6.1 The File menu (File)

New project... *builds a new project. Also available as a button on the standard tool bar*

Open project... *opens an existing project (a *.lun or *.isl file). Also available as a button on the standard tool bar*

Save project *saves the current project as a *.lun or *.isl file. Also available as a button on the standard tool bar*

Save project as... *saves the current project as a new *.lun or *.isl file. Also available as a button on the standard tool bar*

Close project *closes the current project*

Project properties... *examines / modifies the properties of the current project*

Export... *generates a *.isl file representing the selected objects of the current window:*
- *no selected object, generates an empty project*
- *for each selected data schema in a process or the project, generates the data schema with its contents*
- *selected objects in a data schema, generates the data schema with the selected object and their components*

Import... *imports selected schemas from a *.isl file in the current process*

Execute Voyager... *runs a compiled Voyager 2 program (*.oxo)*

Continue Voyager... *continues an interrupted Voyager 2 program*

Rerun Voyager... *reruns the last loaded Voyager 2 program*

User tools *list of Voyager 2 programs or menu items defined in the db_main.ini file (maximum 5)*

- **item 1** *... executes the first program or menu command*
- **item 2** *... executes the second program or menu command*
- **etc** *...*

The File menu (continued)

Extract *builds a physical/logical schema describing the data structures extracted from:*

- ***SQL...*** *... an SQL source text file (structure definition script)*
- **COBOL...** *... a COBOL source text file (file and record structures)*
- **IDS/II...** *... an IDS/II DDL text file (schema & sub-schemas)*
- **IMS...** *... an IMS source text file*

the text is then added to the project as a product, and a primitive process ("Extract ...") is created between the two products.
If the current window is a data schema window, then the extractor adds the extracted specifications to this schema ("Incremental" mode); if the current window is the project or a process window, the extractor creates a new schema comprising the extracted specifications ("New" mode)

- **XML...** *... an XML (DTD) source text file (the XML extractor is the EXTRACT_XML.OXO program)*

*This extractor is written in Voyager 2. Use **File/Configuration** (DDL extractors option) to indicate where to find the program*

Generate *generates an executable DDL program corresponding to the current data schema, according to various DBMS and styles :*

- **Standard SQL...** *Standard SQL-2 ...*
- **Vax Rdb/VMS 4.2...** *Vax Rdb/VMS ...*
- **Academic SQL...** *Academic SQL (possible forward references) ...*
- **Standard SQL (check)...** *Standard SQL-2 (with check predicates) ...*
- **Vax Rdb/VMS 4.2 (check)...** *Vax Rdb/VMS (with check predicates) ...*
- **Academic SQL (check)...** *Academic SQL (possible forward references; with check predicates) ...*

*The next three generators are written in Voyager 2. Use **File/Configuration** (code generators option) to indicate where to find the programs*

- **COBOL...** *a (simple) COBOL generator is the COBOL.OXO program*

The File menu (continued)

CODASYL... *there are two "CODASYL" generators. The first one is a CODASYL DBTG-71 generator CODASYL.OXO and the second one is a IDS/II generator (IDS2.OXO)*

Other generators can be developed as Voyager 2 programs.

XML... *a XML (DTD) genrator is the GENERATE_XML.OXO program*

Edit text file... *opens the MS Windows Notepad*

Report... *generates a dictionary report of the current schema or source file window.*

Textual view... *generates a dictonary report of a schema (only textual view) or source file window in a file (*.dic). The semantic and/or technical descriptions with separators can be added as well and the report can be add to the current process.*

Rtf... *a Voyager 2 program generates a standard RTF report (only for schema windows). Other RTF report generator can be developed as Voyager 2 programs.*

Custom... *a Voyager 2 program generates a custom report (only for schema windows). The custom report generator must be developed by the users as Voyager 2 programs.*

*The last two report generators are written in Voyager 2. Use **File/Configuration** (report generators option) to indicate where to find the programs*

Print... *prints the content of the current process, schema or source file window (textual or graphical views) to the chosen printer*

Printer setup... *chooses and configures the printer*

Configuration... *sets some general DB-MAIN options (e.g. the patterns, SQL extractor, generator, dependency graph, assistant libraries options, default directories and technical identifier). Uses the DB-MAIN.INI file*

The File menu (continued)

Exit	*exits from DB-MAIN. Saves the current project if needed*
File1	*opens the last opened file (*.lun or *.isl)*
File2	*opens the next to last opened file (*.lun or *.isl)*
etc	...

6.2 The Edit menu (Edit)

Save point		*in a data schema view: saves a copy of the current schema*
Rollback		*in a data schema view: restores the last copy of the current schema (the current version is lost)*
Undo		*in a data schema view: undo the last action*
Copy	**<Ctrl>+C**	*in a schema view: copies the selected objects to the clipboard; they can then be pasted in any schema of the same project. It can be used to quickly define similar attributes, processing units, groups, entity types, rel-types or sub-schemas, in the same schema, or in different schemas* *in a source file view: copies the selected lines to the clipboard*
Paste	**<Ctrl>+V**	*in a schema view: pastes the contents of the clipboard to the current schema*
Copy graphic		*in any graphical window (project, process or schema): copies the selected objects to the clipboard as graphical objects to be included in another document (e.g. Word, Powerpoint). Useful to document reports. Also available as a button on the tool bar.*
Select all	**<Ctrl>+A**	*in any graphical view: selects all the objects of the current schema, project or process*
Mark selected		*in any view: marks all the selected objects of the current schema, project, process or source file*
Select marked		*in any view: selects all the marked objects of the current schema, project, process or source file*
Color selected		*in any view: colors all the selected objects of the current schema, project, process or source file*
Remove color		*in any view: colors in black all the selected objects of the current schema, project, process or source file*
Delete	**<Del>**	*deletes the selected object (schema, text file, process, entity type, rel-type, attribute, group, collection, constraint, processing unit, internal data object).* *Deleting a text file only removes its reference from the project or process*

The Edit menu (continued)

Change color... *in any textual or graphical view of a schema, in the window of a project or a process and in any text file: changes the color used to color the selected objects.*

Change font... *in any textual or graphical view of a schema, in the window of a project or a process and in any text file: changes the font and character size. It can be used to shrink a large schema on the screen or in a document.*

6.3 The Product menu (Product)

Functions related to the manipulation of the products of the current project. A product is any document used or produced in the current project, and which is under the control of the CASE tool. Currently, the products comprise the schemas, the views, the source text files, and the generated files (e.g. SQL scripts). The products and their relationships are presented in the project window or in a process window.

New schema... *creates a new data or processing schema*

Add text... *adds an existing external file to the project or a process; can be used, e.g., to extract a logical schema; equivalent to drag&drop*

New set... *creates a new product set*

Open... *opens the product (schema or text file) selected in the project windows. Same as double-clicking on the product icon in the project or process window*

Properties... *examines/modifies the properties of the selected product*

Copy product... *generates a new schema or text with the same contents as the current one. Can be asked from the project or process window, or from the current schema window*

View *View management:*

- **Define view...** *defines a view comprising the marked objects of the current schema*
- **Generate view...** *generates a view defined on the current schema*
- **Mark view...** *marks the objects belonging to a view of the current schema*
- **Remove view...** *deletes a view defined on the current schema*
- **Copy view...** *copies a view defined on the current schema*
- **Rename view...** *renames a view defined on the current schema*

The Product menu (continued)

Meta *modifies / examines the schema of the repository (a.k.a. the meta-schema):*
- **Objects...** *adds / deletes / modifies / examines meta-objects (not yet implemented)*
- **Properties...** *adds / deletes / modifies / examines meta-properties*
- **Relations...** *adds / deletes / modifies / examines meta-relations (not yet implemented)*

User-domains... *adds / deletes / examines / modifies user-defined domains*

Lock/Unlock *puts or removes the lock on a product (when a product is locked, no change can be made)*

6.4 The New menu (New)

This menu makes it possible to add new objects to the current schema.

