

Language Aspects of Conceptual Database Design

Peter Bellström, Sten Carlsson

Department of Information Systems

Faculty of Economic Sciences, Communication and IT, Karlstad University

S-651 88 Karlstad, Sweden

Email: Peter.Bellstrom@kau.se, Sten.Carlsson@kau.se

Abstract

We view the database as a part of a living language and therefore put focus on the dialectical relation between the spoken language and a database represented as a conceptual database schema. We assert that the discrepancy between the spoken language and a database needs to be addressed and minimized. We also assert that all concept names need to be retained as long as possible in conceptual database design because otherwise we may end up with a state of concept name compression and semantic loss.

Keywords: Language, Conceptual Database Design, View Integration, Semantic Loss, Understandability

1 Introduction

An information system means that someone is informed by something of the information of the system (Goldkuhl 1980). A database is also often used to store this information, but not as whole messages, but as conceptual pieces of the information of the system. As whole messages of information cannot be stored in databases, the conceptual pieces have to be put together in ways, which are predefined by the users of the system. The database management system (DBMS) then is putting these messages together according to the programmed algorithms for producing this information. Being a part of the communication between people therefore the database should be seen as a living language and not as a collection of data in a word book. So, even if the information in a database is reduced into pieces because of technical reasons this, however, does not give us the reason not bothering about the problems of interpretation of the database according to this reduction. The database is still a part of a living language and should always be treated like that.

Viewing the database as a part of a living language makes it even more important to address the dialectical relation between the language and the database as early as possible during the database design process. Therefore the aim of this paper is as follows. Firstly, we point out some ontological and epistemological prerequisites in order to handle the questions of language in relation to databases. Secondly, we put focus on *conceptual database design* which is the first and most demanding (Engels et al. 1992), and critical (Batini et al. 1992; Dey et al. 1999) phase in database design. One of the big challenges during conceptual database design is to create a global conceptual schema that is *semantically correct, complete and easy to use* (Shoval & Shiran 1997) using the primitives and constructs of the chosen modeling language (Storey et al. 1995). However, we also emphasise that at the same time we need to keep up the possibilities to pay regard to making the meaning of the concepts fully understandable. One way to define a conceptual database schema is first to involve the end users and define views for each end user or user group and secondly to integrate all views into one final and global conceptual schema during a task called view integration. Nevertheless, traditionally the applied methods and models do not fully help us to interpret and understand the meaning of the used concept names during the conceptual database design process. In the end this could instead result in a global and final conceptual schema that is *semantically incorrect, incomplete or hard to use*. The nature of the technical restrictions of databases also

prevents mutual interpretation because of the endeavour of minimizing the number of concepts and relationships between the concepts.

This paper is organized as follows. In chapter two we present our ontological and epistemological aspects of language. In chapter three we describe the five phases of the view integration process and in chapter four we both describe and discuss the problem of concept name compression and the line of mutual interpretation. Finally, in chapter five we summarize the paper with conclusions and a short discussion.

2 A Linguistic Comparison between an Utterance and the Contents in a Database

The aim of making this comparison between an utterance and the contents in a database is to make clear what is missing concerning an expression in a database to be fully understandable as a linguistic conception. Why is this so important? If we assume that a database is living language we would be able to arrange our design of databases much better, so we can simplify the interpretation of the messages of the database. This is applicable both during the design of the views and schema for the database and the reconstruction of their meaning afterwards (Bellström & Carlsson 2004), if there are any problems or misunderstanding of these messages. Maybe, somebody thinks that this is to overdo the difficulties of interpreting a message printed out by means of the computer. We can agree on that, if you are talking about an invoice. But how is it for instance about a statistical message? If you get a message from your boss with a lot of statistical figures about the sales in the company, he/she maybe expects you to act according to what you can find out by means of the figures. Now the interpretation of the message is not quite so simple compared with that one of the invoice.

The idea regarding communication as communicative actions, when analysing information systems, is not new at all. During the 1980's Goldkuhl and Lyytinen wrote several papers about specification of information systems based on the language action view (Goldkuhl & Lyytinen 1982a, 1982b, 1983). The Human-infological research group with the leadership by Göran Goldkuhl together with his colleagues forced this question very strongly. Therefore we do not write this paper in order to blow in open doors. Instead we do this to remind the information system world especially that one concerning the object-oriented modelling world, that we cannot make a reduction of the communication in information system into object as pieces, claiming that it is natural to understand the world as objects. This is a qualified nonsense (Carlsson & Christiansson 1999). This understanding we constitute as actions in the world (Schutz & Luckmann 1974). Let us now start with the comparison.

2.1 An Utterance as an Event of Meaning

What are the prerequisites to fully interpret an utterance? According to Ricoeur (1976) the utterance or the discourse is the event of language. It is the utterance which gives the language its reliance in time, context and coupling to a subject, who is saying something. This does not concern language as a system. An utterance is personal, a language is not. You can of course use a language whenever you want, but when you do it; you are making an utterance as an event of meaning. Someone can tell you, that you had said something yesterday and ask you what you meant by saying it. And you cannot deny that you have uttered it, if you of course do not tell a lie. Accordingly we assert that the contents in a database also are speech events.

