

Dokumentation des Programms Xdobry

Artur Trzewik

Dokumentation des Programms Xdobry

by Artur Trzewik

Xdobry ist ein System für die automatische Erstellung von DB-Formular Anwendungen zu relationalen Datenbanken. Die Formulare unterstützen direkt folgende Abstraktionen: Assoziation (Formular-Links) , Aggregation (eingebettete Formulare) und Spezialisierung (Notebook Formular Elemente). Das Programm entstand ursprünglich als praktischer Teil einer Diplomarbeit¹ und beinhaltet viele neue Ansätze der Entwicklung der DB-Applikationen. Wurde aber im Laufe der Zeit überarbeitet und weiterentwickelt. Xdobry Homepage www.xdobry.de²

Table of Contents

1. Xdobry Spezifikation	1
Hauptziel des Programms	1
Besondere Merkmale	1
Eigenschaften des Programms	1
Zielgruppe	2
Technische Daten	2
Version Neuheiten	3
2. Benutzer Handbuch.....	5
Systemkomponenten	5
SchemaEditor	5
Reverse Engineering.....	7
Definieren von Abstraktionen	8
DB-Schema Anlegen und Verändern	9
DB-Schema und Daten migrieren	12
DB-Schema und Daten Exportieren und Importieren.....	12
FormEditor	13
GUI Editor für Formulare.....	14
FormServer	19
Navigieren mit Formularen	20
Eigene Makros in Formularen	22
Programm-Parameter.....	23
Schema vs. Formularaussehen	24
Darstellung der Fremdschlüssel.....	24
Assoziation (Formular-Links).....	24
Spezialisierung (Vererbung).....	25
Aggregation	26
Fehlende Funktionalitäten bei Xdobry.....	26
3. Tutorial	29
Relationales Schema.....	29
Erstellung des Repositories.....	30
Erstellung der Formulare	31
Benutzung der Formularen.....	31
Andere Beispiele.....	31
4. Entwickler Informationen.....	33
Programmierwerkzeuge.....	33
XML.....	33
Weiterentwicklung	33
Ideen für die Zukunft	34
Datenbankanbindung.....	35
WWW-Verweise für Entwickler	35
5. Produktiver Einsatz von Xdobry.....	37
Lizenz	37
Einsatz von Xdobry.....	37
Vorteile gegenüber bestehenden Systemen	37
Kontakt mit Autor	38
6. Unterstützung	39
7. Xdobry aus Quelldateien installieren.....	41

Kapitel 1. Xdobry Spezifikation

Dieses Kapitel beschreibt den Verwendungszweck des Programms und seine Haupteigenschaften. Es werden die Fragen beantwortet wem und wozu es nützlich sein könnte. Die Zielgruppe und ihre Anforderungen werden genannt.

Hauptziel des Programms

Dieses Programm ermöglicht schnelle und einfache Erstellung von DB-Formular-Anwendungen. DB-Formulare sind die graphischen Schnittstellen einer Datenbank, die die komfortable Bearbeitung von Dateninhalten erlauben. (Eingabemasken). Mit Xdobry ist es möglich ein existierendes Schema einer Datenbank: Auszulesen, Modifizieren, Anlegen und Daten zwischen verschiedenen Datenbanken-Systemen zu migrieren. Die Erstellung der Formulare verläuft automatisch oder halb-automatisch durch Hilfe der Assistenten. Es werden keine Programmierkenntnisse benötigt. Die erstellte Formulare sind nicht einfach die Abbildungen der Tabellen in relationalen Datenbank. Sie entsprechen vielmehr einem höheren Abstraktions-Niveau bekannt aus konzeptionellen Modellen.

Besondere Merkmale

- Plattformunabhängig: binäre Pakete für Windows und Linux (Intel)
- Unterstützt die alle Hauptdatenbanken: Oracle, MS SQL-Server, mysql, postgres, MS Access, allgemein ODBC
- Erlaubt einfache Migration des Schemas und Daten zwischen unterschiedlichen Datenbanksystemen.
- Kleine Laufzeitumgebung, die keine besondere Installation oder besondere Voraussetzungen benötigt.
- Spezielle Assistenten, die Migration, Schemamanipulation und Erstellung der Formulare erleichtern
- Reiches Umfang an datenbankspezifischen Oberflächenelementen

Eigenschaften des Programms

Das Programm wurde entworfen als ein offenes System für die Erstellung von formularbasierten DB-Applikationen. Im Gegensatz zu vorhandenen Systemen wurde die Erstellung der Formulare als mehrstufiger Prozess aufgefasst. Der Hauptgrund für die mindere Qualität der automatisch erstellten Formulare, bei herkömmlichen System, ist die Benutzung des relationalen Schema als Informationsquelle für Formulargenerator. Solche Formulare müssen dann aufwendig manuell angepasst werden. Relationales Schema (Tabellenstruktur) beinhaltet nur wenige Informationen über die Semantik der Daten. Es sind einfach zu wenige Informationen um gute Formulare zu erstellen. Bei Xdobry verläuft die Erstellung der Formulare in drei Stufen

1. Genaue Spezifizierung der DB-Schema auf der Stufe des konzeptionellen Modell, Benutzung von Reverse Engineering Techniken. Benutzer DB-Administrator
2. Automatische Erstellung der Formulare. Anpassung des Aussehen im GUI-Editor. Benutzer DB-App. Entwickler
3. Benutzung der DB-Formulare für die Bearbeitung der Daten. Endbenutzer

Auf diese Weise kann man leicht und schnell qualitativ sehr gute Formulare erstellen. Formulare unterstützen direkt höhere Konzepte der Datenbankmodellierung ohne der speziellen Programmierung.

- Die Assoziationen (Relationship oder Beziehungen) zwischen den Objekten werden für die Navigationshilfen benutzt. Auf diese Weise kann man durch Formulare etwa wie im Hypertextdokumenten navigieren (Hyperformulare). Die Assoziationen können so leicht erforscht werden, ohne der Kenntnis der Datenbankschema
- Aggregation und Spezialisierung werden direkt als eingebettete (geschachtelte) Formulare realisiert. Die Formulare können dadurch als Objekte und nicht Tabellen angesehen werden.
- Die Konzepte der Datenbank-Modellierung wie Fremdschlüssel müssen bei dem Endbenutzer nicht vorhanden sein. Das System übernimmt die benutzerfreundliche Anzeige und Handhabung von Fremdschlüssel.

Für jedes Schritt ist ein getrenntes Programm zuständig, das für andere Zielgruppe entwickelt wurde:

1. SchemaEditor
2. FormEditor, Formgenerator und GUI-Drag and Drop Editor
3. FormServer

Xdobry wurde als ein Teil einer Diplomarbeit¹ entwickelt. Es ist ein Programm, das theoretisch in der Diplomarbeit entworfen wurde, aber nachträglich modifiziert und erweitert wurde.

Zielgruppe

Xdobry ist ein hochspezialisiertes Programm für die Entwicklung von Datenbank Applikationen. Entsprechend hoch sind die Anforderungen an die Benutzer. Ich habe mir nicht vorgenommen ein System zu entwickeln, das komplexe Systeme, wie Datenbanken, einfach macht. Es soll nur die Bedienung von solchen Systemen einfach machen. Es ist ein Werkzeug für DB-Spezialisten, womit die Entwicklung von DB-Formularen effektiv und schnell sein kann. Entsprechend wird es nicht auf die Fachsprache verzichtet.

Das System Xdobry wurde so entworfen, dass seine Komponenten von 3 unterschiedlichen Benutzergruppen ausgeführt werden.

1. DB-Administrator: Kenntnisse: Relationale Datenbanken, DB-Entwurf, relationales Schema. konzeptionelles Schema
2. DB-Applikation Entwickler: Kenntnisse: GUI-Entwurf. Besonders die Erstellung der Oberflächen unter Verwendung von Geometry-Manager (grid und pack)
3. Endbenutzer: grundlegende Computerkenntnisse

Um das Programm einzusetzen, sollten folgende Begriffe geläufig sein: relationales Datenbank, SQL, mySql, konzeptionelles Modell, Entity, Relationship, Aggregation, Schema, DB-Formular, Datenbankmodell, Abstraktion, Spezialisierung (Vererbung), Datenbank-Schema.

Technische Daten

Das Programm wurde unter Linux mit Programmiersprache Tcl (und ihre Erweiterungen: Tk,Tix,tDom,XOTcl) entwickelt. Wegen der vielen Komponenten kann die Kompilation und Konfiguration schwierig sein. Als haupt Testdatenbank wurde MySQL relationale Datenbank benutzt. Es existiert auch eine Schnittstelle zu anderen Datenbanken, die zahlreichen herstellereigenen SQL Erweiterungen werden jedoch oft nicht unterstützt. Das System orientiert sich an SQL-92 Standard.

Version Neuheiten

Version 0.12

- Grid Packer (Tabellen Geometry Manager) wird unterstützt
- Formular-Link wurden erweitert, so dass sie für die Navigation von kompletten Datenbanken benutzt werden können. Man kann auch über die n:m Beziehung Navigieren
- Es können in FormServer leicht eigenen Zusatzfunktionen eingebaut werden. Etwa eine API für FormServer. Es können leicht eigenen Prozeduren in Tcl für spezielle Datenverarbeitung, Datendarstellung oder Integritätsüberprüfung hinzugefügt werden.
- Der Eigenschaften-Dialog im FormEditor wurde effizienter programmiert.
- Die Aktionen an Formularen können eingeschränkt werden (kein Löschen, kein Modifizieren, kein Hinzufügen).
- Sortieren der Tupel in Formularen ist möglich.

Version 0.14

- Diverse Bugs wurden korrigiert
- Besseres Error-Handling (Die Sql Fehler werden abgefangen)
- Umstellung auf mysqltcl2.0 (nicht voll kompatibel zu mysqltcl1.53 aber deutlich schneller) und xotcl8.3
- Die Datenbank-Schnittstellen werden getrennt in Modulen nach Bedarf geladen.

Version 0.15

- Getestet mit RedHat 7.0, mysql 3.23 und postgresql 7.0
- Die Veränderungen betreffen eigentlich nur den FormServer
 - Nur Einfügen Modus wird unterstützt
 - viele Programmooptionen
 - Überprüfung der Notnull Bedingung
 - Navigieren auch durch Tasten-Kombinationen
 - Bessere Benutzerfreundlichkeit (Deaktivieren von leeren Funktionen)
- Unterstützung von dbitotcl Schnittstelle. Es können theoretisch alle Datenbanken benutzt werden, die eine Perl DBI Schnittstelle haben.

Version 0.16

- Programm Quelltext Refactoring.
- Die Formular Filter können benannt und gespeichert werden..
- Formulare unterstützen die s.g. Schablonen. Die Schablonen erlauben schnelle Ausfüllen (Voreinstellen oder Verändern) von Datenfelder.
- Datenfelder können automatisch überprüft werden (Validation)
- Zu jedem Datenfeld kann eine Beschreibung angegeben werden, die bei Drücken der F1-Taste erscheint.
- Ein paar Fehler wurden behoben.

