Documentation of the Xdobry application

Artur Trzewik

Documentation of the Xdobry application

by Artur Trzewik

This documentation is computer translated from German language. I need your help to correct this docs

Xdobry is a new approach database form generator for relational databases. The forms are generated not from relational-schema (by reading data dictionary) but from conceptional-schema made by using reverse engineering. The forms supports rich semantic abstraction: association (relationship) as Form-Link, aggregation as nested Forms and object inheritance as Tab-Widget and foreign-keys as multi-columns lists. Therefore form are not another representation of tables but object at higher abstractions level. There are three programs: SchemaEditor for making a conceptional schema, FormEditor GUI Drag and Drop Editor for DB-Form and FormServer. The System is programmed in Tcl-Language (with extension Tk Tix XOTcl tDOM), used XML as Format for describing Form-GUI and DB-Schema (Repository).

Table of Contents

Chapter 1. Xdobry Specification

This chapter describes the intended purpose of the program and its main features. There you can find the answered, whom and for what it can be useful. The target user-group and their demands are named.

Principal purpose of the program

This program makes possible fast and simple production of DB-form-applications. DB-forms are the GUI interfaces to a databases which allow the comfortable treatment of data contents. The program sets in front an existing relational databases (SQL-databases) to him the forms should be created. The production of the forms run automatically or semiautomatic by help of special assistants. No programing knowledge is demanded. The created forms are not simply the another form of the tables in relational databases. They correspond rather to a higher abstraction-level confesses from conceptual models.

Features and Benefits

- platform independent: binary packages for Windows and Linux (Intel) accessible
- supports all main databases: Oracle, MS SQL-Server, mysql, postgres, MS Access, general ODBC
- Easy migration of data and schema between all supported databases.
- small runtime system. No special installation or further requirements are needed
- special wizards and assistant make easy migrate the data, manipulate schema or build GUI-forms
- reach set of special database GUI-elements

Characteristics

The program was developed as an open system of the creation of DB-applications, but only DB-forms applications are fully implemented yet. It is a CASE-tool for the databases area. In contrast to existing systems the production of the forms was understood as a multi-level process. The principal reason for the bad quality of the automatically created forms, with conventional system, is the use of the relational schema (data dictionary) as a source of information for form generator. Then such forms must be adapted manually. Relational schema (table structure) contains only few information about the semantics of the data. It are simple too few information for creating good forms. With Xdobry the production of the forms are made in three steps

1. Exact specification the DB-schema on the step of the conceptual model, use of reverse engineering technology. User DB-administrator
2. Automatic production of the forms. Adaptation of the appearance in the GUI-editor. User DB-application developer
3. Use of the DB-forms for the treatment of the data. End user

In this way man can produce quickly and hight quality DB-form. The forms are in this system more a representation of objects then tables.

1. The relationship among the objects are used to build a link-navigation for browsing the data domain. The form can be used like hypertext documents (hyper forms). One can discover the objects association (relationship) without the knowledge about data structure.
2. The forms support hight level abstraction as aggregation or heritage. This is performed as nested forms.
3. The complex modeling concepts like foreign-keys are hidden from user. The system handles the user friendly display of foreign keys and check the integrity.

For every step a separated program is responsible which was developed for other target user group:

1. SchemaEditor
2. FormEditor, form generator and GUI-Drag and drop editor
3. FormServer

Xdobry It was developed as a part of a dissertation. It is still , extended and modified

Target user-group

Xdobry is a high-specialized program for the development of databases applications. Correspondingly high are the demands for the users. I have not developed this program to make complex systems, such as databases, simple. It should make only the handling of such systems easy. These are tools for DB-specialist with which the development of DB-forms can be effective and can be fast.

The system Xdobry was developed in 3 components for 3 different user's groups.

1. DB-administrator: knowledge: relational databases, DB-developing, relational schema. Conceptual schema
2. DB-application developer: knowledge: GUI-developing
3. End user: fundamental use of computer

Technical data

The program was implemented under Linux with programing language Tcl (and its extensions: Tk, Tix, tDom, XOTcl). Because of a lot of components the compilation and configuration can be difficult. MySql relational databases was used as a primary test database-manager. The was designed to work with standard SQL-92 databases.

Version Changes Log

Version 0.12

- Grid Packer (Geometry Manager) is supported
- Form-Links are extended, one can browse with forms like with hypertext-documents
- The is new API for user-macros for FormServer
- Property-Dialog in FormEditor was new programmed.

- You can reduce the forms functionality (no delete, no insert, no update).
- Sorting. You can view sorted tuples in forms.

Version 0.14

- Many Bugs was killed
- Better Error-Handling (SQL Error from databases are catch)
- adapt for mysqltcl2.0 and xotcl8.3
- the DB-interfaces are loaded on demand in separates files.

Version 0.15

- tested with RedHat 7.0, mysql 3.23 and postgresql 7.0
- The most changes are done by FormServer.
  - insert-only modi is supported
  - many start options
  - checking of not null condition
  - keybord accelator can be used for navigation
  - many user friendly gui-logic
- Support of dbitotcl interface. It allows to user all databases that have a Perl DBI interface (experimental tested only with mysql).

Version 0.16

- Program code refactoring.
- Form filters can be named and saved.
- Form supports the templates. You can fill up quickly the data fields with specified data.
- All input can be validated by data checker
- You can specify the help text to all widgets and get it by F1-Key.
- many bugs are killed.

Version 0.20

- extensive refactoring by FormEngine and FormFactory packages
- Filters becomes own windows by FormServer (more flexible filters are possible)
- The is new API for user-macros for FormServer
- Property-Dialog in FormEditor was new programmed.
- Xdobry is developed with XOTclIDE. My own IDE for XOTcl (www.xdobry.de/xotclIDE) it means Xdobry has new internal structure.

Version 0.30

- Supports sqlite database

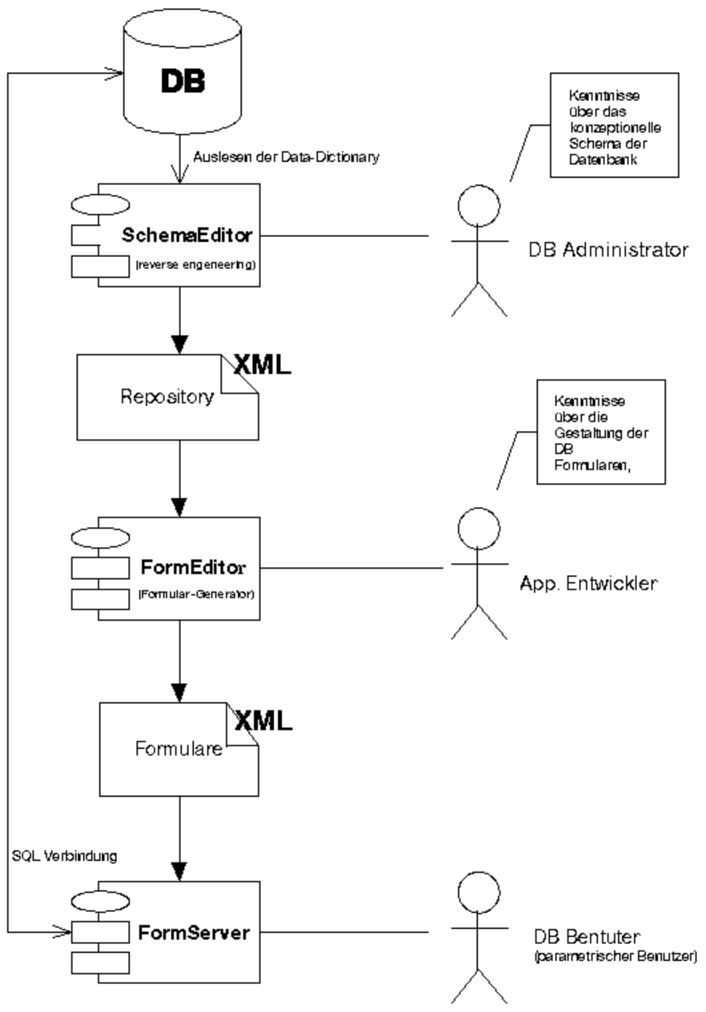
- Deployed also as starkit for Windows and Linux. No further requirements for installation.
- context sensitive menus activation

Version 0.40

- Supports MS SQL-Server, MS Access and Oracle
- SchemaEditor can create and modify DB-Schema
- Xdobry can migrate data and schema between all supported databases
- For internationalisation tcl package msgcat is used
- imporved documatation
- many bug fixes
- tclx is not more used

Chapter 2. User manual

System components


DB
Auslesen der Data-Dictionary
Kenntnisse über das konzeptionelle Schema der Datenbank
SchemaEditor
(reverse engeneering)
DB Administrator
XML
Repository
Kenntnisse über die Gestaltung der DB Formularen,
FormEditor
(Formular-Generator)
App. Entwickler
XML
Formulare
SQL Verbindung
FormServer
DB Bentuter
(parametrischer Benutzer)

Das system consists 3 executable programs **SchemaEditor**, **FormEditor** and **FormServer**. There are only one starkit package but at the start of Xdobry you must choose which module do you want to start. The interfaces between these separate programs are to 2 XML-documents types; the Repository (standard file-extension *.xmldbschema*) and form-description (standard file-extension *.xmlforms*). The treatment of both XML-documents is taken over from corresponding programs, however, can take place also manually in an editor. You can find the fitting DTDs (Document Type Description) in catalog`dtd`.

