starER: A Conceptual Model for Data Warehouse Design

Nectaria Tryfona, Frank Busborg, and Jens G. Borch Christiansen

Department of Computer Science, Aalborg University,
Fredrik Bajersvej 7E, DK-9220, Aalborg Øst, Denmark

{tryfona, dux, jbc}@cs.auc.dk

Abstract. Modeling data warehouses is a complex task focusing, very often, into internal structures and implementation issues. In this paper we argue that, in order to accurately reflect the users requirements into an error-free, understandable, and easily extendable data warehouse schema, special attention should be paid at the conceptual modeling phase. Based on a real mortgage business warehouse environment, we present a set of user modeling requirements and we discuss the involved concepts. Understanding the semantics of these concepts, allow us to build a conceptual model–namely, the starER model–for their efficient handling. More specifically, the starER model combines the star structure, which is dominant in data warehouses, with the semantically rich constructs of the ER model; special types of relationships have been further added to support hierarchies. We present an evaluation of the starER model as well as a comparison of the proposed model with other existing models, pointing out differences and similarities. Examples from a mortgage data warehouse environment, in which starER is tested, reveal the ease of understanding of the model, as well as the efficiency in representing complex information at the semantic level.

Keywords: data warehouse, conceptual modeling, star structure, ER model.

1 Introduction

A data warehouse is a collection of consistent, subject-oriented, integrated, time-variant, non-volatile data and processes on them, which are based on available information and enable people to make decisions and predictions about the future [7]. Over the last years, data warehouses enjoy a lot of attention both from the industrial and the research community. The reason lies in their great importance: making predictions about the (near) future, has always been desirable for business companies.

Data warehouse design has hitherto focused on the physical data organization (i.e., the "internal" structure) and quite understandable so, because of the volume and the complexity of data. Following the logical structure of data, as described in a data warehouse, several schemas have been developed emphasizing on the star-oriented approach; data unfolds around facts occurring in businesses. The star [1], the starflake [12], and the snowflake schema [8] are used widely for this purpose. Although all of these schemas provide some level of modeling abstraction that is understandable to the user, they are not built having his/her needs in mind.

Our position is that data warehouse modeling–as exactly databases do, many years now–should be exposed, to a higher level of design, that is understandable to the user, independent of implementation issues, and that does not use any computer metaphors, such as "table" or "field". The result of this process will be a schema that is formal and complete, so that it can be transformed into the next logical schema without ambiguities. This is the *conceptual* or *semantic modeling phase*, and the benefits of its use have been praised a lot: communication between the designer and the user, early detection of modeling errors, and easily extendable schemas are among them. The conceptual modeling phase is part of a design methodology–which is classical in the database area, and has been already proposed [6] for the data warehouse area– following the user requirements analysis and specifications phase and, is followed by the logical design focusing on workload refinement and schema validation.

In this paper we firstly address the *modeling* requirements of a data warehouse, from the user point of view. For this purpose, we use a real mortgage business environment. The understanding of the requirements reveals a set of concepts that need to be accommodated at the conceptual modeling phase. We propose the use of a new model, namely the starER model, for the conceptual modeling of data warehouses. The starER model combines the semantically rich constructs of the well-known Entity-Relationship (ER) model [4] with the star structure that rules the data in data warehouses. Our starting point is that, the ER model has been tested for years, and proved powerful enough to model complex applications such as spatiotemporal, and multimedia. In all cases, when new modeling techniques are needed to capture the new demands, new constructs are added to ER; but the core of the model is the same. So, there is no reason to change such a model, which designers are familiar and happy with. On the other hand, as mentioned before, data warehouses impose a new modeling structure: facts about businesses are central, and all the data is unfolded around them.

We should make clear that we do *not* propose a set of new concepts and terminology. We rather put all the necessary concepts together under the same data warehouse framework, and try to understand their semantics, in order to, later, build an efficient, in terms of expressive power, conceptual model.

The presented starER model is tested in a mortgage warehouse; examples and experiences are listed here.

