Report No. 3
December 1993

Accessing Information in Large Corporate Databases

– The Intuitive Approach

Peter Rosengren
Ulf Wingstedt
Peeter Kool
Marie Bern

Swedish Institute for Systems Development

SISU

Accessing Information in Large Corporate Databases – The Intuitive Approach[1]

Peter Rosengren

Ulf Wingstedt

Peeter Kool

Marie Bern

Swedish Institute for Systems Development (SISU)
Electrum 212, S-164 40 Kista, Sweden

Abstract

This report introduces the fundamental principles of the Intuitive project. The objective of the Intuitive project is to provide efficient and easy-to-use solutions to enable end users to access heterogeneous information in corporate databases. This report shows how results from three areas - DBMS Front Ends, Information Retrieval and Hypertext - are integrated into a system that provides the user with a natural and homogeneous system for accessing large corporate information resources.

[1] A shorter version of this report has published in the Proceedings of the 12:th International Conference on the Entity Relationship Approach, 1993.

ISSN 1103-1700
ISRN SISU-REP--03--SE

Contents

1. Introduction 1

2. Related Work 4

3. Intuitive Tools Approach 5

3.1 Hypertext Browsing with Conceptual Feedback 7
3.2 Integrated ER-model 9

4. Tools Description 12

4.1 Navigator 12
4.2 Selector 12
4.3 Browser 13
4.4 Presenter 13

5. Architecture 15

5.1 Data Dictionary 15
5.2 Link Database 16
5.3 Conceptual Query Language 16
5.4 Database Query Engine 19

6. Prototype 20

7. Future Work 24

References 25

1. Introduction

This report is the first in a series of reports from the Intuitive Project. Intuitive is an ESPRIT III project and these reports document the work carried out by SISU during the first year of the project.

The objective of the Intuitive project is to provide efficient and easy-to-use solutions to enable end users to access heterogeneous information in corporate databases. 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 by searching a text retrieval system. Additional sources of information may be picture and drawing archives. Each information source might have its own data model and access mechanism, or query language, causing severe usability problems.

The research and development of principles for accessing information has traditionally been carried out in three different areas – *DBMS Front Ends*, *Information Retrieval (IR)* and *Hypertext*.

Over the past years research into innovative and easy-to-use *DBMS front-ends* has been very active. Both visual interfaces and natural language interfaces have been proposed. The most common approach has been to use a Visual Query Language based on the Entity Relationship Model (ER). Batini et al [Batini91], Kuntz and Melchert [Kuntz89a], and Czedjo et al [Czedjo90] all provide overviews of contemporary research in this area. Kool et al complement these surveys with overviews of existing commercial products [Kool93a], [Kool93b].

Database systems impose an important restriction with regards to information access - a predefined data model completely describes the contents in the database. This means that the data in the database have already been interpreted according to the data model and that all user queries are expressed in terms of this data model. In this sense queries as well as data are *precise*.

In the *Information Retrieval* research area this restriction has been relaxed. Focus has been mainly on retrieving text documents, and since textual information is often ambiguous and can be interpreted differently, both queries and data are *vague* and *imprecise*. This has required that intelligent techniques be applied to interpret users' requests and to evaluate which documents match a request.

A third approach for accessing information has been introduced by the hypertext/hypermedia research [Conklin87]. In a hypermedia system information is structured into *nodes* and *links*. The nodes contain pieces of information and the links represent some kind of association between two

information items. A user retrieves information from a hypermedia system by browsing through the hyperdocument, inspecting each node of interest until he has found the information he is looking for. Hypertext seems to be a promising solution for handling large amounts of unstructured information, but hypertext systems suffer from severe management problems if applied in large organisations with many users.

However, as was explained earlier, a user might be faced with all three types of system. We therefore need to consider a user's total need for accessing information in his daily work. Within the Intuitive Project our aim is to combine the *structuring and querying* principles from the database world with *intelligent dialogue techniques* from Information Retrieval and *exploratory search* from hypertext systems.

The partners in the Intuitive Project are:

- Cap Gemini Innovation, France
- INRIA, France
- SISU, Sweden
- Lloyds Register, UK
- City University, UK
- Brameur, UK
- Ibermatica, Spain

This report introduces the fundamental concepts of the Intuitive project and shows how we integrate results from the three areas - DBMS Front Ends, Information Retrieval and Hypertext - into a system that provides the user with a natural and homogeneous system for accessing large corporate information resources.

Within the project we are investigating users' and application requirements to define a generic architecture for Information Retrieval from heterogeneous databases. The four key components in the architecture of the Intuitive System are:

- Multimodal Interaction Manager
- End-User Tools
- Intelligent Dialogue Manager
- Data Access Layer

The role of the Multimodal Interaction Manager is to allow the user to input his request using a combination of speech and pointing. The End-User Tools provide the user with a visual interface and functionality for utilising large heterogeneous databases. The Dialogue Manager is responsible for interpreting user requests according to the user's task. Finally, the Data Access Layer is the glue that binds the information resources together.

The project also addresses 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 specific information retrieval applications.

This report concentrates on the ideas behind the End-User Tools and their underlying architecture. The End-User Tools support several types of interaction - formalised querying, exploratory ad-hoc browsing, schema navigation and entity inspection. We are suggesting a combined approach of Hyperlinks and ER-models where the ER-model will compensate for the navigation problems from which most hypertext systems suffer.

Thus, we believe that the combination of ER-based formal querying and hyperlinks will provide increased support for information access.

Rosengren et al have already given an overview of this work [Rosengren93a], [Rosengren93b]. Wingstedt et al discuss the requirements for the functionality of the End-User Tools 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 report is organised as follows. Chapter 2 relates our work to existing research. Chapter 3 presents the Intuitive Tools approach, this leads into a more detailed description of each tool in Chapter 4. In Chapter 5 the architecture that the End-User Tools are built upon is presented, this includes a description of the Dictionary and the Conceptual Query Language used to represent user queries. To validate and test our approach a prototype has been developed within the first year of the project. This prototype is described in Chapter 6. Finally, Chapter 7 describes our future work.

2. Related Work

Our research has been inspired by many different sources. Research on ER-based query interfaces started in the early eighties. Two of the first research systems to use an ER-schema as a main component in the user interface were GUIDE [Wong82] and gql/ER [Zhang83]. Another pioneering approach for database retrieval was presented by Fogg in combining ER-models with a browsing interface [Fogg84]. The list of early contributions is a long one, some examples are [Larson84], [Elmasri85], [Catarci87], [Rogers87].

More recent research makes use of other data models such as the universal relation model [Kim88] or extensions of the ER-model with specialisation, generalisation and aggregation. Examples of information retrieval systems exploiting extended ER-models are Candid [Schneider89], Pasta-3 [Kuntz89a], [Kuntz89b], and Super [Auddino91]. Czedjo et al also describe a query interface based on extended ER-models [Czedjo90]. Visual querying and browsing interfaces for object-oriented data models have also been proposed [Leong89], [Staes91]. The reader is further referred to the surveys mentioned in the introduction.

Our architecture has been inspired by the hypermedia system Intermedia [Yankelovich88]. Its major advantage is that it separates the link structure from the document database. This makes it possible to have different sets of links, called *webs*, for different users. Although Intermedia is designed for hypermedia applications many ideas can be brought into a database environment.

Our idea to combine hyperlinks and ER-models stems from an earlier collaboration project between Swedish Institute for Systems Development (SISU) and Infologics. In that project an electronic manual was represented by an ER-schema. Users could traverse the manual by following hyperlinks while getting feedback in the ER-schema on the concept they were reading about [Andersson93].

Our main motivation for carrying out this research comes from the Hybris Project in which an ER-based graphical query tool for SQL-databases was developed [Lundh89]. Since 1990 this tool has been in operation at Swedish Telecom. Experience gained from that and results from various evaluations have shown that ER-based query interfaces are very promising from a usability point of view. Users report that they understand the information structure much better and that they feel more secure when retrieving information from the database [Sahlin90].