Version 0.20

- Die Module FormEngine und FormFactory wurden massiv bereinigt (refactoring).
- Die Filter im FormServer werden im eigenen Fenster spezifiziert. Die Eingabe von spezielle Filters (z.B. >2) ist möglich.
- Xdobry wird mit xotclIDE entwickelt. Eine von mir entwickelte IDE für XOTcl Programmiersprache. Die interne Struktur hat sich sehr verändert. Die Module werden per package require geladen.

Version 0.30

- Unterstützung für sqllite Datenbank
- Auslieferung als Starkit für Windows und Linux. Keine weitere Voraussetzungen oder besondere Installation.
- Kontextabhängige Aktivierung der Menüs

Version 0.40

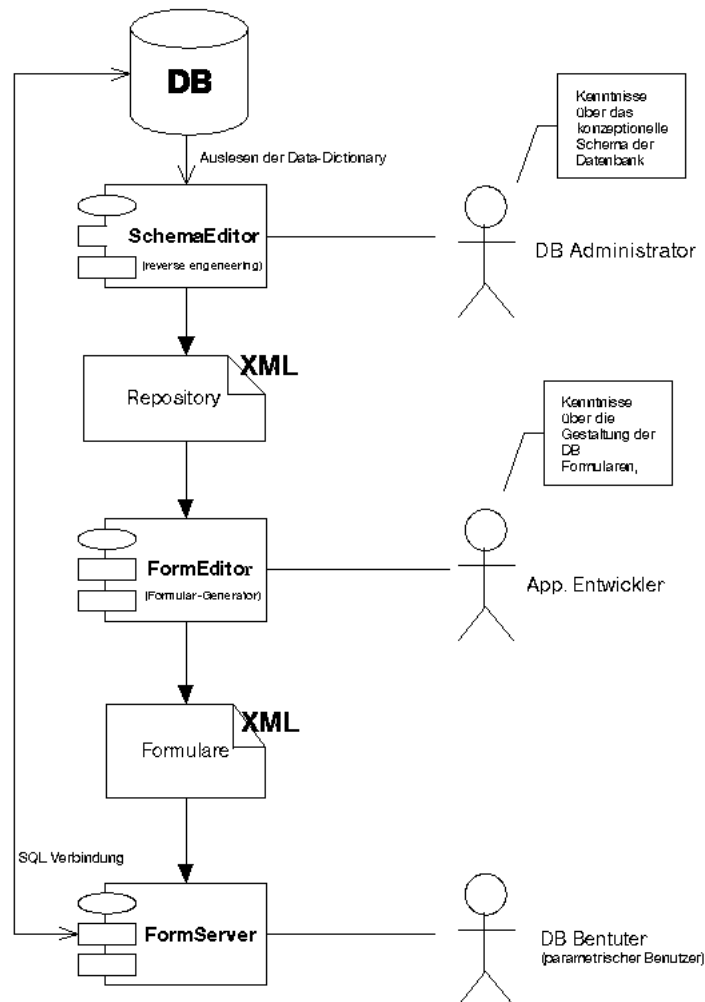
- Unterstützung für MS SQL-Server, MS Access und Oracle
- Mit SchemaEditor kann das DB-Schema anlegen und modifizieren
- Xdobry kann die Daten der verschiedenen Datenbanken migrieren
- Für Internationalisierung wird der standard Tcl Modul msgcat benutzt
- Überarbeitung der Dokumentation
- Zahlreiche Fehlerbehebungen
- Tclx wird nicht mehr eingesetzt

Fußnoten

1. <http://www.xdobry.de/artur/diplomarbeit.html>

Kapitel 2. Benutzer Handbuch

Systemkomponenten



Das System besteht aus 3 ausführbaren Programmen **SchemaEditor**, **FormEditor** und **FormServer**. Als Starkit-Distribution wird zwar ein Programm gestartet aber zum Start muss entschieden werden welcher Komponenten ausgeführt werden soll. Die Schnittstellen zwischen den einzelnen Programme sind 2 XML-Dokumenten Typen; das Repository (Standard Datei-Erweiterung *.xmldbchema*) und Formular-Beschreibung (Standard Datei-Erweiterung *.xmlforms*). Die Bearbeitung von beiden XML-Dokumenten wird von entsprechenden Programmen übernommen, kann allerdings auch manuell in einem Editor erfolgen. Die passenden DTDs (Document Type Description) finden Sie in Verzeichnis *dtd*.

SchemaEditor

Die Hauptaufgabe des Schema-Editor ist eine XML-Repository zu schaffen und zu verwalten. Weiterhin kann das Schema-Editor benutzt werden um Daten zwischen verschiedenen Datenbanken zu migrieren und Datenabzüge als XML oder SQL zu erstellen. Es wird ein Benutzer des Schema-Editor als ein Datenbank-Administrator oder auch ein Daten-Verwalter vorausgesetzt. Der Benutzer hat Kenntnisse über die

Struktur der Daten und ihrer Repräsentation im relationalem Modell. Das Schema-Editor besitzt folgende Funktionalitäten:

1. Die Data Dictionary (Schema) einer relationalen Datenbank auslesen und in XML-Repository umzuwandeln.
2. Hinzufügen von semantische Informationen zum Repository. Dabei werden die Reverse Engineering Techniken (siehe. 3.3) benutzt, die entweder ganz automatisch oder durch Befragung des Benutzers verlaufen. Der Benutzer kann auch die semantische Informationen selber hinzufügen .
3. Hinzufügen und Editieren der Meta-Informationen der Attributen.
4. Anlegen oder Löschen von Tabellen
5. Anlegen oder Löschen von Taballenspalten
6. Anlegen einer DB-Schema aus Xdobry-Repository
7. Generieren von XML, SQL Datenbankabzügen
8. Importieren und Exportieren von DB-Schema für alle unterstütze Datenbank
9. Automatische Migration der Daten und Datenschema zwischen allen unterstützten Datenbanken

Wichtig: Im Gegenteil zu Anderen DB-Manager-Programmen arbeitet Schema-Editor nicht direkt auf einer Datenbankschema. Das Datenbankschema wird ausgelesen und in eigene Datenstruktur (Repository) umgewandelt, die auch als XML Datei (.xmlschema) gespeichert und wieder geladen werden kann. Diese Struktur unabhängig von dem Datenbanksystem. Auf diese weise ist eine Verbindungslose Bearbeitung des Schemas möglich. Aus diesem Schema können DDL-Sql für jedes Datenbanksystem erzeugt werden. Auch die Änderungen des Schemas können als DDL-Sql generiert werden.

Die Idee der graphischer Oberfläche basiert auf der Darstellung des Schema als einer Baumstruktur, die von dem Benutzer manipuliert werden kann. Diese Baumstruktur hat folgende Knotentypen.

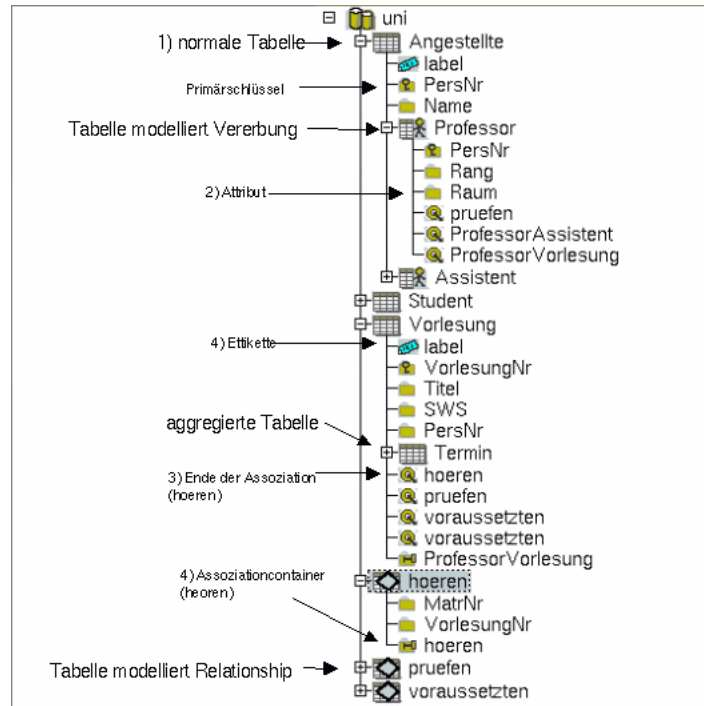


Abbildung 2-1. Anzeige von DB-Schema als ein Baum

1. Tabelle (Knoten)
2. Attribut (Blatt)
3. Assoziationscontainer
4. Assoziationsziel
5. Tabellen-Etikette
6. Attribut-Gruppen (strukturierte Attribute)

Reverse Engineering

Das relationale Modell ist streng wertorientiert. Es kennt keine Verknüpfungstypen. Die Verknüpfungen werden als Fremdschlüssel repräsentiert. Die Tabellen können als Entities, Relationen, Teilobjekte (bei Generalisierung) oder Eigenschaften (z.B. Listen der Attribute) verwendet werden. Bei Reverse Engineering wird die semantische Bedeutung, also die Information, wie das relationale Modell zu interpretieren ist, wiedergewonnen. Diese Informationen werden durch die Namensgebung der Attribute wiedergespiegelt. Weder MySQL noch PostgreSQL unterstützen die Definition von Fremdschlüssel was bei der Erstellung der Formulare großes Nachteil ist. Die Reverse-Engineering Algorithmen basieren bei Xdoby nur auf den relationalen Schema, der Dateninhalt wird nicht ausgewertet. Es wurden drei Reverse-Engineering Algorithmen implementiert. Die Fremdschlüssel-Findung ist eine Basis für andere Algorithmen (außer des Spezialisierung-Findung).

Fremdschlüssel-Findung

Finde alle Attribute die genau so heißen wie die Primärschlüssel der anderen Tabellen aber nicht die Prim-Attribute sind oder nicht einziges Prim-Attribut sind oder den Namen als id mit Präfix der Tabellennamen haben.

Beschränkung: Schlüssel der Objekttabellen bestehen aus einzigen Attribut. Keine Rekursive Beziehungen! Aktion: Es wird eine einfache Referenz erstellt. Assoziation-Container erhält den Fremdschlüssel. Assoziation-Target erhält den Primärschlüssel. Es ist eine Basis für die weitere Algorithmen.

Spezialisierung-Findung

Suche die Spezialisierung. Finde alle Tabellen die einen gleichnamigen Primärschlüssel haben. Sie müssen die Vaternabelle selbst ermitteln. Bei Mehrstufiger Vererbung (Enkelobjekte) muss die Aktion mehrfach durchgeführt werden. Beschränkung: Schlüssel der Objekttabellen bestehen aus einzigen Attribut.

Finde Assoziation-Tabellen

Dieses Reverse Engineering Technik basiert auf dem Schritt "Finde Fremdschlüssel". Es sollten die Tabellen ermittelt werden, die Relationship modellieren (z.B n:m Beziehung). Algorithmus: Finde alle Tabellen, dessen Primärschlüssel aus mehreren Fremdschlüssel besteht oder mehrere Fremdschlüssel und kein eindeutiges Primärschlüssel haben. Aktion: Die Relationship-Tabellen werden besonders gezeichnet. Die n:m oder n:m:z.. Beziehungen werden erkannt. Beziehungen dürfen eigene Attribute haben.