The paper is organized as follows: Section 2 gives the modeling requirements of a data warehouse, showing the concepts that need to be modeled. Then, in Section 3, the starER model is proposed. For each concept, we show the one-to-one correspondence with a starER construct. Small examples of use are given. Section 4 focuses on the design of a larger excerpt from a mortgage and loan business. The efficiency and ease of use of starER is demonstrated. Section 5 includes an evaluation of starER and a comparison of the proposed model with other existing models, pointing out differences and similarities. Finally, Section 6 concludes and discusses the future research plans.

2 Conceptual Modeling of Data Warehouses: Requirements and Concepts

In this Section we focus on the special modeling needs of a data warehouse at the conceptual phase, as they are drawn from theoretical [1], [2], [7] and practical experience [3] for a mortgage company. The listed user requirements reveal a set of new modeling concepts that need to be handled. Based on these, later on, well-known models and schema are to be combined and extended with new constructs, improving their ability to conveniently design data warehouse environments.

Consider the following example taken from a mortgage business environment. Customers have loans on buildings. They pay the loans in terms of repayments (i.e., "small" amounts) at specific dates. The mortgage company is interested in keeping track of the repayments and analyzing them in terms of customer profiles, loan profiles, and time periods. At the conceptual modeling phase of such a data warehouse there is the need to:

(a) represent *facts* and their *properties*.

Facts are central to data warehouses. They show actual facts of the real world and can be seen as processes further generating data over time. They are characterized by properties. In our example, "repayment" is the fact, showing the payback of a loan. "repayment" has "repayment amount" (i.e., the amount paid every month), and "installment" (i.e., the remaining amount of the loan to be paid) as properties.

Fact properties are usually numerical data, and can be summarized (or aggregated) in various ways in order to extract further information. For this reason, the numerical properties are also called *summary properties* ([9] refers to them as *summary attributes*). For example, the "repayment amount" is a summary property, as it can be summarized through the months, presenting the paid amount up to the current time point. This characteristic of the fact properties is important to data warehouses, and it is called *summarizability*. There are three different types of properties, with respect to summarizability: *stock*, *flow* and *value-per-unit*.

A property of type *stock* records the state of something at a specific point in time; for example, "installment" is of type stock. Alternatively, properties of type stock can be thought of as snapshots of the current state of some parameter in the environment monitored by the data warehouse. Summary properties of type *stock* can be summarized over the temporal dimension.

A property of type *flow* records the commutative effect over a period of time for some parameter in the environment monitored by the data warehouse. Properties of type flow record the change or, the commutative effect of a parameter over a period of time. Summary properties of type *flow* can always be summmarized. The "repayment amount" of our example is of type flow.

A property of type *value-per-unit* is similar to a property of type stock, in the respect that it is measured for a fixed time, but that the units of the property is different; because of that, the resulted measures can not be summarized. A property of type value-per-unit always describes the recording of the parameter in relation to some unit in the environment monitored. An example of this property type is the "interest rate per repayment". This is different from summary properties of type *stock*, since the "interest rate" can only be viewed in the context of its unit, i.e. "repayment".

A list of how the different property types behave in relation to the different summery functions can be found in [9].

(b) connect the temporal dimension to facts.

Facts can be seen as processes evolving over time, generating data. Based on this, "time" is an important aspect, and is *always* associated to a fact. For example, the "repayment" is done at specific dates, or within specific (pre-specified) time-periods.

(c) represent objects, capture their properties and the associations among them.

Information connected to facts can be analyzed–as in classical applications–in terms of objects, their properties and associations among objects, capturing the semantics of the data warehouse environment. In the example of the mortgage company, a "customer" is an object, with "customer identification number" (or, "customer id"), "name" and "income level" as properties. The "customer" "has" (i.e., association), an "address", with "zip code" and "street number" as properties.

Object properties can, like the fact properties, be numeric and are, also, called *summary properties*. Accordingly, they can be of type stock, flow, and value-per-unit. For instance, the "interest rate" of a loan is of type value-per-unit, as its average, minimum or maximum per year can be calculated, but it can not be summarized (i.e., with the *sum* function).