SchemaEditor

The major task of the schema-editor is to create a XML-Repository and to manage it. A user of the schema-editor is assumed as a databases-administrator. The user has knowledge about the structure of the data and its representation in the relational model. The **SchemaEditor** possesses the following functional characters:

1. To convert Data Dictionary of a relational databases to a XML-Repository.

2. Add semantic information to the Repository. By using the reverse engineering colleges, which can run either completely automatically or by questioning the user. The user can also add the semantic information separately.
3. Add and editing the Meta-information of the attributes.
4. Create and drop of tables
5. Create and drop of table rows
6. Create database from XML-schema
7. Create XML and SQL database dumps
8. Import and export schema between all supported databases
9. Migrate data and schema between all supported databases

Important: In the opposite to other DB-manager SchemaEditor does not work directly on data base schema. The data base schema is picked out and converted into own data structure (Repository), which can be stored also as XML file (xmlschema) and loaded again. This structure is not data base system specific. In this way the editing of DB-repository without db-connection is possible. SchemaEditor can produce Data Definition SQL for each data base system supported by Xdobry Also the changes of the db-schema can be generated as DDL-Sql statement.

The idea of the GUI is based on the representation of the schema as a tree structure which can be manipulated by the user. This tree structure has the following edges types.

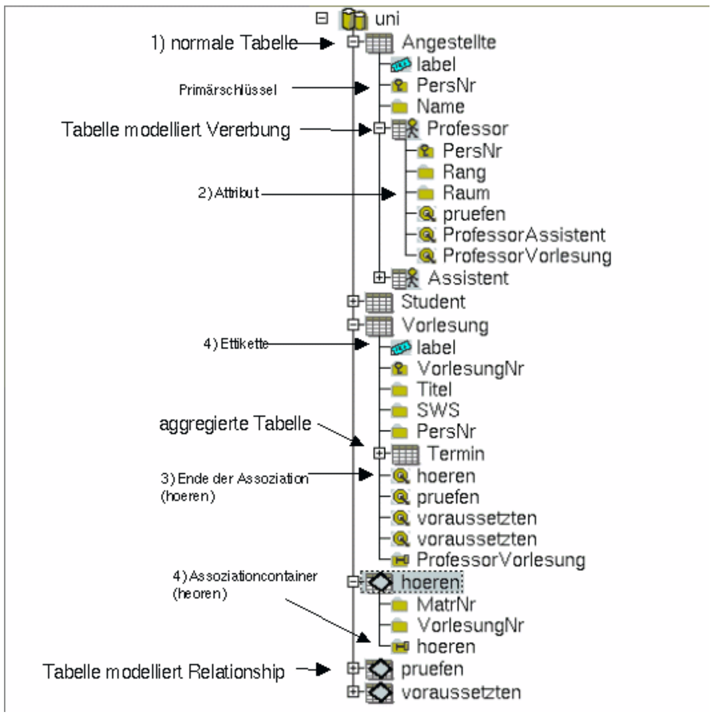


Figure 2-1.

1. Table
2. Attribute

3. Associationcontainer
4. Assoziationtarget
5. Tables-Label
6. Attribute-groups (structured attributes)

Reverse engineering

The relational model is strong value oriented. It knows no association (relationship) types. The association are represented as a foreign key. The tables can represent an Entities (Objects), relations-ship, partial objects (by subtypes modeling) or properties (e.g. , lists of the attributes). With reverse engineering the semantic meaning, also the information how the relational model is to be interpreted, are achieved. This information is coded only as the names of the attributes. Neither MySql nor Postgresql support the definition of foreign-keys. This is big disadvantage by the production of the forms. The reverse-engineering algorithms are based only on the relational schema, the data content is not evaluated. It were implemented three reverse-engineerings algorithms. The foreign-key-search is a basis for other algorithms (out of the subtype-search).

Foreign key-search

Find all attributes with the same name as the primary keys of the other tables, however, the Prim-attributes are not Prim-attribute or are not a single Prim-attribute. Limitation: object tables consist only single prim-attribute. No Recursive relations! Action: a simple reference is constructed. Association-container gets the foreign key. Association-Target gets the primary key. It is a basis for the other algorithms.

Subtype-search

Search for the subtype. Find all tables having one primary keys of the same name. They(you) must determine the father's table themselves. With Multi-level inheritance (grandson's objects' the action must be executed repeatedly. Limitation: object tables consist only single prim-attribute.

Find association-tables

These reverse engineering technology is based on the step "foreign-key-search". The tables should be determined which model Relationship (e.g., n:m Relationship). Algorithm: Find all tables whose primary key exists(consists) of several foreign keys or several foreign keys and no unambiguous primary key have. Action: the Relationship-tables are particularly drawn. N:m or n:m:z. Relations are recognized. Relations may have own Attributes.

Suggest Aggregation

These reverse engineering technology is based on the step "foreign-key-search". Suggest the Aggregation (composition) of the tables (Embedded tables). The algorithm shows only the suggestions, the Aggregation-semantics can be determined only by used. Suggestions: 1. All tables only a foreign key have. It can exist still farther Aggregation. If they check 1:n associations

Define abstractions

Three sorts can be defined by abstractions of the conceptual databases modeling: association, Aggregation (composition) specialization (subtypes). The definition takes place gradually with the help of assistants.

Association

Is the most difficult to define. It corresponds to the Relationship of the ER-model. It are placed the following questions.

- Is the association recursive? There are relations (Relationship) between the objects of the same type.
- Is there a table with references? If the association with help had to be illustrated by additional tables what is relations always the case with the N:M, or how with 1:N relations there is only a reference in one of the table
- Degree of the Relationship: with the N:M it is 2 with the N:M:O it is 3
- Rolle-names: (not obligatorily)
- Existence dependence: Do not become till present farther supported, however, can be defined here

For the n:m relationship must become clear: which objects (tables) participate in the relationship and which table contains the references?

An association is understood as container (collection) with references to objects. The collection can be modeled either as a separated table with references or as a reference in an object (1:n). Around an association become to model two new edeg-types added:*Assozitaioncontainer* With references and *Assoziationtarget* With objects. An association container and at least two Assoziatontargets belongs to every association (relationship). The associations (association container) possesses an unambiguous name. In this way can also be modeled complex association such as recursive 1:n and n:m relations and relations the higher Granularity n:m:s:r. Correspondence in ER-modelFigure 2-8

Aggregation

Here it is about the possibly modeling of nested tables. They are shown as embedded (nested) forms from **FormServer**. One must specify: the container-table, which element-table and reference, foreign key in the element table which points at primary key in container-table. Correspondence in ER-modelFigure 2-10

Subtypes (inclusion dependency)

Also known as inheritance or generalization. There is always a father object and a child object which are modeled in two tables (primary key heritage). One must also give the inherited primary key. Correspondence in ER-model Figure 2-9.

DB-Schema create and modify

Xdobry can be used for modellung data base schema. Tables and columns can be created, modified and deleted. Xdobry does not work directly Dictionary (schema) but has own data base system independent representation of the schema, which can be put down as XML file. It is possible to develop schema without connection data base and install many databases from one XML-repository. The Xdobry schema is based on ER-diagram and contains more semantic information than the Data-dictionary of data base.

Creating new schema

For that data base connection is not necessary. By the menu **Schema**⟶**New** new data base schema is created. Use context sensitive menus (select a leaf and right mouse button click) can be used to creat tables and columns.

Xdobry own has data types system, in addition, which is set on Mysql data base system but supports other types (like Money or Boolean).

Table 2-1. supported DB-Types and type conversion

Xdobry	Mysql	MS Access	MS SQL	Postgres	Oracle
decimal	Decimal, numeric	Currency, money	monay	money	number
double	double	float	float	double	double
float	float	real	real	real	float
int	int	Integer, int	int	Integer, int	int
smallint	Smallint, tinyint	smallint	smallint	smallint	smallint
Boolean	smallint	boolean	boolean	Boolean, bool	Number(1)
text	Text, mediumtext, tinytext	Memo, LONGTEXT, LONGCHAR, MEMO, NOTE, NTEXT	image	text	clob
datetime	Datetime, date, time	datetime	timestamp	timestamp	timestamp
timestamp	timestamp	datetime	datetime	timestamp	timestamp
enum	enum	Varchar(50)	Varchar(50)	Varchar(50)	Varchar(50)
set	set	Varchar(50)	Varchar(50)	Varchar(50)	Varchar(50)

Xdobry	Mysql	MS Access	MS SQL	Postgres	Oracle
varchar	varchar	TEXT(n), ALPHANU-MERIC, CHARAC-TER, STRING, VARCHAR, CHARAC-TER VARYING, NCHAR, NATIONAL CHARAC-TER, NATIONAL CHAR, NATIONAL CHARAC-TER VARYING, NATIONAL CHAR VARYING	varchar	text	varchar2
char	char	char	char	char	char
longblob	Longblob, medium-blob, tinyblob	IMAGE, LONGBI-NARY, GENERAL, OLEOBJECT, BINARY	image	bytea	blob

Table 2-2. autoincrement rows

Database	Kind
MS Acces	Typ Counter
MS SQL	Indent(1,1)
mysql	autoincrement
Postgres	Sequencer
Oracle	Sequencer
Sqlite	Not supported

Xdobry can also create autoincrement rows or sequencer.

