EcrinsDesign: A Graphical Editor for Semantic Structures

Françoise Adreit* ** Michel Bonjour* (1)

* Université de Genève (Switzerland)

** Université de Montpellier (France)

Centre Universitaire d'Informatique (CUI)
12, rue du Lac
CH-1207 Genève (Switzerland)

francoise@macmail.unige.ch, bonjour@uni2a.unige.ch

(1) This work is part of the Rebirth project granted by the Swiss National Fund for Scientific Research (FNRS no. 1 603-0.87)

Abstract

The *EcrinsDesign* system is a graphical tool for data schema design. It is based on the Ecrins semantic data model. It supports interactive data schema design by direct manipulation of graphical objects. Its main contribution is the definition and the management of the dynamic aspects of the graphical representation: the type of each object may evolve, depending on the user's actions and conforming to a set of integrity rules.
The legibility of the edited schemas, the flexibility and the high level of interaction are the main characteristics of the system.
Moreover, the system is composed of an Ecrins Data Definition Language generator, enabling the communication with the target Ecrins/BD DBMS, which implements the Ecrins data model.
The *EcrinsDesign* system is implemented in Think C™ on the Apple® Macintosh™ II family computers.

1 Introduction

The design of a data schema is a complex task. The large amount of conceptual elements (attributes, relations and associations) composing a data schema complicates both its understanding and its control. Therefore, numerous tools have been developed to help the different people participating to the design of a data schema. Those tools have two main goals:

- To support the design process, by providing high-level models, integrity rules control, interfaces with target DBMS and so on.

- To represent a data schema in an expressive way, to permit the communication between the designer and other people.

Moreover, the nature of the design activity (creativeness, evolution, trail-error) has induced other requirements for design tools:

- High level of interaction.
- Strong support of evolution.
- Flexibility of use.

Endless, the user interface paradigm supported by most of the present workstations plays a great part in the definition of new features found in this kind of applications.

In this context, we have developed a graphics-based data schema editor called *"EcrinsDesign"* [Bonjour 89]. The system is based on a semantic data model, Ecrins [Junet 86]. It supports interactive data schema design and modification by direct manipulation of graphical objects, and provides an Ecrins Data Definition Language (DDL) generator.

The *EcrinsDesign* system is a part of the Rebirth project [Falquet 88]. In this project, the Database group of the CUI is realizing an open DBMS which supports dynamic restructuring. Many elements of the project are in a final or advanced stage: a semantic data model, a process model, high-level data management languages, a data restructuring tool, a graphical query tool and a graphical design tool.

In the first stage of the project, the Ecrins data model has been defined. Then, a Data Base Management System (DBMS) called Ecrins/BD has been developed, which implements the concepts of the Ecrins data model. The project is currently in its third stage, dedicated to the development of different tools, all based on the Ecrins data model. The *EcrinsDesign* system has been developed in this stage. It is currently improved to better support the schema evolution process and the definition of semantic contexts [Falquet 89].

After an overview of the Ecrins data model (section 2), we present the main characteristics and the static aspects of the *EcrinsDesign* system in section 3. The behaviour of the graphical elements of a schema is detailed in section 4. In section 5, we give some elements to use the tool and we focus on some ergonomic principles we took into account during the system's design. We conclude in section 6 by underlining the original features of our system and presenting some future research directions concerning the development of graphics-based tools for data schema design.

2 The Ecrins Semantic Data Model

The Ecrins semantic data model [Junet 86] is based on the Entity/Relationship (E/R) model of Chen [Chen 76]. It extends the semantics of the basic E/R model by using the concept of multi-valued role and the abstraction mechanisms of generalization/specialization [Smith 77] and grouping. In this section we present an overview of the Ecrins data model and we illustrate its major concepts with an example.

2.1 Basic Concepts

The Ecrins data model distinguishes three basic concepts: relation, attribute and role, which correspond to the essential features of semantic data models.

A *relation* is a named set of *entities* (objects of the application field). A relation is defined by a set of attributes. A subset of a relation's attributes forms its primary key. There are four types of relations: entity relations, relationship relations, weak relations and sub-relations.

- An *Entity Relation* (ER) corresponds to the entity type concept in the E/R model.
- A *Relationship Relation* (RR) is defined by a set of reference relations, each reference relation playing one or many *roles* in the association.
- A *Weak Relation* (WR) defines entities which depend on the existence of entities in an other relation.
- A *Sub-Relation* (SR) corresponds to the concept of sub-type in the RM/T data model [Codd79] and to the abstraction concept of generalization/ specialization defined in [Smith 77]. A SR is defined by one relation (called its "generic relation") from which it inherits all the attributes. A specialization condition is defined on the value of one of those attributes (called a "generic attribute").

In a data schema, all the relations of the set {ER, RR, WR} are called *basic relations*. The set of those basic relations forms a forest of relations.

An *attribute* A is a function from a relation R to a set of values {a1,...,an} called the *domain* of A. Any domain must be of an unique *type*. An attribute is atomic: each attribute of an entity can be set to only one value of its domain at a time. Finally, an attribute owns three properties to indicate if it is a key attribute, if its value can be modified and if its value is mandatory.

