SISU dokument

A Tool for Design of Multimedia Information Retrieval Systems

– requirements and functional specification

Tapani Kinnula

nr
14

1. Introduction

Today corporate users need to access and combine information from a number of information sources. Some business information is stored in relational databases, while other important information is accessed through searches in a text retrieval system. Additional sources of information may be picture and drawings archives. Each information source might have its own data model and access mechanism, or query language, causing severe usability problems.

The objective of the Intuitive project is to provide efficient and easy to use tools for end users to access information in heterogeneous corporate databases.

Within the project we are investigating users and application requirements to define a generic architecture for Information Retrieval from heterogeneous databases.

The project also address the needs of application designers in developing methods and tools for customising the generic software architecture for different applications. In this sense Intuitive can be seen as both a generic retrieval system as well as a system for creating information retrieval applications.

Rosengren et al have already given an overview of this work [Rosengren93a], [Rosengren93b], [Rosengren93c]. Wingstedt et al discuss the requirements on the functionality of end-user tools for Information Retrieval and their user interface [Wingstedt93].

Bern et al address the needs of application developers in describing a first prototype implementation of the Tools applied in three different applications [Bern93]. The reader that is unfamiliar with the Intuitive Project is referred to these reports.

The purpose of this document is to further investigate application developer's requirements on tools and methods for developing multimedia information retrieval applications.

An Intuitive application is largely based on the conceptual model of business information, the physical structure of databases containing the business information and mapping between conceptual model and the database structure. In addition to this structural information, graphical definitions for the end user interface is needed.

These two types of information together, the structural and the graphical, are called metadata. Metadata is specified and entered in the dictionary by the application designer when creating or maintaining an Intuitive application. The process of creating an Intuitive application is referred in this paper as *application design process*.

To support the application designer in creating Intuitive applications in an efficient and flexible manner, a tool for creation and management of metadata has to be developed in the Intuitive project. This tool for the application design process, called Designer Tool, is described in this document.

A first prototype version of the Designer tool has been developed during the analysis and specification of application development support. This paper gives, in addition to the Designer tool functional requirements, also a short description of the process of Intuitive

application design. The application design process is exemplified with screens dumps from the Designer tool prototype.

Chapter 2 gives an overview of the Intuitive dictionary as it is currently specified. The dictionary is the designer tool's only connection to the rest of the Intuitive environment. The Designer Tools sole purpose is actually to populate and maintain the Intuitive dictionary with metadata defining an Intuitive application.

Chapter 3 gives the functional requirements that the Designer tool should meet. These requirements are basically derived from needs arising during the application design process but also from some of the implications of functionality in Intuitive's information retrieval tools. For instance, if the Selector tool [Wingstedt93] will have to support pre-defined queries (i.e. standard reports), then the Designer tool should support specification of pre-defined queries and put them into the dictionary and available for the Selector.

Chapter 4 describes the application design process at a rather general level. Nevertheless, this chapter still gives a good overview of the entire design process and exemplifies the usage of the Designer tool as design process support. Chapter 5 outlines our future work.

Finally, Chapter 6 gives a short summary of this paper and some conclusions from work with Designer tool specification and prototype development.

2. The Intuitive Dictionary

When discussing metadata, we distinguish between the structural information and the graphical information. The structural information describes how tables and columns in a relational database or objects in text or picture databases are related to entities, attributes and relationships in the conceptual model. The graphical information is about the visual representation of the conceptual model. The conceptual model may have several visual representations showing various views to the model. Views form the users' "graphical query environment" and are normally adapted to the information needs in specific end user tasks.

The graphical information mainly specifies the layout and appearance of views, entities, attributes and relationships.

Figure 2.1 shows a conceptual model of the Intuitive Dictionary containing the metadata.

The majority of all information stored in the dictionary is inserted during the design phase of end user application development. Correctness, consistency and completeness of the information stored in the dictionary is probably the most significant factor that determines the robustness and quality of an end user application. Therefore, the tool support for populating and analysing the dictionary is crucial in application design and maintenance.