Table 2-3. Autoincrement Spalten

Database	Typ
MS Acces	Typ Counter
MS SQL	Indent(1,1)
mysql	autoincrement
Postgres	Sequencer

Database	Typ
Oracle	Sequencer
Sqlite	Not supported

MS database can with [] maksing create tables with space. This would by converted to _.

Table 2-4. Leerstelle in Tabellennamen

Datenbank	Tabellenname
MS Access	[Order Details]
MS SQL	[Order Details]
mysql	Order_Details
postgres	Order_Details
Sqlite	[Order Details]
Oracle	Order_Details

For create Schema on specific data base use menu **Schema**⟶Operations on Schema⟶**Create DB from Schema**. By the menu **Schema**⟶Operations on schema⟶**Show SQL definition for schema** can you Create sql for the specific data base product. This SQL can be supplemented by data base-specific procedures, that Xdobry does not support. The in such a way adapted SQL must be up-played with data base-own tools.

Data base schema modifing

If you want to change the data base already existing, you must first create Xdobry representation of schema. Use for it the menu **Schema**⟶New from DB. They must specify then the data base connection. The changes on that schema are transferred not immediately to the data base. By the menu **Schema**⟶Schema operations⟶**Apply Schema changes to DB** the change on the data base are applied. By the menu **Schema**⟶Schema operations⟶**Show schema changes** can you the see the corresponding DDL-SQL statments.

Tip: This SQL-Statments can be used to update many database synchron (development, Integration, production).

If the Xdobry XML schema already exist load it first. SchemaEditor offers by loading XML-Schema automatically to connect source database. If another connection is desired or the schema was not developed from a data base, you can connect database with **Schema**⟶Connect with data base In the case that schema of the data base is not consistently with that is schema of Xdobry e.g. the schema of the data base reworked on with other tools, you can with the function **Schema**⟶Operations on schema⟶**Synchronize schema with data base** synchronize XML-repository with data base schema. Afterwords the schema can being adapted like accustomed.

Important: One cannot record the changes to schema between into several meetings. So the changes on the data base must be up-played, so that you are not lost through terminating the meeting. Schema editor does not possess a function for synchronizing that data base schema to Xdobry Schema.

SchemaEditor and data migrate

Schema editor possesses various possibilities to migrate the schema and the data between different data base system. Schema editor can keep at the same time 2 data base connections open and transport sentence by sentence the data from data base another. The range of the data too migrated can be adapted. Both tables can be excluded or also individual columns from the migration.

Accomplish the migration

1. Schema from the source data bank produce. The connection with source data bank must be present
2. The tables or columns, which are not to be migrated, are to be removed from that schema (you are not actually deleted in source data bank)
3. The migration start by menu **Schema**⟶Migration⟶**Migrate to data base**
4. Arise during the migration errors. These are indicated. The migration can be broken off or continued in this case. In the case of abort the target data bank is not deleted

SchemaEditor and data exporting and importing

Schema editor knows the data on 2 points to export. As XML dump **Schema**⟶Migration⟶**Database to XML-SQL** the SQL dump **Schema**⟶Migration⟶**Database to XML-Dump**. For SQL dump the type of target data bank must be indicated.

Important: The export or import of XML-dump is suitable only for smaller data bases. At now with XML-dumpe into the main memory the entire data base is illustrated at short notice (DOM-object). In this way the Xdobry can with larger data sets fast overflow. An iterative production of XML is possible and becomes for the next version of Xdobry planned. The SQL-dump is provided iterative and causes no excessive demand of main memory.

With the function **Schema**⟶Migration⟶**XML dump to SQL convert** one can convert a XML departure to SQL departure.

FormEditor

The purpose of the component is to generate a collection of the forms to a databases and secondly an user's interface for the adaptation of these forms. The product of the form-editor is a description of the forms as a XML-document. Associations (Relationship) and other abstractions are supported in the forms directly. The user of the form-editor (application-developer) must not know the schema of the data. His task is only to adapt the automatically produced forms under the formative face point or to specify the input fields of the form.

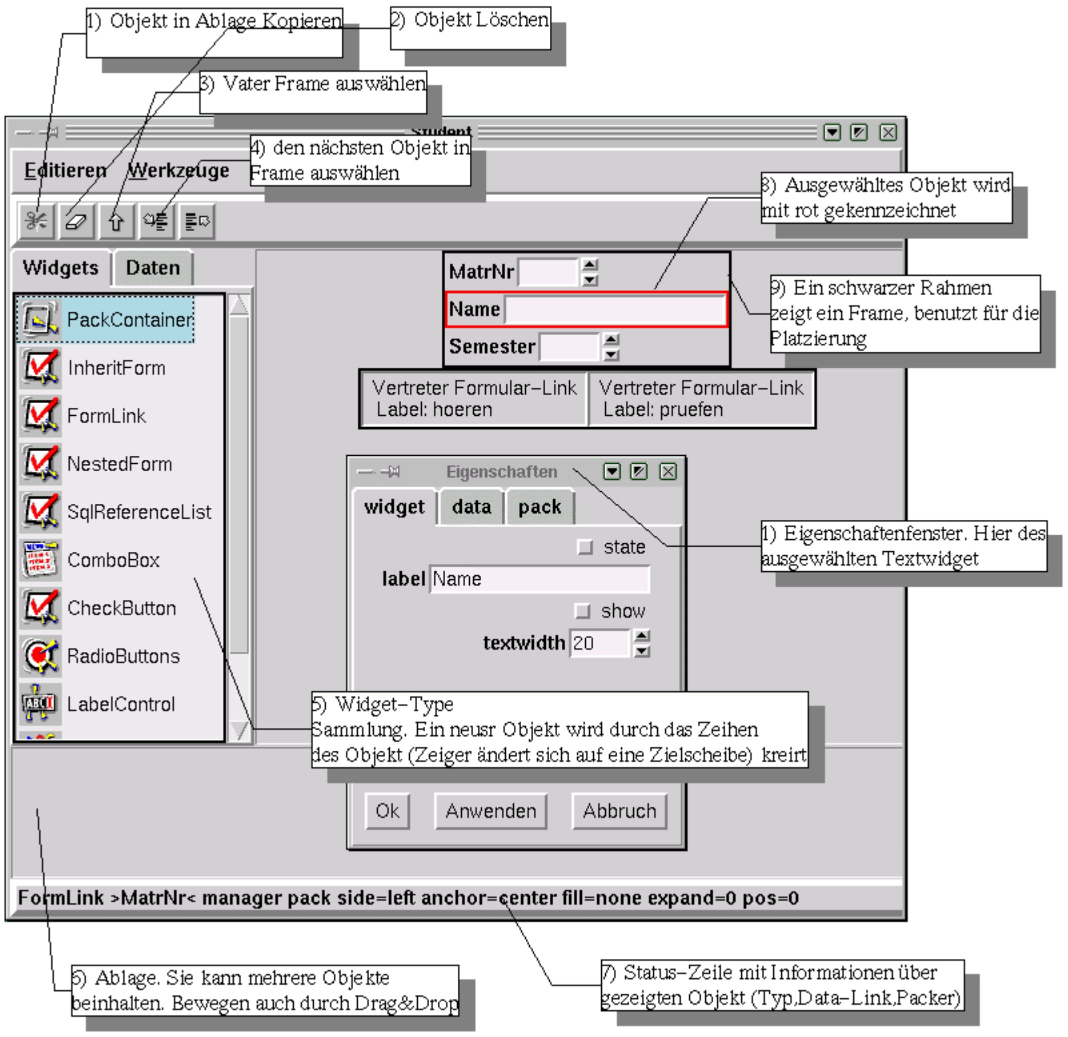
A form becomes as a collection of different input fields, their placement, and their correspondence to objects in the databases (rows in a table). The following types of the input fields were realized:

Frames. Frames for the placement of the elements
Simple text fields (one-line)
Input fields for whole numbers with control arrows
Listsbox
Radiobuttons
Checkbox fields (Yes / No switches)
Text fields consisting of several lines
Multicolumn Listboxes for the representation of the foreign key (references)
FormLinks
Object for the nesting forms

The embedded forms and the formlinks belong to special elements of the forms. Both are a different visualization of the relationship. The FormLinks appearing along corresponding paths (association or also Aggregation). Besides, one can reach from a form a form of the object which has a foreign key on the first form. Besides, the foreign key is faded out with the tied up form (the value is determined by the first form) and the files are filtered after the foreign key. It are shown by the filtering only the objects which are in touch with the shown object in the first form. With embedded forms the tied up form is a part of the parent's form. With form-links a command button is inserted, only to open of the tied up form serves. The foreign keys are not treated as normal attributes but are represented as special diagrammatic elements. First the domain of the foreign key is limited by integrity conditions on the amount of primary key values in the corresponding table. This can be represented by a listbox. Then the user knows that he has only a certain number of possibilities at which he can look and select. Secondly the values of the key are often statement-empty for the user or unsuitably for the identification of the object. For it the user expects other attributes (e.g. name and first name besides of the PersonalIDs). The Aggregation can be realized as the embedded forms. The association is realized as form-links. The specialization can also be realized as a linkage by forms. Besides, a form is applied for a parent's object (high type) and one form for the children's object (under type).