When you get your invoice there is a date of transcription marked on it. You get it from a company as an event of meaning. The invoice therefore is an utterance from the company.

The database, on the other hand, contains the language, which is producing the utterance from the company to the customer.

Another important remark about an utterance is how small an utterance as a speech event can be. In systems development, according to Langefors (1995), the concept information component is used as the smallest piece of information when designing databases. A customer's name and address can be such a concept of information. This is however not the same as an utterance as a speech event. According to Ricoeur (1976) the smallest piece of speech event is a subject and a predicate. By a predicate he means something that "designates a kind of quality, a class of things, a type of relation, or a type of action" (ibid. p. 11). The function of the subject is to make the universal predicate more un-universal by identifying the speaker of the message. That's the only way according to Ricoeur to be able to interpret the meaning. And this interpretation is not possible to do by analysing the structure of the sentence of the utterance. Ricoeur puts it like this. The possibility to interpret the utterance is due to the reveal of the intertwining interplay between the identification as a subject and what the subject means with the uttered predicate. That is a question between the dialectic of utterance as an event and its meaning (ibid.).

A special case of a speech event is that what is printed out by means of a computer. When you say something you often just do it like a comment. On the contrary a computer based speech event mostly concerns something more. The sender of these types of events also expects some actions from the receiver. Invoices do not just mean that you tell the customers that they owe the company money. They are also expected to act and pay.

Considering the idea of the action mentioned earlier, conceptual database design does not take into account the aim of the concept information component. It is not mentioned in relation with that component. If we are going to understand an action, we have to know the aim of it and where it starts and begins (Schutz & Luckmann 1974).

Summing up the differences and similarities so far, an utterance and an output like an invoice are both speech events. They might be confirmed in a way according to time, a sender and a context. The smallest piece of a speech event has a subject and a predicate and when designing a conceptual view or schema the information component is that smallest piece. But in this case the subject is missing in the view or schema. Even if you can track the sender of an invoice and other messages from the computer by means of something printed on the paper, the subject of the speech event documented in a conceptual view or schema is not available in that. The aim of the language action is not either notified in the conceptual view or schema.

2.2 The Dialectic Relation between the Utterance as an Event and the Meaning of it

In order to understand the meaning of a speech event we have to, as Ricoeur puts it, reveal the intertwining interplay between the identification as a subject and what the subject means with the uttered predicate. The utterance as an event, the objective side of the event as we hear or can see, is, as said before, not interpretable by means of the structure. It is the synthetic meaning of the utterance we have to understand. What are the prerequisites for that?

When somebody is saying something it is the utterer's meaning. This utterance is a self-reference to the subject i.e. what the speaker intends to say, which is a new, so to say, criterion of the difference between an utterance and a linguistic code (Ricoeur 1976). People speak, a language does not.

An utterance as a self-reference also implies an obligation of what the speaker is saying to do. The performatives are concepts like promises, commands, wishes, questions or assertions. When you promise something you put yourself under obligation to do what you promised. These concepts tell us that the utterance is addressed to someone. Their functions are to knit some sort of relation between the speaker and the listener in a dialogue. It was Austin and Searle who according to the speech-act theory launched the linguistic terms the locutionary act, the illocutionary and the perlocutionary act. These terms means according to Ricoeur (ibid. p 14), “saying something (the locutionary act), do something in saying (the illocutionary act), and yield effects by saying (the perlocutionary act)”. The illocutionary act has also another dimension in the relation between the speaker and the listener. This act also put some sort of question to the listener. If you performatively are wishing something from somebody the listener has to agree upon what you wish from him or her to do. Otherwise the yielded effects will not occur. Of course the listener also has to understand what is meant by the wish. Therefore there has to be a mutual understanding about both what your wish is about, that is to say the prepositional content, and what the illocutionary and the perlocutionary acts means according to the corresponding intention of what is wished. All these three acts are dialectically intertwined with each other in the utterance. They constitute as Ricoeur puts it, some sort of psychological condition of the speech act. So the question now is: how can the speaker and the listener understand each other?

The answer to the question is the open dialogue. It is by means of the dialogue all the intentions of the locutionary act, the illocutionary act and the perlocutionary act might be understandable. As “the intention of saying is itself communicable to a certain extent” (Ricoeur 1976, p. 18), this implies that these intentions has to be recognized of all the participants in the dialogue (ibid.). This dialogue should be characterized as dialogism and not as monogolism (Linell 1994). The listener cannot be just a listener: He or she must be an active participant in the dialogue putting questions and making statements if something has to be explained in order to be able to identify, recognize and acknowledge the intentions.