Aggregation Vorschlagen

Dieses Reverse Engineering Technik basiert auf dem Schritt "Finde Fremdschlüssel". Schlage die Aggregation (Komposition) der Tabellen vor. (eingebettete Tabellen). Das Algorithmus zeigt nur die Vorschläge, die Aggregation-Semantik kann nur von Benutzer bestimmt werden. Vorschläge: Alle Tabellen die nur einen Fremdschlüssel haben. Es können noch weiter Aggregation existieren. Überprüfen sie die 1:n Assoziationen

Definieren von Abstraktionen

Es können drei Arten von Abstraktionen der konzeptionellen Datenbankmodellierung definiert werden: Assoziation, Aggregation und Spezialisierung. Die Definition erfolgt stufenweise mit Hilfe von Assistenten.

Assoziation

ist am schwierigsten zu definieren. Es entspricht den Relationship des ER-Modells. Es werden folgende Fragen gestellt.

- Ist die Assoziation rekursiv? Es gibt Beziehungen (Relationship) zwischen den Objekten des gleichen Typs.
- Gibst es eine Tabelle mit Referenzen? Musste die Assoziation mit Hilfen von zusätzlichen Tabellen abgebildet werden, was beim N:M Beziehungen immer der Fall ist, oder wie bei 1:N Beziehungen gibt es nur ein Verweis in einer der Tabelle
- Granularität der Beziehung: beim N:M ist es 2 beim N:M:O ist es 3
- Rollennamen: Ein Objekt Student bekommt durch die Beziehung zu Objekt Prüfung einen Rollennamen Prüfling; (nicht obligatorisch)
- Existenz Abhängigkeit: Werden bis jetzt nicht weiter unterstützt können aber hier definiert werden

Beim n:m Beziehungen muss klar werden: welche Objekte (Tabellen) nehmen an der Beziehung teil, welche Tabelle beinhaltet die Verweise.

Eine Assoziation wird als Container (Sammlung) mit Verweisen auf Objekte aufgefasst. Die Sammlung kann entweder als getrennte Tabelle mit Verweisen oder

als ein Verweis in einem Objekt (1:n) modelliert werden. Um eine Assoziation zu modellieren werden zwei neue Knotentypen hinzugefügt: *Assoziationcontainer* bei Verweisen und *Assoziationtarget* bei Objekten. Zu jeder Assoziation gehört ein Assoziationcontainer und mindestens zwei Assoziationtargets. Die Assoziationen (Assoziationcontainer) besitzen einen eindeutigen Namen. Auf diese Weise können auch komplexe Assoziationen modelliert werden wie: rekursive 1:n und n:m Beziehungen und Beziehungen der höheren Granularität n:m:s:r. Entsprechung in ER-Modell Abbildung 2-8

Aggregation

Hier handelt es sich um etwa die Modellierung von eingebetteten oder geschachtelten Tabellen. Sie werden näher von **FormServer** als eingebettete Formulare dargestellt. Man muss spezifizieren: die Behälter Tabelle, die Element Tabelle und Referenz, Fremdschlüssel in Elementtabelle, das auf Primärschlüssel in Container-Tabelle zeigt. Entsprechung in ER-Modell Abbildung 2-10

Spezialisierung

Auch als Vererbung oder Generalisierung bekannt. Es gibt immer einen Vater Objekt und ein Kind Objekt, die in zwei Tabellen modelliert werden. Man muss auch den vererbten Primärschlüssel angeben. Entsprechung in ER-Modell Abbildung 2-9.

DB-Schema Anlegen und Verändern

Xdoby kann benutzt werden um Datenbank zu modellieren. Es können Tabellen und Tabellenspalten angelegt, verändert und gelöscht werden. Xdoby arbeitet nicht direkt auf der Datenbank Dictionary (Schema) sondern hat eigene Datenbanksystem unabhängige Repräsentation des Schemas, die als XML-Datei abgelegt werden kann. Durch diesen Einsatz ist es möglich eine Schema ohne Datenbank zu entwickeln und auf mehrere unterschiedliche Datenbanksystem aufzuspielen. Die Xdoby spezifisches Schema basiert auf ER-Diagramm und enthält mehr semantische Information als die Datadictionary einer Datenbank.

Anlegen einer neuen Schema

Für das einlegen einer neuen Schema ist keine Datenbankverbindung nötig. Durch das Menü **Schema** → **Neu** wird neues Datenbankschema angelegt. Benutzen Sie die kontextsensitive Menüs (einen Knoten selektieren und rechte Maustaste klicken) um neue Tabellen und Spalten anzulegen.

Xdoby hat eigenes Datentypen System, der an Mysql-Datenbanksystem angelegt ist aber auch andere Typen (wie Money oder Boolean) unterstützt.

Tabelle 2-1. Unterstützte Typen und Typenkonventionierung

Xdoby	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

Xdobry	Mysql	MS Access	MS SQL	Postgres	Oracle
smallint	Smallint, tinyint	smallint	smallint	smallint	smallint
Boolean	smallint	boolean	boolean	Boolean, bool	Number(1)
text	Text, mediumtext, tinytext	Memo, LONGTEXT, LONG- CHAR, 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)
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, LONGBINA- RY, GENERAL, OLEOBJECT, BINARY	image	bytea	blob

Tabelle 2-2. Autoincrement Spalten

Database	Kind
MS Acces	Typ Counter
MS SQL	Indent(1,1)

Database	Kind
mysql	autoincrement
Postgres	Sequencer
Oracle	Sequencer
Sqlite	Not supported

Xdoby kann auch die autoincrement-Spalten entsprechend der Datenbank anlegen. Falls kein Autoincrement-Typ vorhanden ist wird ein Sequencer angelegt.

Tabelle 2-3. Autoincrement Spalten

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

MS Datenbank können auch mit [] Maskierung Tabellen mit Leerzeichen anlegen. (Wird oft in Beispieldatenbank der Firma verwendet). Diese werden falls nötig zu _ konvertiert.

Tabelle 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

Um das Schema auf konkreten Datenbank anzulegen benutzen Sie das Menü **Schema**→Operationen auf Schema→**DB aus Schema anlegen**. Durch das Menü **Schema**→Operationen auf Schema→**Zeige SQL-Definition für Schema** können Sie die CREATE-SQL für die spezifische Datenbank erzeugen. Dieses SQL kann durch datenbankspezifische Prozeduren ergänzt werden, die von Xdoby nicht unterstützt werden. Die so angepasste SQL muss mit datenbankeigenen Werkzeugen aufgespielt werden.

Datenbankschema Ändern

Falls Sie die bereits existierende Datenbank verändern wollen, müssen Sie zuerst die Xdoby Repräsentation der Schema erzeugen. Benutzen Sie dafür das Menü **Schema**→Neu aus DB. Sie müssen dann die Datenbankverbindung spezifizieren. Die Änderungen auf der Schema werden nicht sofort auf die Datenbank übertragen. Durch das Menü **Schema**→Schema Operationen→**Änderungen auf DB**

aufspielen werden die Änderung auf die Datenbank aufgespielt. Durch das Menü **Schema**→**Schema Operationen**→**Zeige Schema Änderungen** können Sie sich die Änderung-SQL anschauen.

Tipp: Sie können die Änderung-SQL benutzen um die Änderungsskript für mehrere Datenbanken (Entwicklung, Intergration, Produktion) zu erzeugen.

Falls die Xdobry XML-Schema bereits existiert, soll Sie als erstes geladen werden. Schema-Editor bietet an nach dem Laden der Schema automatisch sich mit der Datenbank zu verbinden, aus dem das Schema entstanden ist. Falls eine andere Verbindung gewünscht ist oder das Schema nicht aus einer Datenbank entstanden ist, kann mit der Funktion **Schema**→**Verbinde mit Datenbank** die Verbindung erstellt werden. Ist das Schema der Datenbank mit der Schema von Xdobry nicht konsistent. Z.B wurde das Schema der Datenbank mit anderen Werkzeugen bearbeitet, kann man mit der Funktion **Schema**→**Operationen auf Schema**→**Synchronisiere Schema mit Datenbank** Das Xdobry Schema mit der Datenbank synchronisieren. Danach kann das Schema wie gewöhnt angepasst werden.

Wichtig: Man kann die Änderungen an Schema nicht zwischen in mehrere Sitzungen festhalten. So müssen die Änderungen auf der Datenbank aufgespielt werden, damit Sie nicht durch das Beenden der Sitzung verloren gehen. Schema Editor besitzt keine Funktion für das Synchronisieren der Datenbankschema an Xdobry Schema.

DB-Schema und Daten migrieren

Schema-Editor besitzt vielfältige Möglichkeiten das Schema und die Daten zwischen verschiedenen Datenbanksystem zu migrieren. Schema-Editor kann gleichzeitig 2 Datenbankverbindungen offen halten und die Daten satzweise von einen Datenbank auf eine andere zu transponieren. Dabei kann der Umfang der zu migrierten Daten angepasst werden. Es können sowohl Tabellen oder auch einzelne Spalten aus der Migration ausgeschlossen werden.

Durchführen der Migration

1. Eine Schema aus der Quelldatenbank erzeugen. Die Verbindung mit Quelldaten Bank muss vorhanden sein
2. Die Tabellen oder Spalten, die nicht migriert werden sollen, sollen aus der Schema entfernt werden (Sie werden nicht tatsächlich im Quelldatenbank gelöscht)
3. Die Migration starten durch Menü **Schema**→**Migration**→**Migriere Datenbank**
4. Treten während der Migration Fehler auf. Werden diese angezeigt. Die Migration kann in diesem Fall abgebrochen oder fortgesetzt werden. Bei Abbruch wird die Zieldatenbank nicht gelöscht

DB-Schema und Daten Exportieren und Importieren

Schema-Editor kann die Daten auf 2 weisen exportieren. Als XML-Abzug **Schema**→**Migration**→**Datenbank als XML dumpen** der SQL-Abzug

Schema→Migration→**Datenbank als XML dumpen**. Bei SQL-Abzug muss der Zieldatenbank-Typ angegeben werden.

Wichtig: Der Export oder Import mittels XML-Abzug ist nur für kleinere Datenbanken geeignet. Bei XML-Abzug wird in den Arbeitsspeicher kurzfristig die gesamte Datenbank abgebildet (DOM-Objekt). Auf diese Weise kann das Xdoby bei größeren Datenmengen schnell überlaufen. Eine Iterative Erzeugung von XML ist möglich und wird für die nächste Version von Xdoby geplant. Der SQL-Abzug wird iterativ erstellt und verursacht keine übermäßige Beanspruchung von Arbeitsspeicher.

Mit der Funktion **Schema**→Migration→**XML-Dump zu SQL umwandeln** kann man ein XML-Abzug zu SQL-Abzug konvertieren.