GUI-Editor for forms

An GUI-Editor is opened by the double click on a form in main window. With it the appearance of the form and the linkages to table-columns in the database can be specified.


1) Objekt in Ablage Kopieren
2) Objekt Löschen
3) Vater Frame auswählen
4) den nächsten Objekt in Frame auswählen
8) Ausgewähltes Objekt wird mit rot gekennzeichnet
9) Ein schwarzer Rahmen zeigt ein Frame, benutzt für die Platzierung
1) Eigenschaftenfenster. Hier des ausgewählten Textwidget
5) Widget-Type Sammlung. Ein neusr Objekt wird durch das Zeihen des Objekt (Zeiger ändert sich auf eine Zielscheibe) kreirt
6) Ablage. Sie kann mehrere Objekte beinhalten. Bewegen auch durch Drag&Drop
7) Status-Zeile mit Informationen über gezeigten Objekt (Typ,Data-Link,Packer)
Student
Editieren
Werkzeuge
Widgets
Daten
PackContainer
InheritForm
FormLink
NestedForm
SqlReferenceList
ComboBox
CheckButton
RadioButtons
LabelControl
MatrNr
Name
Semester
Vertreter Formular-Link Label: hoeren
Vertreter Formular-Link Label: pruefen
Eigenschaften
widget
data
pack
state
label
Name
show
textwidth
20
Ok
Anwenden
Abbruch

FormLink >MatrNr< manager pack side=left anchor=center fill=none expand=0 pos=0

Figure 2-2. GUI Editor

The Properties of the elements are slitted to three types

Widget

The Properties, which belong to GUI-object such length, width (visual appearance).

Data

Linkage with tables-column. Default value and not-null

Packer

Information to placement of the element

The placement of the elements is not indicated definitely (with coordinates), however, under use by Geometry-manager. Most standards-Widget are displayed like they are (WYSIWYG). The other special Widget like (Nested Form, form link, SQL-Reference) are indicated with editor GUI by the representatives. Their size does not correspond to the reality (caution with the NestedForm). If one wants to see the real appearance of the forms, one must use **FormServer**, the DB-connection is not necessary for such purposes.

Drag and Drop in FormEditor

The GUI-Editor is slitted in 3 area.

Widget set
Workbench with your edited form
scratch depository

The 3 drag & drop operations are shown in Figure 2-3 . The Drag & Drop is signalized by special mouse-pointer-icon. The object will be not really moved. All catted objects are moved to depository. The depository can contain many objects

> **Tip:** By inserting new widget, choose one widget in widget-set (the background is displayed colored) window and drop it on workbench.

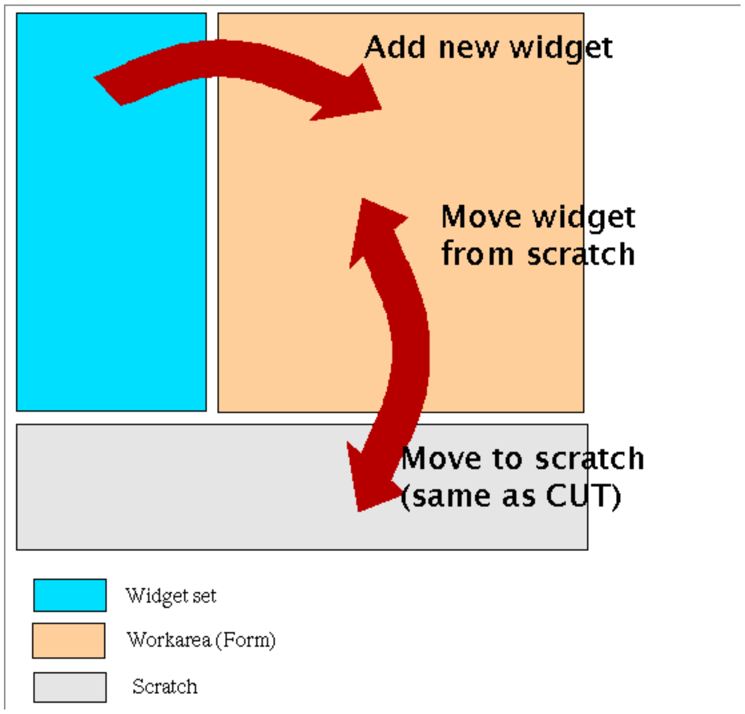


Figure 2-3. Drag and Drop operations

work with Pack-Frame

The elements are places in Frames (also nested). The order of the elements in a frame is crucial. The pack-geometry-manager takes one after another the elements and places them(her) in the frame the remaining place it is applied for the next elements. The following qualities specify the placement:

Side - in which page(side) of the frames the object should be added
Anchor - anchor to which page side
Fill
Expand - the object should be adapted by change of the window size

Tip: One can see the current pack options in the status bar, after pointing the Widgets with mouse's pointer.

The position of new drooped object is computed related to the object you have target. When no object are targeted, the widget will be added at last pack position with standard option (-side top -anchor center -fill none -expand 0). The exact position can be specify by target the object, which is already in the frame.

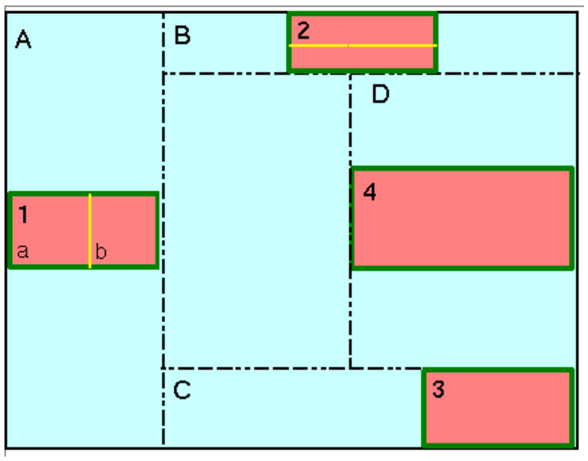


Figure 2-4. Positioning in Pack-Frame (schematic)

Figure 2-4 show an example of Pack-Geometry-Manager composition. The are a part of Tk pack manual.

For each master the packer maintains an ordered list of slaves called the packing list. The -in, -after, and -before configuration options are used to specify the master for each slave and the slave's position in the packing list. If none of these options is given for a slave then the slave is added to the end of the packing list for its parent. The packer arranges the slaves for a master by scanning the packing list in order. At the time it processes each slave, a rectangular area within the master is still unallocated. This area is called the cavity; for the first slave it is the entire area of the master. For each slave the packer carries out the following steps:

1. The packer allocates a rectangular parcel for the slave along the side of the cavity given by the slave's -side option. If the side is top or bottom then the width of the parcel is the width of the cavity and its height is the requested height of the slave plus the -ipady and -pady options. For the left or right side the height of the parcel is the height of the cavity and the width is the requested width of the slave plus the -ipadx and -padx options. The parcel may be enlarged further because of the -expand option (see "EXPANSION" below)
2. The packer chooses the dimensions of the slave. The width will normally be the slave's requested width plus twice its -ipadx option and the height will normally be the slave's requested height plus twice its -ipady option. However, if the -fill option is x or both then the width of the slave is expanded to fill the

width of the parcel, minus twice the -padx option. If the -fill option is y or both then the height of the slave is expanded to fill the width of the parcel, minus twice the -pady option.

3. The packer positions the slave over its parcel. If the slave is smaller than the parcel then the -anchor option determines where in the parcel the slave will be placed. If -padx or -pady is non-zero, then the given amount of external padding will always be left between the slave and the edges of the parcel.

Once a given slave has been packed, the area of its parcel is subtracted from the cavity, leaving a smaller rectangular cavity for the next slave. If a slave doesn't use all of its parcel, the unused space in the parcel will not be used by subsequent slaves. If the cavity should become too small to meet the needs of a slave then the slave will be given whatever space is left in the cavity. If the cavity shrinks to zero size, then all remaining slaves on the packing list will be unmapped from the screen until the master window becomes large enough to hold them again.

working with Grid-Frame

The row and column are computed dynamic.