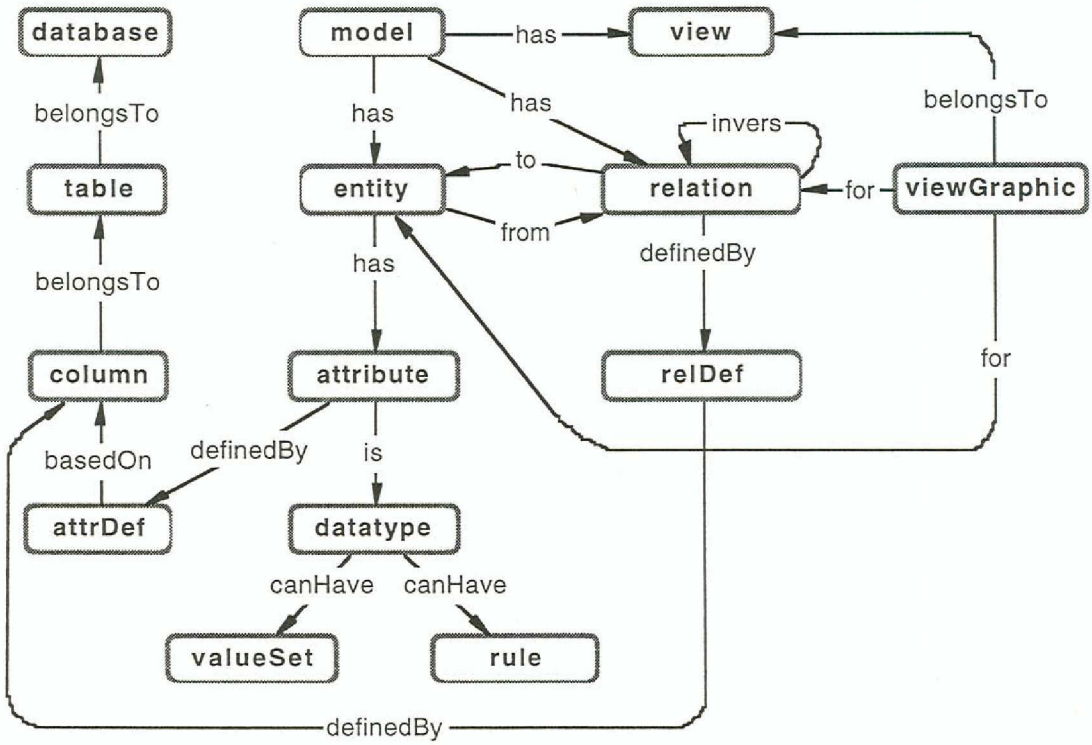

Figure 2.1. The structure of the Intuitive Dictionary

Objects in the dictionary are various types of specifications constituting an end user application. Figure 2.1 shows the basic specification objects and relationships between them. For example, the figure states that entities have attributes of certain data types. Attributes are defined by attribute definitions (the attrDef object) which in turn are based on columns belonging to tables in a database.

The dictionary structure as shown in the figure 2.1 does not take heterogeneous multi-media databases into consideration.

3. Functional Requirements on the Designer Tool

The Designer Tool will support the application designer in tailoring an Intuitive application to meet both the end user requirements [Rosengren 93d] and requirements originating from the structure of existing corporate databases to which the Intuitive application will act as a user interface. These requirements are covered by following six Designer Tool tasks:

1. Create and manage Intuitive specifications of the physical structure (database schema) of corporate databases.
2. Support creation of conceptual models and views describing the structure and contents of databases in terms of business concepts.
3. Manage and present information about mappings between the physical database structure (tables, fields, relationships) and the conceptual model (entities, attributes, relations).
4. Support specification of end user information and interaction support, e.g. entity and attribute definitions, explanations and rules that restrict and validate attribute values.
5. Support the design of the visual appearance of the conceptual model and its views.
6. Support specification of predefined queries.

End user support information mentioned in item 4 above provides help and guidance for end users and also prevents them from making mistakes. There are three kinds of user support information:

- Textual descriptions of entities, attributes and relationships.
- Intuitive data types and value sets of attributes.
- Definition of rules that constrain attribute values and instance relationships.

3.1. Definition of the physical database structure

An Intuitive application's physical database structure is specified as a set of databases and tables in those databases. Each table in turn is specified as a set of columns with column data type specifications.

The database specifications serve two purposes. Firstly, it is used by the CQL (Conceptual Query Language) translator to enable translation during translation of queries expressed in CQL into the current database managers' native query language (e.g. SQL) [Rosengren 93e]. Secondly, it serves as a natural starting point for generation of the conceptual model, since the tables and columns in a database often correspond to entities and attributes in the conceptual model.

The Designer tool must support specification of databases with tables and columns. The lowest level of database specification support in the Designer tool is allowing an application designer to enter the specifications manually. A preferable, but not always possible, approach is of course automatic generation of database definitions using database schema definition produced by a database manager or database design tool. The Designer tool should support both manual definitions and automatic generation from imported database schemes. The Designer tool should at least be able to import and interpret database definitions expresses in SQL ("create table" statements).

3.2. Creating and managing conceptual descriptions

Creation of the conceptual model may start with a more or less automated generation of a preliminary model from the database schema as already described above. After that the Designer Tool should support the application designer in building a model that more closely corresponds to the users' conception of the information stored in the database. This can be done by allowing the designer to construct entities from table definitions. Entities may also be a result from combining or splitting tables motivated by the need to hide the physical structure motivated by technical considerations. Entities, attributes, relationships and attribute data types generated from the database schema are replaced with appropriate natural language terms from the current business domain.

When the entire conceptual model is created, the Designer Tool should support creation of various views to the conceptual model. The views may represent parts of the entire model or show selected amounts of information at various detail and abstraction levels.