FormEditor

Der Zweck der Komponente ist es zuerst eine Sammlung der Formulare zu einer Datenbank zu generieren und zweitens eine Benutzerschnittstelle für die Anpassung der Formulare zu liefern. Das Produkt des Formular-Editor eine Beschreibung der Formulare als ein XML-Dokument. Dabei wird es im Gegenteil zu vorhanden ähnlichen Systemen nicht einzeln zu jedem Tabellen ein Formular generiert, vielmehr wird eine ganze Sammlung der Formularen in einem Schritt generiert. Dabei werden die Assoziationen im Datenbank in der Formularen direkt unterstützt. Der Benutzer des Formular-Editor (Applikation-Entwickler) muss das Schema der Daten nicht kennen. Seine Aufgabe ist es höchstens die automatisch erzeugte Formulare unter dem gestalterischen Gesichtspunkten anzupassen oder die Eingabefelder des Formulars noch weiter zu spezifizieren.

Ein Formular wird als eine Sammlung von verschiedenen Eingabefelder, ihrer Platzierung, und ihrer Entsprechung zu Objekten in der Datenbank verstanden. Folgende Typen der Eingabefelder wurden realisiert:

- Frames. Rahmen für die Platzierung der Elemente
- einfache Textfelder (einzeilig)
- Eingabefelder für Ganzzahlen mit Steuerungspfeilen
- Listenfelder
- Radiobuttons
- Checkbox Felder (Ja/Nein Schalter)
- mehrzeilige Textfelder
- Mehrsplätige Auswahllisten für die Darstellung der Fremdschlüssel (Referenzen)
- Formular Links
- Objekt für die Einbettung von Formularen

Zu speziellen Elementen der Formulare gehören die eingebettete Formulare und die Formular Links. Beide sind verschiedene Visualisierung der Formularverknüpfungen. Die Verknüpfungen entstehend entlang der Beziehung Pfaden (Assoziation oder auch Aggregation). Dabei kann man aus einem Formular ein Formular des Objekts erreichen der ein Fremdschlüssel auf das erste Formular hat. Dabei wird bei dem verknüpften Formular der Fremdschlüssel ausgeblendet (Der Wert wird von ersten Formular bestimmt) und die Dateien werden nach dem Fremdschlüssel gefiltert. Es werden durch die Filterung nur die Objekte gezeigt, die mit dem gezeigten Objekt in dem ersten Formular in Verbindung stehen. Bei eingebetteten Formularen ist der verknüpfte Formular ein Teil des Elternformulars. Bei Formular-Links wird eine Schaltfläche eingebaut, die erst zum öffnen des verknüpften Formulars dient. Die Fremdschlüssel werden nicht als normale Attribute behandelt sondern als spezielle graphische Elemente dargestellt. Dabei sind zwei Tatsachen zu berücksichtigen. Zuerst ist das Wertebereich des Fremdschlüssel durch Integritätsbedingungen auf die

Menge von Primärschlüsselwerte in der korrespondierenden Tabelle begrenzt. Das kann durch eine Auswahlliste repräsentiert werden. Der Benutzer weiß dann, dass er nur eine bestimmte Anzahl von Möglichkeiten hat, die er ansehen und auswählen kann. Zweitens sind die Werte des Schlüssel oft für den Benutzer aussageleer oder ungeeignet für die Identifizierung des Objektes. Dafür erwartet der Benutzer weitere Attribute (z.B Name und Vorname außer des PersonalIDs). Die Aggregation kann als die eingebetteten Formulare realisiert werden. Die Assoziation wird als Formular-Links umgesetzt. Die Spezialisierung kann auch als eine Verknüpfung von Formularen realisiert werden. Dabei wird ein Formular für ein Elternobjekt (Ober-Typ) und je ein Formular für den Kindobjekt (Unter-Typ) verwendet.

GUI Editor für Formulare

Durch das Doppelklick auf ein Formular in Hauptfenster wird ein GUI Editor für Formulare geöffnet. Damit kann das Aussehen des Formulars und die Verknüpfungen der einzelnen Elemente zum Datenbank spezifiziert werden.

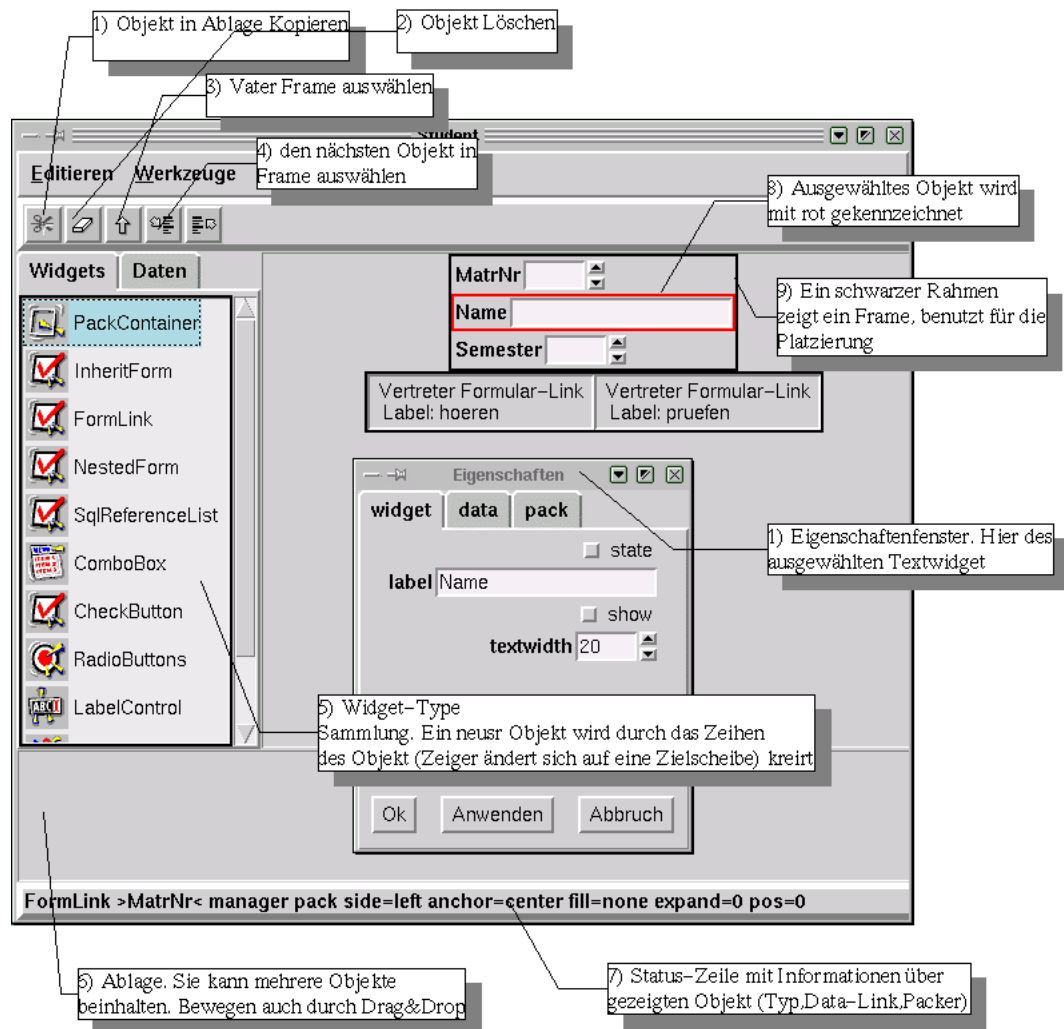


Abbildung 2-2. GUI-Form-Editor (screenshot)

Die Eigenschaften der Elemente werden auf drei Typen

Widget

Die Eigenschaften, die zu eigentlichen GUI-Objekt gehören wie: Länge, Art.

Data

Verknüpfung mit Tabellen-Spalte. Defaultwert und NotNULL

Packer

Angaben zu Platzierung des Elements

Die Platzierung der Elemente wird nicht absolut (also mit Koordinaten) aber unter Verwendung von s.g Geometry-Manager angegeben. Es werden der Pack-Geometry-Manager und Grid-Widget-Manager der Tk-Widgetset unterstützt. Die Elementen können in s.g. Frames (Rahmen) gestellt werden (auch geschachtelt).

Tip: Die aktuellen Pack Optionen kann man in der Statusleiste nach dem Anzeigen des Widgets mit Mauszeiger auslesen.

Die meisten Standard-Widget werden angezeigt wie sie sind. Die anderen speziellen Widget wie (NestedForm, FormLink, SQLReference) werden bei GUI Editor durch die Vertreter angezeigt. Ihre Größe entspricht nicht der Wirklichkeit (Vorsicht beim NestedForm). Will man das tatsächlich Aussehen der Formulare haben, muss man den **FormServer** benutzen, die DB-Anbindung ist für solche Zwecke nicht nötig.

Drag and Drop im FormEditor

Die Fläche der FormEditor wird in drei Flächen geteilt.

Widget Sammlung

Arbeitsfläche mit dem gezeigten Formular
Ablage (untere Bereich)

Die Mögliche Drag & Drop Operationen werden in Abbildung 2-3 gezeigt. Die Drag & Drop wird gezeigt durch das veränderte Maus-Zeiger Icon. Das Objekt wird selbst nicht bewegt. Die Ablage ist hier etwas neues und ungewohntes. Meist verschwindet ein Objekt bei Ausschneiden in einen unsichtbaren Puffer. Hier wird es einfach in die Ablage bewegt. Das gleiche kann man mit Drag & Drop auf die Ablage erreichen. Die Ablage kann mehrere Objekte beinhalten. Deswegen gibt es auch keine Einfüge-Operation.

Tip: Bei Einfügen der Widget zu Formular wird immer das Typ genommen der in Widgetsammlung ausgewählt wurde, also durch dunkles Hintergrund markiert ist.

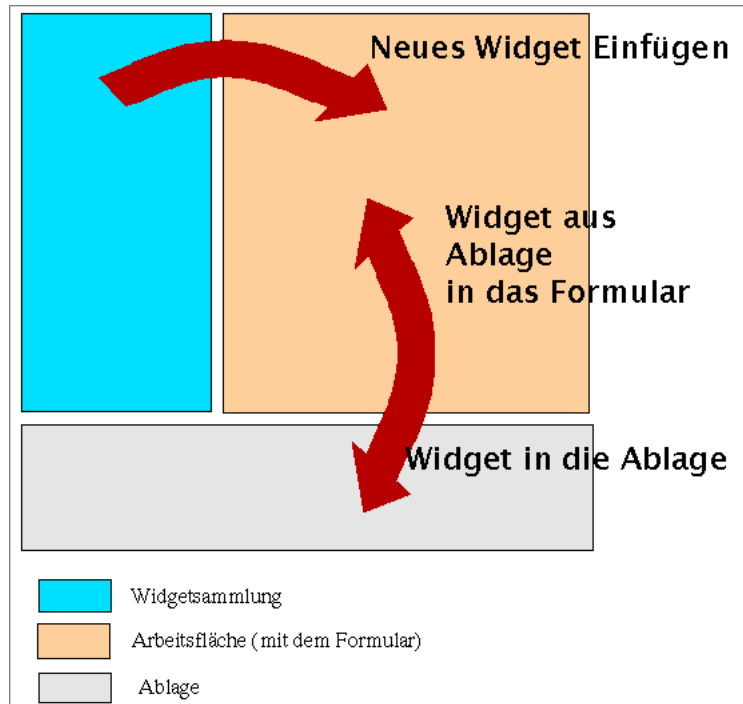


Abbildung 2-3. Drag and Drop Operationen (schematisch)

Platzieren in einem Pack-Frame

Die Reihenfolge der Elemente in einem Frame ist entscheidend. Der Pack-Geometry-Manager nimmt nacheinander die Elemente und stellt sie in den Rahmen der restliche Platz wird für nächste Elemente verwendet. Folgende Eigenschaften spezifizieren die Platzierung:

- side - An welcher Seite der Rahmen soll der Objekt hinzugefügt werden
- anchor - Anker zu welcher Seite
- fill - Soll der Objekt in eine Richtung zu den verbleibenden Platz ausgefüllt werden
- expand - Soll bei Änderung der Fenstergröße der Objekt angepasst werden

Wichtig in diesem Zusammenhang ist die Vorgehensweise wie beim Drag and Drop die Platzierung der Elemente vorgenommen wird. Zeigt man auf ein Frame allgemein wird das Element in das Frame an letzten Stelle mit Standard Optionen angefügt (-side top -anchor center -fill none -expand 0). Die genauere Platzierung wird durch das Anzeigen (Drop) auf ein Element, das bereits in der Frame ist ermöglicht. Die Reihenfolge wird relativ zu dem Angezeigten Element ermittelt, die -side Option wird von dem Element übernommen. Andere Optionen können nur im Eigenschaftfenster verändert werden.