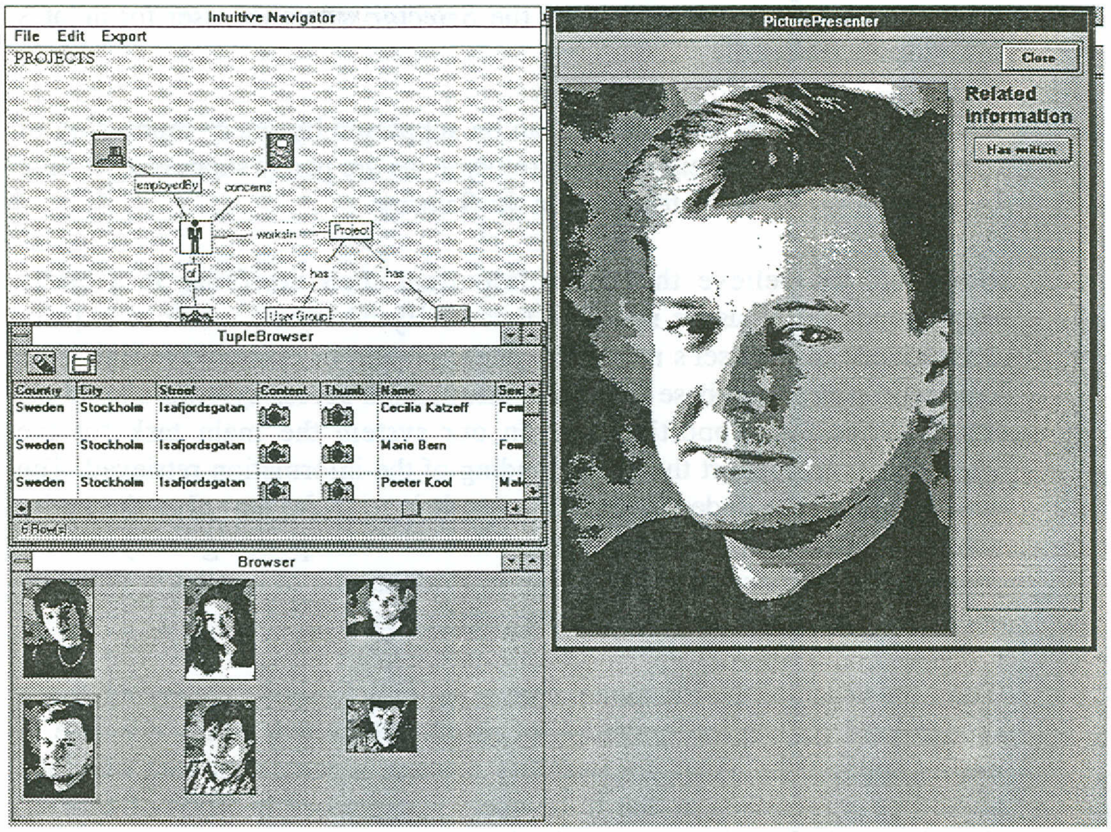
3. Intuitive Tools Approach

In this chapter we will describe the key concepts in our approach and how a user works with the system. Batini et al divide the user query process into three steps - *understanding*, *query formulation* and *testing* and survey various strategies proposed in research systems related to this model [Batini91]. Understanding is defined as identifying which concepts of the schema are useful for the query, query formulation is to formally express the operands involved in the query, with their related operators, and testing is to verify that the query expression precisely matches the user intention. The same model applies for multimedia databases and we will use it to relate our approach to existing research.

Most Visual Query Systems imply a system model where the user first navigates in the ER-schema and selects a subschema that specifies the entity classes of interest and the appropriate relationships. Then, the user adds attribute constraints to further restrict the search. Finally, the query is evaluated by the system and the results are presented on the screen.

However, in order to support access to multimedia objects, it is necessary to allow a much more flexible user interaction with support for ad-hoc exploratory searches of the database's contents in combination with formal querying.

In our system a user works with a set of four tools that are closely integrated with each other. The four tools are - *Navigator, Browser, Selector* and *Presenter*. The figure below gives an idea of how the default user interface for the tools looks:



Country	City	Street	Content	Thumb	Name	Sex
Sweden	Stockholm	Isafjordsgatan	📷	📷	Cecilia Katzeff	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Maria Bern	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Peeter Kool	Mal

Figure 3.1 The Navigator, Browser and Presenter. The Selector is hidden behind the Presenter.

The *Navigator's* main task is to enable the user to express his or her information requirements by providing an overview of the available classes of information, their relationships, and their semantic definitions. In that way the Navigator implements schema browsing or *intensional browsing* as defined by Batini et al. The Navigator will guide and encourage the user to explore data abstractions and meta-data to obtain a general understanding of the contents of the databases.

The role of the *Browser* is to give the user the ability to navigate (browse) at an entity instance level as opposed to the conceptual level. This corresponds to *extensional browsing* as defined by Batini et al. The Browser is also responsible for giving an understandable and interpretable view of a retrieved entity set to the user.

In this way our system supports the user in *understanding* both with intensional and extensional browsing. But note that in the case of multimedia databases, browsing is also an important technique for searching for information since users seem to be using this technique intuitively. For instance, to select a picture from a picture library usually involves selecting a subject and afterwards browsing through the picture instances in order to find the one required.

Query formulation is supported by the *Selector* where the user formulates his information need.

According to Batini et al *testing* can be supported by query rephrasing or query visualisation. In our current prototype we are partly supporting testing by giving a visual query feedback in the Selector.

However, we believe that more important than verifying if a query corresponds to the user's intention is to verify whether a result of a query corresponds to the user's need. The interpretation of retrieved results is an important part of database usage and information access systems should offer facilities for supporting this. In our system the main task for the *Presenter* is to support the understanding of the information retrieved. The first step towards understanding the result is to make sure that the entity shown is indicated in the Browser and that the corresponding entity class is indicated in the Navigator and Selector. Presentation of entities is done based on the presentation model for the entity class.

Thus, the usage of the ER-model here is not only limited to querying, but it is also used for the interpretation of results which was rarely the case in earlier approaches. Also, for usability reasons, a consistent user interface should be emphasised in that if the query is expressed in terms of entities, then the result should be presented using the same concepts.

3.1 Hypertext Browsing with Conceptual Feedback

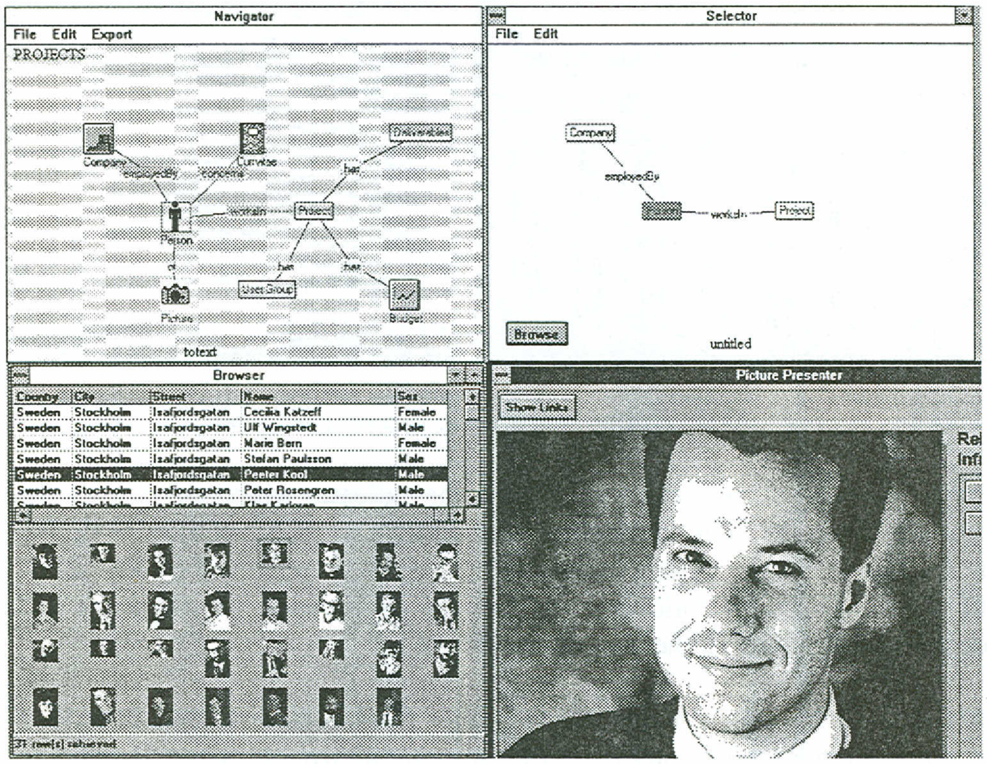
In our system we are not only trying to support structured querying but also exploratory search. The exploratory search that we consider is characterised by a user that wants to know as much as possible about a subject but does not know exactly what he wants to know until he sees what he can get. This kind of search is supported in our system by hyperlinks between the information instances rendered in the Presenter tool.