The entire conceptual model cannot normally be automatically generated from corresponding database definitions. This is because of the limitations in native database manager's expressiveness and lack of appropriate data types. Especially multi-media data is not well supported by the data types in today's database managers. Pictures, document images and video data are often stored in separate files referred to from table fields or in binary fields without information of the actual type of data. The knowledge and functionality necessary to manage multi-media data is instead included in the application programs interfacing to the database managers. Accordingly, it will often be the case that the application designer must manually specify the appropriate conceptual data types (e.g. picture or video) and specify how the Intuitive system will have to manage these data when obtained in answers to queries.

3.3. Mappings between the physical and the conceptual layers

The Designer Tool must create and keep track of mappings between the database schema and the conceptual model since the mapping information is necessary for CQL translation. There are two kinds of mapping information to be created and maintained:

- Mappings between table columns and entity attributes. An attribute is defined as an arithmetic expression of one or more database fields or constants.
- Mapping between entity relationships and table relationships. An entity relationship is derived from database joins between tables which the involved entities are based on.

An additional requirement on the mappings information is that the origin of entities and the type of resource it represents must be registered. This is needed to allow the CQL translator formulate correct database queries in appropriate query languages.

The attribute data type should be easy to understand for the end users. Therefore, column data types should be mapped into more conceptual and expressive attribute data types. If the initial conceptual model is automatically generated from the database schema definitions, a default translation from column data types to attribute data types occurs (e.g. string to Text, char[10] to Text with max size 10, double to Number etc.).

When no appropriate attribute data types are available during attribute definition or re-definition, the Designer tool must allow definition of new attribute data types. An example could be when the entity "Person" has an attribute "Photo". Because of technical reasons, underlying table column does not contain the picture itself but merely refers to a file containing a picture. Since the file name is just a string, the column data type probably will be string. When mapping the column to an attribute, the application designer defines a new conceptual data type "Picture" for the "Photo" attribute and specifies the possible picture formats and eventually also the tools that can be used to display the picture.

The attribute data type definitions may include specification of valid attribute values as value sets or rules expressing the conditions the attribute values must meet. Value sets and rules are used to allow end users to choose among possible values or prevent them from enter incorrect or inconsistent values.

It is desirable that the Designer Tool also be able to detect and prevent mappings that cause ambiguities that the CQL translator may not be able to solve (correct SQL generation requires a unique mapping from the conceptual model to the database schema). Before such support can be implemented in the Designer tool there are, however, both theoretical and practical problems to be solved when the tables are split and combined into entities and when attributes are defined as numerical or logical expressions including columns, constant values and values of other attributes.

3.4. Design of applications' visual interfaces

Design of an application's visual interfaces consists mainly of specifying layout and visual symbols (icons, bitmaps, pictures, labels etc.) that will be used to represent entities, relationships and views. A visual symbol is a graphical object that is used to represent an entity or a relationship in a view or a view in an other view. An entity may have several alternative visualisations in different views. The Designer tool should allow an easy way to connect and modify existing connections visual symbol to these elements.

3.5. Specification of predefined queries

The functional requirements considering specification of predefined queries is not established yet why we refrain from specifying corresponding tool support. We believe however, that predefined queries largely can be supported by regarding them as a special form of restricted views, i.e. views with restricted entity attribute set and pre-set values or value constraints for some attributes.

4. The Design Process

The design process is here presented as a series of consecutive steps. The entire design process does not proceed as single sequence of these step explicitly in given order. There may be several parallel sequences each focusing on its own set of design objects. Sequences may arise during the process and they may join for result integration. A normal design sequence iterates through the steps several times, and in any step new knowledge, problems that arise or new ideas may motivate a jump to an earlier step. The order of the design steps for a given design object or set of design objects is still motivated since it is the *logical causal order* given by the information needs in each step. Results from one step are needed in later steps. This also implies that a jump to an earlier step for modifications may make it necessary to re-do all the following steps and make changes reflecting the modifications.

Normally, an Intuitive application is designed as a high-level visual query interface to