Collection... *in a data schema: creates a new collection; also available from the toolbar*

Entity type... *in a data schema: creates a new entity type (interactively or from the current source text); also available from the toolbar*

Rel-type... *in a data schema: creates a new relationship type; also available from the toolbar; also available from the toolbar*

Attribute *in a data schema: adds a new attribute (interactively or from the current source text) ; also available from the toolbar*

- **First...** *as first child (of the selected parent object)*
- **Next...** *as next sibling (of the selected attribute)*

Role... *in a data schema: adds a new role to the selected relationship type; also available from the toolbar*

Group... *in a data schema: adds a new group (primary/secondary id, access key, coexistence, referential, ...) to the selected object (entity type, relationship type or compound attribute); also available from the toolbar*

Constraint... *in a data schema: adds a new constraint involving the selected group*

Processing unit... *in a data schema: adds a new processing unit to the selected parent (entity type, relationship type or schema) or as next sibling (of the selected processing unit)*
in a processing schema: adds a new processing unit to the schema

Data object... *in a processing schema: adds a new internal data object to the selected parent (schema or internal data object) or as next sibling (of the selected internal data object)*

Call... *in a processing schema: adds a new call relation between two processing units*

Decomposition... *in a processing schema: adds a new decomposition relation between two processing units*

The New menu (continued)

In/out... *in a processing schema: adds a new in/out relation between a processing unit and an internal or external (entity type, relationship type, attribute, collection of a data schema) data object*

Note... *adds a new note to the selected parent (entity type, relationship type, attribute, group, processing unit, collection or schema)*

In the textual views:

When an object is created, the corresponding dialogue box appears and the properties can be changed.

In the graphical data schema views:

*When an entity type (rel-type or collection) is created (through menu **New** or the toolbar), the cursor changes into the object icon and the entity type (rel-type or collection) is created where the mouse points when its left button is pressed.*
*To create a role (use the item **New/Role** or the toolbar), draw a line with the cross cursor from the entity type to the rel-type;*
This role cursor can also be used to create a rel-type: draw a line between the two entity types.
To create a multi-ET role, draw a line between a role and an entity type.
To exit the creation mode click a second time on the button in the toolbar or press on the <Esc> key or click on another creation button in the toolbar.
*To create a note by menu **New/Note** or the tool bar, the mouse cursor changes and the note is created where the mouse points when its button is pressed. If the current object is an entity type, rel-type, attribute, group, collection or processing unit, the note is linked to this object. Otherwise the note belongs to the schema.*
To create an identifier with the selected attributes and/or roles, click on the ID button in the toolbar.
To create a group with the selected attributes and/or roles, click on the GR button in the toolbar.

The New menu (continued)

In the graphical processing schema view:

*When a processing unit (internal data object) is created (through menu **New** or the toolbar), the cursor changes into the object icon and the processing unit (internal data object) is created where the mouse points when its left button is pressed.*
*To create a call (decomposition) relation (use the item **New/Call**, **New/Decomposition** or the toolbar), draw a line with the cross cursor between two processing units.*
*To create a in/out relation (use the item **New/In/out** or the toolbar), draw a line with the cross cursor between a processing unit and an internal or external (entity type, relationship type, attribute or collection from a data schema) data object.*
To exit the creation mode click a second time on the button in the toolbar or press on the <Esc> key or click on another creation button in the toolbar.
*To create a note by menu **New/Note** or the tool bar, the mouse cursor changes and the note is created where the mouse points when its button is pressed. If the current object is an entity type, rel-type, attribute, group, collection or processing unit, the note is linked to this object. Otherwise the note belongs to the schema.*

***New/Entity type** and **New/Attribute** can be used to create objects from COBOL statements as follows.*

To create an Entity type:	*- a source text and its corresponding logical data schema are opened* *- selects a data definition statement in the COBOL text (generally a "01" record definition)* *- executes a **New/Entity type** command*
To create Attributes:	*- a source text and its corresponding logical data schema are opened* *- selects the parent object (entity type, rel-type, attribute) or the sibling attribute in the data schema window* *- selects one or several field definition statements in the COBOL text* *- executes a **New/Attribute/First** or **New/Attribute/Next** command*

6.5 The Transform menu (Transform)

Carries out transformations (most of them are semantic-preserving) on the selected object of a data schema. This toolbox will be expanded according to the needs of the different database engineering activities. The current functions allows for, e.g.:

- *the production of relational, CODASYL, standard files, IMS, TOTAL/IMAGE, (and the like) databases*
- *optimization of database schemas*
- *conceptual restructuring*
- *reverse engineering*

Entity type — *Transforms the selected **entity type** E*

- **® Rel-type** — *... into a relationship type*
- **® Attribute** — *... into an attribute*

- **→ is-a → Rel-type** — *... if E is a supertype, by replacing the is-a relations into one-to-one rel-types*
- **® Rel-type → is-a** — *... by replacing one-to-one rel-types with is-a relations, therefore making E a super-type*

- **→ Split/Merge** — *... into two entity types linked by a one-to-one rel-type, or by migrating attributes/processing units/roles/is-a between E. and another entity type, or by merging E with another entity type*
- **® Add Tech ID** — *... adds a technical primary identifier (replacing the current primary id., if any)*

Rel-type — *Transforms the selected **relationship type***

- **® Entity-type** — *... into an entity type*
- **® Attribute** — *... into a reference attribute (foreign key)*

- **® Object att.** — *... into an object-attribute (in OO schemas)*

Attribute — *Transforms the selected **attribute***

- **→ Entity-type** — *... into an entity type*
  - ***instance representation*** — *by representing the (possibly duplicate) attribute instances*
  - ***value representation*** — *by representing the distinct attribute values*

The Transform menu (continued)

Disaggregation — *..., if compound, by replacing it with its components*

Multi → Single — *..., if multivalued, into a single-valued attribute (= value concatenation)*
Single → Multi — *..., if single-valued, into a multivalued attribute (= value slicing)*
Multi → List Single — *..., if multivalued, into a list of single-valued attributes (= instantiation)*
Multi Conversion — *changes the collection type of the selected multivalued attribute (set, bag, list, array, etc.); provides both semantics-preserving and semantics-changing techniques;*

Domain Materialization — *materializes the domain of the selected atomic attribute = replaces a user-defined domain with its definition*

Object Att. → Rel-type — *..., if object-type, into a rel-type*

Role — *Transforms the selected **role***
Multi-ET → rel-types — *..., if multi-ET, into a series of similar relationship types*

Group — *Transforms the selected **group***
→ Rel-type — *... if referential (e.g. foreign key), into a relationship type*
Aggregation — *... into a compound attribute*

→ Multi-valued — *... of single-valued attributes into a multivalued attribute (= de-instantiation)*

Change prefix... — *detects the largest prefix of the components of the selected object and proposes its replacement with a new prefix (or absence thereof)*

The Transform menu (continued)

Name processing... *processes the names and short names of objects in the current schema*
- **scope** *definition of the scope of the process (which global, selected or marked objects, which names)*
- **substitution patterns** *definition of substitution patterns (list of "old string ® new string")*
- **change case, capitalize, remove accents and shorten names** *additional specific transformations*
- **Load/Save substitution patterns** *saves/loads the substitution patterns (file *.pat)*

Relational model *replaces (by applying ad hoc transformations) the current schema by its relational logical version, carries out no optimization*

Quick SQL *(To be avoided by real professional, including students!)*
This function offers a quick, fast-food style and rather lazy way to generate the executable code that implements the data structures corresponding to the current (conceptual or logical) schema. They result into correct, clear, but unsophisticated DDL (SQL) programs. The schema do not have to be DMS-compliant. This function has been included to comply with most commercial CASE tools. Same result as by executing:

Product/Copy schema(product)...
Transform/Relational model
File/Generate/Standard SQL
select schema
Edit/Delete

6.6 The Assist menu (Assist)

This menu offers a series of expert assistants dedicated to specific classes of problems. At the present time, six assistants have been developed (and still are continuously evolving). More are in preparation. Most assistants can be extended and enriched with user-defined predicates and/or actions developed in Voyager-2. Some of them have scripting facilities.

First Assistant: **Global transformations...**

*Carries out selected actions on selected objects in order to solve structural problems. For each outstanding class of constructs (the **problem**), the assistant proposes one or several transformations that replace them by equivalent constructs (the **solution**). The assistant proposes other additional global functions. In addition, the user can build (then save and reuse) customized **transformation scripts** dedicated to specific complex problems.*
For each operation in the list below, we suggest some standard design processes in which it can be most usefu. The standard processes are coded as follows:

CN:	conceptual normalization
RE:	reverse engineering
LD:	logical design (DBMS-independent)
RLD:	relational logical design
FLD:	standard file logical design (e.g. COBOL)
CLD:	CODASYL logical design
OOD:	OO-DBMS design
OPT:	optimization
PD:	physical design

For each outstanding class of constructs, we have also added the actions Mark and Unmark (except Group Names, Generate and Name processing). These actions put or remove a mark on the constructs that belong to the class.