Summing up the differences and similarities between the utterances in this part we firstly can notice that the utterances and the output from the database are the objective sides of speech events. We can also say that the contents of this output correspond to the locutionary act according to speech act theory. The other acts, the illocutionary act and the perlocutionary act, are however missing in the views or schema when we are designing the database and mostly also on the outputs from the database. On an invoice you can find sentences which represent some sort of illocutionary act and perlocutionary act, informing that the company wants you to pay the amount within 30 days. Other outputs do not contain anything concerning these two acts. Goldkuhl & Lyytinen brought up these ideas in analysis of information systems already did the 1980’s. Therefore we want to remind database designers that a better understanding and development of databases might take into account a better understanding of what makes an utterance and a language action more understandable. Therefore we should document all these aspects during the design in order to enlightenment to the meaning of the contents of a database.

When we are using some output from the database as an utterance we mostly do not do this in the dialogue with subject of this utterance. These dialogues are performed instead when the conceptual views and schema are designed. This emphasizes the importance of mutual dialogues during this part of the design of databases.

2.3 The Meaning of the Utterance and its Reference

A speech event concerns what we are talking about. Our talking is directed against something. The words in an utterance do not refer to other words in a language system or a word book. They are directed beyond themselves in the world (Ricouer 1976). It is the reference which relates the utterance to the world (ibid.). Language helps us to orient ourselves comprehensively according to different daily life situations. It is there we bring experience to language (ibid.). How is it about the reference concerning the output from a database? When we for instance get an invoice from a company this company has uttered a message that you have to pay for something you have bought from it. To be able to agree with this you have to agree with the company about the reference what it is all about; for instance an article with its price. So, when we get an output extracted from a database it has also a reference to something. Therefore there is accordance with an utterance and an output from a database. They refer to something in the world. In this case we also should notify the conceptions in our views or schema what they refer to. Especially this is important when we in database design compress different synonyms into one concept. After the compression one concept therefore might refer to several different contexts using this concept in different ways.

2.4 Some Finishing Conclusions of this Part

The comparison analysis between an utterance and the contents in a database has made it clear that some speech aspects might be missing in documentation of the analysis of the database, aspects, which in turn are necessary to fully interpret an utterance. So, if we assume that a database is a part of a living language, these speech aspects have to be documented in order to make it possible to simplify the interpretation of the concepts in a database. Further more this documentation might even simplify the reconstruction of the concepts contextual origin. These aspects are such as the subject of the message, the aim of the speech action, the illocutionary and perlocutionary acts and the references. A dialogue as we asserted above is the key for a better understanding of an utterance. But these dialogues are normally carried out when we design the database. This underlines the importance that the end-users of the database have to take part in the design dialogues in order to get a better understanding of the concepts in the database. It is during these dialogues the end-user has the possibilities to recognize that meaning.

3 Conceptual Database Design

Conceptual database design is the first of three phases in database design, logical and physical are the other two (e.g. Connolly & Begg 2005; Elmasri & Navathe 2004). As been told in the introduction in this paper we put focus on conceptual database design henceforward defined as “the process of constructing a high-level and implementation-independent description, a conceptual schema, of the structure of the database, based on the information used in the enterprise.” (Bellström 2006a, p. 6). Besides that we also view conceptual database design as divided into two distinct parts: *view design (modeling)* and *view integration* and in this section we put focus on the second part: view integration. In view design, views for each end user or user group are defined and graphically illustrated using a modeling language such as the Entity-Relationship (ER) modeling language (Chen 1976), the Unified Modeling Language (UML) (e.g. Martin & Odell 1998) or Enterprise Modeling (EM) (e.g. Gustas & Gustiené 2004). In view integration the views are integrated into one final and global conceptual schema. Batini et al. (1986) define schema integration, the origin of view integration, as “[...] the activity of integrating the schemas of existing or proposed databases into a global, unified schema.” (p. 323).

Dividing conceptual database design into the two previously described phases is important because the views preserve and highlight differences between the end users views of the organization while a global schema may instead mask these (Parsons 2002). Similar reasoning to apply a two-phase design approach is given in Nuseibeh et al. (2001) where the authors point out that sometimes it is even desirable or necessary to tolerate inconsistency in and between different views and descriptions which may not only prevent premature design decisions but also ensure that all end users views of the organization are taken into account. Finally, this two-phase approach is also one way to manage the complexity of designing large databases (Mannino 2006).

A conceptual view or schema may also have many roles (e.g. Olivé 2004) however in this paper we mainly treat a view or schema as a communication tool between end users and designers (e.g. Ambrosio et al. 1997; Hoppenbrouwers et al. 2005). This is motivated since we first of all emphasize the importance of addressing the language and use of concept names during conceptual database design.

3.1 The View Integration Process

In the study performed by Batini et al. (1986) the authors concluded that a view integration process is composed of, or at least is a mixture of, the following four phases: *pre-integration*, *comparison of the views*, *conforming the views* and, *merging and restructuring*. In Bellström (2006b) the author concluded that one phase was still missing. *Pre-conforming the views* was therefore proposed and incorporated as a standard phase into the view integration process. In Figure 1 the view integration process is illustrated and it is shown that the process is iterative moving back and forth between the five phases. The arrows moving from left to right illustrate feed-forward and the arrows from right to left feed-back.