Additionally, three special types of associations–apart from the usual ones–among objects exist very often:

(i) specialization/generalization, showing objects as subclasses of another object. For example, a "company" and a "physical person" are both "customers" for a "mortgage company",

(ii) aggregation, showing objects as parts of a larger object; for example, a "mortgage company" consists of the "financial department" and the "administration department",

(iii) membership, showing that an object is member of another "higher" object class[1] with the same characteristics and behavior; for example, "branches" of a mortgage company are members of the "company".

[1] The term "object class", here, has a broader meaning than the specific object-oriented one.

The membership association is of special interest in data warehouses, since it appears very often and in connection to dimensions gives further results (see below, (e)). It is characterized by its *strictness* (or, not) and *completeness* (or, not). *Strict membership* means all members belong to *only* one higher object class. For example, a "branch" is a member of only one "company". *Complete membership* means that *all* members belong to one higher object class and that object class is consisted by that members *only*. For example, all "branches" and only them, belong to the "company". Thus, the membership branch-company is strict and complete.

(d) record the *associations* between objects and facts.

Facts are semantically connected to objects. For example, a "customer" (i.e., object) who has a loan, "pays" the "repayment" (i.e., fact) which is associated to the "loan" (i.e., object).

(e) distinguish *dimensions* and categorize them into *hierarchies*.

When a fact is connected to an object, the data that can be retrieved and analyzed via this association is of great importance. For example, analysis between "repayment" and "customer" shows the connection between the size of the repayment and the income level of the customers. Objects that are connected via associations to facts are called *dimensions*, and they are usually the focus of the data warehouse analysis. As said before (in (b)), a fact is always connected to the temporal dimension.

Dimensions are usually governed by associations of type membership forming *hierarchies* that specify different granularities. Thus, the time dimension can be decomposed into year, month and day, showing that days are members of months, which are members of years, which are members of time (i.e., membership). In that way, the sum of the repayments per customer can be summarized at quarterly basis of the year. In a similar way, a year can be summarized by its season-parts.

3 The starER Conceptual Data Warehouse Model

After analyzing the user requirements for data warehouse modeling, we proceed to the conceptual (or semantic) modeling phase of the application development cycle. Here, the main goal is to translate user requirements into an abstract representation, understandable to the user, that is independent of implementation issues, but is formal and complete, so that it can be transformed into the next logical schema without ambiguities.

The Entity-Relationship model (ER) [4] is the most widely used conceptual model. Its main advantages are the ease of use that it provides and the small set of supported constructs. Its main constructs are (a) the entity sets, capturing real world objects, (b) the relationship sets, capturing associations among objects and, (c) the attributes representing properties of entity or relationship set.

On the other hand, a set of new schemas (and *not* models) have been recently employed to capture the structure of data warehouses: the star, the snowflake and the starflake schemas. The reason for this is the star-structured data of a warehouse environment: facts about companies are centered, and data unfolds around them.

We believe that none of the aforementioned models and schemas is adequate to meet the data warehouse requirements, although they provide the constructs and the structure for this purpose. Our position is that the semantically rich ER model and the structure-efficient star schema have to be combined under the same model. For this purpose, we enrich ER with all the concepts showed in Section 2, as they are the result of the understanding and experiment [3] of the data warehouse nature as well as the star structure, in which facts are central. We show the one-to-one translation of these concepts to modeling constructs of the combined starER (i.e., star and ER) model.

3.1 The constructs of the starER model

Summarizing the set of concepts presented in Section 2, with respect to the ER constructs, the starER needs to accommodate: (a) facts, which show actual facts of the real world application environment, based on on-going processes, (b) entities, which represent autonomous objects of the environment, (c) relationships, which represent associations or links among entities, or among entities and facts, and (d) attributes, representing characteristics or properties of entities or relationships, or facts.

Based on these concepts, the model has the following constructs:

- *Fact set*: represents a set of real-world facts sharing the same characteristics or properties (see below). Semantically, a fact set points to the process of generating data over time, i.e., data is generated in terms of facts, each time an event related to the fact takes place. For that reason, a fact set is *always* associated to time (see below). A fact set is represented as a circle.