The Assist menu (1st assistant, continued)

Entity type	*problems related to entity types*
Rel. entity types → Rel-types	*the entity type seems to be the representation of a relationship type (CN)*
Att. entity types → Attributes	*the entity type seems to be the representation of an attribute (CN)*
Missing id. → Add a technical id	*the entity type will need an identifier when rel-types are transformed into foreign keys (RLD)*
Rel-type	*problems related to relationship types*
With attributes → Entity types	*relationship type with attributes not allowed: replace with entity type (LD)*
Complex → Entity types	*too complex relationship types (N-ary or with attributes) not allowed: replace with entity type (LD)*
Binary 1-1 → Is-a	*transform the one-to-one relationship type into a is-a relation if it does not conflict with another one-to-one relationship type (CN, RE)*
Binary 1-N → Referential attributes	*each one-to-many relationship type is replaced by reference attributes (RLD,FLD)*
Binary N-N → Entity types	*many-to-many binary relationship type not allowed: replace by an entity type (LD)*
Binary w/o att. → Object attributes	*each binary relationship type without attribute is replaced by an object-attributes (LD, OOD)*
Cyclic → Entity types	*cyclic relationship types not allowed: replace by entity types (CLD)*
Cyclic 1-N → Referential attributes	*one-to-many cyclic relationship types not allowed: replace with reference attributes (CLD)*
Multi-ET roles → Split rel-types	*multi-ET roles not allowed: split them (LD); if the rel-type contains more than one multi-ET roles, this global transformation must be carried out as many times*
Is-a	*problems related to IS-A relations*
All → Rel-types	*each IS-A relation is replaced by a one-to-one relationship type (LD)*

The Assist menu (1st assistant, continued)

Attributes	*problems related to attributes*
Compound → Disaggregation	*compound attributes not allowed: replace by their components (RLD)*
→ Entity types	*... or by entity types (CN,LD)*
Multivalued → Entity types	*multivalued attributes not allowed: replace by entity types (instance repr.) (LD)*
Object → Rel-types	*object-attributes not allowed: replace by relationship types (RE)*
User-defined → Materialize	*user-defined attributes are replaced by their definition (LD)*
Comp. att, 1 comp. → Disaggregation	*compound attributes with only one component: disaggregate (LD)*
Single comp. att., FK → Disaggregation	*single-valued compound attributes with foreign key: disaggregate (RE)*
→ Entity-type	*single-valued compound attributes with foreign key: entity type (RE)*
Multi. comp. att., FK → Entity type	*multi-valued compound attribute with foreign key : replace by an entity type (RE)*
Groups	*problems related to groups*
Referential → Rel-types	*each reference group (foreign key) is replaced by a one-to-many rel- type (RE)*
Id & ref w/o AK → Make access key	*an access key is associated with each id or reference group (PD)*
Id >1 comp. → Add technical id	*primary ids with more than 1 component are replaced by a technical id (OPT,PD)*
Id >2 comp. → Add technical id	*primary ids with more than 2 components are replaced by a technical id (OPT,PD)*
Id >3 comp. → Add technical id	*primary ids with more than 3 components are replaced by a technical id (OPT,PD)*
Access keys → Remove	*remove access keys (RE)*
Multi-att. identifiers → Aggregate	*makes a compound attribute with multi-attribute id (FLD)*
Multi-att. access keys → Aggregate	*makes a compound attribute with multi-attribute access key (FLD)*
Prefix access key → Remove	*removes each access key which is a prefix of another access key (RLD,FLD)*
Coexistence → Aggregate	*isolate the components of a coexistence group (CN, LD)*
Names → Rename	*renames the groups with unique standardized names (PD) (Mark/Unmark not available)*
Miscellaneous	*problems related to other objects*
Technical descript. → Remove	*removes technical descriptions (RE)*
Collections → Remove	*removes the collections (RE)*

The Assist menu (1st assistant, continued)

Generate — *generates executable DDL code for the current schema (must be compliant with the DBMS)*

- **Standard SQL** — *standard SQL-2 program*
- **Vax Rdb/VMS 4.2** — *Vax RDB SQL program*
- **Academic SQL** — *simplified SQL program (with possible forward references)*
- **Standard SQL (check)** — *standard SQL-2 program (with check predicates)*
- **Vax Rdb/VMS 4.2 (check)** — *Vax RDB SQL program (with check predicates)*
- **Academic SQL (check)** — *simplified SQL program (with check predicates)*
- **COBOL** — *ENVIRONMENT DIVISION and DATA DIVISION*
- **CODASYL** — *CODASYL schema DDL program*
- **XML** — *XML (DTD) file*

Name processing — *processes the names of selected objects in the current schema*

- ***scope*** — *definition of the scope of the process (which objects, which names)*
- ***substitution patterns*** — *definition of the substitution patterns (list of "old string → new string")*
- ***change case, capitalize, remove accents or shorten names*** — *additional specific actions*
- ***Load/Save substitution patterns*** — *saves/loads the substitution patterns (file *.pat))*

The scripts — *a script is a sequence of operations chosen among those described above. It implements simple transformation plans. The current script, if any, appears in the Script window.*

- **Add** — *adds the selected action at the end of the current script*
- **Insert** — *inserts the selected action before the selected operation in the script*
- **Remove** — *deletes the selected operation from the script*
- **Edit** — *edits the selected **Name processing** operation in the script*
- **Clear** — *deletes the current script*

The Assist menu (1st assistant, continued)

Pre-defined	*loads a pre-defined script, the following scripts are available:*
Binary	*no is-a; no complex rel-types*
Bachman	*binary + no many-to-many, or cyclic rel-types*
Flat binary	*binary + single-valued, atomic attributes only*
Flat Bachman	*Bachman + single-valued, atomic attributes only*
Relational rev. eng.	*rebuilds a conceptual schema from a (simple) relational logical schema*
COBOL rev. eng.	*rebuilds a conceptual schema from a (simple) COBOL logical schema*
Pseudo-relational	*in most cases, generates an acceptable physical relational schema from a conceptual one*
Logical pseudo-relational	*in most cases, generates an acceptable logical relational schema from a conceptual one*
Physical pseudo-relational	*in most cases, generates an acceptable physical relational schema from a logical one*
Pseudo-COBOL	*in most cases, generates an acceptable logical COBOL schema*
Pseudo-IDS/II	*in most cases, generates an acceptable logical IDS/II schema*
Load	*loads a previously saved script (file *.scr)*
Save	*saves the current script (file *.scr)*
Copy	*copies the current script in the clipboard*
Confirm button	*automatic or step by step processing*
OK button	*if the script window is empty, executes the selected action* *if the script window contains operations, executes them*
Cancel button	*closes the assistant without taking any actions*
Help button	*explains the meaning and use of the global transformations assistant*

The Assist menu (continued)

Second Assistant: **Advanced global transformations...**

This one is a sophisticated version of the Global Transformation Assistant providing more flexibility and power in script development. A script consists of transformations and control structures. A transformation has the form A(P) where A is an action (transform, remove, mark, etc.) and P is a predicate that select specific objects in the data schema. The meaning is obvious: apply action A on each object that satisfies predicate P. The control structures include scope restrictions and loops. A library of advanced global transformations can be defined and reused in the definition of new ones.

Transformations

A transformation is designed to perform a given action on a set of objects. A default set is defined for each transformation. This set may be refined to a subset defined by a predicative rule. This rule is a search rule of the ***Schema Analysis Assistant****. For instance, the* `RT_into_ET` *transformation is defined to transform all rel-types of a schema into entity types. But this transformation may be refined to transform complex rel-types (i.e. with attributes and/or with more than 2 roles) only:*

```
RT_into_ET(ATT_per_RT(1 N) or ROLE_per_RT(3 N))
```

This specific transformation can be renamed as "TRANSFORM-COMPLEX-RT" for clarity, and reused in scripts.