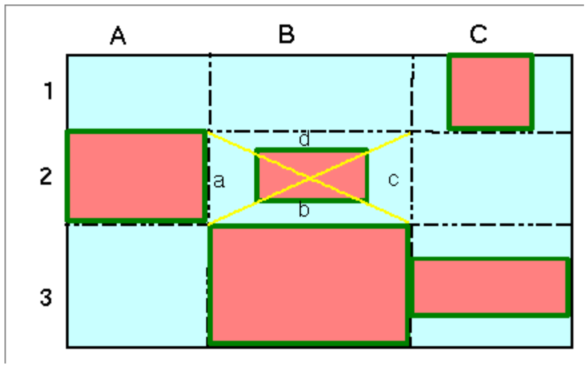


Figure 2-5. Grid-Frame

Figure 2-5 show one Grid-Frame. The cells A2,B2,B3,C1,C3 are occupied. By dropping in the free cell this cell will be used for placement. If you target one occupied cell one row or column will be added. See Example. If you target on cell B2, area d one row above the cell will be added.

The another grid option as sticky or rowspan can be edited by widget-propries dialog.

FormServer

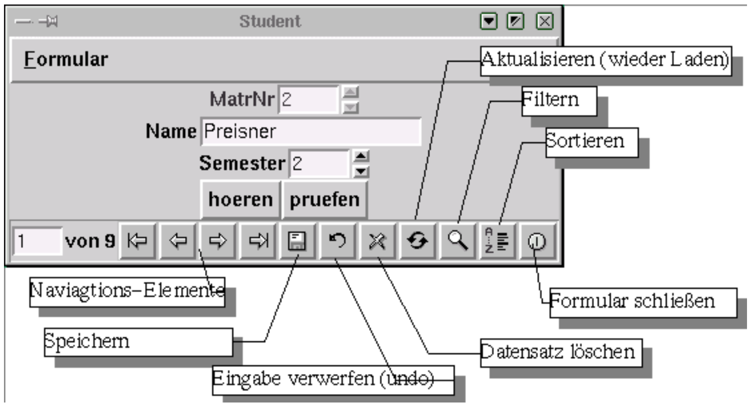
It is a program with which the real end user of the databases has to do. This program can evaluate the form descriptions which come from form-editor, display the forms and connect to database. This program must give a receipt for the manipulations on the forms correctly in in the instructions SQL to the databases. Following functions are supported:

- Data View
- Data modifying (update)
- Data create

- filter or search Data
- order, sort Data
- (new feature) data navigating (Browsing)

By the form-links a navigation is also possible by data contents along the linkage paths (relationship-path). The user can navigate like in hypertext-documents and reach every information. The user must not know the schema to work with Form-Server.

The forms themselves support directly higher abstraction concepts like association, Aggregation and specialization. The knowledge about the concepts of the relational modeling like foreign key and primary key must not exist by user.



Browsing Database with forms

With Form-Links you can brows the database like hypertext-documents.

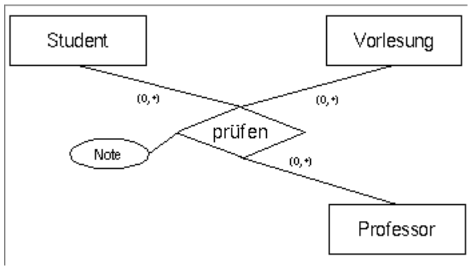


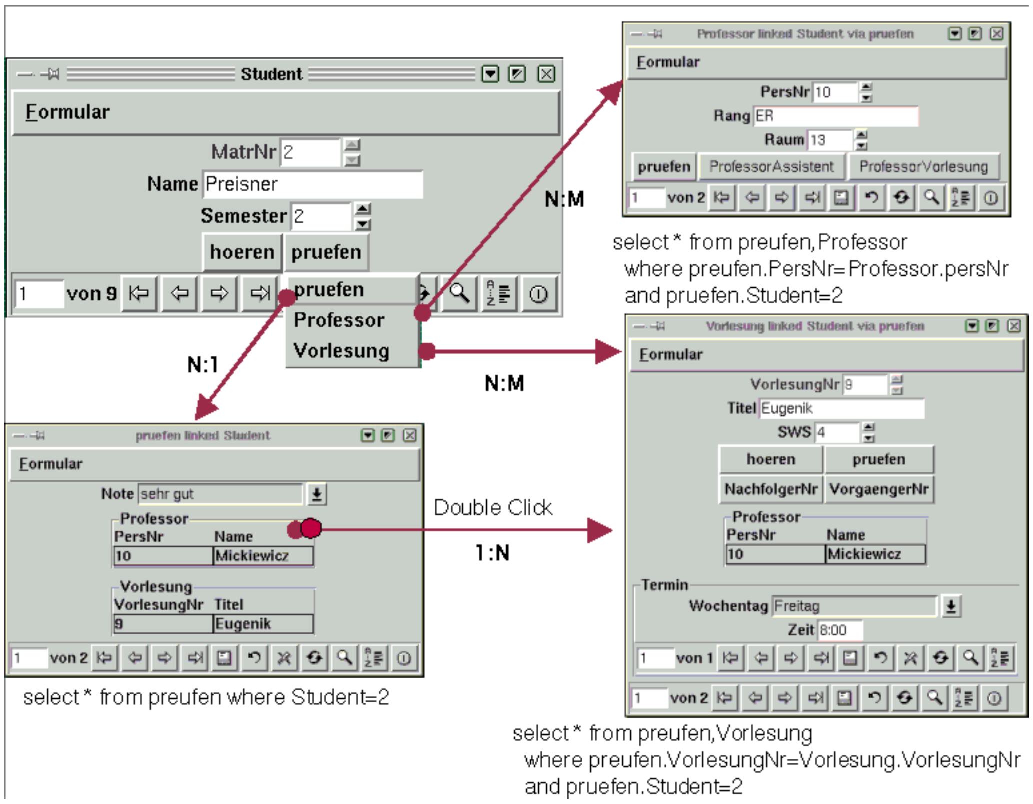
Figure 2-6. Association der Kardinalität 3 mit einem Attribut

Consider following association Figure 2-6. There are three objects (Student, Vorlesung (eng. lesson), Professor) in the relationship Prüfung (eng. exam). The result tables structure.

```
Professor     {PersNr,Rang,Raum}
Student       {MatrNr,Name,Semester}
Vorlesung     {VorlesungNr,Titel,SWS,PersNr}
pruefen       {MatrNr},VorlesungNr,PersNr,Note}
```

With form-links man can brows or navigate throw Database Information. From Student form you can reach per one click the information which exams he must do.

The Problem: Association has reacher semantic the hyper-links. The Association is transitive. Hyper-Links are only one-way references. The association can have more then 2 object (the degree of association can be greater than 2). One object can be associated with more than one another objects.



MatrNr 2
Name Preisner
Semester 2

PersNr 10
Rang ER
Raum 13

select * from preufen,Professor
where preufen.PersNr=Professor.persNr
and pruefen.Student=2

Note sehr gut

Professor	
PersNr	Name
10	Mickiewicz

Vorlesung	
VorlesungNr	Titel
9	Eugenik

select * from preufen where Student=2

VorlesungNr 9
Titel Eugenik
SWS 4

Professor	
PersNr	Name
10	Mickiewicz

Termin
Wochentag Freitag
Zeit 8:00

select * from preufen,Vorlesung
where preufen.VorlesungNr=Vorlesung.VorlesungNr
and pruefen.Student=2

Figure 2-7. navigate (browse) with forms-links

Figure 2-7 shows all form-links types, they correspond to Relationship-Types from Er-model.

1:N

The Association is displayed as choose-list. This is exactly the same as foreign-keys attributes. You can trigger the link by double-click on the reference-list title.

N:1

The Association is displayed as action-button. This is just another site of 1:N relationship.

N:N oder N:M:O:...

The Association is displayed as menu-button. This is just another site of 1:N relationship. The first menu-item is an N:1 Relationship and it link to the relationship-table. The next item links to objects.

After open the linking form, the linking are conserved. This mean, the form are chained.

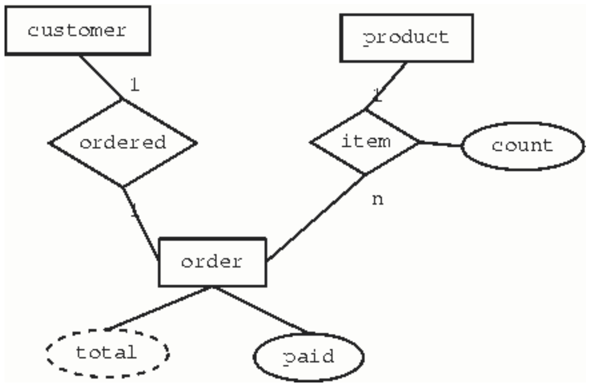
user macros in form

There are simple API to extend the form-server with user-procedures (macros). The FormServer is implemented in Tcl and it allow very ease to dynamic change of program-code.