Consider the example from the mortgage company environment, in which the customers pay repayments on their building loans each month. In this case the event is the repayment of the loans. Figure 3.1 shows a "repayment" as a fact set.

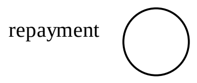

Figure 3.1: "repayment" as a fact set.

- *Entity set*: represents a set of real-world objects with similar properties; it has the same meaning as in traditional application modeling. The graphical representation of an entity set is a rectangle.

Typical entity sets from the mortgage company example are the "customer" and "loan".

customer
loan

Figure 3.2: "customer" and "loan" as entity sets in the mortgage company example.

- *Relationship set*: represents a set of associations among entity sets or among entity sets and fact sets. Its cardinality can be many-to-many (N:M), many-to-one (M:1) or one-to-many (1:M). The graphical representation of a relationship set, in both cases, is a diamond.

Figure 3.3 shows the relationship set between "customer" and his/her "address".

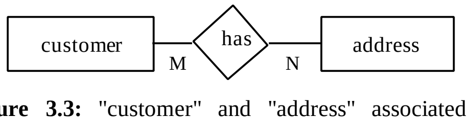

Figure 3.3: "customer" and "address" associated with the relationship set "has".

An example of a relationship set among entity sets and fact sets is (Figure 3.4) the association between "customer" and "repayment".

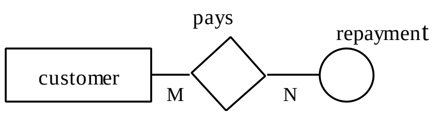

Figure 3.4: "customer" and "repayment" associated with the relationship type "pays".

Relationship sets among entity sets can be of type specialization/generalization, aggregation and membership, as described in Section 2. Figure 3.5 gives examples of these types together with their graphical representation.

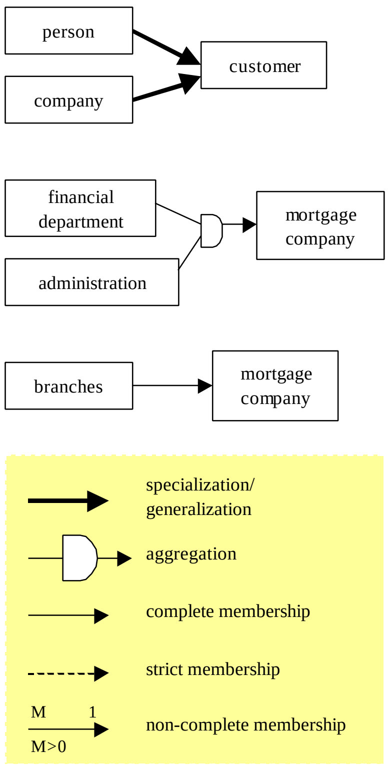

Figure 3.5: Examples of specialization, aggregation and membership relationship sets.

As mentioned in Section 2, in data warehouses, it is important to know if a membership is strict or not and complete, or not. The strictness or not of a membership is shown by cardinality of the membership and the accompanying constraint (i.e., M:1 and M>0 means strict). A complete membership is illustrated by a solid arrow, while a non-complete is given by a dashed arrow.

An important relationship among fact sets and entity sets, is the one between any fact and the "time" this fact is performed. "time" and any further component of it such as "day", "month", "season", is recorded as entity set. That allows us to model, and later on summarize, summary properties of that fact (described below as fact attributes) according to the time granularity (Section 2).

Additionally, "time", as well as the other entity sets associated to a fact, are the *dimensions of that fact*. Dimensions consist of hierarchies and/or other relationships among other entity sets. Consider the example of "repayment"-"payback"-"loan" and "repayment"-"at"-"time" presented in Figure 3.6.

Notice that the hierarchies of a dimension show the different granularities at which the connected fact set can be summarized. So, "repayment" can be summarized in terms of months, or years. Exactly for the reason of the summary calculation, it is important to know if a hierarchy is strict and/or complete.