The following is a table of the available transformations:

ET_into_RT	*Transforms all entity types satisfying the preconditions of the elementary transformation into rel-types.*
ET_into_ATT	*Transform all entity types satisfying the preconditions of the elementary transformation into attributes.*
ADD_TECH_ID	*Add a technical identifier to all entity types. This transformation should never be used without refinement of the scope by a predicate.*

The Assist menu (2nd assistant, continued)

SMART_ADD_TECH_ID	*Add a technical identifier to all entity types that do not have one but should have, in such a way that all rel-types can be transformed into foreign keys.*
ISA_into_RT	*Transform all is-a relations into binary one-to-one rel-types.*
RT_into_ET	*Transform all rel-types into entity types. This transformation should never be used without refinement of the scope.*
RT_into_ISA	*Transform all binary one-to-one rel-types that satisfy the preconditions of the elementary transformation into is-a relations if it can be done without dilemma (the remaining is-a relations can be transformed with the elementary transformation later on).*
RT_into_REF	*Transform all rel-types into referential attributes.*
RT_into_OBJATT	*Transform all rel-types into object-attributes.*
REF_into_RT	*Transform all referential attributes into rel-types.*
ATT_into_ET_VAL	*Transform all attributes into entity types using the value representation of the attributes. This transformation should never be used without refinement of the scope.*
ATT_into_ET_INST	*Transform all attributes into entity types using the instance representation of the attributes. This transformation should never be used without refinement of the scope.*
OBJATT_into_RT	*Transform all object-attributes into relationship types.*
DISAGGREGATE	*Disaggregate all compound attributes.*
INSTANCIATE	*Transform all multivalued attributes into a list of single-valued attributes.*
MATERIALIZE	*Replace all user-defined attributes with their definition.*
SPLIT_MULTIET_ROLE	*Split all the rel-types that contain one or more multi-ET roles.*
AGGREGATE	*Aggregates all groups. This transformation should never be used without refinement of the scope.*
GROUP_into_KEY	*Add the access key property to all groups.*
RENAME_GROUP	*Give a new meaningful name to each group. This name is unique in the schema. Note that the old name is lost forever.*
REMOVE_KEY	*Remove all access keys.*
REMOVE_PREFIX_KEY	*Remove all access keys that are a prefix of another one.*
REMOVE_TECH_DESC	*Remove the technical description of all the objects of the schema.*

The Assist menu (2nd assistant, continued)

REMOVE	*Remove all the objects that are in the specified scope. The deleted objects are lost forever. Note that this transformation is very special, it does not exactly conform to the definition of a transformation since there is no default scope.*
NAME_PROCESSING	*Processe the name of objects that are in the specified scope (define substitution patterns; change case, capitalize, remove accents and shorten names; load and save substitution patterns).*
MARK	*Mark all objects that are in the specified scope.*
UNMARK	*Unmark all objects that are in the specified scope.*
EXTERN	*Call an external Voyager-2 function (user-defined action).*

Control structures

ON (<predicate>) ENDON

The predicate serves as a filter for the embedded operations. All the objects that satisfy the predicate forme a set. This set is used as the scope of the following operations. That is, all the transformations between ON and ENDON will be carried on on the objects of that set rather than on the objects of the whole schema.

LOOP...ENDLOOP

Through this structure the embedded actions several times until a fixpoint is reached. The LOOP keyword is just a label; when it is encountered it does nothing. All the transformations that follow it are performed until the ENDLOOP keyword is reached. Then, if one or more transformations has effectively modified the schema, all these transformations are performed once more. This will continue until the schema has reached a fixpoint for these transformations, i.e. none of them modifies it. Be careful, it is a nice way to develop never-ending scripts!

Library

The library is a list of user defined transformations. Such a transformation has a name (which appears in the list box) and a definition. The definition is made of one or more transformations of the list above with their scope. The library can be saved for further reuse. It can be edited by pressing on the **Edit library** *button. It is a god way to give frequent complex actions a readable name in the language of the analyst.*

The Assist menu (2nd assistant, continued)

Script management

- **Add** — *adds the selected transformation or control structure at the end of the current script*
- **Insert** — *inserts the selected action before the current operation in the script*
- **Remove** — *deletes the current operation from the script*
- **Edit** — *edits the parameters of the current operation in the script*
- **Clear** — *deletes the current script*
- **Pre-defined** — *loads a predefined script; the following scripts are available:*
  - **Binary** — *no is-a; no complex rel-types*
  - **Bachman** — *binary + no many-to-many, or cyclic rel-types*
  - **Flat binary** — *binary + single-valued, atomic attributes only*
  - **Flat Bachman** — *Bachman + single-valued, atomic attributes only*
  - **Relational rev. eng.** — *rebuilds a conceptual schema from a (simple) relational logical schema*
  - **COBOL rev. eng.** — *rebuilds a conceptual schema from a (simple) COBOL logical schema*
  - **Pseudo-relational** — *in most cases, generates an acceptable physical relational schema from a conceptual one*
  - **Logical pseudo-relational** — *in most cases, generates an acceptable logical relational schema from a conceptual one*
  - **Physical pseudo-relational** — *in most cases, generates an acceptable physical relational schema from a logical one*
  - **Pseudo-COBOL** — *in most cases, generates an acceptable logical COBOL schema*
  - **Pseudo-IDS/II** — *in most cases, generates an acceptable logical IDS/II schema*
  - **Pseudo-XML** — *in most cases, generates an acceptable logical XML schema*
- **Load** — *loads a previously saved script (file *.tfs)*
- **Save** — *saves the current script (file *.tfs)*
- **Copy** — *copies the current script in the clipboard*

The Assist menu (2nd assistant, continued)

Edit library	*opens the transformation library dialog box*
New	*creates a new library entry*
Delete	*deletes an entry from the library*
Rename	*renames a library entry*
Add	*adds a new transformation to the current library entry*
Insert	*inserts a new transformation to the current library entry*
Remove	*removes a transformation from the current library entry*
Edit	*edits the parameters of a transformation of the current library entry*
Load library	*loads a library*
Save library	*saves the current library*
Close	*closes the current dialog box*
Help	*explains the meaning of the transformation library*

Using a script

Confirm button	*automatic or step by step processing.*
Ok button	*the transformations are performed. During execution, an* **Abort** button *can be pushed to stop the execution.*
Cancel button	*closes the window without keeping the script and without executing it.*
Help button	*explains the meaning and use of the advanced global transformations assistant.*

The Assist menu (continued)

Third Assistant: **Schema Analysis...**

This assistant is able to detect, in the current data schema, specified structural patterns of any complexity. A structural pattern is defined by an object type and properties which the objects have to satisfy. Some examples are "entity types without attributes", "attributes which are compound but not multi-valued", "rel-types with more than 2 roles", "names which appears in a list of reserved words". The assistant proposes more than 200 rules or constraints, but only some constraints are listed below. User-defined rules can be developed in Voyager 2.

*The ANALYSIS assistant can be used in two ways, namely to **validate** the current schema, and to **search** the schema for specified objects.*

***VALIDATION** (press button **Validate**)*

The assistant performs the analysis of the current schema in order to evaluate its compliance with a sub-model. A sub-model is a restriction of the generalized E/R model proposed by DB-MAIN. It is defined as a collection of structural patterns, specified by rules. Such a rule is a logical expression concerning a given type of objects where the terms are predicative constraints on objects of this type.

If all the objects of the schema satisfy the rules of the sub-model, this schema is said to be compliant with this sub-model. If the analysis results in a failure, then some objects do not satisfy some rules, and the assistant presents them in a diagnostic window which can be used as a notepad. When a diagnostic message is selected, the assistant makes the offending object current in the schema.

***SEARCH** (press button **Search**)*

The assistant searches the current schema for all the objects that satisfy the specified rules. It presents them in a diagnostic window which can be used as a notepad. When a diagnostic message is selected, the assistant makes the corresponding object current in the schema.

The set of rules can be saved and loaded later on. Some predefined sets of rules are available.

The Assist menu (3rd assistant, continued)

The object types *(when a type is selected, the related constraints appear in the second list box)*

Schema
Collections
Entity types
Is-a
Rel-types
Roles
Attributes
Groups
Entity type identifiers
Rel-type identifiers
Attribute identifiers
Access keys
Referential constraints
Processing units
Names

The elementary constraints, classified by object types (excerpts)

Schema

ET_per_SCHEMA <min> <max>	*the schema includes at least <min> and at most <max> entity types*
RT_per_SCHEMA <min> <max>	*the schema includes at least <min> and at most <max> rel-types*
COLL_per_SCHEMA <min> <max>	*the schema includes at least <min> and at most <max> collections*

Collections

ET_per_COLL <min> <max>	*a collection includes at least <min> and at most <max> entity types*

The Assist menu (3rd assistant, continued)

Entity types

ATT_per_ET <min> <max>	*an entity type has at least <min> and at most <max> attributes*
GROUP_per_ET <min> <max>	*an entity type has at least <min> and at most <max> groups*

Is-a

SUPER_TYPES_per_ISA <min> <max>	*an entity type has at least <min> and at most <max> supertypes*

Rel-types

ATT_per_RT <min> <max>	*a rel-type has at least <min> and at most <max> attributes*
GROUP_per_RT <min> <max>	*a rel-type has at least <min> and at most <max> groups*
ROLE_per_RT <min> <max>	*a rel-type has at least <min> and at most <max> roles*
N_ROLE_in_RT <min> <max>	*a rel-type has at least <min> and at most <max> roles with maximum cardinality > 1*

Roles

MIN_CON_of_ROLE <min> <max>	*the minimum cardinality of a role is at least <min> and at most <max>*
ET_per_ROLE <min> <max>	*the number of entity types per role is at least <min> and at most <max>*