The macros are loaded as Tcl-scripts and they become a part of FormServer. The are some simply designed API to control some main function of program. You can change the behavior of some form like (delete checking, update checking ...). The idea; you create new XOTcl class derived from FormEngine class and you overwrite some empty methods. This are empty macros for myform form.

```
Class myform -superclass FormEngine
myform instproc update_check klvalues_ref {
    return 1
}
myform instproc delete_check {} {
    return 1
}
myform instproc insert_check klvalues_ref {
    return 1
}
myform instproc after_delete {} {
}
myform instproc reload_form {} {
}
myform instproc position_check {pos} {
    return 1
}
FormEngine instproc filling_form klvalues_ref {
}
```

See the example *accountancy* how to write your macros. The are such er-schema.



The macros are used to build following functionality.

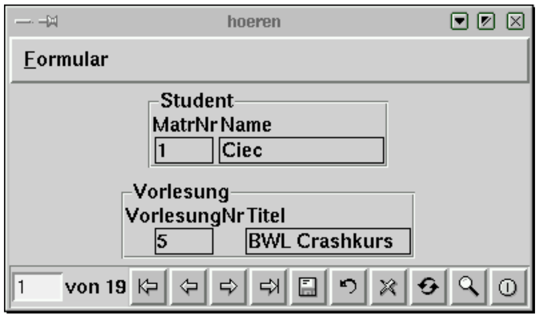
- A Customer (Klient) can be not deleted if he has order he has not paid. By deleting customer set all foreign-keys by cust_order table to NULL
- The macro compute a derivated attribute total price for oder and sum price for oder item
- Check if the date format is 2000-01-01

see the file `sample/accountancy.tcl` how to buid such macros (for example how to buil sql-queries in Xdobry).

Schema vs.Form appearances

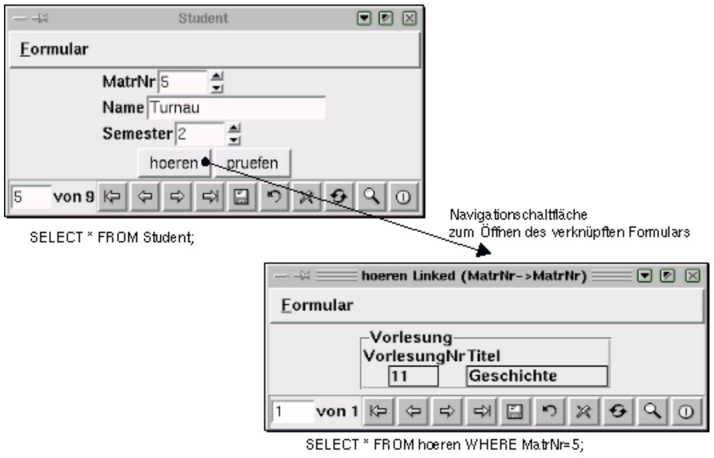
The system was developed after the motto: the appearance of the forms should correspond to data semantics. Here the examples how the drafts of the modelling have their(her) influence on the structure of the forms. The Screen shoots comes all from the Tutorial example.

Representation of the foreign keys



The foreign keys attributes are represented as multi-column-listboxes. Here one can see on the first place the value of the primary key and an additional attribute (name) which has the role of the object-labels.

Association (form-link)



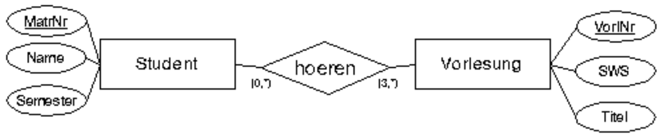
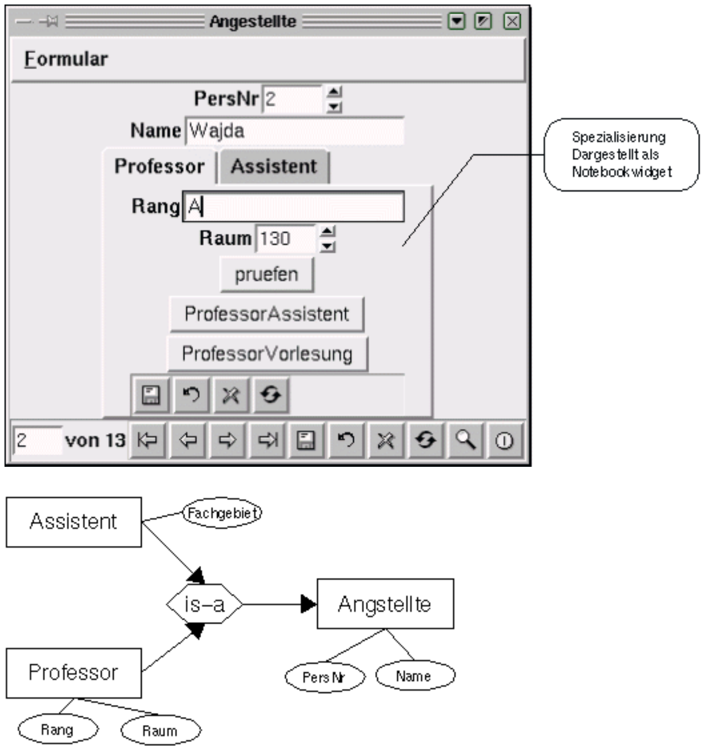


Figure 2-8. Association in ER-model

The form-links function possibly like instruction: Open the form with all objects which are tied together with this object (a foreign key on it have). They are opposed to foreign key references

Specialization (inheritance)

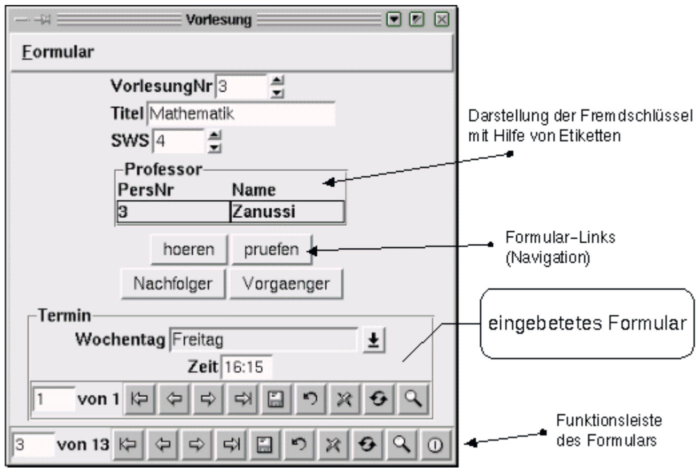


PersNr 2
Name Wajda
Rang A
Raum 130
2 von 13

Figure 2-9. Generalization in ER-model

The specialization forms (children's forms' are embedded in a notebook Widget of the parents-Form.

Aggregation



VorlesungNr 3
Titel Mathematik
SWS 4

Professor

PersNr	Name
3	Zanussi

Termin
Wochentag Freitag
Zeit 16:15
1 von 1
3 von 13

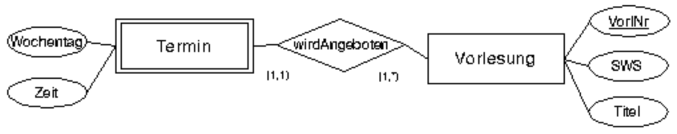


Figure 2-10. Aggregation in ER-model

Aggregation functions as embedded forms, besides, foreign keys are used as a linkage paths; unnoticed from the used.

Missing functionality be Xdobry

I also want to name the possible tarps and pitfalls with the use of the the program.

- With the Aggregation, Subtypes and foreign-key only one-attribute-primary-key per object are supported. An object is identified only by one attribute.
- With empty tables the insertion of the first Tuples is possible only with the command buttons save (diskettes-icon).
- The data in a form are sent only after the change the Tipple-position (with navigation buttons) or by the command button save.
- There are problems with operations (Delete, Change, Create), they become only a visible after the action reload. The condition that the data will not could be current by the red-colored command button reload indicated.
- The number of open SQL-connections is limited system-specifically.
- Only once the Widget (Multi-Column-Listbox) for the announcement of foreign key are initialized while opening the form. If the condition of the objects should change at which the foreign keys point, it is not indicated.
- The forms which can be used with Aggregation separately are opened, however, the foreign keys are indicated as a normal Widget. This can manually in**FormEditor** Are changed, however, it is not shown by form assistants automatic.
- Form-links functioning if one does not point at the root-form, the foreign key, however, in a children's form is. See Tutorial (form Professor link on assistant). Such FormulaLinks need a special treatment which was not programmed yet.

If you have similar problem or if you have consider to other functional ways, let me about it know. How the next version will look, depends on you.

Program Options

You can specify some options by starting FormServer.

FormServer -- [-help] [-debug] [-viewonly] [-ignorexmlpar | -noconnectdialog | -connect `connectpar` | -loadmacros `macrofile` | -openform `formname` | `xmlfile`]

-debug

Activate some menus for debugging: SQL-Monitor, XOtcl class-browser, Tk-Window Browser, interactive mods

-viewonly

The database contens can be only viewed. No delete, modify or insert operation are allowed.

-ignorexmlpar

Ignore all parameters codded in XML file.

-noconnectdialog

Do not show database connect dialog. All parameters for database connection should be specified in XML file or by -connect option.

-loadmacros

load a Tcl-script after initializations.