A major usability problem in hyperlink systems is the navigation among the instances [Katzeff92]. Without a visible structure it is difficult for the user to understand how the information is structured. In our case we already have a structure imposed on the information in the ER-schema displayed in the Navigator. This navigation map is an integrated part of our system and will compensate for the drawbacks of the hyperlink system, while letting the user benefit from support for exploratory searches.

Using the four tools, it is possible to click at an entity class in the Navigator and drag it into the Browser for immediate browsing of entities in that class. By doing so the user can go on to inspect each entity in detail in the Presenter and choose to follow associative links directly between entities. In that way the user can wander away from the starting entity class into other classes, for instance in reading a document which belongs to the attribute class "Project Description" the user might follow a hyperlink which leads him into a document belonging to attribute class "Person Curriculum Vitae" and from that the user might end up in a Picture belonging to attribute class "Person Description".

In this type of interaction the Navigator will keep track of the entity classes visited and always give the user feedback on which entity class he is visiting at that time. See Figure 3.2 and 3.3.

In Figure 3.2, the user has first selected the sub schema Person employed by Company, Person works in Project, from the Navigator (top, left) and exported it to the Selector (top, right). In the Selector, he clicks the Browse button and the result of the query is displayed in the Browser (bottom, left) that shows both textual data in a text Browser and thumbnail pictures in a picture Browser.

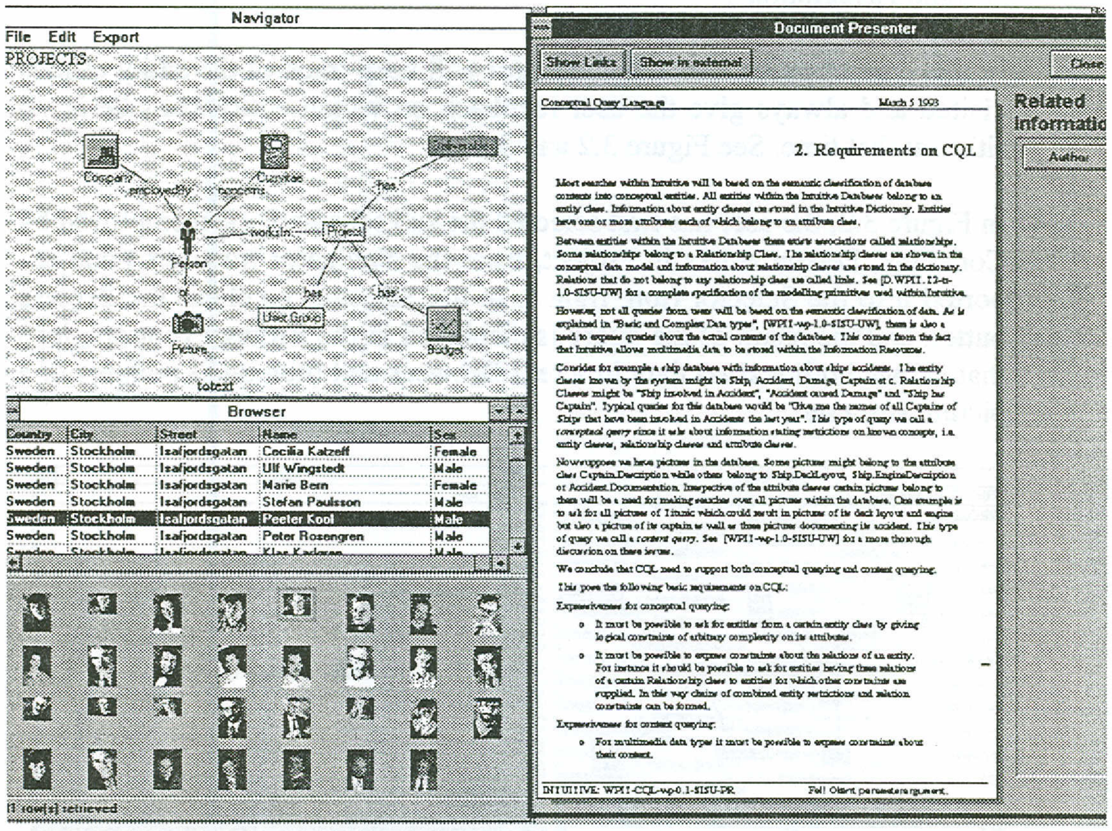


Country	City	Street	Name	Sex
Sweden	Stockholm	Isafjordsgatan	Cecilia Katzeff	Female
Sweden	Stockholm	Isafjordsgatan	Ulf Wingstedt	Male
Sweden	Stockholm	Isafjordsgatan	Marie Bern	Female
Sweden	Stockholm	Isafjordsgatan	Stefan Paulsson	Male
Sweden	Stockholm	Isafjordsgatan	Peeter Kool	Male
Sweden	Stockholm	Isafjordsgatan	Peter Rosengren	Male

Fig. 3.2 Example of system usage

In the Browser, the user has asked for presentation of a certain entity of Person. The Presenter then presents the entity according to its presentation model, i.e. displays the picture of the person in this case. At the same time, the corresponding entity classes in the Navigator and Selector are highlighted.

The user then asked the Presenter to follow a link displayed with the picture. The link, in this case, connects a document to the picture (see Fig. 3.3).



Country	City	Street	Name	Sex
Sweden	Stockholm	Isafjordsgatan	Cecilia Katzeff	Female
Sweden	Stockholm	Isafjordsgatan	Ulf Wingstedt	Male
Sweden	Stockholm	Isafjordsgatan	Marie Bern	Female
Sweden	Stockholm	Isafjordsgatan	Stefan Paulsson	Male
Sweden	Stockholm	Isafjordsgatan	Peeter Kool	Male
Sweden	Stockholm	Isafjordsgatan	Peter Rozengren	Male
Sweden	Stockholm	Isafjordsgatan	Klas Karlgren	Male

Conceptual Query Language March 5 1993

2. Requirements on CQL

Most searches within Intuitive will be based on the semantic classification of database contents into conceptual entities. All entities within the Intuitive Databases belong to an entity class. Information about entity classes are stored in the Intuitive Dictionary. Entities have one or more attributes each of which belong to an attribute class.

Between entities within the Intuitive Databases there exists associations called relationships. Some relationships belong to a Relationship Class. The relationship classes are shown in the conceptual data model and information about relationship classes are stored in the dictionary. Relations that do not belong to any relationship class are called links. See [D.WPI1.T2-tr-1.0-SISU-UW] for a complete specification of the modelling primitives used within Intuitive.

However, not all queries from users will be based on the semantic classification of data. As is explained in "Basic and Complex Data types", [WPI1-wp-1.0-SISU-UW], there is also a need to express queries about the actual contents of the database. This comes from the fact that Intuitive allows multimedia data to be stored within the Information Resources.

Consider for example a ship database with information about ships accidents. The entity classes known by the system might be Ship, Accident, Damage, Captain et c. Relationship Classes might be "Ship involved in Accident", "Accident caused Damage" and "Ship has Captain". Typical queries for this database would be "Give me the names of all Captains of Ships that have been involved in Accidents the last year". This type of query we call a *conceptual query* since it asks about information stating restrictions on known concepts, i.e. entity classes, relationship classes and attribute classes.

Now suppose we have pictures in the database. Some pictures might belong to the attribute class Captain.Description while others belong to Ship.DeckLayout, Ship.EngineDescription or Accident.Documentation. Irrespective of the attribute classes certain pictures belong to there will be a need for making searches over all pictures within the database. One example is to ask for all pictures of Titanic which could result in pictures of its deck layout and engine but also a picture of its captain as well as three pictures documenting its accident. This type of query we call a *content query*. See [WPI1-wp-1.0-SISU-UW] for a more thorough discussion on these issues.

We conclude that CQL need to support both conceptual querying and content querying.

This pose the following basic requirements on CQL:

Expressiveness for conceptual querying:

- It must be possible to ask for entities from a certain entity class by giving logical constraints of arbitrary complexity on its attributes.
- It must be possible to express constraints about the relations of an entity. For instance it should be possible to ask for entities having three relations of a certain Relationship class to entities for which other constraints are supplied. In this way chains of combined entity restrictions and relation constraints can be formed.

Expressiveness for content querying:

- For multimedia data types it must be possible to express constraints about their content.

INTUITIVE: WPI1-CQL-wp-0.1-SISU-PR Fel! Okänt parameterargument.

Fig. 3.3 Example of system usage (continued)

The document is retrieved and the Navigator indicates which entity class the document belongs to by highlighting the entity class Deliverable. Note that the document did not belong to the original result set retrieved by the query, but was retrieved by following the link. In such cases, the Browser will keep its position within the original result set.