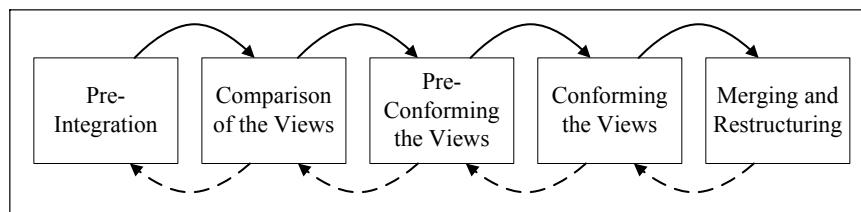


Figure 1. View Integration Process as described in Bellström (2006b p. 197)

As also Figure 1 illustrates the first phase in the view integration process is *pre-integration*. Song (1995) emphasize that pre-integration has three main tasks that should be carried out. The first is to translate all views into a canonical modeling language, the second is to check for conflicts and inconsistencies in each view, and the third is to select integration strategies.

The second phase that should be applied is *comparison of the views*. The main task to perform in this phase is to compare the views aiming to identify not only similarities but also differences between the views. According to Johannesson (1993), comparison of the views is divided into three main activities: 1. identification of name conflicts, 2. identification of structural conflicts, and 3. identification of inter-schema properties.

The third phase that should be applied is *pre-conforming the views*. The main task in this phase is to simplify not only conflicts but also inter-schema properties using inference rules to deduce new dependencies from conflicting dependencies or where an inter-schema property exists between two concepts (Bellström 2006b).

The fourth phase that should be applied is *conforming the views*. The main task in this phase is to resolve conflicts and inter-schema properties identified in the second phase. This means that not only conflicts such as name and structural conflict but also inter-schema properties such as generalization hierarchies (Batini et al. 1992) are addressed.

The fifth and last phase that should be applied is *merging and restructuring*. The main task in this phase is to merge all the views into one global schema and to conduct restructuring on the global schema. Restructuring has to be carried out since several redundant dependencies and concepts may exist after the views have been merged. It is also very important that restructuring is carried out carefully and without semantic loss (Spaccapietra & Parent 1994) which otherwise could change the meaning of some part of the schema.

In the rest of this section we briefly describe the most ordinary resolution techniques for name conflicts, *homonyms* and *synonyms*, and the inter-schema *hypernym-hyponym* property also known as generalization-specialization. At the same time we indicate problems with these resolution techniques in connection with the language, concept names, used in the organization and in the designed conceptual views and schema. A homonym occurs if one concept name is used for several concepts, a synonym occurs if several concept names are used for one concept and finally a hypernym-hyponym property occurs if one concept name is defined as a generalization of another concept name which at the same time is defined as a specialization of the other concept name. The motivation for describing the three problems is that these may all be when trying to resolve them at high risk to end up in a state of concept name compression.

The most ordinary resolution technique for name conflicts is simple renaming (e.g. Batini et al. 1986; Johannesson 1993; Lee & Ling 2003). This means that one or several concept names are changed. However, how concept name renaming should be carried out is unclear and renaming is therefore a resolution technique that is vague and may cause a state of concept name compression which may result in language impoverishment, semantic loss and interpretation problems. For the homonym conflict other resolution techniques such as prefixing the concept names (Parent & Spaccapietra 1998) and standardization of names (Lawrence & Barker 2001) have also been proposed. Nevertheless, these resolution techniques may still cause a state of name compression and semantic loss. An illustrating description of the concept name problems is given in Martin & Odell (1998) as follows: "If concept synonyms are headache for data administrators, homonyms are migraine material." (p. 20)

Finally, the hypernym-hyponym property is addressed. The hypernym-hyponym property has been given many names such as inter-schema property (e.g. Batini et al. 1992; Johannesson 1993) and semantic conflict (Spaccapietra & Parent 1991) and it is mainly resolved by introducing a subset or a generalization hierarchy into the view or schema (e.g. Batini et al. 1992; Johannesson 1993). The term cyclic generalization is in Song (1995) used in connection with concept names and the two relationships: *B is-a A* and *A is-a B*. However, a cyclic generalization may often indicate a synonym conflict instead of a hypernym-hyponym property.

4 Concept Name Compression and the Line of Mutual Interpretation

In traditional conceptual database design methods the designers try to resolve many of the problems connected with the language used in the organization and the views by minimizing the number of concept names and dependencies in the conceptual schema. Minimization of

concept names is even more obvious in the view integration approach because then the end users different ways of naming and defining concepts are focused and integrated. In this section we therefore discuss and describe resolution techniques, to the problems described in the previous section, that instead of minimizing the concept names retain these as long as possible during the view integration process. We also describe and discuss a phenomenon we call *the line of mutual interpretation*. The goal of that discussion is to emphasize and illustrate consequences of applying the traditional resolution techniques compared with our proposed resolution techniques in a perspective that focuses not only on how to counteract impoverishment of the language used in the views and schema but also how to simplify the interpretation and understanding of the views and schema.

4.1 Concept Name Compression

The problem of concept name compression was first described, discussed and criticized in Bellström & Carlsson (2004) and later on defined in Bellström (2006a) as “a state which may occur if several concept names are merged (compressed) into one concept name, for example when choosing one of the concept names to represent a concept when trying to resolve a synonym conflict.” (p. 46).