-connect

Specify database connect parameter. The parameters are codded as TclX keyed list.

```
-connect "{interface mysql} {user root} {dbank test} {password geheim} {socket /var
```

-openform

Open a form after initialising.

Warning

All option must be tipped after --.

Example usage.

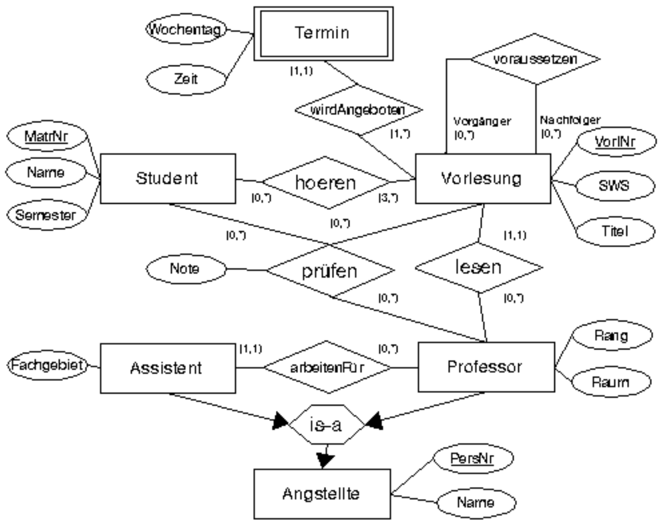
```
./FormServer -- -debug -viewonly
 -connect "{interface mysql} {dbank uni}
          {user root} {socket /var/lib/mysql/mysql.sock}"
 -noconnectdialog -openform Student uni.xmlforms
```

Chapter 3. Tutorial

Here it a example of using the system. A school-book-databases was chosen, so that all qualities of the system can be shown. The step consequence corresponds possibly to the production by real DB-applications. First a ER-diagram of the system is constructed and is illustrated in relational schema. Then this schema is converted with the help of reverse engineering colleges of technology into a XML-Repository.

Relational schema

Accepted a databases-schema with the help of ER-model design. This schema gets 5 Entities. Besides, is specialized office workers to assistant and professor. With appointment it is about a Weak-Entity. It is here the modeling the Aggregation (better composition) from sets-attribute of the object. There is also different take from relations. Examine a connection(affair) the degree 3 with own attribute is a mark(note).



This ER-diagram is converted to a relational schema.

```
Angestellte   {PersNr,Name}
Professor     {PersNr,Rang,Raum}
Assistent     {PersNr,Rang,Raum,Professor}
Student       {MatrNr,Name,Semester}
Vorlesung     {VorlesungNr,Titel,SWS,PersNr}
hoeren        {MatrNr,VorlesungNr}
voraussetzten {VorgaengerNr,NachfolgerNr}
pruefen       {MatrNr},VorlesungNr,PersNr,Note}
Termin        {VorlesungNr,Wochentag,Zeit}
```

FileUnibank.sql In the directory sample of the program contains the SQL-instructions for MySql databases for the creation of the databases. To put on the databases one must first University databases create.

Warning

The MySql databases have to be installed on your system. It runs and you have to put on corresponding rights the databases and tables.

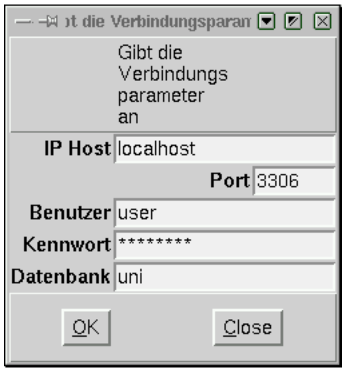
```
[User@localhost] mysql mysql
>CREATE DATABASE university
```

Attach from tables

```
>cat sample/unibank.sql | mysql-u username university
```

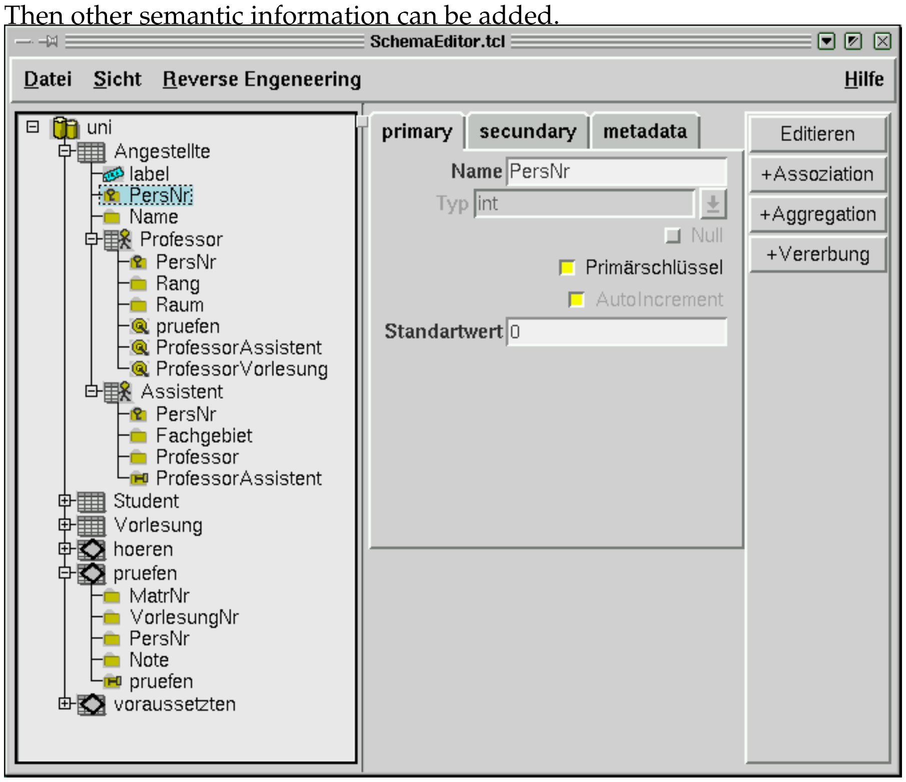
Production of the Repositories

For the production of the Repositories which represents an extension of the relational schema around semantic information, becomes the program**SchemaEditor** Applies. The program can select the relational schema by the SQL-connection. In addition one applies the menu point**File**⟶**Case schema from DB** . In the following dialog the correct data accessibility must be given(indicated).



IP Host localhost
Port 3306
Benutzer user
Kennwort ********
Datenbank uni

Then other semantic information can be added.



Name PersNr
Typ int
Null
Primärschlüssel
AutoIncrement
Standartwert 0

This can happen in three sorts

1. Use of reverse engineering menu**Reverse engineering** .
2. Add from abstraction concepts by the use of assistants: +Association (possibly foreign key), +Aggregation, +Suptype
3. Edit properties of the attributes (schema-elements).

4. Add from other schema-elements such as attribute-group, label. It happens by use of pop-Up Menu (right mouse button)

The complete example of XML-Repository show the file `sample/uni.xmldbschema`.

Production of the forms

The creation of the forms happens with the program**FormEditor** . First becomes the DB-schema `sample/uni.xmldbschema` Loaded **File**⟶**DB of schema shop** . Automatic creation of DB-forms happens with **Tools**⟶**Form assistant** . The created forms can designed with a GUI-editor by command button**Form Edit** . The finished forms are in `sample/uni.xmlforms`.

Use of forms

The finished forms are stored as XML-files. The program **FormServer** Can represent the forms correctly and communicate with the databases. Call with argument from program catalog.

```
./FormServer sample/uni.xmlforms
```

Other examples

In the directory `sample` are two other examples too.

1. An example from publication: Roger H.L.Chiang; Terence M.Barron; Veda C.Storey. Reverse engineering of relational databases: Extraction of in EER model from a relational database. Data and Knowledge engineering 12. Elsevier. in 1994. Pages 107-142. Under the name*Bussines* . It is a databases schema of a hypothetical firm. There are the following files.

`Business-ER.dia`	ER-diagram of the databases in format of a GNOME of program DIA
`Bussines-er.eps`	ER-diagram as a PostScript file
`Bussines.sql`	create tables Sql-instructions for mySQL database
`Bussines.xmldbschema`	Repository of the schema
`Bussines.xmlforms`	Forms

2. An example of a course-databases. It is offered different course-types. The realization of a course (entity Kurs) is named offer (with date) (entity Angebot). The courses are structured hierarchically. Untertypes A1 A2 A3 belong also to course type A, etc. . To every course a predecessor (can be named vice versa(conversely) successor). Z. B the participation in the course B sets in front the earlier participation in the course to A (B is deepening A).