Attributes

SUB_ATT_per_ATT <min> <max>	*a compound attribute has at least <min> components and at most <max> components*
DEPTH_of_ATT <min> <max>	*the decomposition level of a compound attribute must be at least <min> and at most <max>*
TYPES_ALLOWED_for_ATT <list>	*List of allowed types of attributes, <list> is the list of the allowed types*
TYPES_NOTALLOWED_for_ATT <list>	*List of all forbidden types of attributes, <list> is the list of the forbidden types*

Groups

COEXIST_in_GROUP <yn>	*coexistence groups are allowed <y>/not allowed <n>*
EXCLUSIVE_in_GROUP <yn>	*exclusive groups are allowed <y>/not allowed <n>*
ATLEASTONE_in_GROUP <yn>	*at-least-one groups are allowed <y>/not allowed <n>*

The Assist menu (3rd assistant, continued)

Entity type identifiers

COMP_per_EID <min> <max>	*an entity type identifier has at least <min> and at most <max> components*
COMP_per_EPID <min> <max>	*an entity type primary identifier has at least <min> and at most <max> components*

Rel-type identifiers

COMP_per_RID <min> <max>	*a rel-type identifier has at least <min> and at most <max> components*
COMP_per_RPID <min> <max>	*a rel-type primary identifier has at least <min> and at most <max> components*

Attribute identifiers

COMP_per_AID <min> <max>	*an attribute identifier has at least <min> and at most <max> components*
COMP_per_APID <min> <max>	*an attribute primary identifier has at least <min> and at most <max> components*

Access keys

COMP_per_KEY <min> <max>	*an access key group has at least <min> and at most <max> components*

Referential constraints

COMP_per_REF <min> <max>	*an constraint has at least <min> components and at most <max> components*

Processing unit

ALL_PROCUNIT	*finds all processing units*

Names

NONE_in_LIST_NAMES <list>	*<list> is a list of words. None of them can be used for any name of any object in the schema*
NO_CHARS_in_LIST_NAMES <list>	*The names of schema, entity types, rel-types, attributes, roles or groups cannot include any character of the list <list>*
ALL_CHARS_in_LIST_NAMES <list>	*The names of schema, entity types, rel-types, attributes, roles or groups must be made of the characters of the list <list> only*

The Assist menu (3rd assistant, continued)

The library *a library entry is a complex rule. It has a name (which appears in the list box) and a definition. The definition is a single rule made of one or more predicates with their parameters. The library can be edited by pressing the* **Edit library** *button. It can be saved.*

Script management *an Analysis script defines a sub-model as a sequence of rules. Each rule is associated with one of the object types of the model. A rule is either an elementary constraint, among those described above, or a boolean expression of elementary constraints. The boolean expression uses the logical operators* ***and***, ***or*** *and* ***not****. Parentheses are not allowed, but more than one rule can be defined for the same object type. The current script, if any, appears in the Script window.*

Add	*adds the selected constraint at the end of the current script*
Insert	*inserts the selected constraint before the current one in the script*
Remove	*deletes the current constraint from the script*
Edit	*edits the argument values*
Clear	*deletes the current script*
Pre-defined	*loads a pre-defined sub-model. The following scripts are available:*
VAX RDB/VMS 4.2	*Vax RDB relational model*
COBOL	*logical COBOL model*
CODASYL	*logical CODASYL model*
ER-normalization	*conceptual model*
Practical UML	*UML model extended with DB-Main facilities*
Strict UML	*UML model strictly respected*
XML	*XML (DTD) model*
Full-ER (future)	*correct Entity-Relationship model; conceptual constructs only*
simple-ER (future)	*correct Entity-Relationship model - no is-a, no complex constraints*
Binary (future)	*binary model: no is-a; no complex rel-types*
Bachman (future)	*Bachman model*: *Binary + no many-to-many, or cyclic rel-types*
Flat binary (future)	*Binary model + single-valued, atomic attributes only*

The Assist menu (3rd assistant, continued)

Flat Bachman (future)	*Bachman + single-valued, atomic attributes only*
logical (future)	*logical model*
relational (future)	*logical relational model*
IMS (future)	*logical IMS model*
Load	*loads a previously saved script (file *.scr)*
Save	*saves the current script (file *.scr)*
Copy	*copies the script to the clipboard*
Edit library	*opens the analysis library edition dialog box.*
New	*creates a new library entry*
Delete	*deletes an entry from the library*
Rename	*renames a library entry*
Add	*adds a new predicate to the current library entry*
Insert	*inserts a new predicate to the current library entry*
Remove	*removes a predicate from the current library entry*
Edit	*edits the parameters of a predicate of the current library entry*
Copy	*copies the script to the clipboard*
Load library	*loads a library from a disk*
Save library	*saves the current library to a disk*
Close	*closes the current dialog box*
Help	*explains the meaning of the analysis library*

OK button	*if the script window is empty, executes the selected constraint* *if the script window contains rules, evaluates them*
Cancel button	*closes the assistant without taking any actions*
Help button	*explains the meaning of the constraint, and the format of its arguments. Help can also be called when the assistant asks for the arguments*

The Assist menu (continued)

Fourth Assistant: **Integration**

This assistant offers a set of tools for integrating data schemas and objects.

Schemas... *integrates an data schema from the project into the current data schema.*
In the list of the data schemas (except the current one), selects the data schema that must be integrated into the current data schema. The schema integration rules can be found in the help file.

Objects... *integrates two objects (entity types, relationship types or compound attributes) in the same data schema or between two different data schemas (from the slave to the master). There are six integration strategies. Attributes, processing units, roles, is-a relations and their properties can be migrated selectively.*

Copy slave into master	*copies the slave object (if not empty) and its components into the master schema.*
Merge slave into master	*merges the slave object with the master object.*
Create a 1-1 link	*copies the slave entity type and its components into the master schema and create a rel-type between the master and slave. Its cardinalities are (1-1/1-1) or (0-1/1-1).*
Slave is-a master	*copies the slave entity type and its components into the master schema and creates an is-a relation between the master and slave entity types. The master ET is the supertype.*
Master is-a slave	*copies the slave entity type and its components into the master schema and creates an is-a relation between the master and slave entity types. The slave ET is the supertype.*
Create common supertype	*copies the slave entity type and its components into the master schema and creates a common supertype for the master and slave entity types.*

Remark: The identifiers made of transferred attributes or roles are also integrated.

The Assist menu (continued)

Fifth Assistant: **Text Analysis**

This assistant offers a set of tools for text analysis. The text to analyse can be an external text (such as a source program) or the contents of semantic or technical descriptions of the current schema.
The analysis is based on a pattern engine which can search the selected textual materials for specific patterns. The patterns are defined in PDL, a Pattern Definition Language whose specification can be found in the help file.
The patterns used to analyse the texts are stored in pattern libraries (.PDL files). In most cases, two libraries are used, the main library, and the secondary library. The main library contains the patterns to be selected by user (e.g. COBOL or SQL statements), while the secondary library contains the definitions of the low-level patterns used, but undefined, in the main library (e.g. COBOL separators, C name syntax).*

Load pattern... *loads and edits pattern files (*.pdl) to be used in text analysis (source, SEM, TECH)*
The complete name of these files are stored in the WINDOWS\DB_MAIN.INI file. In further uses, these files will be automatically loaded when the assistant is used. This tool will be used either the first time the assistant is used, or when you want to use other libraries.

- **Secondary** — *Concerning the secondary library*
  - **Edit** — *changes its contents*
  - **Browse** — *selects another secondary library*
- **Main** — *Concerning the main library:*
  - **Edit** — *changes its contents*
  - **Browse** — *selects another main library*
- **OK** — *Confirms the changes*
- **Cancel** — *Discards the changes*
- **Help** — *calls the help file (specification of the Pattern Definition Language)*

The Assist menu (5th assistant, continued)

Search... *interactive pattern matching processor: searches specified texts for the next instance of a pattern, or for relationships between text components defined by patterns. The text to be searched can be the source text in the text browsing window, the semantic or technical descriptions or the names of the components of the current schema.*

Pattern	*name of the pattern and its definition*
Value	*value to be given to the selected pattern variable*
variable list box	*list of the variables of the pattern. They can have a value*
Sem	*searches the semantic descriptions in the current schema*
Tech	*searches the technical descriptions in the current schema*
Name	*searches the names in the current schema*
Case sensitive	*indicates if the search is case sensitive or insentive*
Select all	*selects all the lines or the objects that contain the pattern*
Change	*assigns the value (in the **Value** field) to the selected variable*
Clear	*clears the value of the selected variable*
Clear All	*clears the value of all the variables*
OK	*executes the search for the first instance of the pattern*
Cancel	*abandons the actions*
Help	*calls the help file*

Search next **<F3>** *searchs for the next instance of the pattern into the specified text*

Execute... *prepares and executes a Voyager 2 function with the variables of a pattern as parameters. The function works on the current instance of the pattern*

Pattern	*name of the pattern*
Value	*value to be given to the selected pattern variable*
variable list box	*list of the variables of the pattern. They can have a value*
Sem	*searchs the semantic descriptions in the current schema*
Tech	*searchs the technical descriptions in the current schema*

Case sensitive	*indicates if the search is case sensitive or insentive*

The Assist menu (5th assistant, continued)

Change	*assigns the value (in the **Value** field) to the selected variable*
Clear	*clears the value of the selected variable*
Clear All	*clears the value of all the variables*
Program V2	*name of the selected Voyager 2 program*
Procedure V2	*name of the selected Voyager 2 procedure*
parameter list box	*list of the variables to be used as parameters by the Voyager 2 procedure*
First	*adds the selected variable to the parameter list box (1st position)*
Next	*adds the selected variable to the parameter list box (next position)*
Remove	*removes the selected parameter*
OK	*executes the procedure with the specified parameters*
Cancel	*abandons the actions*
Help	*calls the help file*

Dependency... *builds the dependency graph of all the variables of the source program.*
The graph is built by considering relationships between variables defined by statements (such as assignment, comparisons, etc.) described by specific patterns to be chosen.
The source text can then be queried by selecting a variable definition; all the variables which depend on it (and conversely) are highlighted.