However, a state of concept name compression may also occur after resolving other conflicts and inter-schema properties. We therefore extend the discussion of the problem to include resolution techniques for homonyms and the hypernym–hyponym property.

As been told earlier the most ordinary resolution technique for synonyms and homonyms is simple renaming which means that one or several concept names are renamed. However, how renaming should be carried out is seldom stated or explained. Although, in Mannino (2006) the rename procedure is explained a little bit deeper as “[...] rename synonyms the same (or establish an official list of synonyms) and rename homonyms differently.” (p. 442). In this paper we instead assert the importance that both synonyms and homonyms should be resolved without losing any of the used concept names. In Figure 2 our resolution approach to the synonym conflict is illustrated and it is shown that all concept names are retained even after the synonyms have been resolved. Treating synonyms that way is also in agreement with the view on standardization of concept names that Martin and Odell (1998) have and explain as follows: “While standardization is useful, enforcing the one-and-only-one name is impractical in all situations.” (p. 19).

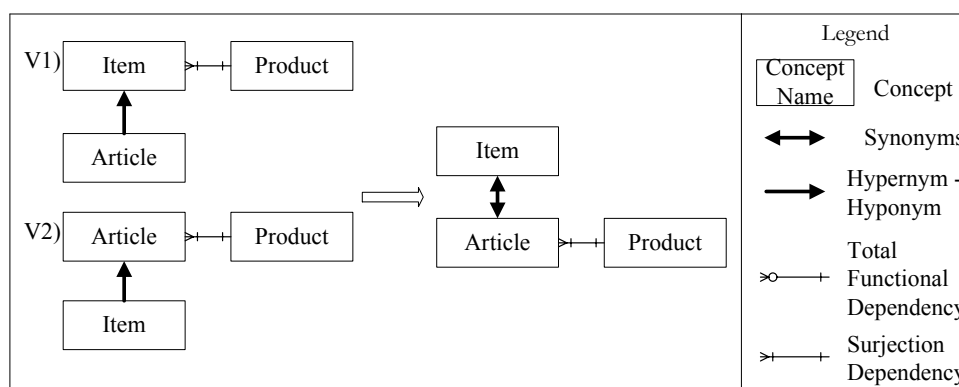


Figure 2. Synonyms before and after resolution.

In Figure 2-Figure 4 the abbreviations V1 and V2 are used. These stand for “view one” and “view two” which are views prepared for integration.

In Song (1995) a name conflict is instead called a semantic conflict and includes not only synonyms and homonyms but also cyclic generalization. However, cyclic generalization is just another example and description of the synonym conflict and should therefore be resolved by applying the mutual inheritance dependency defined as: “A $\leftarrow\rightarrow$ B if and only if A \rightarrow B, B \rightarrow A” (Gustas 1997, p. 182) which is graphically illustrated in Figure 2.

The importance of resolving synonym conflicts without losing any concept names by applying the mutual inheritance dependency has also been stressed in Bellström & Jakobsson, (2006) and Jakobsson & Bellström (2006). In these publications the authors illustrate and point out when and how to apply the mutual inheritance dependency not only for concept names in pure static views and schema but also for concept names and states in dynamic views and schema.

A state of concept name compression may also be the result after trying to resolve homonyms. This is the case since renaming homonyms differently without thinking of the consequences might actually result in several interpretations for one concept name. We instead emphasize that a more detailed resolution technique that not only retain all concepts names but also improve the semantic quality should be applied. In Figure 3 that resolution technique is illustrated and it is shown that instead of just renaming the concept names the context is included in the concept name. The primitives used in Figure 3 are described in Figure 2.

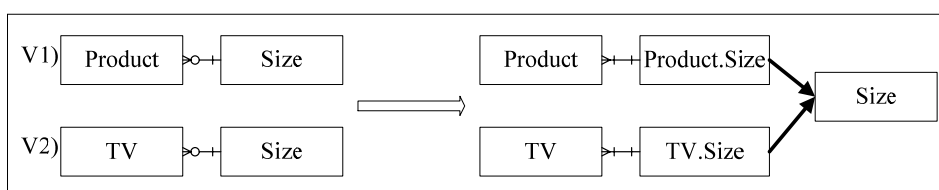


Figure 3. Homonyms before and after resolution (Bellström 2006b, p. 189)

When applying the proposed resolution technique for homonyms an additional concept is created. The concept name for the new introduced concept is composed of both the names of the two involved concepts with a dot between them. The dependencies between the concepts also change. The homonym resolution technique is in agreement with how homonyms should be resolved according to Martin & Odell (1998) whom express it as “[...] we must support the homonym phenomenon in ways that can be understood by humans—through context.” (p. 20). Applying the previously described resolution technique means that all concept names are retained even after resolving the homonyms. This is achieved by introducing a new concept and concept name a resolution technique emphasized in Bellström (2005)

The problem of concept name compression is something we also have to be aware of when dealing with the inter-schema hypernym–hyponym property. Inter-schema properties are not really conflicts; instead they express when two concepts are related to each other by certain constraints (e.g. Batini et al. 1992; Johannesson 1993) such as in our case a hypernym–hyponym dependency. At first sight a hypernym–hyponym property might appear to be a synonym conflict but when analyzed a bit closer differences between the concept names can be identified and these should therefore be treated differently. We therefore assert that this type of property should be resolved by introducing a hypernym–hyponym dependency between the two concept names as illustrated in Figure 4. Identification and resolution of inter-schema hypernym–hyponym properties have also been analyzed in Bellström et al.