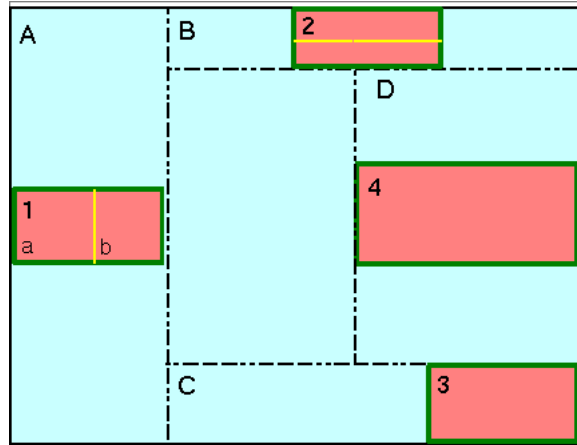


Abbildung 2-4. Platzieren in einem Pack-Frame (schematisch)

Die Abbildung 2-4 zeigt an einem Beispiel das Prinzip wie der Pack-Geometry-Manager arbeitet und wie bei einer Drag & Drop Operation die Platzierung des Widget ermittelt wird. Die Widgets haben folgende Platzierungsoptionen. Die Reihenfolge ist wichtig. Der Algorithmus platziert zuerst das erste Widget (markiert mit 1), der Bereich A wird besetzt und kann weiter nicht mehr verwendet werden. Die Optionen der weiteren Widgets.

1. -side left
2. -side top
3. -side bottom -anchor e
4. -side right

Wie das Widget in seinem Bereich platziert wird hängt von -anchor Attribut. Standardmäßig wird es zentriert (-anchor center). Will man ein neues Widget in das Frame einfügen, muss man die Position relativ zu bestehenden 4 Widgets angeben. Zeigt man in den Raum (Blau) zwischen den Widgets, wird es am Ende der Packliste (Position 5) mit Standardoptionen eingefügt (-side top -anchor center). Also das blaue Raum in der Mitte. Zeigt man auf ein Widget, wird der -side Attribut von dem Widget übernommen, die Paste wird relativ zu dem gezeigten Widget ermittelt. Zeigt man auf die Fläche (a) des ersten Widgets wird die Position auf 0 gewählt. Man kann auf diese Weise die Widgets schnell aneinander reihen. Die speziellen Optionen können erst in einem Dialog spezifiziert werden.

Platzieren in einem Grid-Frame

Das Grid-Manager platziert die Widgets in einem Gitter (Tabelle). Die Zellen-Größen werden entsprechend der Bedürfnisse der einzelnen Widgets ermittelt. Ein Widget kann auch mehrere benachbarte Zellen besetzen (-columnspan -rowspan). Im Gegenteil zu anderen GUI-Editoren werden die Spalten und Zeilen dynamisch gebildet. Man muss also die Zeilen- und Spaltenzahl nicht angeben oder keine neuen Zeilen

oder Spalten einfügen.

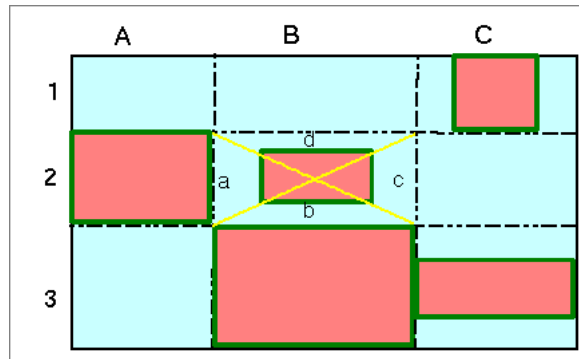


Abbildung 2-5. Platzieren in einem Grid-Frame (schematisch)

Die Abbildung 2-5 zeigt ein beispielhaftes Grid-Frame. Die Zellen A2, B2, B3, C1, C3 sind bereits besetzt. Beim Drag&Drop Operation wird es zuerst ermittelt ob eine Zelle frei ist. Ist das der Fall wird sie einfach besetzt. Wird auf eine besetzte Zelle gezeigt (z.B. B2), wird angenommen, dass eine neue Spalte oder Zeile hinzugefügt werden soll um eine neue Zelle zu schaffen. Wird (siehe Zelle B2) der Bereich d (oben) angezeigt wird eine Zeile über der Zelle B3 hinzugefügt. Das neue Widget wird über das Widget im Zelle B2 platziert.

Die speziellen Optionen wie Spaltenbreite und Zeilenhöhe der Zelle können erst in einem Dialog spezifiziert werden.

Pack Geometry Manager

Es wurden zwei geometry Manager pack und grid für die Platzierung der Widget benutzt. Beide sind sehr mächtig und erlauben sehr gute Feststellung von Windows Platzierung auch beim wechselnden Windowsgrößen. Ähnliche oder gleiche Geometrymanager werden auch bei Java, Gtk und Qt benutzt. Grid ist am schnellsten zu verstehen und entspricht etwa der Tabelle.

Mehr dazu Lesen Sie in Tk-Dokumentation *man pack*, *man grid* Hier ein Ausschnitte aus pack Man-Dokumentation.

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.

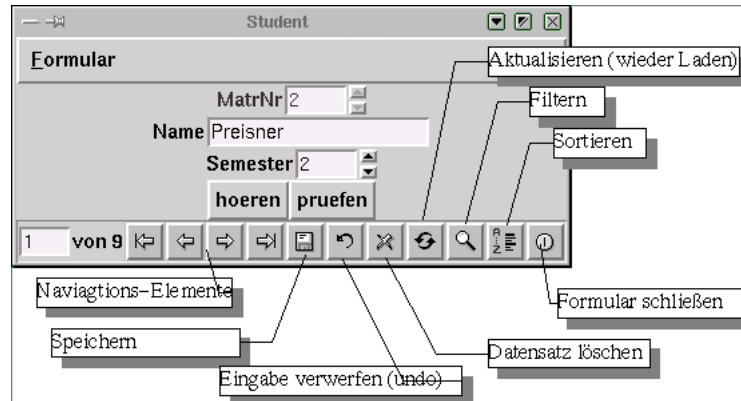
FormServer

Es ist ein Programm, mit dem der eigentlicher Endbenutzer der Datenbank zu tun hat. Dieses Programm kann die Formularbeschreibungen, die von Formular-Editor stammen, richtig auswerten und die Formulare darstellen. Dieses Programm muss die Manipulationen auf den Formularen richtig in die SQL Anweisungen umsetzen und an die Datenbank quittieren. Dabei werden folgende Funktionen unterstützt:

- Daten Anzeigen (View)
- Daten Modifizieren (Update)
- Daten Anlegen (Create)
- Daten Filtern oder Durchsuchen
- Daten Sortieren
- durch Daten Navigieren (Browsing or Navigate)

Durch die Formular-Links ist auch eine Navigation durch Dateninhalte entlang der Verknüpfungspfade möglich. Der Benutzer kann ähnlich wie in Hypertext-Dokumenten navigieren und jede Information auf verschiedenen Wegen erreichen. Vor allem ist die Frage, was ist mit dem Objekt überhaupt verknüpft ist, schnell ohne der Kenntnis des Schemas beantwortet.

Die Formulare selbst unterstützen direkt höhere Abstraktionskonzepte wie Assoziation, Aggregation und Spezialisierung. Die Kenntnisse über die Konzepte der relationalen Modellierung wie Fremdschlüssel und Primärschlüssel müssen nicht vorhanden sein. Der Benutzer kann die verständliche und allgemeinere Konzepte wie Aggregation und Assoziation wahrnehmen, ohne auf die besondere Techniken der relationalen Modell eingehen zu müssen.



Navigieren mit Formularen

Die Benutzung der Formulare zum Navigieren durch eine Daten-Bank war die Idee, die mich zur Entwicklung des System verleitet hat. Mit dem s.g Formular-Links (Verweisen) kann man in einer Datenbank wie in Hypertextdokumenten navigieren. Die Idee der Hyperlinks sollte inzwischen jedem Benutzer bekannt sein. Die Links entstehen entlang der Assoziation-Pfaden.

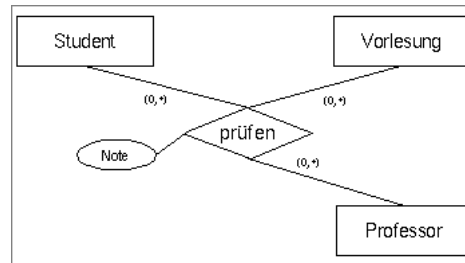


Abbildung 2-6. Assoziation der Kardinalität 3 mit einem Attribut

Die Abbildung 2-6 zeigt eine Assoziation zwischen Drei Objekten (Student, Vorlesung, Professor). Die entsprechende Tabellen-Struktur ist wie folgt:

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

Diese Assoziation (Beziehung, Relationship) entsteht durch das Faktum, dass die drei Objekte durch eine Prüfung zueinander assoziiert werden. Die Grundidee war, dass man aus dem Student-Formular einfach nachschauen kann (durch ein Klick) bei welchen Prüfungen der Student teilnimmt. Zusätzlich kann man durch die Formular-Links die Datenbank erforschen, ohne die genaue Datenstruktur zu kennen. Man sieht unmittelbar, dass ein Student an Assoziation Prüfung teilnimmt.

Die Assoziationen haben aber eine komplexere Semantik als Hypertext-Links. Die Assoziation ist immer transitiv (zweiseitig). Die Hyper-Links haben nur einen einseitigen Verweis-Charakter. Die Assoziationen können zwischen mehr Objekten bestehen. Man sagt, die Granulität der Assoziation kann größer als 2 sein. Ein Hyper-Link verbindet nur 2 Dokumente. Die Kardinalität der Assoziation kann mehr als 1 betragen. Ein Student nimmt an mehreren Prüfungen teil.