Pattern	*name of a pattern (should have 2 variables named var_1 and var_2).*
pattern list box	*list of the pattern used to define the relationships between variables.*
Add	*adds the selected pattern to the pattern list box.*
Delete	*removes the selected pattern.*
directed	*the relationship are directed from var_1 to var_2 or not.*
Case sensitive	*indicates if the computing of the dependency graph is case sensitive or insentive.*
Separator	*name of the pattern defining the separators enclosing variable names in the text.*
Save dependency graph	*saves the dependency graph in the given file by (chosen by the **Browse** button).*

The Assist menu (5th assistant, continued)

OK *computes the dependency graph.*
Cancel *abandons the operation.*
Help *calls the help file*

Program slicing *colorizes the lines belonging to the slice computed with respect of the selected statement in the source file window; if the statement uses several variables or if they are compound, the user is requested to select the component according to which the slice is computed (the so-called "slice criterion");*

Mark slice *marks the lines in the source file window that belongs to the last computed slice*

Slicing assistant *computes several slices with options: shows their union or intersection, computes forward or backward slice, computes on the current line, on the marked lines or with start instruction containing condition.*

Dependency / SDG d*isplays the variables depending of the selected variable*

Load / Save *manages the System Dependency Graph*

- **Load SDG** *loads the System Dependency Graph of the current source file*
- **Save SDG** *saves the System Dependency Graph of the current source file*
- **Load parsing** *loads the parsing tree of the current source file*
- **Free structure** *frees the parsing tree and the System Dependency Graph of the current source file*

Settings *changes the color of the dependency graph and the slice, stores or not the line number into the clipboard.*

Clear color *puts in black the colored lines of the program slice and dependency graph*

The Assist menu (5th assistant, continued)

Goto *shows the mapping between a source text file and its corresponding logical schema. Both must be opened. The active mapping can be used in both ways:*

1. *in the schema window:*
   select an object in the schema (in any graphical/textual view)
   ***Assist/Text analysis/Goto**: → the origin source text line of the selected object is highlighted in the text window*
2. *in the source text window:*
   select a statement in the source text
   ***Assist/Text analysis/Goto**: → the object extracted from this statement is selected in the schema window*

Using the dependency graph:

*click on the name of a variable with the **right button,** all the instances of all the variables linked to the selected variable are highlighted (if the variable does not appear in the dependency graph no variable are highlighted). Pressing the Tab key positions the next instance in the middle of the screen.*

Searching for patterns, computing dependency graphs and program slices use very complex algorithms that can take some time when applied to large and intricated programs. However,displaying the dependency graph of a variable is immediate.

The Assist menu (continued)

Sixth Assistant: **Reference key...**

This tool proposes some popular heuristics to find foreign keys. The analyst chooses one of the two strategies:

- *given a candidate foreign key (i.e. a group), finds the possible target record types;*
- *given a group (usually an identifier), finds the field (an existing group or an attribute) of the schema that could reference the group.*

*To use the reference key assistant, execute **Assist/Reference key**. A first dialog displays in its top part the choosen strategy. Check one of the radio button on the left to choose the groups used as the starting point of the search (selected, marked, primary identifier, any identifier, any group or find by a Voyager 2 program). Those groups are on the top if the first strategy was chosen, otherwise they are on the bottom. The other part of the window displays the groups that match the search criteria. To set the search criteria to their default value, click on the **Reset** button. To change the search criteria, click on the **Search** button. The dialog box of the search engine appears.*
If the first strategy is selected it is possible to choose the type of the target group of the constraint (primary identifier, any identifier or any kind of group). If the second strategy is selected, it is possible to accept mere attributes (i.e. that do not form a group) as candidate reference key and to accept multivalued reference key.

The other options are common to both strategies.
*The **structure matching** rules defines the relation between the structure of the candidate matching two groups:*

- *they have the same overall length;*
- *they follow roles (hierarchical foreign keys);*
- *considered pairwise, the components have the same length and/or the same type.*

*The **name matching** rules are only applicable if the two groups have only one attribute. It states the relation between the name of the attributes of the two groups. It checks if the name of the reference key includes:*

- *a specified keyword (ref, id, code, ...)*
- *some (or all) characters of the target entity type name (besides the key word)*
- *some (or all) characters of the target attribute name (besides the key word)*
- *the comparison can be case sensitive or not*

The Assist menu (6th assistant, continued)

*Click on the **More** button to mention a Voyager-2 procedure that checks whether two groups are matching.*
*Click on the **OK** button to return to the first dialog box. Now the list of matching groups appears in the combo box, each group is prefixed by its entity type name. If you select a group, then its entity type and its attributes appear. The groups marked with a "*" are already target or origin (depending of the strategy) of a constrain. To create a constraint, select the entity type and the group and then click on the **Create** button. The button **Create All** allows you to create all the proposed constraints.To remove a constraint from the list, click on the **Remove** button.*
*Click on the button **Advanced** to apply more sophisticated actions through Voyager 2 functions .*

6.8 The Engineering menu (Engineering)

The history of a database engineering process contains the trace of all the activities that were performed, all the products involved, all the hypotheses that were made, all the versions of the products resulting of those hypotheses as well as all the decisions taken. Naturally, the result is a complex graph. The engineering menu is the place where the history can be managed.

Use primitives *updates the selected products by using the primitive functions of the CASE tool.*

Copy schema & use primitives *copies the selected schema and updates the copy by using the primitive functions of the CASE tool.*

End use of primitives *terminates the use of the primitives.*

New engineering process *creates a new engineering process using selected products in input.*

End current process *terminates the current engineering process using selected products as output.*

Continue process *allows a terminated process to be continued if resulting products are still to reused.*

Take decision *takes the decision of continuing with one or some of the selected products.*

Properties *shows the properties of the selected process.*

Edit input/output/update products *edits input/update/output produts for the selected process.*

Control *allows DB-MAIN to adapt the level of control of user actions according to the methodology or history coherence.*

6.9 The Log menu (Log)

Logs (records) the actions carried out by the analyst, and replays them selectively. The logs are recorded into the schema. In the future, these functions will be replaced by more sophisticated History management and Replay processors.

Trace *enables/disables the recording of user actions in the current schema log*

Add check point... *inserts a checkpoint label in the current schema log*

Add schema... *inserts a copy of the current schema windows in the current schema log*

Add desc... *inserts a user message in the current schema log*

Clear log... *clears the current schema log*

Save log as... *saves the current schema log into a *.LOG file*

Replay *replays selected actions recorded in a log file (schema copies and messages are ignored) onto the current schema*

- **Automatic...** *carries out all the actions between selected checkpoints*
- **Interactive...** *carries out the actions under the control of the user*

6.10 The View menu (View)

Chooses the way the current schema (process or project) will be displayed in its schema (process or project) window.