(2006) where the resolution technique illustrated and described in Figure 4 is emphasized. The primitives used in Figure 4 are described in Figure 2.

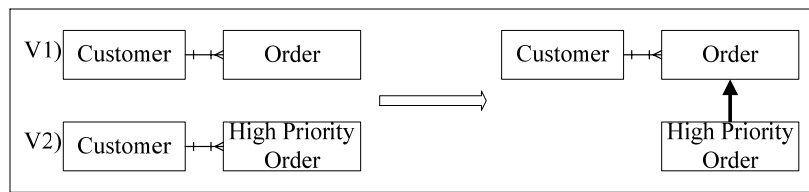


Figure 4. Hypernym-hyponym property before and after resolution.

Applying the resolution technique described and illustrated in Figure 4 means that all concept names are retained even after resolving the inter-schema hypernym–hyponym property. In our case the concept name *Order* is a more general concept name (hypernym) than *High Priority Order* which at the same time is a more specialized concept name (hyponym) than *Order*.

Summing up, applying the traditional resolution techniques for synonyms, homonyms and the inter-schema hypernym–hyponym property may result in a state of concept name compression. However, applying the proposed resolution techniques may instead counteract that the language used in the views may become impoverished when the views are integrated into one global schema. In the next section we therefore introduce the line of mutual interpretation and, discuss and describe what that line means for resolution of conflicts and inter-schema properties in the view integration process.

4.2 Maximizing the Line of Mutual Interpretation

In section three we mentioned that a conceptual view or schema may have many roles in conceptual database design and that we mainly treat them as a communication tool between designers and end users. It is therefore important to design and integrate the views into one global conceptual schema without any loss of concept names which also was illustrated and emphasized in the pervious section. In this section we discuss and describe what loss of concept names and concept name compression means for the interpretability and understandability of the concept names used in the views and schema during the whole view integration process. Our discussion regarding the problem on how to maximize the line of mutual interpretation is conducted describing Figure 5.

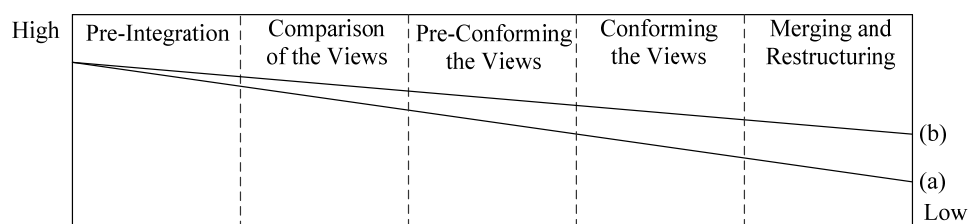


Figure 5. The line of mutual interpretation with traditional resolution techniques (a) and with proposed resolution techniques (b).

The view integration process is composed of five phases (see section 3.1) starting with pre-integration and ending with merging and restructuring. When the designers and the end users enter the pre-integration phase they have several views that overlap and therefore should be integrated into one conceptual schema. However, when comparing the views the designers often identify not only semantic heterogeneity (e.g. Song 1995) but also semantic equivalence

(e.g. Sheth 1991) between the views. This means that both conflicts and inter-schema properties need to be resolved or at least simplified. As pointed out earlier, applying one of the traditional resolution techniques for synonyms, homonyms or the inter-schema hypernym–hyponym property may not only impoverish the language used in the views but also end up in a state of concept name compression. The loss of concept names may also result in semantic loss and interpretation problems.

Interpretation problems in connection with traditional conflict resolution techniques are illustrated by line (a) in Figure 5. We assert that applying the original concept names used in the end user views results in high interpretability which is illustrated by the keyword *High* and the starting points of both line (a) and line (b) in Figure 5. At the same time we assert that minimizing the concept names in the global schema results in low interpretability which is illustrated with the keyword *Low* and the end point of line (a) in Figure 5. Finally, with line (b) in Figure 5 we wish to point out that applying the proposed conflict resolution techniques the level of interpretability is also raised because all concept names are retained even after resolution. Maximizing the line of mutual interpretation in conceptual database design and view integration may result in a conceptual schema that is semantically correct, complete and easy to use. This is the case since all concept names are retained and therefore the end users may use their own language and concept names when interpreting and discussing the conceptual database views and schema.