The concept of *role* describes the role played by a relation (called "reference relation") inside a relationship relation. A role can be *single* or *multi-valued*; the notion of multi-valued role is introduced, allowing to associate many entities playing the same role in a relationship relation entity. The maximum (minimum) number of entities which can be associated is the maximum (minimum) degree of the role; the maximum cardinality of a role indicates the maximum number of relationship relation entities which can be related to the same entity of the reference relation through this role.
A role defines an existential dependency between a relationship relation and its reference relations. Thus, a relationship relation entity could not exist without its different reference relation entities [Léonard 88].

2.2 Example of an Ecrins Data Schema

In this section, we illustrate the concepts of the Ecrins data model using an imaginary Airline Company. The example below describes a part of the "Small_Air" Company's information system (the complete data schema is shown in Figure 6). A description of the company assumes the following points:

EMPLOYEE, CITY and PLANE appear as entity relations (with specific attributes). The employees are specialized in three sub-relations: PILOT, TECHNICIAN, and CONTROLLER using the "Function" attribute of EMPLOYEE (its domain will be the set of values {"PILOT", "TECHNICIAN", "CONTROLLER"}). We can then define a relationship relation CREW, linking PILOT and TECHNICIAN entities through three roles:

- Chief-Pilot, a single role with a maximum cardinality of 1 (a pilot can be chief-pilot in at most one crew).
- Co-Pilot, a multi-valued role (minimum degree: 1, maximum degree: 2; each crew must have at least one co-pilot and no more than two), with a maximum cardinality unknown.
- Radio, a single role with a maximum cardinality unknown.

Likewise, FLIGHT is described as a relationship relation between a crew (CREW), a plane (PLANE) and two airports (AIRPORT).

The role Plane_Of enables a plane (an entity of PLANE) to participate to a maximum of 2000 flights (entities of FLIGHT); the removal of an entity in the relation PLANE will constrain the DBMS to remove all entities of FLIGHT to which was connected the removed entity of PLANE.

Finally, an airport could not exist without its main serving city. This is represented by defining a weak relation AIRPORT associated to the entity relation CITY through a weak role.

2.3 Ecrins Data Definition Language

The Ecrins Data Definition Language (DDL) is a procedural language, like SQL. We present further on (see Figure 1) the definition of the entity relation EMPLOYEE and the relationship relation CREW as an example, to illustrate the Ecrins DDL syntax. The complete description of the language is given in [Junet 90].

The Ecrins semantic data model has been implemented in the Ecrins/BD DBMS. A prototype of this DBMS is used by the Database Group members as an experimental system. *EcrinsDesign* uses Ecrins/BD as a target DBMS: An Ecrins DDL generator is part of the system, converting the internal data schema model into DDL files.

```
declare relation EMPLOYEE
key is
     Name char (30);
     FirstName char (20);
with properties
     Address char (60);
     Function generic word (sr PILOT sr TECHNICIAN sr CONTROLLER);
     Hiring_Date date;
     Salary real (2000:10000);
end-declare

declare relationship relation CREW
association of
     PILOT
          (Chief_Pilot) with maxcard 1
          (Co_Pilot) multi-valued of degree 2
          with maxcard unknown
     TECHNICIAN with maxcard unknown
key is Crew-Id integer;
with properties Number_Of_Flights integer mandatory;
end-declare
```

Figure 1 Ecrins DDL definition of EMPLOYEE and CREW relations

3 Presentation of the *EcrinsDesign* system

The *EcrinsDesign* system permits to define graphically an Ecrins data schema. In this section, we present the main characteristics and the static aspects of the system. We explain the dynamic aspects in the next section. First, we indicate the rules which guided the development of the system (see 3.1) and we compare it with other graphics-based systems, from a data schema design point of view (3.2). Then, we present the architecture and the different components of the system (see 3.3). In 3.4, we indicate the static aspects of the graphical representation. Endless, we focus on the optimization of the representation.

3.1 Introduction

Classical DBMSs support text-based DDL. This approach presents some problems:

- When the conceptual schema is usually designed in a graphical form (during a previous analysis stage based on a formal methodology, like Merise [Tardieu 86] or SADT [Ross 77]), it must be translated into a text-based DDL depending on the target DBMS.
- The designer has to remember both the language's concepts and exact syntax.
- The schema can't be viewed as a whole but must be read sequentially.

To avoid these problems, many systems exploit other kinds of languages (mainly natural or graphical languages), closer to the user's mental model. The *EcrinsDesign* system is based on a graphical language whose main advantage is to be the language used in the previous analysis stage: the designer does not have to translate from graphical to text-based language. The graphical language is also well-known for its own advantages: it is a global, synthesizing and non-linear language which permits progressive reading of a data schema.

We have also chosen the direct manipulation metaphor which is known to be very suitable when used on top of a graphical language (see [Shneiderman 83]). This choice has guided us to define a set of features it will have to implement:

- Systematic use of graphical objects to represent data schema elements.
- Direct manipulation operations on data schema elements.
- Coherent and consistent interface for all operations and object types.
- Effective management of a data schema screen-area, to make it as easy to read as possible.
- Permit the user to design a schema in a very intuitive way.
- Support data schema evolution, based on the successive designer's decisions.
- Provide the user with an ergonomic editing environment, using the capabilities of the Macintosh graphical interface.