This idea of conceptual feedback resembles the interaction style described by Andersson [Andersson93]. However, our approach is not limited to hypertext browsing but also allows for database querying based on the conceptual schema.

3.2 Integrated ER-model

As can be seen from the examples above, the ER-schema is used as a unifying framework for holding the database contents together and provides the user with an orientation and overview of the information space.

The extended ER-model is the glue of the system. It provides:

- a visual overview and explanation of the available information to the user
- a visual ER-query language for expressing database queries
- a description of the information space that allow translation of the ER-query language into database specific query languages
- a communication protocol for data exchange between different tools.

We assume a three schema level type of architecture - database level, conceptual level and presentation level, see Figure 3.4:

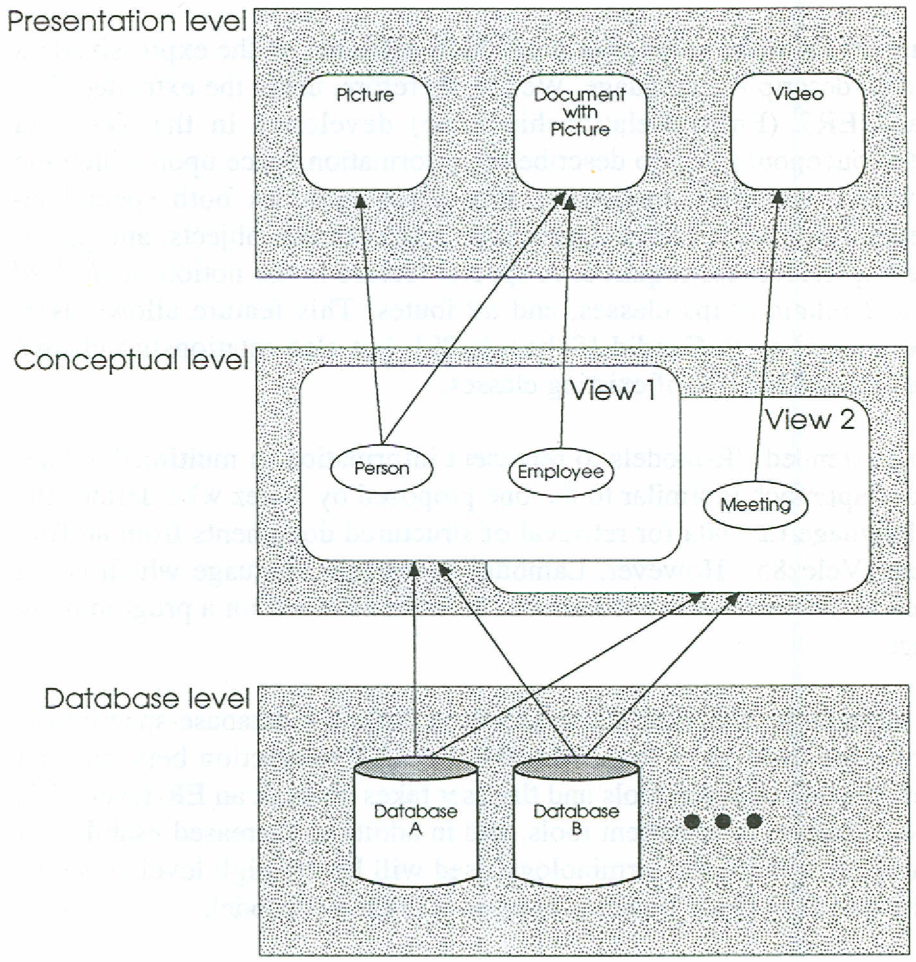


Fig. 3.4 The three schema levels of the architecture

At the *database level* each database is described by a logical database model and has a specific access language - examples are a relational database with SQL, a document category system with free text search and a picture database with keyword searches. The *conceptual level* constitutes the conceptual model of real world objects within a particular application. Information about an object at this level might be distributed over several databases, for instance information about a Project Entity could be project data stored in a relational database, project description found in a full text database and a picture with project participants in a picture database.

At the *presentation level* presentation aspects are described, i.e. depending on a user and his task a Project Entity can be presented at different levels of detail and with different types of presentation methods depending on the presentation model for that entity class.

This is the same type of architecture as described by Klas et al [Klas89]. However, their focus is to provide an object-oriented data model for multimedia databases, while we apply an ER-model for the conceptual level.

The support for multimedia data poses high demands on the expressiveness of the data description language. We are, therefore, using the extended ER-approach ERT (Entity-Relationship-Time) developed in the Tempora project [Loucopoulos91], to describe the information space upon which the tools work. The ERT modelling formalism includes both specialisation/generalisation as well as aggregation into complex objects, and provides the expressiveness required. A special feature is the notion of *derived* entity and relationships classes, and attributes. This feature allows user-defined entities as in Candid [Schneider89], but also relationship classes may be defined in terms of existing classes.

In using extended ER-models to represent information in multimedia databases our approach is similar to the one proposed by Velez who defines the query language Lambda for retrieval of structured documents from an ER-database [Velez85]. However, Lambda is a textual language which is not intended for the end-user, but rather to be embedded within a programming language.

In our system, the extended ER-model used hides the database-specific details from the tools as well as from the user. All interaction between tool components and between tools and the user takes place at an ER-level. This provides database independent tools, and in addition, increased usability in user interaction since the terminology used will be the high level concepts from the language of the business expressed in the ER-model.

4. Tools Description

This section gives an overview of how each individual tool works. It should be noted that here we describe the default versions of the four tools. As is explained by Bern et al and Rosengren et al each tool can be customised for a particular application with a user interface that is appropriate for that particular application [Bern93], [Rosengren93a]. A more detailed description of the tools is given by Wingstedt et al [Wingstedt93].

4.1 Navigator

As is recognised by, for instance, the developers of the SNAP-system [Bryce86], the single most important component in an ER-model management system is the visual presentation of the model. The basic visualisation technique used in the Navigator is a plain network model that shows a simplified subset of the complete ER-model, including entity and relationship classes.

In real business applications, the ER-model describing the data may be extensive and include a large number of different entity classes and attributes. In response to this, the Navigator supports division of the model into views that show pre-defined subsets of the complete model. These pre-defined views may be edited by the user.

While relying on the same ER-structure expressed in the ERT formalism, the visualisation used in the Navigator may vary. In fact, since the visualisation of the ER-model is independent of its structure, various visualisation techniques may be used depending on user profile. For instance, the same ER-model may be visualised as labelled boxes in a network schema or as icons on a map. The Navigator, nevertheless, will always include functionality for exporting sub-models to other tools.

4.2 Selector

The Selector implements the query formulation facilities of our system. For a visual query interface there are two choices - to use a *diagrammatic* approach or an *iconic* approach. The diagrammatic approach visualises the query as a subschema in a traditional way with boxes and arrows. In an iconic approach the user instead formulates visual query sentences using icons. Catarci et al discuss these issues in depth and also present a system where the user can choose between either a diagrammatic interface or an iconic interface [Catarci91].

The iconic interface is attractive but we believe that for real world applications there will be problems finding meaningful symbols to represent all important concepts, especially more abstract ones like "Project". Therefore we have chosen for our prototype a traditional diagrammatic interface which allows a user to edit and restrict a subschema for querying the database. However, in the future we will implement different visual languages and test whether they feel natural to users.

4.3 Browser

The main source of inspiration for the Browser is LID "Living in a database" [Fogg84] although with one important difference: LID uses browsing as the *main* technique for data selection; we do not think this is realistic strategy in real databases because of performance and usability issues. The number of instances of an entity class will rarely be possible to survey on the screen. Browsers in our system will therefore work mainly on retrieved result set, but it is still possible for the user to work in a LID-like manner.

Each of the entity classes in a result set is rendered as visually separated objects in the Browser user interface with the relationships connecting them together. The exact rendering depends on the type of data or media, pictures are rendered as a set of thumbnails, business data as records etc. Only distinct instances for each entity class are displayed. The interaction style is an extension of "Synchronised Browsing" in the Pasta-3 system [Kuntz89b]. When the user selects an instance, the related instances in other entity classes will be highlighted. The difference between Pasta-3 and our system is that we display all distinct instances of all entity classes in the result set whereas Pasta-3 displays only instances related to the one currently selected. However, in some situations our system will utilise the Pasta-3 type of synchronised browsing, for instance when the number of instances is too large.