5 Conclusion and short Discussion

In chapter two we made a comparison between an utterance and the contents in a database. The aim was to reveal what is missing concerning an expression in a database to be fully understandable as a linguistic conception and to suggest some ideas to help up the situation of interpretability of the conceptions in the views and schema of a database. The analysis has shown as follows. Those which are missing to make the conceptions to be fully interpretable in a linguistic sense in the views and schema are such as: the subject of the message, the aim of it, the illocutionary and the perlocutionary acts and finally what contexts the concepts are referring to. A solution to manage these problems might be to make up a meta-database in which these missing concepts for interpretation are coupled to what is normally documented in the views and schema. By means of this meta-database you will be able to ask for such as: who is the subject of the message, what is the aim of the message, in what context do the concepts belong (there might be many), and so on.

In chapter three we focused on the view integration process and what problems traditional conflict and inter-schema properties resolution techniques might cause in connection with the language used in the organization and in the views. In chapter four we discussed the problem of concept name compression and how to avoid it. Conflict resolution techniques that may bridge the problems with the traditional resolution techniques were proposed, discussed and illustrated.

Chapter four also contains a discussion regarding how to maximize the line of mutual interpretation. The conclusions of chapter three and four are as follows. Applying the proposed resolution techniques for synonyms, homonyms and the hypernym–hyponym property the line of mutual interpretation is increased because all concept names are retained even after resolving the conflicts and the inter-schema property. Retaining all concept names may not only be a first step trying to bridge the shortcomings discussed in chapter two but also help the end user interpreting the concepts and making the schema easy to use.

Finally, addressing the shortcomings and problems described and discussed in chapter two, three and four may empower and improve not only the conceptual database design procedure as such but also the interpretability and understandability of the final and global conceptual database schema. In the long run this may also contribute to an implemented database that is semantically correct, complete and easy to use.

References

- AMBROSIO A.P., MÉTAIS E., MEUNIER J-N. (1997) “The Linguistic Level: Contribution for Conceptual Design, View Integration, Reuse and Documentation”, *Data & Knowledge Engineering*, No 2. pp 111-129
- BATINI C., LENZERINI M., NAVATHE B.L. (1986) “A Comparative Analysis of Methodologies for Database Schema Integration”, *ACM Computing Surveys*, Vol 18, No 4. pp 323–363
- BATINI C., CERI S., NAVATHE S. (1992) *Conceptual Database Design An Entity-Relationship Approach*, The Benjamin/Cummings Publishing Company, Inc., Redwood City, California
- BELLSTRÖM P. (2005) “Using Enterprise Modeling for Identification and Resolution of Homonym Conflicts in View Integration”, Vasilecas O. et al. (Eds.) *Information Systems Development Advances in Theory, Practice and Education*, Springer, pp 265–276
- BELLSTRÖM P. (2006a) *View Integration in Conceptual Database Design – Problems, Approaches and Solutions*, Licentiate thesis, Karlstad University Press, 2006:5
- BELLSTRÖM P. (2006b) “Bridging the Gap between Comparison and Conforming the Views in View Integration”, Manolopoulos Y. et al. (Eds.) *Local Proceedings of the 10th ADBIS Conference*, pp 184-199
- BELLSTRÖM P., CARLSSON S. (2004) “Towards an Understanding of the Meaning and the Contents of a Database through Design and Reconstruction”, Vasilecas O. et al. (Eds.) *Proceedings of the 13th ISD Conference*, pp 283-293
- BELLSTRÖM P., JAKOBSSON L. (2006) “Towards a Generic and Integrated Enterprise Modeling Approach to Designing Databases and Software Components”, Nilsson, A.G. et al. (Eds.) *Advances in Information Systems Development: Bridging the Gap between Academia and Industry*, Springer, pp 635- 646
- BELLSTRÖM P., VÖHRINGER J., SALBRECHTER A. (2006) “Recognition and Resolution of Linguistic Conflicts: The Core to a Successful View and Schema Integration“, *Information Systems Development New Methods and Practices for the Networked Society*, Springer (in print)
- CARLSSON S. & CHRISTIANSSON B. (1999) “The Concept of Object and its Relation to Human Thinking: Some Misunderstandings Concerning the Connection between Object-Orientation and Human Thinking”, *Informatica*, Vol 10, No 2. Lithuanian Academy of Sciences, Vilnius
- CHEN, P. (1976) “The Entity-Relationship Model – Toward a Unified View of Data”, *ACM Transactions on Database Systems*, Vol 1, No 1. pp 9-36
- CONNOLLY T., BEGG C. (2005) *Database Systems A Practical Approach to Design, Implementation, and Management*, Addison Wesley, England.
- DEY D., STOREY V.C., BARRON, T.M. (1999) ”Improving Database Design through the Analysis of Relationships”, *ACM Transactions on Database Systems*, Vol 24, No 4. pp 453-486
- ENGELS G., GOGOLLA M., HOHENSTEIN U., HULSMANN K., LOHR-RICHTER P., SAAKE G., EHRICH, H.D. (1992) “Conceptual Modelling of Database Applications using an Extended ER Model”, *Data & Knowledge Engineering*, Vol 9, pp 157-204
- ELMASRI R., NAVATHE S.B. (2004) *Fundamentals of Database Systems*, Addison Wesley, Boston
- GOLDKUHL G. (1980) *Framställning och användning av informationsmodeller*, PhD thesis, University of Stockholm, Department of Information Processing, GOTAB, Stockholm (in Swedish)
- GOLDKUHL G., LYYTINEN K. (1982a) “A disposition for an information analysis methodology based on speech act theory”, Goldkuhl G., Kall C-O (Eds.) *Report of the 5th Scandinavian Research Seminar on Systemeering*, Dept of Information Processing, Chalmers University of Technology, Göteborg
- GOLDKUHL G., LYYTINEN K. (1982b) “A language action view of information system”, *3rd International Conference on Information System*, Ann Arbor