3.2 Related Work

In this paragraph we present a comparison between *EcrinsDesign* and other graphical interactive systems, from a data schema design point of view. The goal of this comparison is to show different approaches chosen to represent semantic data schemas. We have based our comparison on the following three criteria: Sub-relation representation, Attributes representation and Object evolution. For each of them, we first enumerate some existing solutions then we explain and justify our choice.

Sub-relation Representation
Hierarchies of sub-relations are usually represented in an exploded form: each sub-relation is connected to its generic relation by an edge specially decorated to mean the IS-A semantic ([Czejdo 90], Pasta-3 [Kuntz 90], ISIS [Goldman 85]). In contrast, sub-relation hierarchies are represented in the *EcrinsDesign* system by nodes imbrication:

- At the highest level, the basic relations include their sub-relations.
- At the sub-relation levels, each set of sub-relations appears inside its generic relation node (see Figure 5).

With this approach, the visualization of the data schema is easier:

- Edges are reserved to the expression of interactions between relations (rolesand binary associations).

- The direct manipulation of nodes allows the user to modify the relation hierarchy by simple (and very intuitive) objects moving.

The GLAD system [Wu 89] represents only the highest relation level (the basic relations) on a main view; each sub-relation level of a generic relation is displayed in a separate view. The problem of navigation complexity (due to view proliferation on screen) is (partially) resolved by using color-based matching between associated objects.

Attributes Representation
The attributes are represented either directly on the schema, and may be manipulated as graphical objects, or in a separate view in a text-based form. The first alternative, used by (RUBIS [Rolland 88], SNAP [Bryce 86], SUPER [Spaccapietra 90]), provides the user with a complete view of the data schema. We have preferred the second one. Indeed, in the first one:

- The attributes, often represented as nodes on the schema, overload the representation, leading to very confusing pictures for complex data schemas.

- The connection between an attribute and its associated relation is represented by an edge (like a role between two relations) while its semantic is different.

With the second approach, we have chosen a preferential representation of the relations hierarchies (and not of the attributes). So, we have decided to use separate views to both represent and edit the attributes, in a non-graphical form, like Silverrun-MCD. In the Silverrun-MCD [XA 90] system, each node contains a textual list of attributes labels; the size of a node's extent can be modified, showing more or less attributes, while attributes must be edited using separated (non-graphical) views.

Object Evolution
The systems we have studied provide poor support for the objects evolution during the schema design process. Indeed, once a data schema object has been created (with an initial type), its type cannot be changed any more. This lack of flexibility may imply tedious situations. For example, a relationship relation cannot become an entity relation or a sub-relation. To change an object type, the user generally has to destroy the object and then re-create a new one of the desired type (and with the same label). Because of its awkwardness and of the incoherence which may appear (what about the old relationship relation's attributes?), this method is difficult to use during the design activity. To avoid these problems, we have defined a set of object behaviour rules to describe object evolution depending on the actions triggered by the user. The definition and the management of those rules and actions form what we call the *EcrinsDesign*'s dynamics (see section 4).

3.3 Software Architecture

Figure 2 shows the software architecture of the *EcrinsDesign* system. The user builds

the data schema graphically, using the editor. On his request, this graphical schema is then transformed by the translator into an Ecrins DDL file; this DDL description is sent afterwards to the DBMS which generates the corresponding data dictionary.

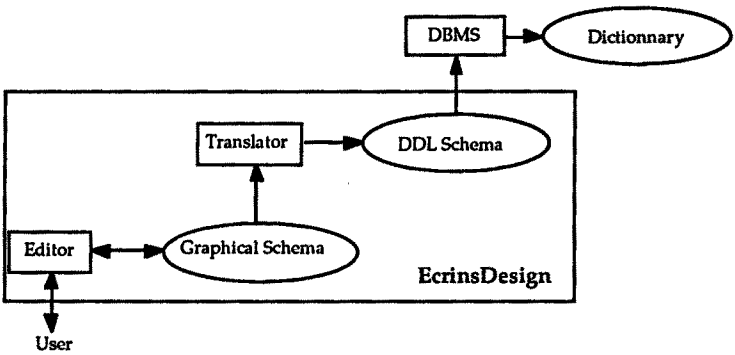


Figure 2 Software Architecture of the EcrinsDesign system

The *EcrinsDesign* system is not integrated to the DBMS, but forms a separate module which communicates with the DBMS through the DDL format. The system is composed of two modules: the Editor and the Translator. This architecture exploits the advantages of the modularity: flexibility and reliability. Mainly, the various modules can evolve separately.

To illustrate this modularity, we can indicate two extensions we are developing:

- we define a new translation module to generate the DDL description for an other target DBMS, Farandole [Falquet 90]
- we add new functions to the graphical editor without any modification of the translation modules.

3.4 Graphical Representation

The data schema is shown on screen as a two-dimension graph. *Nodes* (rectangles with round corners) represent the relations, while *edges* (sets of oriented segments) represent the associations between relations. An example of an Ecrins data schema build with *EcrinsDesign* is shown in Figure 6.