4.4 Presenter

The Presenter decides on how to present an entity based on the *presentation model*, *user preferences* and information in the *link database* (see Chapter 5). Here we will briefly describe two presentation techniques employed to give an example of the many aspects that need to be considered when presenting multimedia data.

The logical organisation of data has considerable influence on how the user perceives and understands the information, especially when handling structured data. The presentation of structured data in our system has been inspired by Kerner and Thiel's box presentation form [Kerner91]. The objective of the box presentation form is to enhance the overview and understanding of a single tuple of structured data.

The presenter tool for text documents presents these on pages instead of the more common presentation as scrollable text. This means that there will be an almost exact correspondence between the on-screen appearance and the appearance of a paper print-out. This similarity will enable the user to benefit from recognition of previously seen documents and use the position cues to understand the document. The position cues will also enhance the search task when dealing with a familiar document structure [Nygren92].

We are combining two different methods of *link visualisation* in our prototype. The first link type is called overlay links and is visualised as a colour indicated area in the entity presented. The second link visualisation method is used to visualise associations that exist between two entities but is not described in the conceptual model. These links are visualised by buttons with descriptive names.

5. Architecture

In this section, we will give a short overview of the most important modules of the overall systems architecture. Our system consists of five main components:

- End-User Tools
- Data Dictionary
- Link Database
- Conceptual Query Language
- Database Query Engine

The End-User Tools were described in the previous chapter. Below, we describe the role of the other four components. Figure 5.1 shows the overall architecture upon which the tools are built.

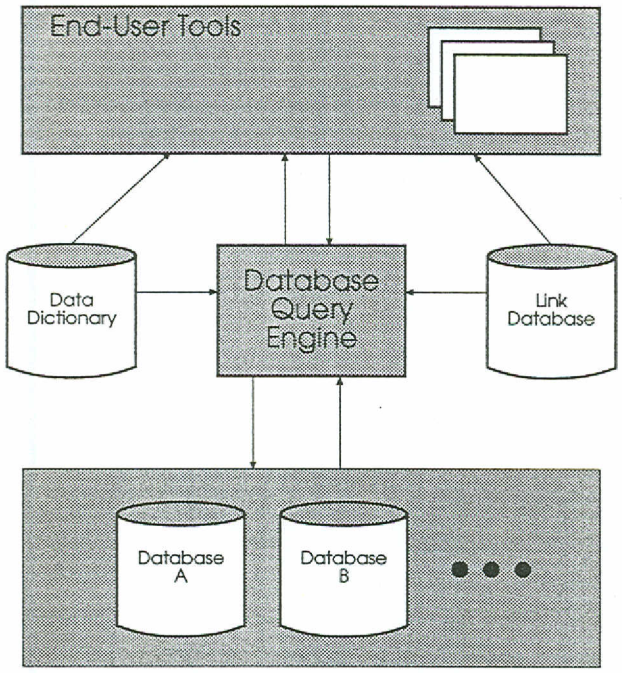


Fig. 5.1 Overall architecture

5.1 Data Dictionary

The ER-model and its mappings to the underlying database schemas are stored in a Data Dictionary accessible by all tools for their various purposes. The Data Dictionary also includes layout information for the visualisation of the model. Figure 5.2 shows the structure of the Dictionary. This also defines the meta information a systems designer needs to specify in order to get a working system. In fact, application building to a large extent consists of populating the Dictionary [Bern93].

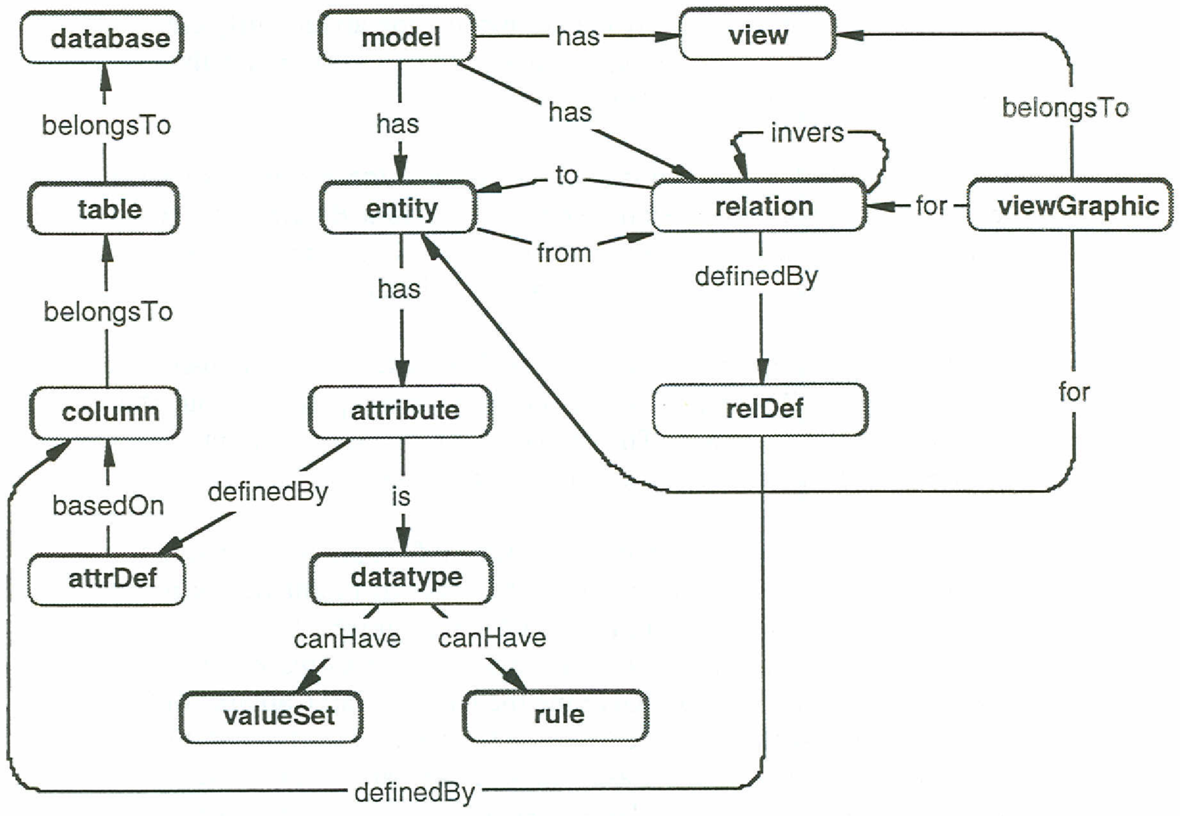


Figure 5.2 Dictionary Structure

5.2 Link Database

The links in our system are span-to-span links according to Halasz and Conklin's definition [Halasz90]. A span is defined as an area within a data instance, such as a picture. Each stored instance can contain several spans. The span table contains information about the position of the spans within their instances and how they should be indicated in the instance. Each span can have several links referring to it.

The hyperlinks are stored in a separate link database implemented as a relational table. The link table contains information on which spans are related, the meaning and purpose of the relation and the access rights.

5.3 Conceptual Query Language

The Conceptual Query Language (CQL) is the internal query language used within Intuitive. The purpose of CQL is to allow the Intuitive Tools and other modules such as the Dialogue Manager to represent user queries in terms of restrictions on entities, attributes, relationships and value contents.

Most searches within Intuitive will be based on the semantic classification of database contents into conceptual entities. All entities within the Intuitive Databases belong to an *Entity Class*. Information about entity classes is stored in the Intuitive Dictionary. Entities have one or more attributes each of which belongs to an *Attribute Class*.

Between entities within the Intuitive Databases there exist associations called relationships. Some relationships belong to a *Relationship Class*. The relationship classes are shown in the conceptual data model and information about relationship classes is stored in the dictionary.

However, not all queries from users will be based on the semantic classification of data. There is also a need to express queries about the actual contents of the database. This comes from the fact that Intuitive allows multimedia data to be stored within the databases.

Consider for example a ship database with information about shipping accidents. The entity classes known by the system might be *Ship*, *Accident*, *Damage*, *Captain* etc. Relationship Classes might be *Ship involved in Accident*, *Accident caused Damage* and *Ship has Captain*. Typical queries for this database would be "Give me the names of all Captains of Ships that have been involved in Accidents in the last year". This type of query we call a *conceptual query* since it asks about information stating restrictions on known concepts, i.e. entity classes, relationship classes and attribute classes.

Now suppose we have pictures in the database. Some pictures might belong to the attribute class *Captain.Description* while others belong to *Ship.DeckLayout*, *Ship.EngineDescription* or *Accident.Documentation*. Irrespective of the attribute classes to which certain pictures belong, there will be a need to make searches through all pictures within the database. One example is to ask for all pictures of The Titanic which could result in pictures of its deck layout and engine but also a picture of its captain as well as three pictures documenting its accident. This type of query we call a *content query*.