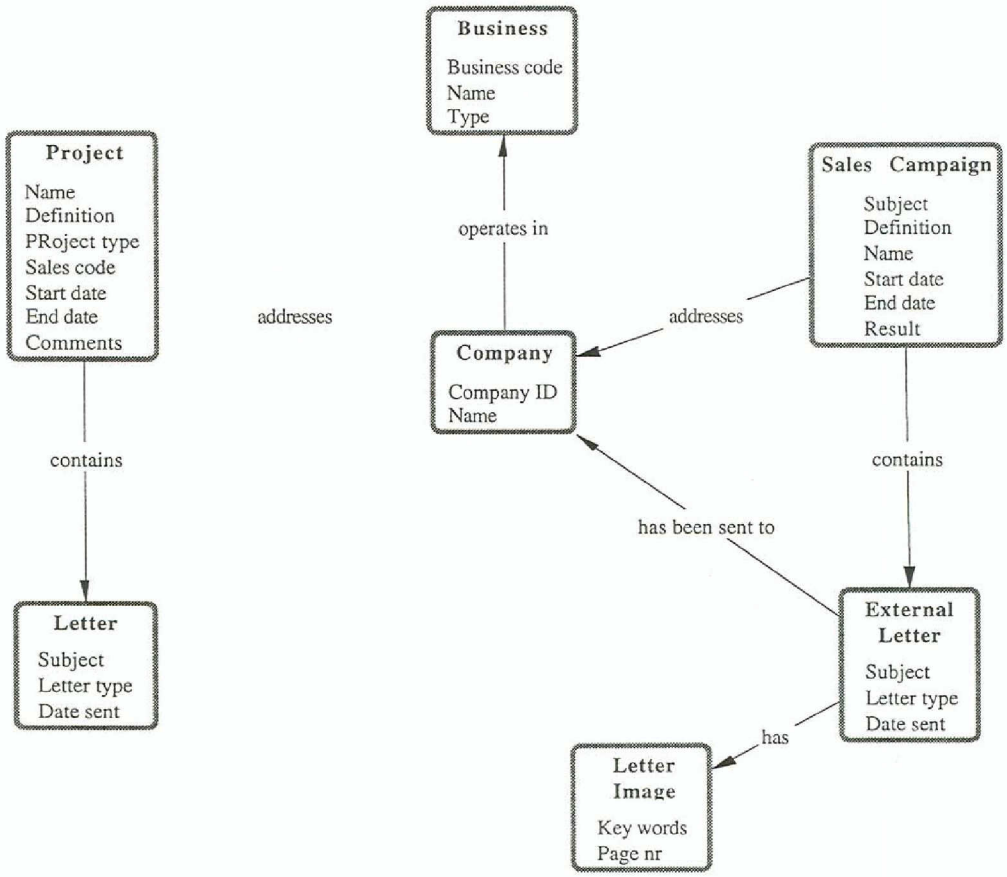

Figure 4.1. An Example Conceptual model of business information.

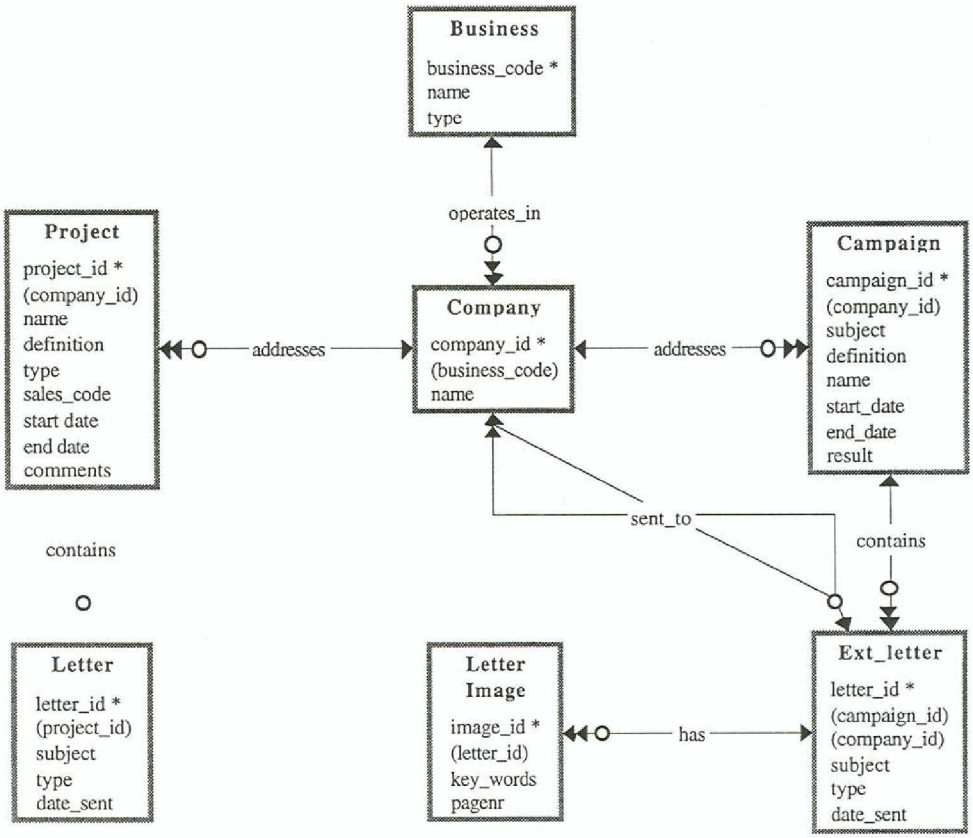

Figure 4.2. Data model describing the physical database structure in a relational database.

existing corporate databases [Bern93]. Sometimes, however, the application database will be designed as an initial activity or in parallel to design of the Intuitive application. The Intuitive development system is not intended to be used for database design, since there are lot of advanced database design tools for practically all existing database managers.

In the following we shortly describe the Intuitive application design process using the Designer tool. The process descriptions are exemplified with screen dumps from the first functional prototype of the Designer Tool implemented in Visual Basic.

1. Analyse the business information structure in co-operation with users and domain experts and construct a conceptual model of the business information. It is preferable to create a conceptual model of the business information. Figure 4.1 shows an example of how a conceptual model from this step may look like.
2. If the needed databases do not exist, design a new database based on the results from business analysis using a suitable database design tool. Observe that the database design does not always closely reflect the conceptual model because of requirements of technical nature (data security, efficiency, database architecture etc.). Figure 4.2 shows a data model for business information as modelled in figure 4.1.