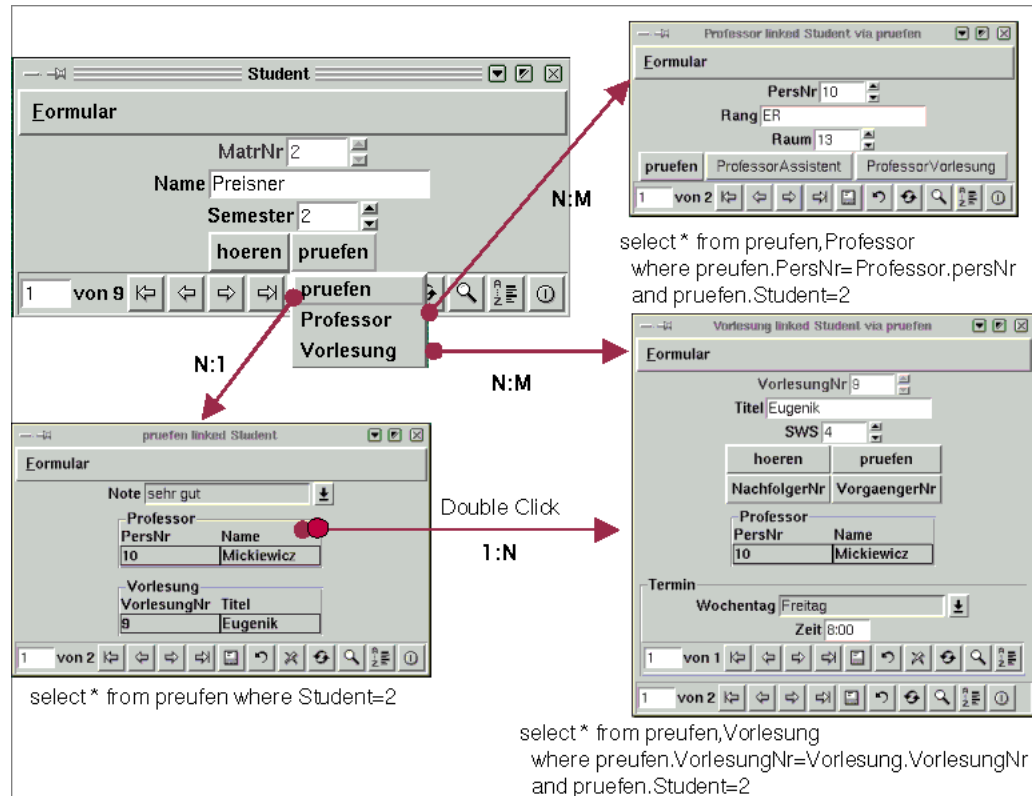


Abbildung 2-7. Navigieren mit Formular-Links

Die Abbildung 2-7 zeigt wie ein Link (Prüfen) aus Formular Student ausgeführt werden kann. Es gibt mehrere Typen von Formular-Link, die auch den Assoziationstypen in ER-Modell entsprechen.

1:1

Die Beziehung wird durch eine veränderbare Referenz dargestellt und deckt sich mit Fremdschlüssel-Attributen der Tabelle. Mit einem Doppelt-Klick kann das Formular des Objekts geöffnet werden, auf das die Referenz Verweis.

N:1

Die Beziehung wird als eine einfache Schaltfläche dargestellt. Es ist die andere Seite der 1:N Beziehung. Die Fragestellung lautet: Suche die Objekte die auf das Objekt zeigen.

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

Die Beziehung wird durch einen Menü-Button dargestellt. Der erste Menüpunkt zeigt immer auf die N:1 Beziehung auf die Referenz-Tabellen (Tabellen, die N:M modellieren, und meistens nur Fremdschlüssel haben). Die nächsten Menüpunkte, zeigen die eigentliche Objekte der N:M:.. Beziehungen. Die Fragestellung: Suche die Objekte die durch die Referenz-Tabelle mit dem Objekt in der Beziehung stehen.

Die Formulare bleiben nach dem Öffnen durch die Links verbunden und passen sich automatisch dem ersten Formular an. Es entsteht eine Link-Kette durch die geöffnete Formulare. Problematisch ist es, dass auch eine Schnittstelle zur Veränderung der Assoziationen angeboten werden muss. Um Missverständnisse zu Vermeiden, werden bei N:M Formular-Links die Funktionen Objekt-Einlegen oder Löschen abgeschaltet. Um Assoziationen zwischen Objekten anzulegen oder zu löschen, muss an der

Referenzen (1:N) gearbeitet werden. In diesem Fall, um einen Student eine weitere Prüfung zuzuweisen, muss ein neuer Datensatz in Formular Prüfung angelegt werden.

Das Konzept der Formular-Links kann auf dem ersten Moment sehr kompliziert erscheinen, ist allerdings intuitiv sehr schnell erlernbar und kann sogar durch die Möglichkeit von Links-Ketten sehr schnell zu Lösung von komplexen Fragen angewendet werden. Z.b. es kann schnell herausgefunden werden, mit welchen anderen Studenten ein Student einer Vorlesung zuhört. Die Fragestellungen. Wie heißen die Assistenten von Professoren, bei denen ein Student ein Prüfung ablegt, sind durch zwei Formular-Links zu lösen. Eine Frage, die bereits einer relativ komplexen SQL-Abfrage benötigt hätte.

Eigene Makros in Formularen

FormularServer besitzt eine Schnittstelle, die es erlaubt, die Funktionalitäten der Formulare um eigene Prozeduren zu erweitern. Dazu können gehören:

Eigene Überprüfung der Integrität-Regel

Besondere Behandlung von Attributen, z.B. Formatierung bei Datenspeicherung oder Daten-Visualisierung

Berechnung von abgeleiteten Attributen

Selbstdefinierte Aktionen

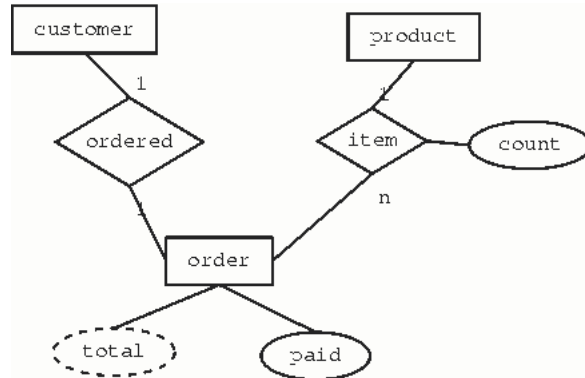
Der FormularServer ist vollständig im Tcl (mit Erweiterungen) programmiert, weil Tcl eine interpretative Sprache ist, erlaubt es, das Verhalten von FormularServer dynamisch zu ändern. Das wird durch Laden von zusätzlichen Tcl-Prozeduren zur Laufzeit erreicht. Es wurde eine Schnittstelle programmiert, die es ermöglicht die Standard-Prozeduren von FormServer zu beeinflussen.

Die Makros (Selbstdefinierte Prozeduren) werden in XOTcl als eine abgeleitete Klasse von FormEngine geschrieben. Der Name der Klasse muss mit dem Namen des Formulars übereinstimmen. Die Makros gelten nur für dieses Formular. Sie werden von FormServer geladen und als ein Teil des Programm benutzt. Hier eine Unterlage die für die eigenen Makros benutzt werden kann. (für Formular myform)

```
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 {
}
```

Die Methoden-Parametern *klvalues_ref* sind Referenzen auf TclX keyd list; mehr dazu in TclX Dokumentation. Durch *_check* Prozeduren kann man eine bestimmte Aktion verhindern (return 0).

Sehen sie das Beispiel *accountancy* das vollständig mit Makros implementiert ist.



Hier wurden Makros benutzt um folgende Eigenschaften zu garantieren

- Ein Customer (Klient) kann nur gelöscht werden, wenn er keine oder nur Bezahlte Rechnungen (orders) hat. Danach wird der Fremd-Schlüssel (custId) in Tabellen cust_order (Bestellungen) auf NULL gesetzt.
- Für jede Rechnung wird der Preis (total) von FormularServer berechnet
- Berechnung des Preises für jedes Produkt (anzahl*preis)
- Überprüfung ob das Datum der Form 2000-01-01 entspricht

Sehen sie die Datei `sample/accountancy.tcl` für die Realisierung (z.B wie sql-Abfragen gebildet werden).

Programm-Parameter

Beim Start von FormServer kann einer Reihe von Programm-Parameter angegeben werden. Es spart die Klick-Arbeit wenn das Programm aus Skripten ausgeführt wurde.

Warnung

Bei der Starkit variante können keine Parameter spezifiziert werden.

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

-debug

Aktiviert Menüs für debugging: SQL-Monitor, XOtcl Klassen-Browser, Tk-Window Browser, Interaktiver Modus

-viewonly

Die Datenbankinhalten können nur angeschaut werden. Kein Löschen, Einfügen und Modifizieren von Tupeln

-ignorexmlpar

Die Parameter, die in XML-Datei kodiert werden (Datenbankanbindung, Makros), werden nicht beachtet (sinnvoll mit -connect parameter)

-noconnectdialog

Es wird versucht eine Datenbankverbindung ohne Befragungs der Benutzer durchzuführen die Verbindungsparameter können entweder in XML oder durch -connect Parameter spezifiziert werden.

-loadmacros

Lädt automatisch ein Tcl-Skript. Sinnvoll um eigene Makros zu programmieren.

-connect

Angeben von Datenbank Verbindungs-Parameter. Wenn keine -ignorexmlpar angeschaltet wurde, werden die XML-codierte Parameter überschrieben. Die Parameter werden in Form einer TclX keyed list angegeben. Die Namen der Parameter hängen von benutzten DB-Schnittstelle z.B. für mysql wäre das.

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

-openform

Das Programm versucht ein Formular nach dem Start sofort zu öffnen. Voraussetzung die Formulare wurden erfolgreich geladen und die Datenbankverbindung steht.

Warnung

Alle FormServer Parameter müssen durch -- von Tcl-Parametern getrennt werden.

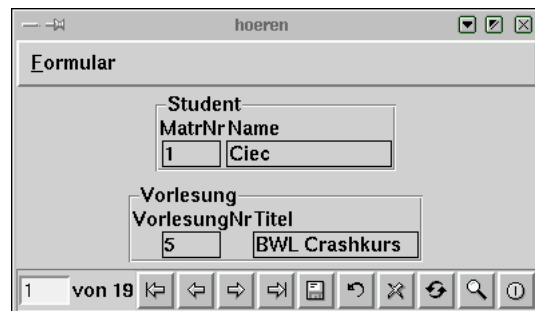
Beispiel (alles in einer Zeile)

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

Schema vs. Formularaussehen

Das System wurde nach dem Motto entwickelt: Das Aussehen der Formulare soll ihrer Semantik entsprechen. Hier die Beispiele, wie die Konzepte der Modellierung ihren Einfluss auf das Aussehen (Struktur) der Formulare haben. Die Screenshoots stammen alle aus dem Tutorial-Beispiel.

Darstellung der Fremdschlüssel



Die Fremdschlüssel Attribute werden als Merhkolumnen-Auswahllisten dargestellt. Hier sieht man auf der ersten Stelle den Wert des Primärschlüssel und einen zusätzlichen Attribut (Name), das die Rolle des Objekts-Etikette hat.

