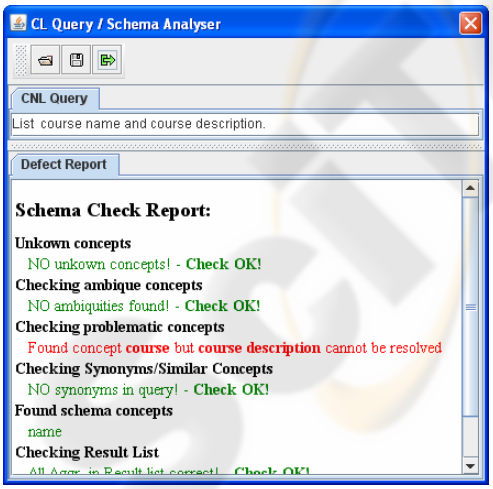

Figure 5: Second query.

The verbalization approaches in (Dalianis, 1992), (Halpin et.al, 2005) aim at getting a better understanding of the conceptual schema by presenting users a natural language translation (verbalization) of the schema.

In (Moody, 2005) an extensive research of literature on the quality of models was made. That author found out that some important factors are missing and he concluded that conceptual modelling must shift from an art to an engineering discipline were quality plays an important role. Any engineering discipline nowadays aims at continuous quality checks of products and intermediate products. Quality is not checked at the end of a process but throughout the process. Thus the approach described here is thought to be such a strategy using controlled language queries for continuous schema checking.

Since queries are based on natural language, another important topic for this work is the research on user centred query languages. User centred query language approaches all aim to query the schema with a technique that is more suitable to the user (e.g. natural language based queries and visual query languages)

Techniques of natural language querying are described in (Berger et.al, 2003), (Hofstede et.al, 1996), (Kardovácz, 2005), (Kapetainos et.al, 2005), (Kao et.al., 1988), (Owei et.al. 1997), (Stratica et.al., 2005). Some operate on a relational schema, others on the conceptual schema. Some uses additional information derived from linguistic lexicons or ontologies. However, the main objective of all is to support the generation of a SQL query that can be executed on the relational schema. In addition it is recommended, that the schema is already complete and consistent. It is thus not the task of the query to validate if something in the schema is missing.

This also holds for visual query tools described in (Bloesch et.al., 1996), (Jaakola, 2003), (Järvelin et.al., 2000) which operate on the conceptual schema. Their main purpose is to produce a good SQL statement. It is not their purpose to check the schema, since the schema must be already stable. Beside these, also form based query languages are used to generate queries (Embley, 1989) and to use them for a better understanding of the data in a database (Terwillinger et.al., 2007).

To summarize the related work: A lot of research results were proposed in the topic of quality of conceptual modelling. The approach described here, is aimed to complement the previous work. It is seen as an additional support for checking the quality of a schema during requirements- and software engineering. Furthermore it is not expected that the complete and stable schema is already given. This approach must be applied during the

construction of a schema as early as possible and then during each refinement iteration step. Therefore the focus is not a generation of SQL but on checking the quality of the schema. It is thought as a help, if it is necessary to check the schema, before any prototype or user interface form can be built. Opposite to the graphical query languages where the query is constructed by navigating through the schema, the end user must not necessarily be confronted with the schema during creation of the queries. This has the advantage that the user is not influenced by the schema but freely names the notions which he needs for the query. The query then helps to check if the designed schema can handle the notions used in the query.

6 SUMMARY

This work uses controlled language queries for continuous schema quality checking. Each query is applied on the schema in order to find defects which can be a basis for discussion and further refinement of the schema.

This strategy was chosen to give both developers and end users the possibility to communicate about the end users retrieval needs. It was of interest how far such a language can be constructed without huge linguistic lexicons. Instead only information which is necessary to define a good schema and information that could support the development of data intensive software itself was allowed.

In future, the approach might be extended to transform column names of an EXCEL sheet into a query. The query language itself might be extended to formulas.

REFERENCES

Assenova, P. Johannesson, P., 1996. Improving the Quality in Conceptual Modelling by the Use of Schema Transformations. In Thalheim, B. (ed.): *Proceedings of the 15th International Conference on Conceptual Modeling, Cottbus, Germany*, Lecture Notes in Computer Science (LNCS), Vol. 1157. Springer Verlag Berlin Heidelberg New York, 1996, pp. 277 – 291.

Berger, H., Dittenbach, M., Merkl D., 2003. Quering Tourism Information Systems in Natural Language, In Godlevsky, M., Liddle, St., Mayr, H.C. (eds). *Informaton Systems Technology and its Applications – Proceedings of the 2nd Conference ISTA 2003*, GI Lecture Notes in Informatics, Vol. p-30, Koellen Verlag,, Bonn, 2003, pp. 153 – 165.

Bloesch, A.C., Halpin, T.A., 1996. ConQuer: A Conceptual Query Language. In Thalheim, B. (ed.): *Proceedings of the 15th International Conference on Conceptual Modeling*, Cottbus, Germany, Lecture Notes in Computer Science (LNCS), Vol. 1157. Springer Verlag Berlin Heidelberg New York, 1996, pp. 121 – 133.