3. Enter the physical database definitions into the Designer tool manually or by importing database schema definitions.
4. Enter the entire conceptual model of the business information into the Designer tool. An initial model can be generated automatically using the physical database definitions as a starting point, and then modifying to correspond the users' view on the business information as expressed in the conceptual model from step 1. The model can also be constructed by manually entering its content and how it is mapped to the physical database structure.

   An other important activity integrated into model creation is specification of end user interaction support. This support consists of descriptions and explanations of entities, attributes and relationships. End user support also includes attribute data types (as opposed to table column data types) and value sets and rules that restrict attribute values and help end users to see and enter valid attribute values in queries.

   Figure 4.4 shows an entity definition dialogue that can be invoked from the table definition dialogue as shown in the figure 4.3 (if the option "Generate - Manual definition" is selected) or by double-clicking on an entity in the model graph. Figure 4.5 shows an initial model graph as generated by the Designer from table definitions. Observe that no relationships between entities exist in the initial model, since the table join information cannot be specified during table definitions. When relationships are drawn in the conceptual model, the Designer tool invokes a Relationship Definition dialogue where relationships are defined as joins between tables which the involved entities are based as illustrated in figure 4.6. During entity and attribute definition, the designer can also invoke the data type definition dialogue, see figure 4.7.

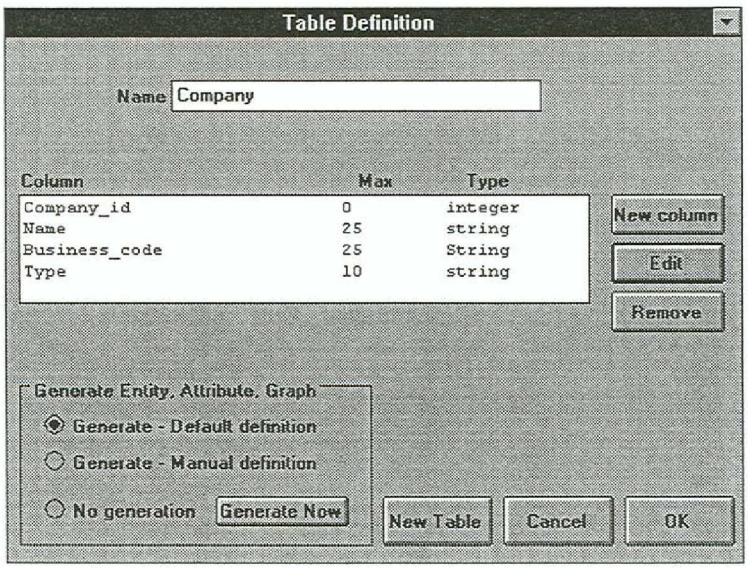

Figure 4.3. A Designer tool dialogue for manual definition of database tables.

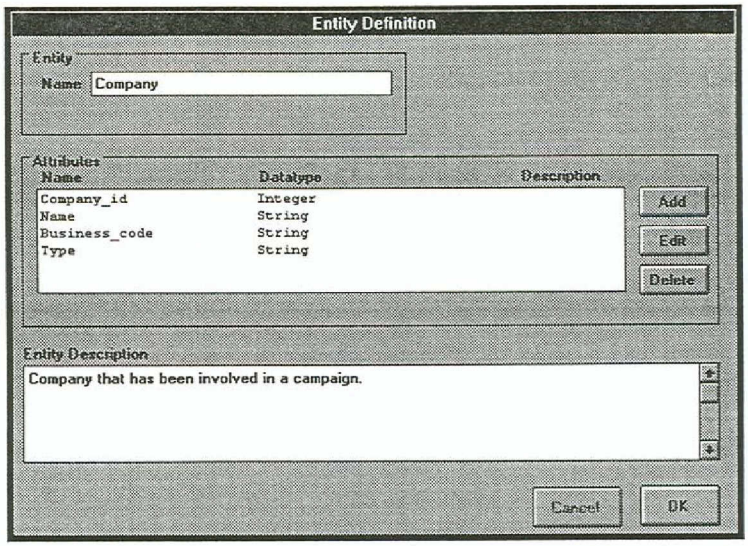

Figure 4.4. Entity definition dialogue. The entity "Company" is in this case automatically from a corresponding table definition.