- GOLDKUHL G., LYYTINEN G. (1983) "Information systems specification as rule reconstruction", *IFIB 8.2 Conference Beyond productivity: Information systems for organizational effectiveness*, Minneapolis
- GUSTAS R. (1997) *Semantic and Pragmatic Dependencies of Information Systems*, Monograph, Technologija, Kaunas
- GUSTAS R., GUSTIENÉ P. (2004) "Towards the Enterprise Engineering Approach for Information System Modelling Across Organisational and Technical Boundaries", Camp O. et al (Eds.) *Enterprise Information Systems V*, Kluwer Academic Publishers, Netherlands, pp 204-215
- HOPPENBROUWERS S.J.B.A., PROPER H.A., VAN DER WEIDE Th.P. (2005) "A Fundamental View on the Process of Conceptual Modeling", Delcambre L. et al. (Eds.) *Proceedings 24th ER Conference*, Springer-Verlag, pp 128–143
- JAKOBSSON L., BELLSTRÖM P. (2006) "Designing Software Components for Database Consistency – An Enterprise Modeling Approach", *Information Systems Development New Methods and Practices for the Networked Society*, Springer (in print)
- JOHANNESSON P. (1993) *Schema Integration, Schema Translation, and Interoperability in Federated Information Systems*, PhD thesis, Department of Computer & Systems Sciences, Stockholm University, Royal Institute of Technology, No. 93-010-DSV, Edsbruk
- LANGEFORS B. (1995) *Essays of Infology*, Dahlbom, B. (Ed), Studentlitteratur, Lund
- LAWRENCE R., BARKER, K. (2001) "Integrating Relational Databases Using Standardized Dictionaries", *16th ACM Symposium on Applied Computing*, ACM Press, pp 225-230
- LEE M.L., LING T.W. (2003) "A Methodology for Structural Conflict Resolution in the Integration of Entity-Relationship Schemas", *Knowledge and Information System*, Vol 5, No 2. pp 225-247
- LINELL P. (1994) *Approaching Dialogue: On Monological and Dialogical Models of Talk and Interaction*, Department of Communication Studies, University of Linköping
- MANNINO M.V. (2006) *Database Design, Application Development, & Administration*, McGraw-Hill, New York
- MARTIN J., ODELL J.J. (1998) *Object Oriented Methods A Foundation*, Prentice Hall, New Jersey
- NUSEIBEH B., EASTERBROOK S., RUSSO A. (2001) "Making Inconsistency Respectable in Software Development", *The Journal of Systems and Software*, Vol 58, pp 171-180
- OLIVÉ O. (2004) "On the Role of Conceptual Schemas in Information Systems Development", Llamosí A., Strohmeier A. (Eds.) *Proceedings of 9th Ada-Europe International Conference on Reliable Software Technologies*, Springer-Verlag, pp 16-34
- PARENT C., SPACCAPIETRA S. (1998) "Issues and Approaches of Database Integration", *Communications of the ACM*, Vol 41, No 5es. pp 166-178
- PARSONS J. (2002) "Effects on Local Versus Global Schema Diagrams on Verification and Communication in Conceptual Data Modeling", *Journal of Management Information Systems*, Vol 19, No 3. pp 155-183
- RICOEUR P. (1976) *Interpretation Theory: Discourse and the Surplus of Meaning*, Forth Worth: Texas University Press
- SCHUTZ A., LUCKMANN T. (1974) *The Structures of the Life-World*, London:Heinemann
- SHOVAL P., SHIRAN S. (1997) "Entity-Relationship and Object-Oriented Data Modeling – an Experimental Comparison of Design Quality", *Data & Knowledge Engineering*, Vol 21, pp 297-315
- SONG W. (1995) *Schema Integration – Principles, Methods, and Applications*, PhD thesis, Department of Computer & Systems Sciences, Stockholm University, Royal Institute of Technology, No. 95-019, Edsbruk
- SPACCAPIETRA S., PARENT C. (1991) "Conflicts and Correspondence Assertions in Interoperable Databases", *ACM SIGMOD RECORD*, Vol 20, No 4. pp 49-54
- SPACCAPIETRA S., PARENT C. (1994) "View Integration: a Step Forward in Solving Structural Conflicts", *IEEE Transactions on Knowledge and Data Engineering*, Vol 6, No 2. pp 258-274
- STOREY V.C., THOMPSON C.B., RAM S. (1995) "Understanding Database Design Expertise", *Data & Knowledge Engineering*, Vol 16, pp 97-124