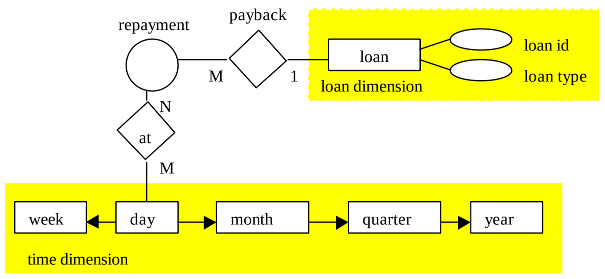

Figure 3.6: Dimensions and hierarchies.

- *Attribute:* static properties of entity sets, relationship sets, and/or fact sets are represented as attributes. Attributes are illustrated by ovals.

Attribute examples of the "customer" entity set are "name", "customer id", and "income level".

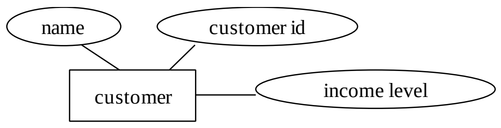

Figure 3.7: "name", "customer id", and "income level" as attributes of "customer".

As said in Section 2, fact properties can be of type stock, flow, or value-per-unit. This is indicated by an "S", "F", or "V" on the left of the attribute illustration, respectively. The attribute examples of the "repayment" fact set are the "repayment amount" and the "installment" (Figure 3.8).

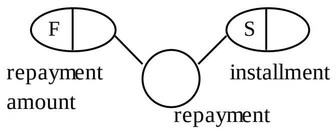

Figure 3.8: "repayment amount", and "installment" as attributes of the "repayment" fact set.

4 Example of Use

In this Section we give an extended excerpt (i.e., the customer-loan view) from the data warehouse of a mortgage business company, using the starER model. As can be seen, the basic structure of the resulted starER schema is star. The fact set "repayment" is central, and around it–indicated by shaded color–are the dimensions "customer", "loan", "real estate", and "time". Each dimension is further modeled by using the constructs of the ER model. Also, the "time" dimension is clearly governed by hierarchies specifying the different granularities at which the fact set "repayment" can be summarized.