`Course-er.dia`	ER-diagram of the databases in format of a GNOME of program DIA
`Course-er.eps`	ER-diagram as a PostScript file
`Kurs.sql`	create tables Sql-instructions for mySQL database
`Kurs.xmldbschema`	Repository of the schema
`Kurs.xmlforms`	Forms

Chapter 4. Developer information

Programming tools

The whole system was programmed in the computer language TCL (tool Command Language). Tcl is interpretative and string based computer language and characterizes itself by a simple syntax and a simple extensibility. These qualities it ideally for glue together of other specialized components (mostly libraries which are written in other languages (C, C ++)). The language are often used for *Rapidly Developments Prototyping* Tcl uses a brilliant, simple syntax. Although Tcl is a script language, the Tcl-applications are much faster than Java-programs and use very little resources. The object-oriented implementing was possible by using XOTcl extension. This makes possible a clear structure of the program. The following additional components of the TCL were used.

Tk and Tix

a library for developing GUIs (Widget set).

XOTcl

object-oriented Tcl extension.

mysqltcl pgtcl ...

interfaces to specific database systems.

tDom

DOM implementation for Tcl.

I also had no ambition to program everything in pure Tcl/Tk if there are so many good libraries.

XML

In the system XML-documents are interfaces between **SchemaEditor** and **FormEditor** (class XML dbschema) and farther between **FormEditor** and **FormServer** (class XML xmlforms). The high independence of the components and their interchangeability is thereby guaranteed. Every component can be replaced with another which can manipulate the corresponding XML-document (can interpret). It means, they must be enough for the interface which is described as a XML-DTD (DTD-document). For the user of the system the internal use of the XML-format is invisible. He can create according to demand the XML-documents also manually or process with the help of other general XML-tools.

The descriptions of the separate XML-classes as documents DTD become in files `dtd/dbschema.dtd` And `dtd/dbforms.dtd`. A small script `validate.sh` If simplifies the use from nsgml Parser which belongs already normally to every LINUX distribution.

The system was developed so that these XML-documents can be used by other applications.

TODO List

1. Interface to other databases.
2. Special Widget for time types, scores, BLOB

3. Support of the BLOB-types which are specified by MIME-types. Possibly accessibility in KDE or gnome System. Announcements of picture data. Exchange with file system
4. Form-server supports cascading delete and modifying of primary-keys
5. Form-server supports relationship-(min.max) and other integrity-rule
6. Direct help to dialogs
7. Balloon or tip-texts to Widgets
8. Alternative representation of forms as tables or table as parts of the forms. Use of tkTable Tcl-extension
9. Support of tabulator consequences and generally better user friendliness
10. Short-cuts (Accelerators)
11. Null-value treatment with Widget. Till present all Widget can not support it well.
12. New more powerful GUI-elements (Widget) and more Options.

I need your feedback.

Ideas for the future

First task is to fix all bugs and improve user usability. The other is to develop really new ideas. Xdobry could serve as a basis for the development of different DB-applications. With XOTcl development too Xdobry can be customized and extended for own needs.

As the smallest denominator for relational databases standard SQL-92 and mysql was chosen. Many of the problems were solved in SQL99 standard. It remains for the moment to wait to the available RDBMS support this standard. Interesting the implementing of object-relational qualities of the popular databases Postgres could be. Many of those cover themselves with standard SQL3.

1. Array attributes
2. Tuple can be identified by OID (Objectidentifier)
3. Versioning
4. databases-rule

2 interesting projects also exist to the description of the GUI interface as XML-documents. One comes of it as a by-product of the development of Open Source Netscape (Mozilla). It is XPToolkit (http://www.mozilla.org/xpfe), the user's interface can independent of platform as XML-documents (user's interface XUL XMl Language). Another project comes with the editor for the development of GUIs GLADEhttp://glade.pn.org[2]. It is a part of the GNOME project, free desktop Enivronment for LINUX. The idea with both projects is that the appearance of the GUI is not party-coded in the program, but is always present as a XML-document. This could allow a better and flexible configuring of the interface by the used without must to recompile the program. Both projects are all the more valuable that the complete source are free and can be applied the available components also with foreign projects. The development of **FormEditor** and **FormServer** could profit from it.

There also exist other interesting colleges of technology which are connected with XML-standard. By the Extensible Stylesheet Language (XSL) the transformation can be automated by XML-documents in other XML-classes or other formats. So one could realize transformation of the Repository in specific SQL-instructions (the type CREATE TABLE) easily.

The main disadvantage the DTD (Document Type Defintion) is no context-systex can be covered. With DTD can not be guaranteed a well-shaped Repository also a valid relational schema represents. The newer developments like XML-schema allow a more exact definition of the document-types and overcoming such disadvantages.

Another direction for the advancement of the system would be the support of other abstractions and semantics in **SchemaEditor** and afterwords with the form-server. Two cases are conceivable here: 1. the system an interface offers for the addition functions the DBMS like versioning, new data types, DB-rule (Trigger) 2. The system implements even new functional characters like versioning, Authentiefizierung, integrity rule, cascading delete. For the conceptual model the support of other abstractions would be possible: derived attributes, derived relations, dependent relations (e. g. , an assistant can only in the project participant if he the special qualifications has). , representation and manipulation of the tree structures which are filed as tables.

Supported Databases

- mysql
- postgres
- sqlite
- MS SQL-Jet (MS Access Format) (durch ODBC)
- MS SQL Server (durch ODBC)
- general ODBC
- Oracle

Xdobry use XOSQL-Adapter.

WWW-Links for developers

1. TCL-Tk Home[3] entry link for Tcl
2. XOTcl[4] Object oriented Tcl
3. Tix[5] Tix Widget library
4. Tclodbc[6] ODBC interface
5. Mysqltcl[7] mysql interface
6. tDom[8] DOM implementation
7. Mysql[9] free realtional database
8. Postgresql[10] another free RDBMS

Notes

1. http://www.mozilla.org/xpfe
2. http://glade.pn.org/
3. http://www.tcl.tk
4. http://www.xotcl.org

5. http://www.sourceforge/projects/tix
6. http://www.sourceforge/projects/tclodbc
7. http://www.xdobry.de/mysqltcl
8. http://www.tdom.org
9. http://www.mysql.com
10. http://www.postgresql.org

Chapter 5. Productive use

License

Of this software is defeated by the license GPL. The exact wording of the license GPL in English is in the main catalog in `LICENSE`. Very briefly it is a free software.

On this software no guarantee is given. It is after German law a present (gift).

Meaningful using of application

This software is still in development. However, this does not mean that it is not usable. Xdobry lacks some important functionality for manage sensible data (transactions, locking, complex validation). It can be good used for ad-hoc application and prototyping

Advantages to opposite existing systems

There are many similar systems which excel this with the functional character largely, nevertheless it is to be enumerated worth a few peculiarities of the software which distinguish the software.

1. It is very small.
2. Existing components with computer language Tcl were already stuck together.
3. Source text is accessible
4. Development with Tcl and XOTcl is 10 times faster than with JAVA (according to SUN)
5. It can easily be extended around special features
6. It uses consistently XML-format

However, the main advantage is a new way of developing the DB-forms. The tables themselves do not place the basis for the forms but special the conceptual model. The conceptual model is won by reverse engineering from databases schema (Data Dictionary). The forms have so better quality and do not need of agile adaptations as with conventional systems. The foreign keys are represented as listsboxes. The inheritance is supported directly. Embedded forms are possible. The developing and use of forms was separated (It is always almost done by different people). Thereby it use very few resources. More of it in the underlying dissertation.

Contact with author

Homepage: http://www.xdobry.de[1] Email: `<mail@xdobry.de>`[2]

Notes

1. http://www.xdobry.de/
2. mailto:mail@xdobry.de

Chapter 6. Support

The program is free and it is a gift. If you problem with it inform me. I promise no guarantee 100per cent of the answer, however, on every bugs report I will be glad. This allows to improve the program. First they visit the homepage of the program, perhaps the mistake is known ready and is repaired in new version. Formulate the mistake description in this way so I can reproduce the bug. (They Copy the mistake message) you can also send their(her) databases structure (as a SQL-Dump) if the data are not confidentially. A ER-diagram or specification of the data would be also meaningful. Support only by email.

Chapter 7. Installation from Source code

Starkit packaged allows so called "copy-installation". Just copy one file no another requirements.

Xdobry System consists only Tcl-scripts and therefore no compile are necessary. However, it uses many Tcl-extension (libraries) which were programmed in C or C ++. There are: Tk, Tix, which belong to Linux-distribution and should be already installed on your computer. Windows users can use free Tcl distribution from Active-State Tcl

Other Tcl-extensions must be installed separately:

1. XOTcl (object oriented extension for Tcl) successor of OTcl and competitor of ITcl.
2. TDom 0.5a2 - is a DOM (Document Object model) interface for treatment of XML-documents.
3. Mysqltcl - is a Tcl interface to MySql databases.
4. Pgtcl - is a Tcl interface to Postgersqll databases. Belongs also to the newest RedHat distribution (greater 6.2).

The installation of above extensions can be problematically, because it demands some experience with compile programs.

Xdobry analysis by starting with Tcl interfaces are available. For checking this Tcl command **package require** will be used. following packages will be search for:

- mysqltcl for mysql
- sqlite for Sqlite
- Pgtcl for Postgres
- ORATcl for Oracle
- tclodbc for ODBC
- tclodbc for MS Access (Windows only)
- tclodbc for MS SQL-Server (Windows only)