Text compact *list of the collections, entity types and rel-types of the data schema*

Text standard *compact textual presentation + collection contents, IS-A, attributes,processing units, roles, groups, constraints and notes*

Text extended *standard textual presentation + additional details*

Text sorted *sorted list of all the names declared in the schema, together with their object type and parent object*

Graph. compact *graphical presentation of IS-A, roles, collections, entity types, rel-types and notes of a data schema*

graphical presentation of all products and derived dependencies of the project

Graph. standard *compact graphical presentation + attributes, processing units, groups and constraints of a data schema*

graphical presentation of processing units, internal or external data objects, relations (call, decomposition and in/out) and notes of a processing schema

compact graphical presentation + sub-processes of a process

Graph. UML *standard graphical presentation with an UML look of a data schema*

Graph. dependency *graphical presentation of all products and derived dependencies of a process*

Graph settings *defines settings for graphical project and process views:*

Zoom factor	*shrinks / expands the graphical representation of the current process or project*
Reduce factor	*reduces / enlarges the current process or project in its graphical space*
Grid format	*draws a grid in the window of the current process or project*
Show objects	*shows / hides the new schema and add text processes*

The View menu (continued)

defines settings for graphical data schema views:

Independent — *defines the ways the tool reacts when an object is moved in the graphical Schema windows*
- when checked**:** *only the moved object is redrawn, all the connected objects are unchanged*
- when unchecked: *all the connected objects are moved proportionally*

Zoom factor — *shrinks / expands the graphical representation of the current schema*

Reduce factor — *reduces / enlarges the current schema in its graphical space*

Grid format — *draws a grid in the window of the current schema*

Show attributes — *shows / hides the attributes of entity types and relationship types*

Show groups — *shows / hides the groups of entity types, relationship types and compound attributes*

Show proc. units — *shows / hides the processing units of entity types, relationship types and schema*

Show att. types — *shows / hides the type, length and decim of attributes*

Show stereotypes — *shows / hides the stereotype of attributes, entity types, rel-types, collections and processing units*

ISA square — *draws the isa relations with horizontal and vertical lines*

ET Shape — *chooses the shape of the entity types (round or square)*

ET Shaded — *chooses the shade of the entity types (when checked: entity types are shaded, otherwise no shades)*

RT Shape — *chooses the shape of the rel-types (round or square)*

RT Shaded — *chooses the shade of the rel-types (when checked: rel types are shaded, otherwise no shades)*

defines settings for graphical processing schema views:

Zoom factor — *shrinks / expands the graphical representation of the current schema*

Reduce factor — *reduces / enlarges the current schema in its graphical space*

Grid format — *draws a grid in the window of the current schema*

Show call relations — *shows / hides the call relations*

Show decomp. relations — *shows / hides the decomposition relations*

Show in/out relations — *shows / hides the in/out relations*

Show stereotypes — *shows / hides the stereotype of external or internal data objects and processing units*

The View menu (continued)

Alignment *aligns the selected object in any graphical view*

- **Left** *aligns selected objects so their left sides are on the left of the rectangle formed by their external edges*
- **Right** *aligns selected objects so their right sides are on the right of the rectangle formed by their external edges*
- **Hor. center** *aligns selected objects so their horizontal centers are in the center of the rectangle formed by their external edges*
- **Hor. space equal** *moves selected objects horizontally so they are spaced evenly within the rectangle formed by their external edges*

- **Top** *aligns selected objects so their tops are at the top of the rectangle formed by their external edges*
- **Bottom** *aligns selected objects so their bottoms are at the bottom of the rectangle formed by their external edges*
- **Vertical center** *aligns selected objects so their vertical centers are at the center of the rectangle formed by their external egdes*
- **Vert. space equal** *moves selected objects vertically so they are spaced evenly within the rectangle formed by their external edges*

- **Horizontal square** *aligns roles of selected rel-types horizontally with their rel-type. If the rel-type is cyclic, first role is aligned horizontally and the other vertically.*
- **Vertical square** *aligns roles of selected rel-types vertically with their rel-type. If the rel-type is cyclic, first role is aligned horizontally and the other vertically.*
- **Top square** *puts the selected rel-types (no cyclic) in a upper corner of the rectangle formed by the external edges of their entity-types*
- **Bottom square** *puts the selected rel-types (no cyclic) in a lower corner of the rectangle formed by the external edges of their entity-types*

Auto-Draw *in graphical views: proposes a new graphical layout of the current schema (useful in reverse engineering)*
in case of unsatisfactory result, do it again

Engineering method *displays a window containing the current engineering method*

6.11 The Windows menu (Windows)

The standard window arrangement functions of Windows

Standard tools	*displays/hides the standard floating tool bar*
Graphical tools	*displays/hides the graphical floating tool bar*
RE tools	*displays/hides the reverse engineering floating tool bar*
Transfo tools	*displays/hides the transformations floating tool bar*
Process tools	*displays/hides the process modeling floating tool bar*
User tools	*displays/hides the user-defined tools floating tool bar*
———————	
Property box	*displays/hides the property box*
Process hierarchy	*displays/hide the process hierarchy of the history*
Status bar	*displays/hides the status bar*
———————	
Tile	*tiles windows on the desktop*
Cascade	*cascades windows on the desktop*
Arrange Icons	*arranges all iconized windows*
Close All	*closes all opened windows*
———————	
1: *window name*	*selects the first window*
2: *window name*	*selects the second window*
etc	

6.12 The Help menu (Help or F1 key)

Help **<F1>** *displays the table of contents for the help system*

About DB-MAIN *displays version, copyright and DB-MAIN project informations*

7 The tool bars

The standart tool bar

	*builds a new project (**File/New project...**)*
	*opens an existing project (**File/Open project...**)*
	*saves the current project as the current *.lun file (**File/Save project**)*
	*saves the current project as a new *.lun file (**File/Save as...**)*
	*in a data schema view: saves a copy of the current schema (**Edit/Save point**)*
	*loads and runs a compiled Voyager-2 program (**File/Execute Voyager...**)*
	*continues an interrupted Voyager-2 program (**File/Continue Voyager**)*
	*reruns the loaded Voyager-2 program (**File/Rerun Voyager**)*
	*Standard textual view for current schema (**View/Text standard**)*
	*Standard graphical view for the current schema (**View/Graph. standard**)*
	*creates an new entity type (interactive or from the source text) (**New/Entity type...**)*
	*creates a new relationship type (**New/Rel type...**)*
	*creates a new attribute (interactive or from the source text) as the first child of the current parent object (**New/Attribute/First...**)*
	*creates a new attribute (interactive or from the source text) as the next brother of the current object (**New/Attribute/Next...**)*
	*creates a new role or a new relationship type or add a member to a role (**New/Role...**)*
	*creates a new identifier with the selected attributes and/or roles*
	*creates a new group with the selected attributes and/or roles*
	*creates a new collection (**New/Collection...**)*

The tool bars (continued)

	*adds a new note to the selected object (**New/Note...**)*
	*adds a new processing unit to the current schema, rel-type or entity type (**New/Processing unit...**)*
	*adds a new internal data object to the current schema (**New/Data object...**)*
	*adds a new call relation between two processing units into the current schema (**New/Call...**)*
	*adds a new decompositon relation between two processing units into the current schema (**New/Decomposition...**)*
	*adds a new in/out relation between a processing unit and an internal or external data object into the current schema (**New/In/out...**)*
	*examines / modifies the semantic description of the current object*
	*examines / modifies the technical description of the current object*
	*examines / modifies the meta-properties values of the current object*
	*colors the selected objects (**Edit/Color selected**)*
	*if checked, the selected objects are marked,if unchecked,the selected objects are unmarked, if greyed, some of the selected objects are marked and others unmarked; when pressed, mark the selected objects if unchecked, unmarked them if checked, and mark those which are unmarked if greyed (**Edit/Mark selected**)*
	*indicates the current marking plane; make one of the three marking planes the current one*

The tool bars (continued)

The graphical tool bar

	*if unchecked, the tools works in the **dependent** mode (when moving an object or multiple selection of objects all the connected objects are also moved), otherwise it works in the independent mode (only the selected object are moved)*
	*copies the graphical view of the selected objects onto the clipboard (**Edit/Copy graphic**)*
	*expands the graphical representation of the current schema (current zoom + 10%)*
	*shrinks the graphical representation of the current schema (current zoom - 10%)*
	*shrinks or expands the graphical representation of the current schema or project (choose a percentage or the fit option)*
	*aligns selected objects so their left sides are on the left of the rectangle formed by their external edges (**View/Alignment/Left**)*
	*aligns selected objects so their right sides are on the right of the rectangle formed by their external edges (**View/Alignment/Right**)*
	*aligns selected objects so their horizontal centers are in the center of the rectangle formed by their external edges (**View/Alignment/Horizontal center**)*
	*moves selected objects horizontally so they are spaced evenly within the rectangle formed by their external edges (**View/Alignment/Horizontal space equal**)*
	*aligns selected objects so their tops are at the top of the rectangle formed by their external edges (**View/Alignment/Top***
	*aligns selected objects so their bottoms are at the bottom of the rectangle formed by their external edges (**View/Alignment/Bottom**)*
	*aligns selected objects so their vertical centers are at the center of the rectangle formed by their external egdes (**View/Alignment/Vertical center**)*
	*moves selected objects vertically so they are spaced evenly within the rectangle formed by their external edges (**View/Alignment/Vertical space equal**)*