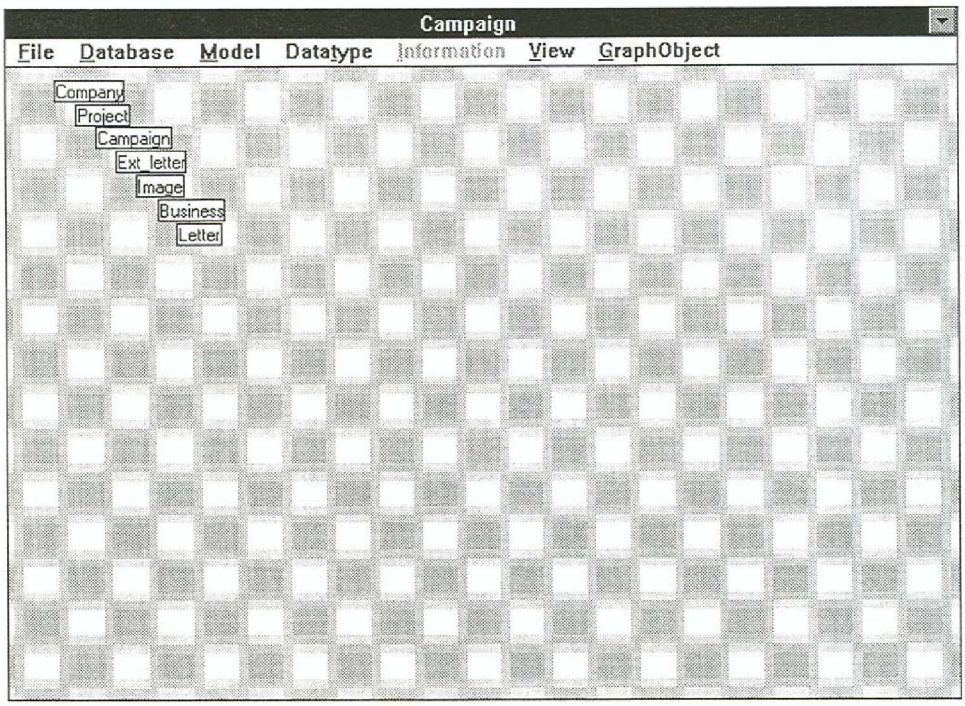

Figure 4.5. An initial conceptual model as generated by Designer tool. No relationships exist yet between entities, since they have to be defined manually.

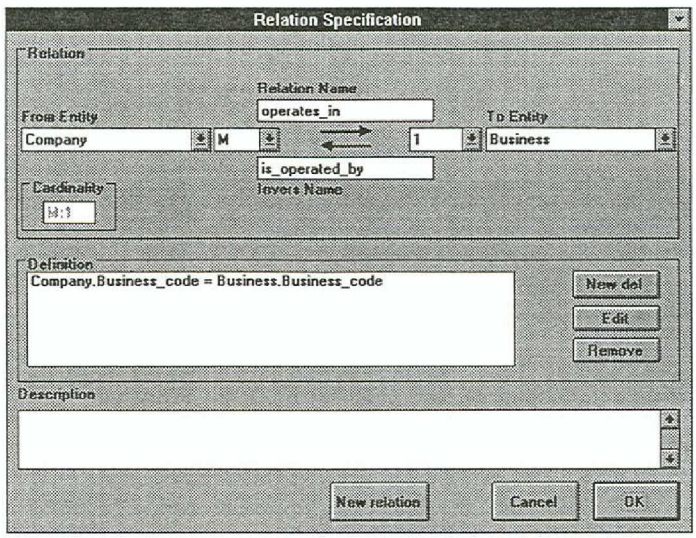

Figure 4.6. Relationship definition dialogue. The relationship is defined by giving the corresponding columns in the tables from which the entities originate.

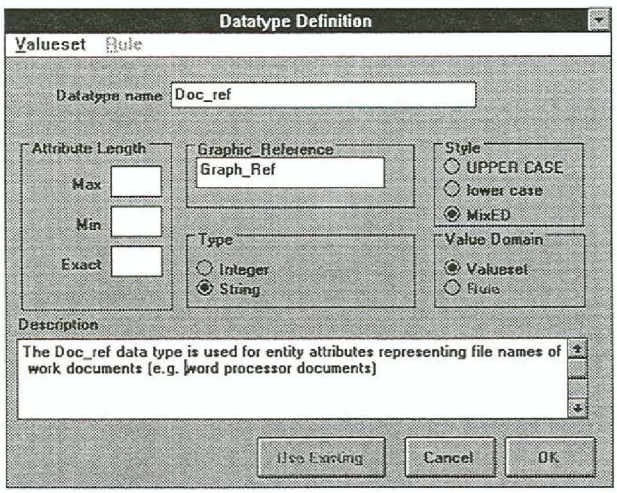

Figure 4.7. Data type definition dialogue. Data types and value sets for data types are defined for entity attributes to provide user interaction support and validation attribute values.

5. Refine the model layout if necessary.
6. Analyse the users information needs in their working tasks and group tasks with similar information needs.