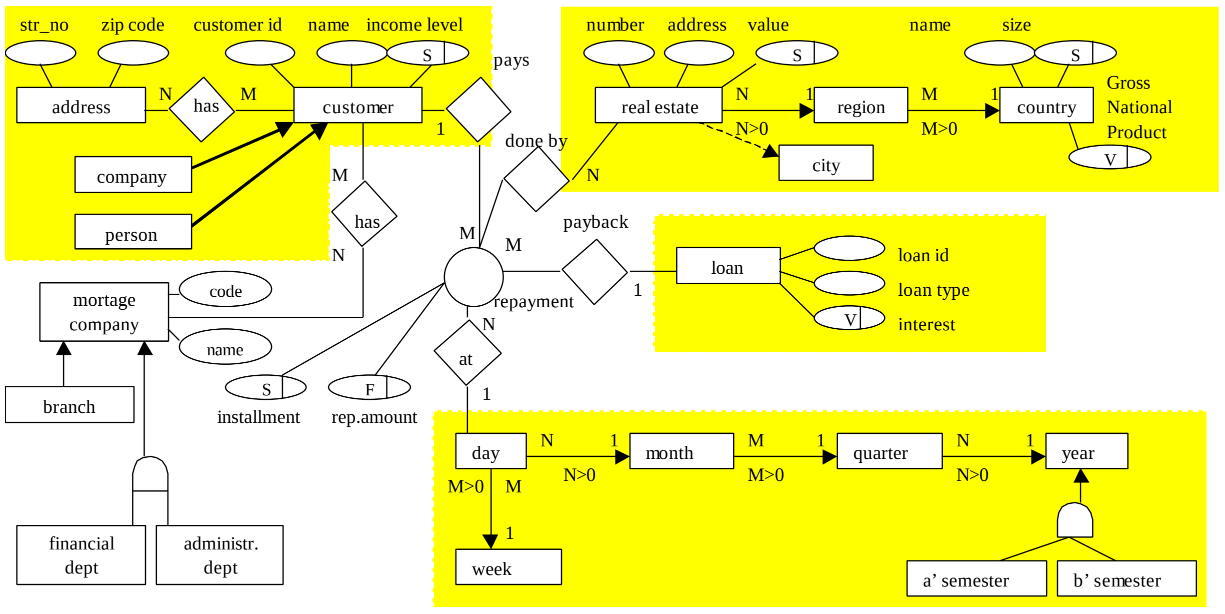

Figure 3.9: An excerpt from a mortgage company data warehouse, using the starER model. The financial and the administration department are outside the customer dimension as this is dependent on design decisions and semantic issues.

5 Evaluation of the starER model

In this section we evaluate the starER model based on a well-defined set of criteria [11] and later on, we compare the proposed model to other conceptual data warehouse models [5].

[11] present a set of evaluation criteria for data warehouse models. We adopt them here to evaluate the starER model. The criteria allow us to decide whether or not a model is adequate for data warehouse modeling, in terms of correctness, modeling power and efficiency in information capturing, independent of the design level the particular model exists. Below we present the criteria as described in [11] and discuss whether or not the starER model meets them. The nine criteria are:

- *Explicit hierarchies in dimensions.* The hierarchies in the dimensions should be captured explicitly by the schema, so the user has available the relation between the different hierarchical level.

The starER model supports explicit hierarchies, in terms of memberships.

- *Symmetric treatment of dimensions and summary attributes (properties).* The model should allow summary attributes to be treated as dimensions and vice versa.

In our interpretation of the starER constructs we have made a conceptual distinction between dimensions and summary attributes, but we do not restrict the user from modeling a dimension as a particular summary property (or vice versa) in order to add extra features to the analysis. In our example, the "repayment amount" is a summary attribute, to allow for computations such as summing up values.

- *Multiple hierarchies in each dimension.*

In our example of the "time" dimension, days can roll-up to months, and days can also roll-up to weeks.

- *Support for correct summary or aggregation.* The data model should give meaningful summaries or aggregations to the user.

The starER model requires the illustration of explicit hierarchies together with the type of the hierarchy (i.e. strict, complete etc.) and the measure types (i.e., stock, flow, or value-per-unit), so that the conditions of summarizability can be properly evaluated.

- Support of non-strict hierarchies.

The starER model allows non-strict hierarchies via the cardinality of the memberships.

- *Support of many-to-many relationships between facts and dimensions.*

An example of this is the relationship between "repayment" and "real estate".

- *Handling different levels of granularity at summary properties.*

The starER model handles different levels of granularity at summary properties via the membership hierarchy. For example, the "repayment amount" of Figure 3.8 can be summarized per year, following the granularity of the "time" dimension.

- *Handling uncertainty.*

The starER at the present state does not accommodate uncertainty. Uncertainty in data warehouses is a research area on its own, and needs extended study. If uncertainty deals with the presence or not of fact sets, entity sets, attributes, or relationship sets, then it can be however handled by starER, by adding probability (i.e., attributes showing probability) at fact sets, entity sets, attributes (i.e., making them composite) or relationship sets, respectively. If uncertainty deals with specific values of instances of starER constructs, e.g., the probability to have "loan interest" more than 3%, then it falls at the logical design level, and its accommodation can not be argued for the starER model, since all the included information is at the conceptual level.

- *Handling change and time.*

This criterion falls in the logical design level, and its accommodation can not be argued for the starER model, since all the included information is at the conceptual level.

As mentioned before, the starER model has the basic constructs and philosophy of the ER model, as well as the structure of the star schema. It is semantically richer that the star, starflake and snowflake schema as it is designed to serve highly abstract representations. The only schema that falls into the area of conceptual design is the dimensional fact schema [5]. Its main components are facts, dimensions and hierarchies. A fact schema is constructed as a tree, with the fact as root.