Assoziation (Formular-Links)

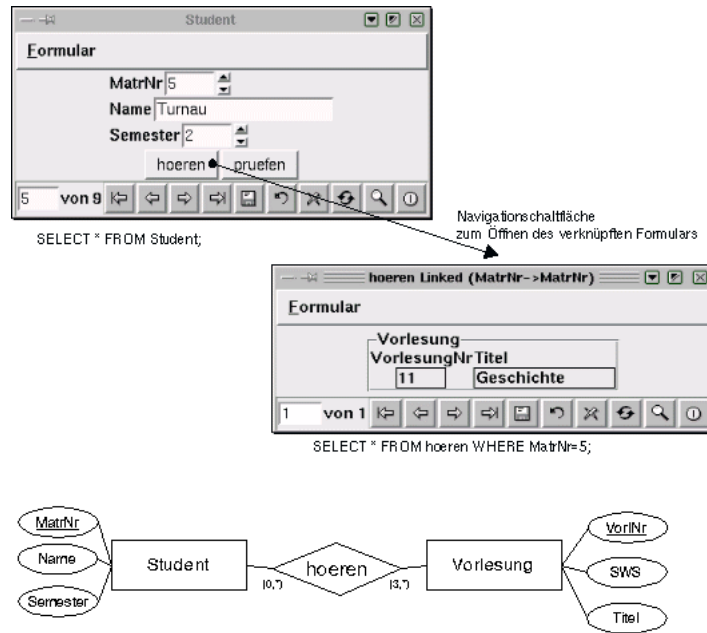
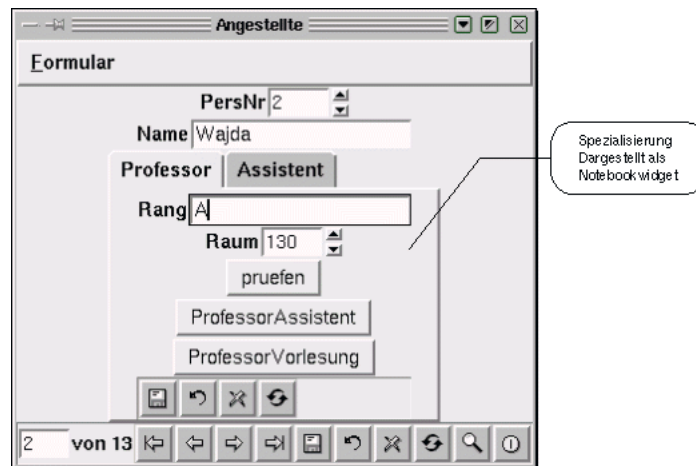


Abbildung 2-8. Assoziation in ER-Modell

Die Formular-Links funktionieren etwa wie Anweisung: Öffne den Formular mit allen Objekten, die mit diesen Objekt verknüpft sind (ein Fremdschlüssel darauf haben). Sie werden entgegengesetzt als Fremdschlüsselverweise

Spezialisierung (Vererbung)



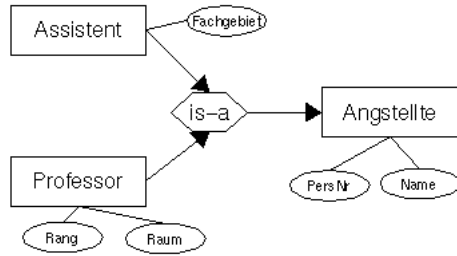


Abbildung 2-9. Generalisierung in ER-Modell

Die Spezialisierung Formulare (Kinder-Formulare) werden in einem Notebook Widget des Eltern-Formulars eingebettet.

Aggregation

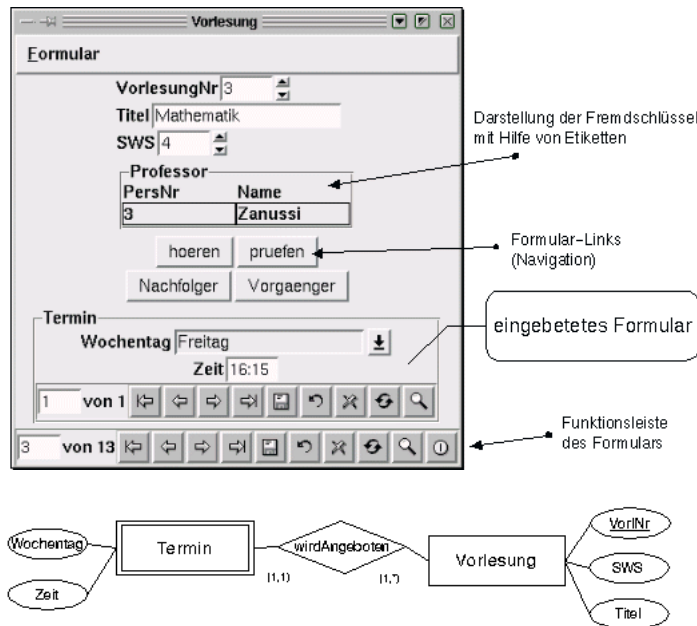


Abbildung 2-10. Aggregation in ER-Modell

Aggregation funktioniert als eingebettete Formulare, dabei werden Fremdschlüssel als Verknüpfung Pfade benutzt; unbemerkt von dem Benutzer.

Fehlende Funktionalitäten bei Xdoby

Ich möchte die mögliche Fallen bei der Benutzung des Programms nennen:

- Bei der Aggregation, Spezialisierung und Fremdschlüssel werden nur Ein-Attributige-Primärschlüssel unterstützt. Ein Objekt wird nur durch ein Attribut identifiziert.
- Bei leeren Tabellen ist das Einfügen des ersten Tupels nur mit der Schaltflächen Speichern (Disketten-Ikon) möglich.

- Die Daten in einem Formular werden erst nach der Änderung der Tuppelposition (mit Navigations-schaltflächen) oder durch die Schaltfläche Speichern zur Datenbank geschickt.
- Es gibt Probleme mit Operationen (Löschen, Ändern, Einfügen), sie werden erst sichtbar nach der Aktion Aktualisieren. Der Zustand, dass die Daten nicht aktuell sein könnten wird durch die rotgefärbte Schaltfläche Aktualisieren angezeigt.
- Die Anzahl von offenen SQL-Verbindungen kann systemspezifisch begrenzt. Es hängt von der Datenbank und der benutzten Schnittstelle ab.
- Die Widget (Multi-Spalten-Listen) für die Anzeige von Fremdschlüssel werden nur einmal bei dem Öffnen des Formulars initialisiert. Sollte sich der Zustand der Objekte ändern, auf die die Fremdschlüssel zeigen, wird es nicht angezeigt.
- Die Formulare, die bei Aggregation benutzt werden, können separat geöffnet werden, die Fremdschlüssel werden jedoch als normale Widget angezeigt. Das kann manuell in **FormEditor** geändert werden, wird aber von FormAssistenten nicht automatisch gezeigt.
- Form-Links funktionieren nicht wenn man auf das Wurzel-Formular zeigt, der Fremdschlüssel sich aber in einem Kind-Formular befindet. Siehe Tutorial (Formular Professor Link auf Assistent). Solche Formular-Links brauchen eine spezielle Behandlung, das noch nicht programmiert wurde.

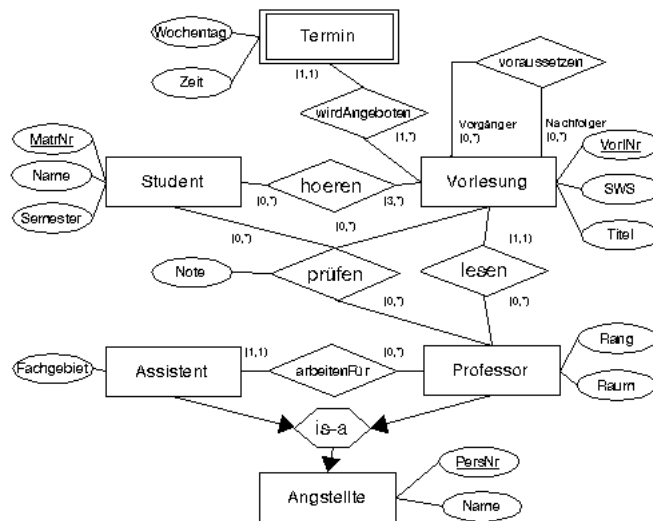
Haben Sie ähnliche Problem festgestellt, oder haben Sie Bedenken zu anderen Funktionsweisen, lassen Sie mich davon wissen.

Kapitel 3. Tutorial

Hier wird es eine Beispielverwendung des System beschrieben. Es wurde eine Lehrbuch-Datenbank gewählt, so dass alle Eigenschaften des System gezeigt werden können. Die Schrittfolge entspricht etwa dem Vorgehen, das bei realen DB-Applikationen Erstellung anzutreffen ist. Zuerst wird ein ER-Diagramm des System erstellt und in relationales Schema abgebildet. Danach wird diese Schema mit Hilfe von Reverse Engineering Techniken in eine XML-Repository umgewandelt. Um die fertige Beispielformulare sofort zu sehen gehen sie zum letzten Schritt.

Relationales Schema

Angenommen, es wurde ein Datenbank-Schema mit Hilfe von ER-Modell entworfen. Dieses Schema enthält 5 Entities. Dabei wird Angestellte zu Assistent und Professor spezialisiert. Bei Termin handelt es sich um ein Weak-Entity. Es ist hier die Modellierung der Aggregation (besser Komposition) von Mengenwertigen Attribut des Objekts Vorlesung. Es gibt auch unterschiedliche Arten der Beziehungen (Relationships). Prüfen ist eine Beziehung der Granularität 3 mit einem eigenen Attribut Note.



Dieses ER-Diagramm wird zum relationalen Schema umgewandelt.

```

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}
  
```

Datei uni.xmldump im Verzeichnis sample des Programms beinhaltet ein XML-Abzug der Datenbank die mit SchemaEditor auf jede unterstützte Datenbank aufgespielt werden kann. Um die Datenbank anzulegen muss man zuerst die uni Datenbank kreieren.

Für die Mysql Datenbank muss man folgende vorbereitende Schritte machen.

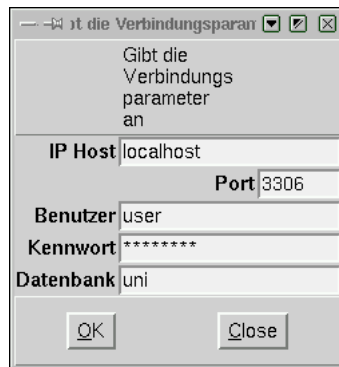
Warnung

Voraussetzung! Sie haben die MySql Datenbank richtig installiert. Es läuft und Sie haben entsprechende rechte die Datenbank und Tabellen anzulegen.