Nodes and edges

The node aspect depends on its relation type (see Figure 3), as the edge thickness depends on its type. We exploit the visual separation power of these two cosmetic variables (aspect and thickness) to better distinguish the various concepts of the Ecrins data model. To complete the graphical description, the graph is decorated with textual items exhibiting the objects labels and the degrees and cardinalities of edges. The "-" token is used to represent an unknown degree or cardinality value.

We introduced the *binary association* notion to represent a relationship relation (without any own attribute) associating two reference relations through two single roles. The relationship relation is not represented on the graph but the two reference relations are connected by a special type of edge, its decoration representing the cardinality of the two single roles. The aim of this graphical abstraction is to visually simplify the graphical representation of such often used structures.
The Figure 4 shows a synoptic description of the various types of edges. The different types of binary associations are presented in figure 4a, while Figure 4b shows their equivalent conceptual structure.

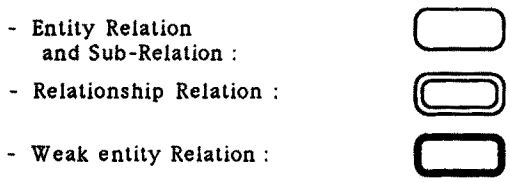


Figure 3 Relations (Nodes) Representation

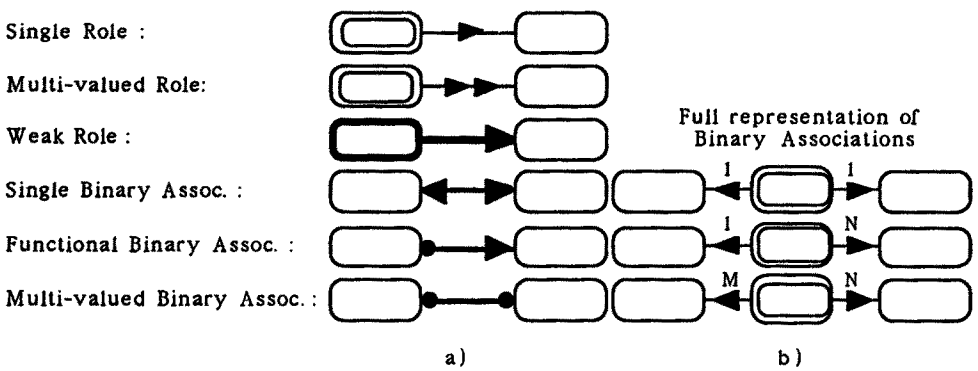


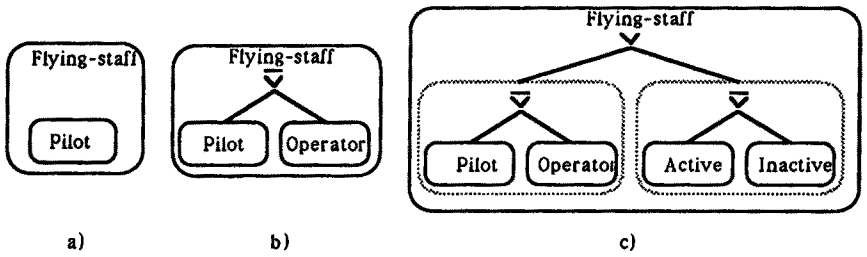
Figure 4 Associations (Edges) Representation

Sub-relations
The sub-relation concept is graphically represented as follows:

- A node GN used to represent a generic relation GR includes all the nodes SNi representing the GR's sub-relations SRi (see Figure 5 a).

- When i>1 (the GR has more than one sub-relation), the following conventions are used (see Figure 5 b):
  - the GR specialization symbol (V for inclusive OR, $\bar{V}$ for exclusive OR) is displayed inside the node GN;
  - the RG specialization symbol is linked to each SNi with a thin segment, automatically drawn by the system.

For an inclusive specialization, (see Figure 5 c), each subset of sub-relations is included in a dimmed graphical object (so called "cluster") which allows sub-relations manipulation as a whole.



a)Single Exclusive Sub-Relation
b)Exclusive Sub-Relations
c)Inclusive Sub-Relations