There are some differences between the dimensional fact schema and the starER model, which make the second one semantically richer:

- Relationships between dimensions and facts in starER are not only many-to-one, but also many-to-many, which allows for better understanding of the involved information. Such an example is the relationship between "repayment" and "real estate".
- Objects participating in the data warehouse, but not in the form of a dimension (i.e., not connected directly to a fact) are allowed in starER, permitting in this way to capture more semantics.
- Specialized relationships on dimensions are permitted, such as specialization/generalization, membership, and aggregation representing more information (see Figure 3.8)

One could argue that the dimensional fact schema requires only a rather straight forward transformation to fact and dimension tables, and this is an advantage of the dimensional fact schema. But, this is not a drawback for the starER model, since well-known rules of how to transform an ER schema (which is the basic structural difference between the two approaches) to relations *do* exist.

6 Conclusions

In this paper we discuss a set of modeling requirements as they are drawn from a real mortgage warehouse environment, from the users' point of view. These requirements reveal a set of concepts that need to be included into conceptual models in order to efficiently design data warehouses. Based on these concepts, we build a new model, the starER model, which combines the semantically powerful constructs of the ER model, with the dominant, in the warehouses, star-structure of data. The model has been tested in a mortgage business environment and we experienced a welcome acceptance from both users, appreciating the ease of use and understanding of starER, and designers, for the model's expressive power and still close relation to tools and terms they are used to.

We see the starER model as part of a data warehouse design methodology, leading from user requirements to physical implementation. We are currently working on building the tool to support in a semi-automatic way such a methodology. Transformation rules from the starER constructs to specific logical models, such as the relational model, which is used by many data warehouse software packages (see for example [13]) and to multidimensional models [10] are considered.

Acknowledgements. This work was supported by the Nykredit Coorporation (Denmark). The authors wish to thank Nykredit Data for providing the environment, as well as the feedback for the mortgage and loan management applications.

7 References

[1] Anahory, S., and Murray, D., 1997. *Data Warehousing in the Real World*. Addison-Wesley.

[2] Busborg, F., Christiansen, J.B., Jensen, K.M., and Jensen, L., 1998a. *A Method fo Data Warehouse Development*. Dat5 Report, part 2. CS Department. Aalborg University.

[3] Busborg, F., Christiansen, J.B., Jensen, K.M., and Jensen, L., 1998b. *Data Warehouse Modeling: The Nykredit Case Study*. Dat5 Report/Part I. Computer Science Department. Aalborg University.

[4] Chen, P.S., 1976. *The Entity-Relationship Model: Toward a unified view of Data*. ACM TODS, 1(1):9-36.

[5] Golfarelli, M., Maio, D., and Rizzi, S., 1998. *Conceptual Design of Data Warehouses from E/R Schemas*. Proceedings of the 13th Hawaii International Conference on System Sciences. Kona, Hawaii.

[6] Golfarelli, M., and Rizzi, S., 1998. *A Methodological Approach for Data Warehouse Design*. Proceedings of the 1st International Workshop on Data Warehouses and OLAP (DOLAP'98). Washington DC. USA.

[7] Immon, W. H., 1996. *Building the Data Warehouse*. Wiley Computer Publishing (2nd Edition).

[8] Kimball, R., 1996. *The Data Warehouse Toolkit*. John Wiley & Sons Inc.

[9] Lenz, H-J., and Shoshani, A., 1997. Summarizability in OLAP and Statistical Databases. 9th International Conference on Scientific and Statistical Database Management.

[10] Oracle Manual, 1998. Oracle Corporation. Oracle Express Server: Delivering OLTP to the Enterprise. White paper at: www.oracle.com/database/documents/express_server_fo.pdf

[11] Pedersen, T. B., and Jensen, C. S., 1998. *Multidimensional Data Modeling of Complex Data*. Proceedings of the 15th IEEE International Conference on Data Engineering (ICDE 99), Sydney, Australia.

[12] Poe, V., 1996. *Building a Data Warehouse*. Prentice Hall.

[13] SAS Manual, 1996. SAS Institute's Rapid Warehousing Methodology (Manual), SAS institute Inc.