We will give some examples of how CQL works. Assume the following simple conceptual model:

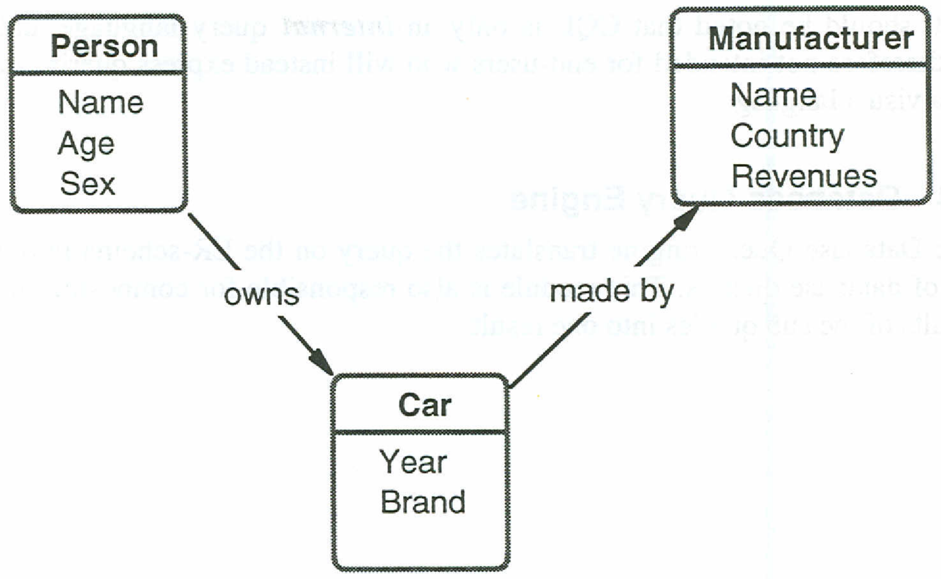


Figure 5.3 Simple conceptual model

Query: Give me all persons older than 20 years named Peter.

CQL-statement: Person(name=Peter AND age > 20)

Query: Give me all persons older than 20 years named Peter, that own a car made after 1985.

CQL-statement: Person(name=Peter AND age > 20 AND owns().Car(year > 1985))

Query: Give me all persons older than 20 years named Peter that own a car made before 1980 by a Japanese manufacturer.

CQL-statement: Person(name=Peter AND age > 20 AND owns().Car(year <1980).madeBy.Manufacturer(country = Japan))

Query: Give me all persons older than 20 years named Peter that own three cars made after 1985 and four cars made before 1980 by a Japanese manufacturer.

CQL-statement: Person(name = Peter AND age > 20 AND owns(3).Car(year > 1985) AND owns(4).Car(year < 1980 AND madeBy().Manufacturer(country = Japan))

It should be noted that CQL is only an *internal* query language and therefore not intended for end-users who will instead express queries in a visual language.

5.4 Database Query Engine

The Database Query Engine translates the query on the ER-schema into a set of database queries. This module is also responsible for composing the results of the sub queries into one result.

6. Prototype

To validate our approach we have built a generic prototype system which has been used for early evaluations to get user feedback. This generic prototype shows the four end-user tools in their default mode. The database chosen as an application for the generic prototype is a database with information about SISU, its projects, employees, documents, results etc. Bern et al give a more detailed description of this prototype and also two medical prototypes [Bern93].

The Navigator gives the user an overview of the information available within the databases. To the left in Figure 6.1, one way of visualising the information structure is shown, where icons are used to illustrate important concepts.

Once the user has navigated in the information structure and found the entity classes he is interested in, he selects the part of the model he wants for further querying. The entity classes of interest are exported to the selector where the user can add further constraints to restrict his query, for instance to add that he is only interested in pictures of persons working in companies in Stockholm. The Selector is shown to the right in Figure 6.1.

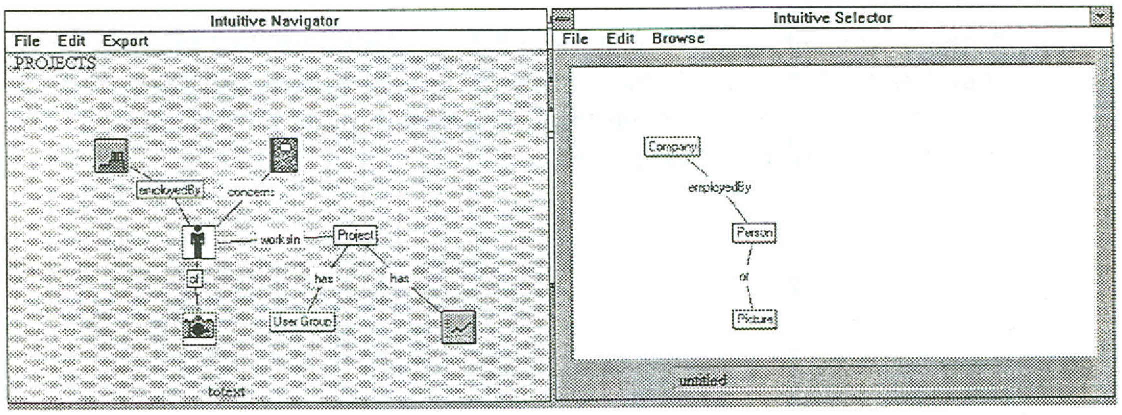
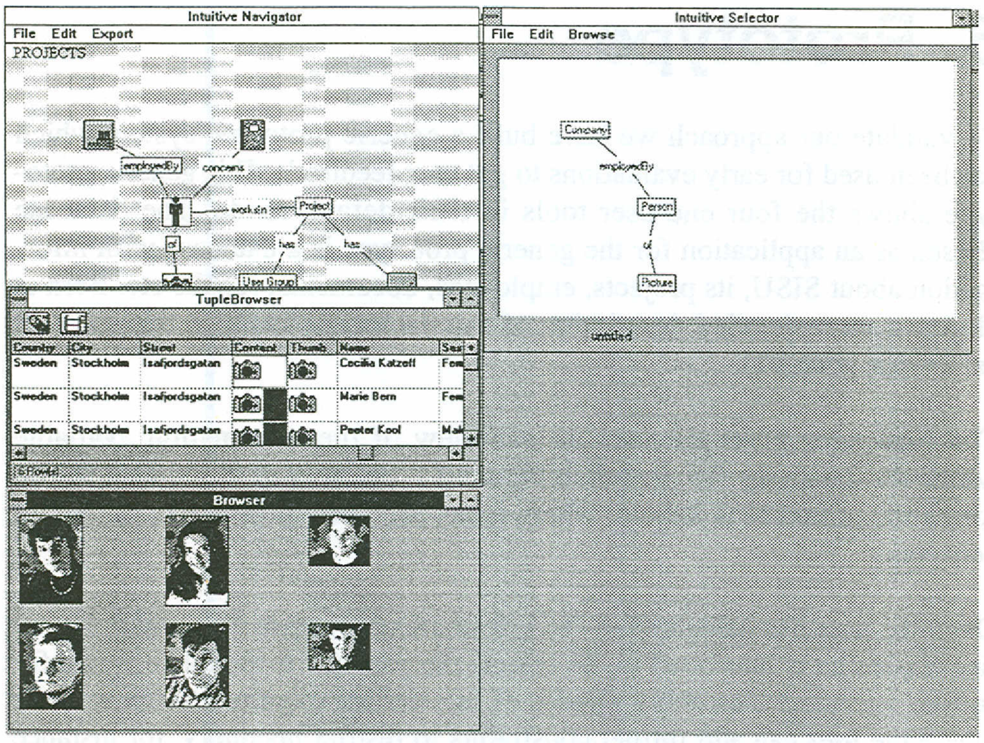


Figure 6.1 User formulates his query with a Selector.