Figure 5 Sub-Relations Representation

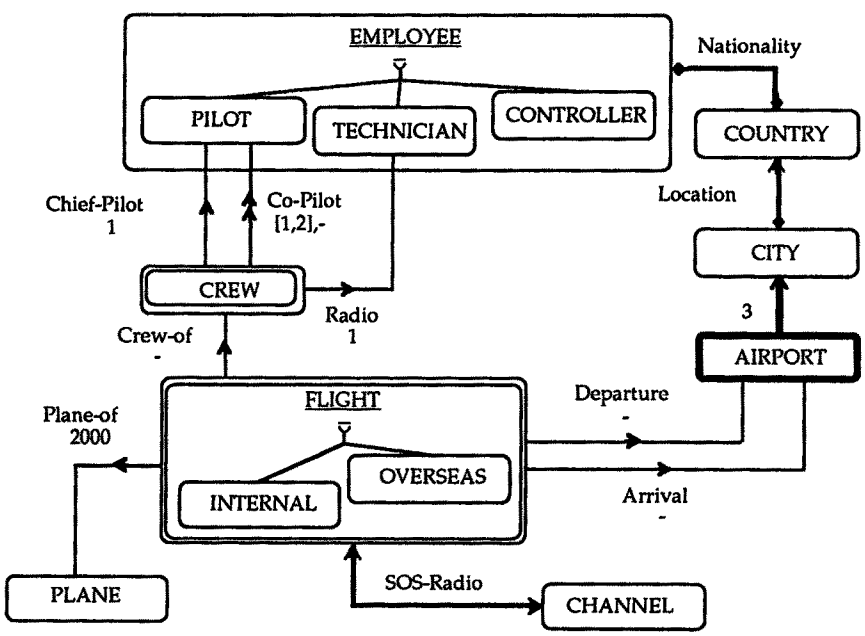


Figure 6 Graphical Representation of the "Small-Air" Data Schema using the EcrinsDesign system

3.5 Graphical Optimization

The semantic of direct manipulations is based on a strong correspondence between the physical position of a graphical object in a schema and its "conceptual position" (hierarchy level, connected relations) inside the data schema structure.

This correspondence between physical and conceptual properties implies a strict management of the graphical area, to correctly interpret each manipulation. For that reason, we have defined a set of optimization rules to control the nodes extent and the edges drawing:

- Node's extent must be minimized: any node must include its sub-relation nodes as close as possible. Nevertheless, the extent minimization is restricted by any edge which cannot be extended or shortened to follow the reduction. This rule (recursively applied to all concerned levels of a nodes hierarchy) guarantees a consistent representation of a relation hierarchy.

- Edge drawing is submitted to the following rules:
  - extremities are always attached to the border of the nodes connected by the edge;
  - any unnecessary point is destroyed; for example, the middle point of three aligned points is destroyed;
  - if the user requests it so, edges are drawn with right angles.

This set of rules, carried out after each graphical manipulation, assists the user by conferring a (real-time) "CAD-like" precision style to the data schema representation.

4 Dynamics of the Graphical Representation

After the main features and the static aspects of the system, we present its dynamic aspects in this section. In 4.2, we indicate the integrity constraints which control the dynamics. Then we develop the dynamic behaviour of nodes and edges (4.3 and 4.4).

4.1 Introduction

The direct manipulation of graphical elements by the user induces an evolution of the data schema, both at the graphical and structural levels. The *EcrinsDesign* system deals with that evolution by implementing the following principles:

Definition and Validation
To better deal with the usually iterative and non-formal schema design task, the tool supports a two-step schema creation process:

- *Definition* of the data schema, using the graphical editor; this activity may be progressive, partially done during multiple work sessions. Each of those sessions may result in a different schema that can be stored in a file.

- *Validation* of the data schema: when requested by the user, the data schema is analyzed and translated by the DDL generator, which alerts the user in case of local incoherence of the structure. Once valid, the DDL description of the schema may be transmitted to the Ecrins/BD DBMS.

Semantic at the Editor's Level
We have incorporated a part of the Ecrins data model's semantic in the functions of the tool, to better assist the user during the design process. For example, an edge must connect two valid relations at any time (no "in the air" extremity).

However, the complete set of constraints defined at the data model level is too restrictive compared with the nature of the design process.

For that reason, we have relaxed some of those constraints at the tool level, to improve its ease of use. This choice permits some local schema incoherence during the definition stage, to be signalled later by the DDL generator during the validation stage.

This aspect will be more detailed in section 4.2.

Behaviour of Graphical Objects
The direct manipulation metaphor has been chosen to interact with the system. This kind of manipulation represents a compromise between the different types of users (beginners, occasional users or experts) (see [Shneiderman 87] for a survey on different interface types). It permits a very intuitive interaction between the user and the tool, based on the materialization of data schema conceptual elements. On the other hand, the implementation of this kind of systems is more complex: the event-driven mode of interaction with the user needs to prepare the system for the user to do anything at any time.

So, we have defined the different events which may be triggered by the user, and for each of them what kind of structural schema modification will be induced. These modifications (change of element types) are managed by the system: the user has only to create, move and modify graphical objects; each type (and graphical look) transition is controlled by *EcrinsDesign* , as the eventual alerts occurring in case of constraint violation. Thus, the global validity of a data schema is assured at all times.

The management of the graphical objects behaviour and their consequences on the data schema forms the *dynamic* aspect of the *EcrinsDesign* system that we'll present in sections 4.3 and 4.4.

4.2 Relaxing of Integrity Constraints

Some of the Ecrins data model constraints have been relaxed at the tool's level, to improve its flexibility:

- Attributes can be "detached", meaning that they (temporary) don't participate in any relation definition. In particular, when a relation is deleted, its attributes (excepting the generic ones) are preserved, but they are detached (their owner relation is set to "empty").

- Objects labelling is mandatory. When created, each object is automatically labelled by the tool using an unique number, if the user does not enter a valid (unique) label of his choice.

- A relation can exist during the design stage without any attribute. This permits a first data schema rough sketch, to be detailed and completed later with the attributes specification.