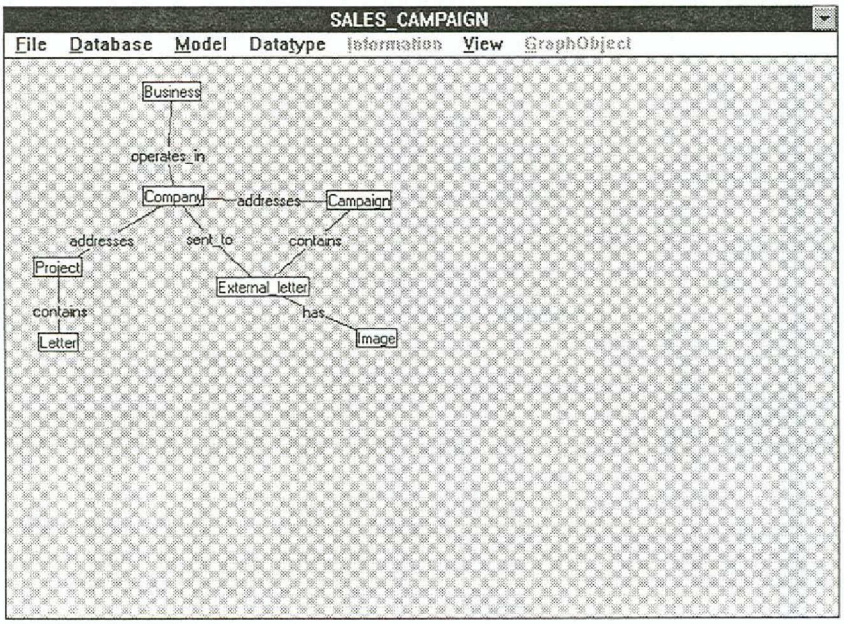

Figure 4.8. The conceptual model as it appears after definition of relationships and some layout work.

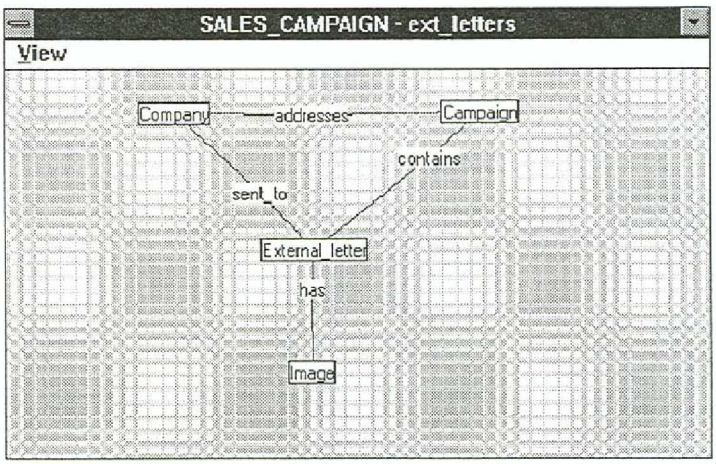

Figure 4.9. A view containing a part of the conceptual model.

7. Design views to the entire conceptual model so that each view cover the information needs of a specific user task or task group. This step should be performed with great care and in close co-operation with the users. A well designed set of views is probably the most important factor that makes the user interaction with an Intuitive application easy and efficient, thereby improving the user productivity. Having too large views cluttered with information inhibits users' perception of and navigation in presented information. Having too many small views makes it difficult to find the right view and

also restrict the user's possibilities to formulate queries expressing unexpected or uncommon information needs.

8. Queries expected or known already during the design phase are specified as predefined queries in the Designer tool. However, frequency of query usage should be high enough to motivate a predefined query. An unnecessarily large set of queries does not ease the users' interaction with an Intuitive application, especially if many of the queries are unclear, inconsistently named or rarely used. This is also true for ordinary views.

   Note: The currently available set of Intuitive tool prototypes [Wingstedt93] does not support predefined queries. Neither does the Designer tool support specification of forms or predefined queries. Accordingly, this step cannot be performed using the existing tool prototypes.

9. Specify the visual appearance of the elements of views and predefined queries. Entities may be presented most intuitively as icons or pictures instead of plain boxes with text. For example, an icon or a picture of a car may be a better visualisation of entity "Car" than a labelled rectangle among ten other labelled rectangles for other entities.

10. Install, test and verify the Intuitive application. Use real business data whenever possible. Re-do earlier steps to make the necessary modifications stemming from knowledge gained during testing and verifying.

Since the last step may initiate any other step at any time, it actually covers the maintenance of an Intuitive application, i.e. further development, enhancement or modifications to reflect changes in business information needs, databases, user tasks or in visual interface requirements.

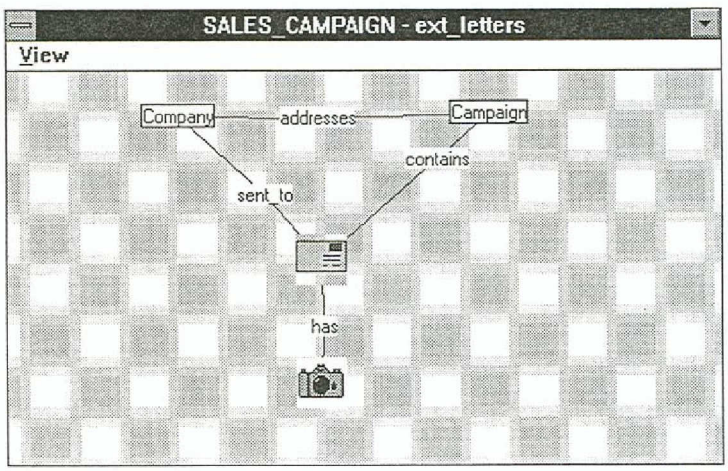

Figure 4.10. A view with an alternative visualisation of entities.