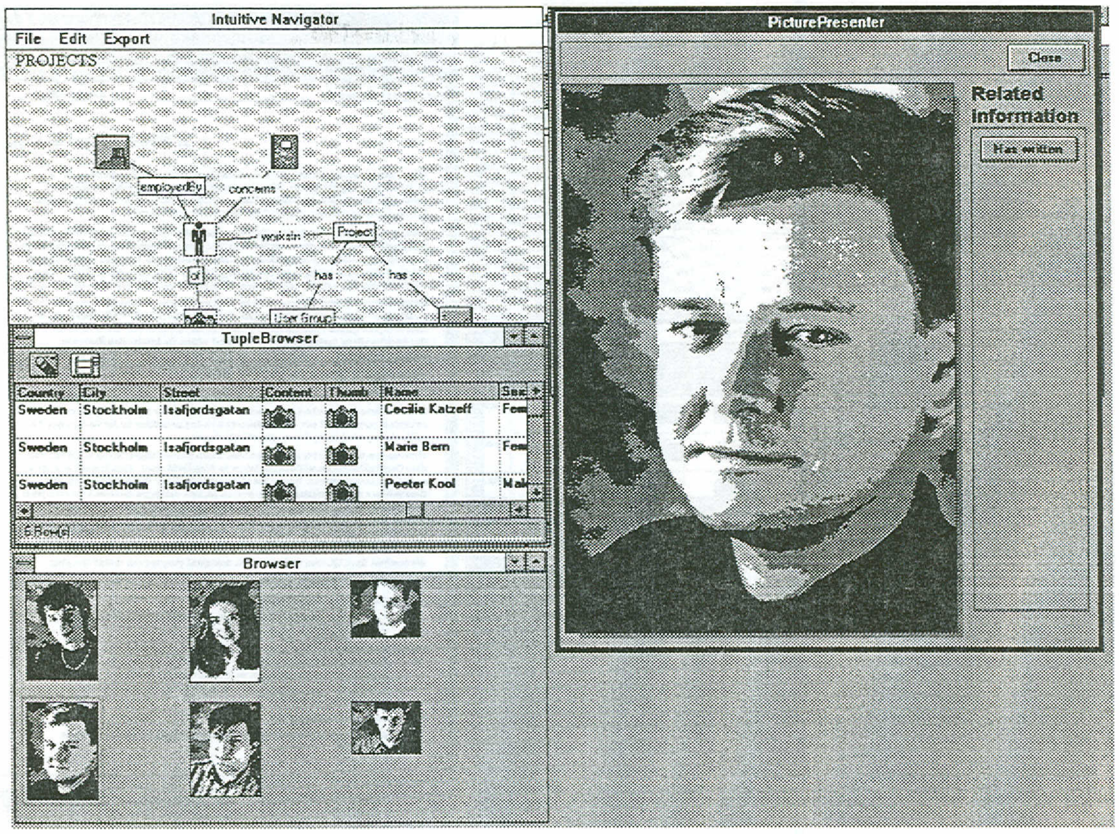
The query is sent to the database. The result is retrieved and displayed in a Browser, where pictures are indicated with camera icons. If the user double clicks on the column with camera icons, the corresponding pictures are displayed as miniatures, called *thumbnails*, see Figure 6.2.



Country	City	Street	Content	Thumb	Name	Sex
Sweden	Stockholm	Isafjordsgatan	📷	📷	Cecilia Katzeff	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Marie Bern	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Peeter Kool	Mal

Figure 6.2 The result of a query can be viewed in different browsers.

If the user wants to see a picture in full resolution he double clicks on a thumbnail. The picture is then displayed in a Presenter, see Figure 6.3. If related information exists, buttons for accessing that information will be displayed together with the picture.



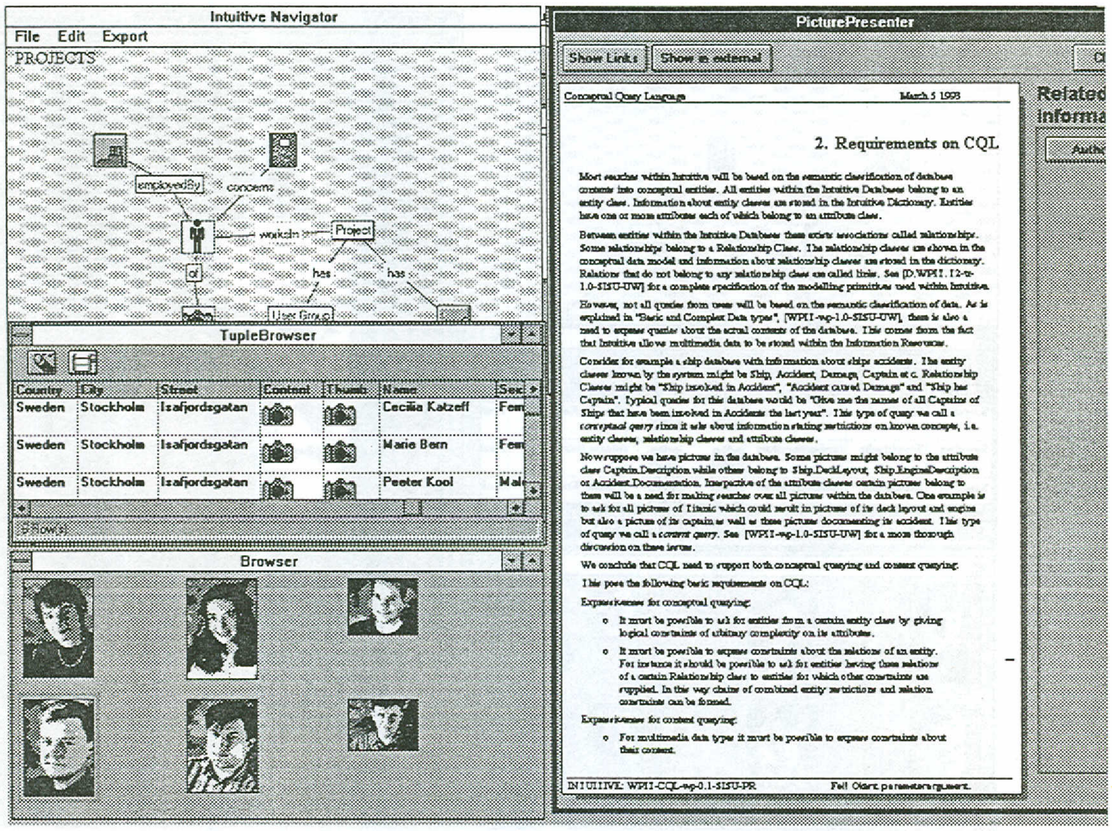
Country	City	Street	Content	Thumb	Name	Sex
Sweden	Stockholm	Isafjordsgatan	📷	📷	Cecilia Katzeff	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Marie Bern	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Peeter Kool	Mal

6 Row(s)

Figure 6.3 Data items can be viewed in full detail.

The user then chooses to follow a link from the picture to a related document by clicking the button "Has Written" in the "Related Information" column in the Presenter.

The document is then displayed in a new Presenter (Fig. 6.4). This document can be exported to Word or any other word processor if the user clicks on the button "Show in external".



Country	City	Street	Content	Thumb	Name	Sex
Sweden	Stockholm	Isafjordsgatan	📷	📷	Cecilia Katzeff	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Marie Bern	Fem
Sweden	Stockholm	Isafjordsgatan	📷	📷	Peeter Kool	Mal

3 Row(s)

Conceptual Query Language March 5 1993

2. Requirements on CQL

Most searches within Intuitive will be based on the semantic classification of database contents into conceptual entities. All entities within the Intuitive Databases belong to an entity class. Information about entity classes are stored in the Intuitive Dictionary. Entities have one or more attributes each of which belong to an attribute class.

Between entities within the Intuitive Databases there exists associations called relationships. Some relationships belong to a Relationship Class. The relationship classes are shown in the conceptual data model and information about relationship classes are stored in the dictionary. Relations that do not belong to any relationship class are called links. See [D.WPI1.12-tr-1.0-SISU-UW] for a complete specification of the modelling primitives used within Intuitive.

However, not all queries from users will be based on the semantic classification of data. As is explained in "Basic and Complex Data types", [WPI1-wp-1.0-SISU-UW], there is also a need to express queries about the actual contents of the database. This comes from the fact that Intuitive allows multimedia data to be stored within the Information Resources.

Consider for example a ship database with information about ships accidents. The entity classes known by the system might be Ship, Accident, Damage, Captain etc. Relationship Classes might be "Ship involved in Accident", "Accident caused Damage" and "Ship has Captain". Typical queries for this database would be "Give me the names of all Captains of Ships that have been involved in Accidents the last year". This type of query we call a *conceptual query* since it asks about information stating restrictions on known concepts, i.e. entity classes, relationship classes and attribute classes.