- In the same way, an attribute's domain can be omitted, or even specified as a personal (unknown at the DBMS level) type (ex: colour, weekday), (it will be adapted during the validation stage).

This set of relaxed constraints improves considerably the semantic abstraction power of the tool, by better supporting both the design process uncertainty and the user's iterative activity.

Contrary to the graphical editor, the DDL generator is based on the complete set of Ecrins integrity constraints. This impartiality is used to detect any local incoherence or rule violation permitted at the tool level.

Currently, these problems are indicated to the user by showing the resulting DDL, annotated with special error tokens underlining each error. A special diagnostic tool is actually under study, to allow a graphics-based error monitoring, without any DDL knowledge needed to correct the data schema.

4.3 Dynamic Behaviour of Nodes

We illustrate the dynamic behaviour of nodes by a states and transitions automaton, based on node types and user triggered events (Figure 7).

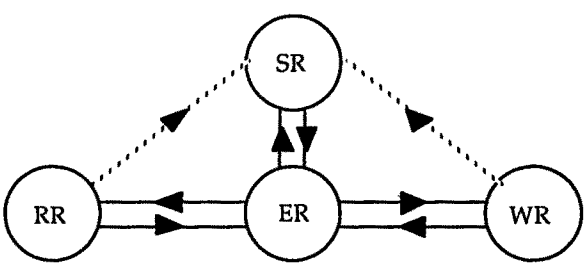


Figure 7 Nodes Dynamic Behaviour

States (types of relations):

- ER:Entity Relation (initial state)
- RR:Relationship Relation
- WR:Weak Relation
- SR:Sub-Relation

Transitions (Events):

- ER -> SR: a basic entity relation is moved inside any other relation R. It keeps its own attributes and inherits attributes from R. R's specialization attribute is updated by the system.

- SR -> ER: a sub-relation is moved to a point which is not inside a basic relation of the schema. It looses inherited attributes but keeps its own attributes. The specialization attribute of its old generic relation is updated by the system.

- ER -> RR: a first (single or multi-valued) role is created from an entity relation to any other valid relation.

- RR -> ER: the last (single or multi-valued) role from a relationship relation is removed.

- ER -> WR: a weak role is created from a basic entity relation to any other valid relation.

- WR -> ER: the weak role from a weak entity relation is removed.

- RR -> SR: a relationship relation is moved to a point inside any other relation R. This transition needs the previous removal of all the roles from the relationship relation. The sub-relation inherits R's attributes, while R's specialization attribute is updated by the system.

- WR -> SR: a weak entity relation is moved to a point inside any other relation R. This transition needs the previous removal of the weak role from the weak entity relation. The sub-relation inherits R's attributes, while R's specialization attribute is updated by the system.

4.4 Dynamic Behaviour of Edges

We illustrate the dynamic behaviour of edges by a states and transitions automaton, based on node types and user triggered events (Figure 8).

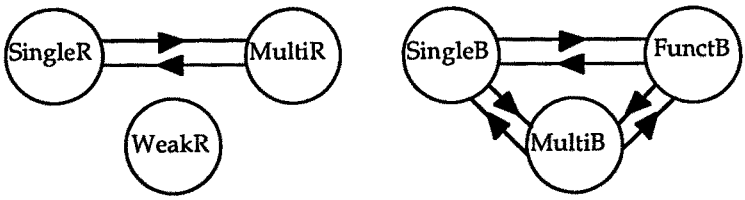


Figure 8 Edges Dynamic Behaviour

States (types of edges):

- SingleR: Single Role
- MultiR: Multi-valued Role
- WeakR: Weak Role
- SingleB: Single binary Association
- FunctB: Functionnal binary Association
- MultiB: Multi-valued binary Association

Transitions (Events), with N > 1 and M > 1:

- SingleR <-> MultiR: a role's maximum degree is modified from 1 to N (and vice-versa).
- SingleB <-> FunctB: the maximum cardinalities of the two single roles of a binary association are modified from (1,1) to (1,N) or (N,1) (and vice-versa).
- SingleB <-> MultiB: the maximum cardinalities of the two single roles of a binary association are modified from (1,1) to (N,M) (and vice-versa).
- MultiB <-> FunctB: the maximum cardinalities of the two single roles of a binary association are modified from (N,M) to (1,N) or (N,1) (and vice-versa).

5 Interface and Use

The *EcrinsDesign* system is implemented on Apple® Macintosh™ II computers. In this section, we present the main elements to use the system. In 5.1, we give some ergonomic characteristics of the system. The paragraph 5.2 is a brief description of how to use the *EcrinsDesign* system. We present in 5.3 the browsing capabilities of the system.

5.1 Ergonomic Remarks

We focused on the integration of ergonomic aspects relative to the schema design activity:

- The *EcrinsDesign* system uses the graphical language: the same language is used both at the conceptual and data schema levels.

- Integrity constraints are weaker in the editor than in the data model, so a schema can be designed gradually, using intermediate states which may violate some of the Ecrins constraints.

- A data schema is designed using a graphics-based editor: all the tools are available on the screen at any time.

- Manipulations which imply object type transformations are supported: The *EcrinsDesign* system manages automatically those transformations.