Dalianis, H., 1992, A method for validating a conceptual model by natural language discourse generation. In P. Loucopoulos (Eds.), *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Enginering*. Lecture Notes in Computer Sciences (LNCS) Vol. 594, Springer Verlag, 1992, pp. 425-444.

Embley, D.W., 1989. NFQL: The Natural Forms Query Language. In *ACM Transactions on Database Systems*, Vol. 14. No. 2., 1989, pp. 168 – 211.

Easterbrook St., Yu, E., Aranda, J. Fan, Y. Horkoff, J. Leica, M., Quadir, R.A., 2005, Do Viewpoints Lead to Better Conceptual Models? An Exporatory Case Study. In *Proceedings of the 13th IEEE Conferences n Requirements Engineering (RE'05)*. IEEE Press, 2005, pp. 199 – 208

Fuchs, N.E., Höfler, S., Kaljurand, K., Rinaldi, F., Schneider G., 2005. Attempto Controlled English: A Knowledge Representation Language Readable by Humans and Machines. In Norbert Eisinger N. and Maluszynski, J. (eds.): *Reasoning Web, First International Summer School 2005*, Lecture Notes in Computer Science (LNCS)mVol. 3564, Springer Verlag, 2005 pp. 213-250.

Halpin, T., Curland, M., 2006. Automated Verbalization for ORM 2. In *Proceedings, OTM 2006 Workshops - On the Move to Meaningful Internet Systems 2006*, Lecture Notes in Computer Science (LNCS 4278), Springer Verlag, 2006, pp. 1181 – 1190.

ter Hofstede, A.H.M., Proper, H.A., van der Weide, Th.P., 1996. Exploring Fact Verbalizations for Conceptual Query Formulation. In. van de Riet, R.P., Burg, J.F.M., van der Vos, A.J. *Proceedings of the Second International Workshop on Applications of Natural Language to Information Systems*, IOS Press, Amsterdam, Oxford, Tokyo, 1996, pp. 40 – 51.

Jaakkola, H., Thalheim, B., 2003. Visual SQL – High Quality ER Based Query Treatment", In. Jeusfeld, M., Pastor, O. (eds.) *Conceptual Modelling for Novel Application Domains*, Lecture Notes in Computer Science (LNCS), Vol. 2814, Springer Verlag, Berlin, Heidelberg, New York, 2003, pp. 129 – 139.

Järvelin, K., Niemi, T., Salminen, A., 2000. "The visual query language CQL for transitive and relational computation". In *Data & Knowledge Engineering*, Vol. 35, 2000, pp. 39 – 51.

Kardovácz, Z.T., 2005. On the Transformation of Sentences with Genetive Relations to SQL Queries. In. Montoyo, A., Munoz, R., Metais, E. (eds.) *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005)*, Lecture Notes in Computer Science (LNCS), Vol. 3531, 2005, pp. 10 – 20.

Kapetainos, E, Baer, D, Groenewoud P., 2005. Simplifying syntactic and semantic parsing of NL-based queries in adavanced application domains. In *Data & Knowledge Engineering Journal* , Vol. 55, 2005, pp. 38 – 58.

Kao, M.; Cercone, N.; Luk, W.-S., 1988. Providing quality responses with natural language interfaces: the null value problem". In *IEEE Transactions on Software Engineering*, Volume 14 (7), 1988, pp. 959 – 984.

Lindland, O., Sindre, G., Solvberg, A..: Understanding Quality in Conceptual Modeling, *IEEE Software*, March 1994, pp. 29 – 42.

Moody, D., 1996. Graphical Entity Relationship Models: Towards a More User Understandable Representation of Data. In Thalheim, B. (Ed.): *Proceedings of the 15th International Conference on Conceptual Modelling*, Cottbus, Germany, Lecture Notes in Computer Science (LNCS), Vol. 1157. Springer Verlag Berlin Heidelberg New York, 1996, pp. 227 – 245.

Moody, D., 2005. Theoretical and practical issues in evaluating quality of conceptual models: current state and future directions. In *Data & Knowledge Engineering Volume 55*, 2005, pp. 243 - 276

Nijssen, G.M., Halpin, T.A., 1989. *Conceptual Schema and Relational Database Design - A fact oriented approach*, Prentice Hall Publishing. Company. 1989.

Owei, V, Rhee, H-S., Navathe, Sh., 1997, Natural Language Query Filtration in the Conceptual Query Language. In *Proceedings of the 30th Hawaii International Conference on System Science*, Vol. 3. IEEE Press, 1997, pp. 539 – 550.

Stratica, N., Kosseim, L., Desai, B.C., 2005. Using semantic templates for a natural language interface to the CINDI virtual library. In D*ata & Knowledge Engineering*, Vol 55, 2005, pp. 4 – 19.

Terwillinger, J.F., Delcambre, L.M., Logan, J., 2007. Queyring through a user interface. In *Data & Knowledge Engineering*, Vol. 63, 2007, 774 – 794.