5. Future Work

The Designer tool will be subject to enhancements and refinements. We do not expect any significant changes in the general tool architecture, rather minor modifications because of problems encountered and additional functional requirements. The functional specification is largely derived from support needed in the design process, and because the design process will continue to evolve as we gain more knowledge and experience, the needs and tool support requirements will change. We believe however, that there will not be any radical changes in the functional requirements.

Not only the design process alone put requirements on the Designer tool. Other Intuitive tools need certain types of information that must be provided by the designer tool. Changed or extended functionality in other Intuitive tools may cause additional functional requirements in the Designer Tool. For example, we do not yet exactly know what kind of presentation information in the dictionary will be necessary for the general Presenter tool, especially when it comes to data composed of voice, video, pictures, text etc. What we do know is that if some of that presentation information is known at design time, the Designer tool should be able to handle it.

The current prototype does not meet all of the functional requirements. The next Designer tool version will be extended with support for import of database schemes and generation of table and column definitions. Also, the support for dictionary analysis and design maintenance should be improved. The current version does not allow specification of forms and predefined queries, which the next version should be able to do.

To conclude the expected work on the designer tool to be carried out in the future, we summarise it as follows:

- Better conversion between column data types and attribute data types.
- Import of database schemes, at least in standard SQL.
- Implementation of support for predefined queries (eventually also forms) if the functional requirements will be established.
- Addition of important dictionary analysis functions.
- Improvements in the interaction interface (drawing, layout of graphical objects, more efficient and flexible dialogues and forms).
- Conversion from Visual Basic to C++ using existing class libraries for applications development.

There are at least two wild cards: the functionality needed for specification of presentation information for the general Presenter tool and specification of moving pictures and voice data. We do not expect at the principal problems with these issues will be solved soon enough to allow implementation of any advanced support during the next six months.

6. Summary

In this paper we have described and discussed the functional requirements on the tool support for the process of developing Intuitive applications. The functional requirements can be summarised as follows

- Create and manage description of the physical structure (database schema) of corporate databases.
- Support creation of conceptual models and views describing the structure and contents of databases in terms of business concepts.
- Manage and present information about mappings between the physical database structure (tables, fields, relationships) and the conceptual model (entities, attributes, relations).
- Support specification of end user information and interaction support, e.g. entity and attribute definitions, explanations and rules that restrict and validate attribute values.
- Support the design of the visual appearance of the conceptual model and its views.
- Support specification of predefined queries.

A first working prototype that meets the most of these requirements has been developed at SISU, except support for predefined queries. Work remains however to make the Designer tool a complete and fully functional tool for application design and maintenance. Practical usage of the Designer tool has been demonstrated in this paper in the context of describing the application design process. The work to be carried out during the next six months can be summarised according to the following list.

- Better conversion between column data types and attribute data types.
- Provided that the functional requirements will be established in time during the period, implementation of support for canned queries.
- Addition of important dictionary analysis functions.
- Enhancing the interaction interface of the Designer tool.
- Moving from the programming environment during the first prototype phase (Visual Basic) to C++ and employing existing software components and advanced class libraries.

7. Acknowledgements

The Intuitive project is sponsored by NUTEK, Telia and Sweden Post.

Significant parts of Designer Tool prototype were developed by Hans Hogedahl, Sweden Post.

8. References

[Bern93] M. Bern, P. Kool, P. Rosengren, U. Wingstedt, "Application Design with the Intuitive Tools - two case studies", SISU Report No. 5.

[Rosengren93a] P. Rosengren, U. Wingstedt, M. Bern, P. Kool, "ER-Based Information Retrieval In a Mixed Database Environment", Proceedings of the 12:th International Conference of Entity Relationship Approach, 1993 (to be published).

[Rosengren93b] P. Rosengren, U. Wingstedt, M. Bern, P. Kool, "A Tools Oriented Visual Interface for Multimedia Databases", NDA'93, International Symposium on Next Generation Database Systems and Their Applications, Japan September 1993.

[Rosengren93c] P. Rosengren, U. Wingstedt, P. Kool, M. Bern, "Accessing Information in Large Corporate Databases - The Intuitive Approach", SISU Report No. 3.

[Rosengren93d] P. Rosengren, "Applications of a Multimedia Retrieval Information System - five case studies", SISU Document 10.

[Rosengren93e] P. Rosengren, "CQL- A query language for retrieving multimedia information", SISU Document 12.

[Wingstedt93] U. Wingstedt, M. Bern, P. Kool, P. Rosengren, "Intuitive Tools for Information Retrieval - Requirements and Architecture", SISU Report No. 4.

SVENSKA INSTITUTET FÖR SYSTEMUTVECKLING
SISU

Electrum 212, 164 40 Kista
Isafjordsgatan 26
Telefon 08-752 16 00 Telefax 08-752 68 00