- To better visualize a data schema, attributes do not appear on the graph, but in a separated window.

5.2 Schema Design

5.2.1 The Palette

The palette provides the set of tools necessary to build a data schema (see Figure 9). These tools (symbolized by icons), are selected by the user with the mouse; each tool represents a graphical object type:

- For the nodes, a new relation is always an entity relation and its type may change *afterwards*, following the dynamics rules presented in section 4.3. So, only an entity relation tool is available within the palette.

- For the edges, each type is represented by a tool. Indeed, it is possible to create an edge of any type (between two nodes), if these nodes exist and if their types are compatible with the edge's initial type. This type may then evolve (section 4.4).

Figure 9 The EcrinsDesign Palette

5.2.2 Graphical Manipulations

During a work session, the graphical objects composing a data schema can be moved by direct manipulation. We distinguish *cosmetic* and *structural* manipulations:

- Cosmetic manipulations do not affect the data schema structure, but are used to make the data schema easier to read; for example, edge decoration items (label, cardinality) can be moved, nodes can be arranged differently up to the user.

- Structural manipulations, in contrast, modify the data schema structure. They consist in moving a graphical object to a point outside the extent of its correlated object:
  - moving a sub-relation node outside the extent of its generic relation. On a graphical point of view, the extent of all the generic relation's tree is adjusted (minimized). On a conceptual point of view, the sub-relation is no more attached to its generic relation;
  - moving one of an edge's extremities outside the extent of the correlated relation to a point inside an other relation.

5.2.3 Objects Labelling and Edges Properties

The *EcrinsDesign* system assigns automatically a number to any relation, (single or multi-valued) role and binary association, at creation time. This number is used to identify the object within the data schema. The user can assign a label to an object, using a labelling window. In this case, the tool controls the label uniqueness (the label is used as identifier).

Furthermore, the properties of an edge can be changed in a separated window. The Figure 10 shows the edition view for a role and for a binary association.

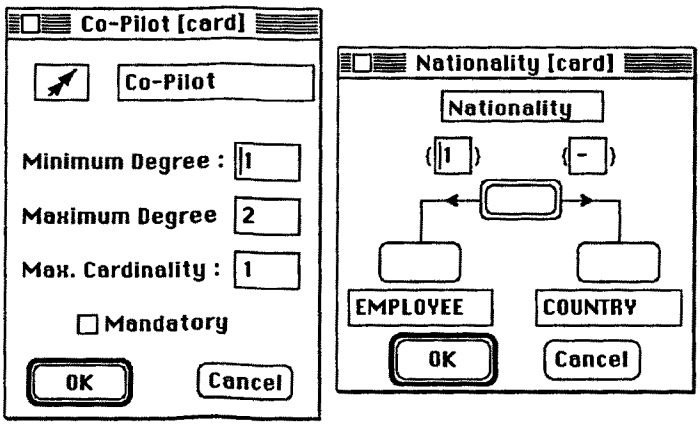


Figure 10 Edition views for a Role and for a Binary Association

5.2.4 Attributes Editing

The attributes do not appear on the data schema graph. They can be edited in separated views, one for each selected relation, a more complete window grouping the complete set of the data schema's attributes.

An attribute can be detached (and has no more owner relation) or attached to an other relation (chosen in a list of current defined relation labels).
The attribute's type may be chosen in a set of standard types (those defined in Ecrins/BD DBMS) or defined by the user (set of personal types).

5.3 Browsing

A data schema graph is drawn in a logical view larger than the user window area; so all the graph may not be visible on the screen. Browsing (using window scroll bar), looking for an object is then a laborious (blind) operation.
To avoid this, a navigation function is proposed to reach an object with its label: The "objects" menu contains the label and the type (represented by an icon) of all the graphical objects, in alphabetical order (see Figure 11). When he looks for an object, the user has to select its label within the menu, then the graph is moved to show the correlated object centered on the screen.

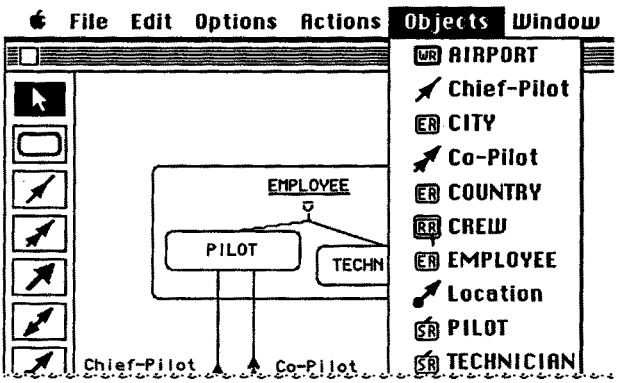


Figure 11 Browsing using the Objects Menu

6 Conclusion

The *EcrinsDesign* system permits to define data schema interactively, by direct manipulation of graphical objects. It is based on the Ecrins semantic data model, an extension of the Entity/Relationship (E/R) model. The Ecrins data model extends the semantics of the basic E/R model by using the concept of multi-valued role and the abstraction mechanisms of generalization/specialization and grouping.