The tool bars (continued)

	*aligns roles of selected rel-types horizontally with their rel-type (**View/Alignment/Horizontal square***
	*aligns roles of selected rel-types vertically with their rel-type (**View/Alignment/Vertical square**)*
	*puts the selected rel-types (no cyclic) in a upper corner of the rectangle formed by the external edges of their entity-types (**View/Alignment/Top square**)*
	*puts the selected rel-types (no cyclic) in a lower corner of the rectangle formed by the external edges of their entity-types (**View/Alignment/Bottom square**)*

The reverse engineering tool bar

	*Searches a text for the first instance of a given pattern (**Assist/Text analysis/Search**)*
	*Searches a text for the next instance of a given pattern (**Assist/Text analysis/Search next**)*
	*When a logical schema and its source text both are visible, the origin source text line of the selected object is highlighted (**Assist/Text analysis/Goto**)*
	*Computes the program slice with respect to the current line (**Assist/Text analysis/Program slicing**)*
	*Finds the possible other groups connected to the current group (**Assist/Reference key**)*

The tool bars (continued)

The transformation tool bar (Menu Transform)

Entity type	*Transforms the current entity type*
→ Rel-type	*... into a relationship type*
→ Attribute	*... into an attribute*
Is-a → rel-types	*Transforms the is-a relation(s) of current entity type into one-to-one relationship type(s)*
Rel-types → is-a	*Transforms the current relationship type(s) of current entity type into is-a relation(s)*
Split/Merge	*Splits the current entity type into two entity types or merges two entity types linked by a one-to-one relationship type*
Add Tech ID	*adds a technical primary id to the current entity type*
Rel-type	*Transforms the current relationship type*
→ Entity type	*... into an entity type*
→ Attribute	*... into reference attributes (foreign key)*
→ Object att.	*... into object-attribute(s)*
Attribute	*Transforms the current attribute*
→ Entity type	*... into an entity type*
Disaggregation	*..., if compound, replacing it by its components*
Multi → Single	*..., if multivalued, into a single-valued attribute*
Single → Multi	*..., if single-valued, into a multivalued attribute*
Multi → List Single	*..., if multivalued , into a list of single-valued attributes*
Multi Conversion	*changes the collection type of the current attribute (with or without loss)*
Materialize domain	*materializes the user-defined domain of the current attribute*
Object att. → RT	*..., if object type , into a rel-type*
Role	*Transforms the current role*
Multi-ET → RT	*..., if multi-ET, into a list of relationship types*
Group	*Transforms the current group*
→ Rel-type	*..., if referential (foreign key), into a relationship type*
Aggregation	*... into a compound attribute*
→ Multi-valued	*... of single-valued attributes into a multivalued attribute*

The tool bars (continued)

The process modeling tool bar (Menu Engineering)

	*updates the selected products by using the primitive functions of the CASE tool*
	*terminates the use of the primitives*
	*creates a new engineering process using selected products in input*
	*terminates the current engineering process using selected products as output*
	*allows a terminated process to be continued if resulting products are still to reused*
	*makes the decision of continuing with one or some of the selected products*

The User tool bar (Menu File/User tools)

The user tools bar contains the list of Voyager 2 programs or menu entries (maximum 5) defined by the configuration dialog box (menu File/Configuration) in the db_main.ini file.

8 Special Mouse and Key actions

F1 key: displays the help file

Actions in a graphical project window

Mouse actions

left-side button (single click):	*selects an object (schema, text file or process)*
<shift> + left-side button (single click)	*adds an object to the selection*
draw a rectangle	*multiple selection*
drag a selected object	*moves all the selected objects in the windows*
left-side button (double click):	*opens the window of an object (graphical schema window for a schema, graphical process window for an engineering process and source file window for a text file)*
right-side button (single click):	*on a process, aligns the process between its products* *on a product, aligns the product between its processes*
"close window" icon	*closes the current engineering process view and shows its parent if there is one, or closes the project.*

Keyboard actions

enter (↵):	*opens the window of the selected object*
up-arrow (↑):	*if an object is selected, moves the object four pixel up* *otherwise scrolls the project one line upward*
down-arrow (↓):	*if an object is selected, moves the object four pixel down* *otherwise scrolls the project one line downward*
left-arrow (←):	*if an object is selected, moves the object four pixel left* *otherwise scrolls the project one screen to the left*
right-arrow (→):	*if an object is selected, moves the object four pixel right* *otherwise scrolls the project one screen to the right*
<ctrl> + arrow (↑, ↓, ← or →):	*if an object is selected, moves the object one pixel up, down, left or right*
<Del>	*deletes the selected objects (schema or text file)*

Special Mouse and Key actions (continued)

Actions in a textual schema window

Mouse actions

left-side button (single click):	*selects an object*
left-side button (double click):	*opens the property box of an object*
<control> + left-side button (single click):	*adds a line to the selection*
<shift> + left-side button (single click):	*selects all the lines from the first selected one to the current one*
right-side button (single click):	*goes to the origin definition of the object the name of which is clicked on e.g. clicking on a role sends back to its entity type*

Keyboard actions

<enter> (↵)	*opens the object property box of the selected object*
<tab>	*displays the next occurrence of the selected object*
up-arrow (↑)	*scrolls the schema one line upward*
down-arrow (↓)	*scrolls the schema one line downward*
left-arrow (←)	*scrolls the schema one half-screen to the left*
right-arrow (→)	*scrolls the schema one half-screen to the right*
<Alt> + ↑	*when an attribute, a processing unit, a group or an internal data object is selected: swaps it with the previous one when a role is selected: swaps it with the previous one*
<Alt> + ↓	*when an attribute, a processing unit, a group or an internal data object is selected: swaps it with the next one. When a role is selected: swaps it with the next one*
<Del>	*deletes the selected object*

Special Mouse and Key actions (continued)

Actions in a graphical schema window

Mouse actions

left-side button (single click):	*selects an object*
<shift> + left-side button (single click):	*adds an object to the selection or removes an already selected object form the selection*
draw a rectangle:	*multiple selection*
drag a selected object:	*moves all the selected objects in the window*
left-side button (double click):	*opens the property box of an object*
right-side button (single click):	*in a data schema: on a rel-type, aligns the rel-type between its entity-types* *on a role, aligns the role between its rel-type and its entity type(s)* *on a cluster, centers the cluster between the super-type and the sub-types* *in a processing schema: no action*

Keyboard actions

<enter> (↵)	*opens the object property box of the selected object*
up-arrow (↑)	*if an object is selected, moves the object four pixel up* *otherwise scrolls the schema one line upward*
down-arrow (↓)	*if an object is selected, moves the object four pixel down* *otherwise scrolls the schema one line downward*
left-arrow (←)	*if an object is selected, moves the object four pixel left* *otherwise scrolls the schema one screen to the left*
right-arrow (→)	*if an object is selected, moves the object four pixel right* *otherwise scrolls the schema one screen to the right*
<ctrl> + arrow (↑, ↓, ← or →):	*if an object is selected, moves the object one pixel up, down, left or right*

Special Mouse and Key actions (continued)

<Alt> + ↑	*in a data schema: when an attribute, a processing unit or a group is selected, swaps it with the previous one* *in a processing schema: when an internal data object is selected, swaps it with the previous one*
<Alt> + ↓	*in a data schema: when an attribute, a processing unit or a group is selected, swaps it with the next one* *in a processing schema: when an internal data object is selected, swaps it with the next one*
<Del>	*deletes the selected object*

Actions in a source file window

Left-side button (single click):	*selects a line*
Left-side button (double click):	*edits the description of the selected line*
<control> + left-side button (single click):	*adds a line to the selection*
<shift> + left-side button (single click):	*selects all the lines from the first selected one to the current one*

Actions in a property box

<enter> (↵):	*same as clicking on the **OK** button*
<Esc>	*same as clicking on the **Cancel** button*
<Tab>	go to the next field

Special Mouse and Key actions (continued)

Actions in the method window

Mouse actions

left-side button (double click):	opens the property dialogue box of the selected product type or primitive process type, or, on an engineering process type, displays its strategy instead of the current one.
right-side button (single click):	on any product type or process type: displays a contextual menu.
'Close Window' icon	closes the the current process type and displays its parent, if there is one; else, closes the window.

Process types contextual menus

Execute	executes a process of the selected type.
Terminate	terminates the execution of the processes of the given type.
Properties	shows the property dialogue box of the selected object (product type or process type).
View	on an engineering process type, displays its strategy instead of the current one.
Back	on the title of the current engineering process types, closes its view and displays the parent process type strategy.