Now suppose we have pictures in the database. Some pictures might belong to the attribute class Captain.Description while others belong to Ship.DeckLayout, Ship.EngineDescription or Accident.Documentation. Irrespective of the attribute classes certain pictures belong to there will be a need for making searches over all pictures within the database. One example is to ask for all pictures of Titanic which could result in pictures of its deck layout and engine but also a picture of its captain as well as those pictures documenting its accident. This type of query we call a *content query*. See [WPI1-wp-1.0-SISU-UW] for a more thorough discussion on these issues.

We conclude that CQL need to support both conceptual querying and content querying.

This pose the following basic requirements on CQL:

Expressiveness for conceptual querying:

- It must be possible to ask for entities from a certain entity class by giving logical constraints of arbitrary complexity on its attributes.
- It must be possible to express constraints about the relations of an entity. For instance it should be possible to ask for entities having these relations of a certain Relationship class to entities for which other constraints are supplied. In this way chains of combined entity restrictions and relation constraints can be formed.

Expressiveness for content querying:

- For multimedia data types it must be possible to express constraints about their content.

INTUITIVE: WPI1-CQL-wp-0.1-SISU-PR Fel! Okänt parameterargument.

Figure 6.4 The user followed a link from a picture which lead him to a related document.

7. Future Work

Within the Intuitive Project we are building two prototypes - The Generic Prototype and a Medical Demonstrator. Besides that, we are currently evaluating our approach by defining and building three other applications - a Knowledge Databank for Salespersons, a Multimedia Repository Interface and a combined Customer Database/File System. The experience gained from these will give us valuable feedback on the design of our system.

Due to the flexible and modular architecture of our system we will be able to experiment with different approaches for individual tools. That allows us easily to test different schema visualisation techniques within the Navigator without affecting the rest of the system. We will also experiment with different visual languages in the Selector without changing the rest of the system. All these opportunities will be exploited in a series of cognitive and usability tests.

We also need to address the needs of the application designer. To give support for fast and efficient construction of highly usable ER-based Information Retrieval Systems we need to develop a design methodology that is easy to follow for application developers. For further discussion about these issues see [Rosengren93a], [Bern93].

One research area not discussed in this paper is how the internal contents of multimedia objects can be described and modelled. This involves modelling of both the internal conceptual structure of a multimedia object and modelling of its presentation aspects. For an enlightening discussion about these issues, see [Klas89].

References

[Andersson93]	J. Andersson, "Using Conceptual Models in Hypermedia, lessons learned", Multimedia-System Architectures and Applications, eds. J. Encarnacao, J. Foley, Springer Verlag, 1993.
[Auddino91]	A. Audinno, Y. Denneboúy, Y. Dupont, E. Fontana, S. Spaccapietra, Z. Tari. "Super: A comprehensive approach to DBMS Visual User Interfaces", Ecole Polytechnique Fédérale, Lausanne, Switzerland.
[Batini91]	C. Batini, T. Catarci, M. F. Costabile, S. Levialdi, "Visual Strategies for Querying Databases", IEEE Workshop on Visual Languages, 1991, pp 183-189.
[Bern93]	M. Bern, P. Kool, P. Rosengren, U. Wingstedt, "Application Design with the Intuitive Tools - two case studies", SISU Report No. 5, December 1993.
[Catarci88]	T. Catarci, G. Santucci, "Query by Diagram: A Graphic Query System", Proceedings of the 7:th International Conference of Entity Relationship Approach, pp 157-174, 1988.
[Catarci91]	T. Catarci, A. Massari, G. Santucci, "Iconic and Diagrammatic Interfaces: An Integrated Approach", IEEE Workshop on Visual Languages, 1991, pp 199-204.
[Conklin87]	J. Conklin, "Hypertext: An Introduction and Survey", IEEE Computer, September 1987.
[Czedjo90]	B. Czejdo, R. Elmasri, M. Rusinkiewicz, D. Embley, "A Graphical Data Manipulation Language for an Extended Entity-Relationship Model", IEEE Computer, March 1990

[Elmasri85] R. Elmasri, J. Larson, "A Graphical Query Facility for ER Databases", Proceedings of the 4:th International Conference of Entity Relationship Approach, 1985.

[Fogg84] D. Fogg, "Lessons from a "Living in a Database" Graphical Query Interface", Proceedings ACM Sigmod, 1984.

[Halasz90] F. Halasz, J. Conklin, "Issues in the Design and Application of Hypermedia Systems", Tutorial CHI '90 Human Factors in Computing Systems, 1990.

[Katzeff92] C. Katzeff. "Överblicksproblemet i hyper-media", SISU Technical Report 18, in Swedish, 1992.

[Kim88] H. J. Kim, H. F. Korth, A. Silberschatz, "PICASSO: A Graphical Query Language", Software - Practice and Experience, March 1988.

[Klas89] W. Klas, E. J. Neuhold, M. Schrefl, "Visual Databases need Data Models for Multimedia Data", Visual Database Systems, T. L. Kunii (editor), Elsevier Science Publishers B. V. (North-Holland), IFIP 1989.

[Kool94a] P. Kool, P. Rosengren, U. Wingstedt, "Program för sökning i databaser - en marknadsöversikt", Triadrapport 18/94.

[Kool94b] P. Kool, P. Rosengren, U. Wingstedt, "Att nå och förstå data - möjligheter och be-gränsningar", Triadrapport 19/94.

[Kuntz89a] M. Kuntz, R. Melchert, "Pasta-3´s Graphical Query Language: Direct Manipulation, Cooperative Queries, Full Expressive Power", Proceedings of the 15:th International Conference on Very Large Databases, 1989.

[Kuntz89b]	M. Kuntz, R. Melchert, "Ergonomic Schema Design and Browsing with more Semantics in the Pasta-3 Interface for E-R DBMSs", Proceedings of the 8:th International Conference of Entity Relationship Approach, 1989.
[Larson84]	J. Larson, J. B Wallick, "An Interface for Novice and Infrequent Database Management System Users", AFIPS Conference Proceedings, National Computer Conference, 1984.
[Leong89]	M. Leong, S. Sam, D. Narasimhalu, "Towards a Visual Language for an Object-Oriented Multi-Media Database System", Visual Database Systems, T. L. Kunii (editor), Elsevier Science Publishers B. V. (North-Holland), IFIP 1989.
[Loucopoulos91]	P. Loucopoulos, B. Wangler, P. McBrien, F. Schumacker, B. Theodoulidis, V. Kopanas, "Integrating Database Technology, Rule Based Systems and Temporal Reasoning for Effective Information Systems: The Tempora Paradigm", Information Systems Journal, Vol. 1, No. 1, April 1991.
[Lundh89]	J. Lundh, P. Rosengren,, "Hybris - A first step towards efficient information resource management", SISU report no. 5, 1989.
[Nygren92]	E. Nygren, M. Lind, M. Johnson, B. Sandblad, "The Art of the Obvious", in CHI '92 Proceedings, ACM Press, 1992.
[Rogers87]	T. R Rogers, R. G. G Cattell, "Entity-Relationship Database User Interfaces", Proceedings of the 6:th International Conference of Entity Relationship Approach, 1987.

[Rosengren93a]	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.
[Rosengren93b]	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.
[Sahlin90]	C. Sahlin, "Erfarenheter från användning av Hybris. - Ett multimedia hjälpmedel för navigering i Televerkets PULS databas", Försvarets Forskningsanstalt, Linköping, Sweden, report 90-5991/S. In Swedish.
[Schneider89]	M. Schneider, C. Trepied, "A Graphical Query Language Based on an Extended E-R Model", Proceedings of the 8:th International Conference of Entity Relationship Approach, 1989.
[Staes91]	F. Staes, L. Tarantino, A. Times, "A Graphical Query Language for Object-Oriented Databases", IEEE Workshop on Visual Languages, 1991, pp 199-204.
[Velez85]	F. Velez, "An Entity-Relationship Based Query Language for the Retrieval of Structured Documents", Proceedings of the 4:th International Conference of Entity Relationship Approach, 1985.
[Wingstedt93]	U. Wingstedt, M. Bern, P. Kool, P. Rosengren, "Intuitive Tools for Information Retrieval - Requirements and Architecture", SISU Report 4, December 1993.

[Wong82]	H. Wong, I. Kuo, "GUIDE: Graphical User Interface for Database Exploration", Proceedings of the 8:th International Conference on Very Large Databases, 1982.
[Yankelovich88]	N. Yankelovich, B.J Haan, N.K Meyrowitz, S.M Drucker, "Intermedia: The concept and construction of a seamless information environment", IEEE Computer, January 1988.
[Zhang83] Z.	Q Zhang, O. Mendelzon, "A graphical query language for entity-relationship databases", Entity-Relationship Approach to Software Engineering, pp 441-448, 1983.