The *EcrinsDesign* system implements a *dynamic* aspect of the graphical representation: depending on user operations, the type of the objects may evolve, conforming to the set of integrity rules defined at the tool level. The system manages all necessary graphical aspect modifications, to guarantee a consistent representation of the data schema at all times. For more flexibility, some of the integrity constraints of the Ecrins data model are relaxed at the tool level.
The legibility of the edited schemas, the flexibility, the strong support of schema evolution and the high level of interaction are the main characteristics of the system.
Moreover, an Ecrins Data Definition Language generator is part of the system, enabling the communication with Ecrins/BD, the target DBMS implementing the Ecrins data model. The members of the Database Group use the system to design their data schemas.
EcrinsDesign is currently implemented in Think C™, on Apple® Macintosh™ II family computers.

We currently work on extending the features of the tool to permit:

- The modification of existing database schemas, implying reorganization of the database at both the schema and data levels.

- The definition of semantic contexts, which provide users with suitable database query environments (see the QFE tool presented in [Estier 90]).

Acknowledgments: We appreciate the helpful suggestions of all the members of the Database Group and its professor, Michel Léonard.

References

[Bonjour 89]BONJOUR, M.
"EcrinsDesign: Un outil d'aide à la conception de graphes de relations", Mémoire de licence en informatique de gestion, Genève, 1989.

[Bryce 86]BRYCE, D., HULL, R.
"SNAP: A Graphics-based Schema Manager"
Proc. of the IEEE 2nd Int'l Conf. on Data Engineering, 1986.

[Chen 76]CHEN, P.P.S.
"The Entity Relationship Model-Toward a unified View of Data".
ACM TODS, Vol.1, Nb.1, 1976.

[Czejdo 90]CZEJDO, B. et al.
"A Graphical Data Manipulation Language for an Extended E-R Model"
IEEE COMPUTER, vol 23, no. 3, march 1990.

[Estier 90]ESTIER, T., FALQUET, G.
"QFE: un générateur d'interface pour l'interrogation des bases de données à l'aide de contextes sémantiques".
Proc. of the Inforsid Conf., Biarritz, France, 1990.

[Falquet 88]FALQUET, G. et al.
"Concept Integration as an Approach to Information Systems Design"
in Computerized Assistance During the Information Systems Life Cycle Conf.,
T.W. Olle et al. (ed.), Elsevier, 1988.

[Falquet 89]FALQUET, G.
"Interrogation de bases de données à l'aide de contextes sémantiques"
Thèse nº 2354, Université de Genève, éditions Systèmes et information, 1989.

[Falquet 90]FALQUET, G.
"F2: an Object-Oriented database model with semantic contexts"
Cahier du CUI no. 52, Université de Genève, 1990.

[Goldman 85] GOLDMAN K.J. et al.
"ISIS: Interface for a Semantic Information System"
Comm. of the ACM, 1985

[Junet 86]JUNET, M.
"Design and Implementation of an Extended Entity-Relationship Data Base Management System (ECRINS/86)".
Proc. of the ACM 5th International Conference on Entity-Relationship Approach, Dijon, France, November, 1986.

[Junet 90]JUNET, M.
"Sémantique des bases de données: un modèle et une réalisation"
Thèse nº 356, Université de Genève, éditions Systèmes et information, 1990.

[Kuntz 90] KUNTZ, M.
"Description et évaluation de Pasta-3, une interface graphique de manipulation directe aux bases de données avancées"
Proc. 6èmes journées BD3, Montpellier 1990.

[Léonard 88]LEONARD, M.
"Structures des bases de données"
DUNOD informatique, Paris, 1988.

[Rolland 88]ROLLAND C. et al.
"The RUBIS System"
in Computerized Assistance During the Information Systems Life Cycle Conf.,
T.W. Olle et al. (ed.), Elsevier, 1988.

[Ross 77]ROSS, D.T.
"Structured analysis (SA): A Language for Communicating Ideas"
IEEE Transactions on Software Engineering, VOL.SE-3, Nb.1, 1977.

[Shneiderman 83] SHNEIDERMAN, B.
"Direct Manipulation-A Step Beyond Programming Languages"
IEEE Computer, Vol. 16, no.8, August 1983.

[Shneiderman 87] SHNEIDERMAN, B.
"Designing the User Interface - Strategies for Effective Human-Computer Interaction"
Addison-Wesley Publ., 1987.

[Smith 77]SMITH,J.M; SMITH, D.C.P.
"Database Abstractions: Aggregation and Generalization"
ACM TODS, VOL.2, Nb.2, 1977.

[Spaccapietra 90] SPACCAPIETRA, S.
"SUPER: Un éditeur de schéma", demonstration
ER'90 Conference, Lausanne 1990.

[Tardieu 86] TARDIEU, H. et al.
"La méthode Merise"
Ed. d'organisation, Paris, 1986.

[Wu 89]WU, Thomas C., HSIAO, David K.
"Implementation of Visual Database Interface Using An Object-Oriented Language (GLAD Project)"
Proc. of the IFIP TC 2/WG 2.6 Conf., Visual Database Systems, 1989.

[XA 90]Silverrun-MCD : © XA Systems Corp., 1990.