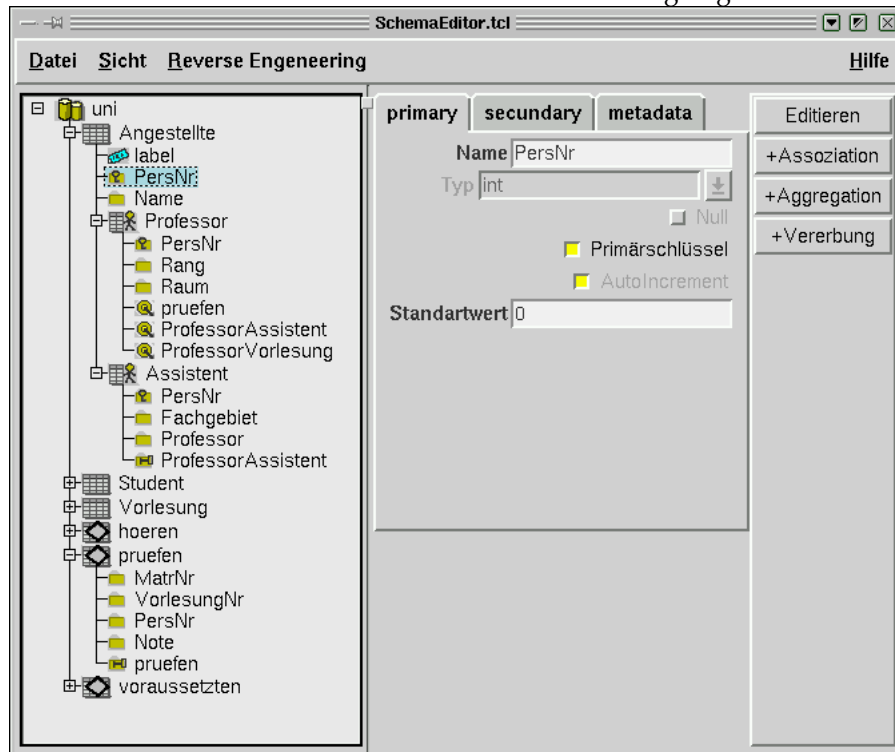
```
[user@localhost]@ mysql
mysql> CREATE DATABASE uni
```

Erstellung des Repositories

Für die Erstellung des Repositories, das eine Erweiterung des relationalen Schemas um semantische Informationen darstellt, wird das Programm **SchemaEditor** verwendet. Das Programm kann durch die SQL-Verbindung das relationale Schema auslesen. Dazu verwendet man den Menüpunkt **Datei** → **Neues Schema aus DB**. In nachfolgenden Dialog muss die richtige Datenanbindung angegeben werden.



Dann können weitere semantische Informationen hinzugefügt werden.



Das kann auf drei Arten geschehen

1. Benutzung von Reverse Engineering Menü **Reverse Engineering**.
2. Hinzufügen von Abstraktionskonzepten durch die Verwendung von Assistenten: +Assoziation (etwa Fremdschlüssel), +Aggregation, +Vererbung
3. Editieren von Eigenschaften der Einzelnen Attributen (Schema-Elementen). Wie auf dem Screenshots können zu jedem Attribut weitere Information spezifiziert werden.
4. Hinzufügen von weiteren Schema-Elementen wie: Attribut-Gruppe, Label (Etiketle). Es geschieht durch Benutzung von Pop-Up Menüis (rechte Maustaste)

Das komplette Beispiel von XML-Repository wird in Datei `sample/uni.xmldbschema` gespeichert. Zu einzelnen Tabellen wird die Semantische Information gespeichert, ob die Tabelle ein Objekt, Spezialisierung, Aggregation oder Assoziation (Relationship) abbildet. Zu Relationship werden können weitere Information eingegeben werden, wie Kardinalität, Rolennamen, Granularität, Existenzbedingungen (beim Löschen).

Erstellung der Formulare

Das Erstellen der Formulare geschieht mit dem Programm **FormEditor**. Zuerst wird das DB-Schema `sample/uni.xmldbschema` geladen **Datei**→**DB Schema Laden**. Automatisches Erstellen von DB-Formularen geschieht mit **Werkzeuge**→**Formular Assistent**. Die Erstellten Formulare können mit einem GUI-Editor durch Schaltfläche **Formular Entwerfen**. Die fertige Formulare sind in `sample/uni.xmlforms` gespeichert.

Benutzung der Formularen

Die fertige Formulare werden als XML-Dateien gespeichert. Das Programm **FormServer** kann die Formulare richtig darstellen und mit der Datenbank kommunizieren. Aufruf mit Argument aus Programmverzeichnis.

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

Wenn Sie tclkit installation benutzen, starten Sie Xdobry in Modus FormServer.

Andere Beispiele

Im Verzeichnis Sample befinden sich noch zwei weitere Beispiele.

1. Ein Beispiel aus Publikation: Roger H.L. Chiang; Terence M. Barron; Veda C. Storey. Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data & Knowledge Engineering 12. Elsevier. 1994. Seiten 107-142. Unter dem Namen *bussines*. Es ist ein Datenbank Schema eines hypothetischen Firma. Es befinden sich folgende Dateien.

```
bussines-er.dia ER-Diagramm der Datenbank in Format eines GNOME Programms DIA
bussines-er.eps ER-Diagramm als PostScript Datei
bussines.xmldump Abzug der Datenbank
bussines.xmldbschema Repository des Schemas
bussines.xmlforms Formulare
```

2. Ein Beispiel einer Kurs-Datenbank. Es werde verschiedene Kurs-Typen angeboten. Die Realisierung eines Kurses wird Angebot (mit Datum) genannt. Die Angebote werden von Referenten in einem Ort durchgeführt. Die Kurse sind

hierarchisch strukturiert. Also zu Kurstyp A gehören Untertypen A1 A2 A3, etc... Zu jedem Kurs kann ein Vorgänger (umgekehrt Nachfolger) genannt werden. Z.B die Teilnahme am Kurs B setzt voraus die frühere Teilnahme am Kurs A (B ist Vertiefung von A).

`kurs-er.dia` ER-Diagramm der Datenbank in Format eines GNOME Programms DIA

`kurs-er.eps` ER-Diagramm als PostScript Datei

`kurs.xmldump` Abzug der Datenbank

`kurs.xmldbschema` Repository des Schemas

`kurs.xmlforms` Formulare

Kapitel 4. Entwickler Informationen

Programmierwerkzeuge

Das gesamte System wurde in der Programmiersprache TCL (Tool Command Language) programmiert. TCL ist eine interpretative und string-basierte Programmiersprache und charakterisiert sich durch einfache Syntax und einfache Erweiterbarkeit. Diese Eigenschaften machen es ideal für das Zusammenkleben von anderen spezialisierten Komponenten (meist Bibliotheken, die in anderen Sprachen (C, C++) geschrieben sind). Die Sprache wird oft für Zwecke des *Rapid Developments Prototyping*s oder benutzt. Tcl benutzt eine geniale, einfache aber gewöhnungsbedürftige Syntax. Obwohl Tcl eine Skriptsprache ist, sind die Tcl-Applikationen deutlich schneller als Java-Programme und verbrauchen sehr wenig Ressourcen. Die objektorientierte Implementierung wurde durch an der UNI Essen entwickelte XOTcl ermöglicht. Das ermöglicht eine klare Strukturierung des Programms. Es wurden folgende zusätzliche Komponenten des TCL benutzt. Das Programm wird mit XOTclIDE Entwicklungsumgebung entwickelt.

Tk und Tix

Eine Bibliothek für das Entwerfen von graphischen Benutzerschnittstellen (Widgetset).

XOTcl

Erweitert Tcl um objektorientierte Eigenschaften.

mysqltcl pgctl ...

Schnittstellen zu Mysql und Postgresql relationalen Datenbanken

tDom

DOM Implementierung für TCL.

Alle Komponenten sind einigermaßen (mit Erfahrung) leicht zu installieren. Ich hatte auch keine Ambition alles in Pure Tcl/Tk zu programmieren, wenn es so viele tolle Bibliotheken gibt.

XML

Im System dienen XML Dokumente als Schnittstellen zwischen Schema-Editor und Formular-Editor (XML Klasse dbschema) und weiter zwischen Formular-Editor und Formular-Sever (XML Klasse xmlforms). Dadurch wird die hohe Unabhängigkeit der Komponenten und ihre Austauschbarkeit garantiert. Jede Komponente kann durch eine andere ersetzt werden, die das entsprechende XML-Dokument manipulieren (interpretieren) kann. Es heißt, sie muss der Schnittstelle, das als eine XML-Kasse (DTD-Dokument) beschrieben ist, genügen. Für den Benutzer des System ist die interne Verwendung des XML-Formats unsichtbar. Er kann nach Bedarf die XML-Dokumente auch manuell erstellen oder mit Hilfe von anderen allgemeinen XML-Werkzeugen verarbeiten.

Die Beschreibungen der einzelnen XML-Klassen als DTD Dokumente werden in Dateien `dtd/dbschema.dtd` und `dtd/dbforms.dtd` beschreiben. Ein kleiner Skript `validate.sh` vereinfacht die Benutzung von nsgml Parser, der bereits standardmäßig zu jeder LINUX Distribution gehört.

Das System wurde so Entworfen, dass diese XML-Dokumente von anderen Applikationen benutzt sein können.

Weiterentwicklung

TODO-Liste:

1. Schnittstelle zu weiteren Datenbanken.
2. spezielle Widget für Zeittypen, Punktzahlen, BLOB
3. unterstützung von BLOB-Typen, die per MIME-Typen spezifiziert werden. Etwa Anbindung an KDE oder GNOME System. Anzeigen von Bilddaten. Austausch mit Dateisystem
4. Formular-Server unterstützt kaskadierendes Löschen und Modifizieren von Primärschlüssel
5. Formular-Server unterstützt Beziehung-Kardinalitäten und andere Integritäts-Regel
6. Direkt-Hilfe zu Dialogen
7. Ballon oder Tip-Texte zu Widgets
8. Alternative Darstellung von Formularen als Tabellen oder Tabelle als Teile der Formulare. Benutzung von tkTable Tcl-Erweiterung
9. Unterstützung von Tabulator Folgen und allgemein Bessere Benutzerfreundlichkeit
10. Befehle per Short-Cuts (Accelatoren)
11. NULL-Wert Behandlung bei Widget. Bis jetzt nicht alle Widget können es gut unterstützen.
12. Neue mächtigere GUI-Elemente (Widget) und mehr Optionen.

Ideen für die Zukunft

Auf dem ersten Plan steht die Fehlerbehebung und Verbesserung der Benutzbarkeit. Das andere ist es wirklich neue Ideen zu verwirklichen. Xdobyry könnte als eine Basis für die Entwicklung von unterschiedlichen DB-Applikationen dienen. Mit XOTclIDE Entwicklungsumgebung ist es möglich Xdobyry auf eigene Bedürfnisse anzupassen.

Als der kleinste Nenner für relationale Datenbanken wurde SQL-92 Standard und mysql gewählt. Viele der Probleme, bzgl. der beschränkt Typbildungsmöglichkeiten, wurden in SQL99 Standard gelöst. Es bleibt vorerst abzuwarten bis die verfügbaren RDBMS diesen Standard unterstützen. Interessant könnte die Implementierung von objektrelationalen Eigenschaften des populären Datenbank Postgres sein. Viele von denen decken sich mit SQL3 Standard.

1. Listen Attribute
2. Tupel können per OID (Objectidentifizier) identifiziert werden
3. Versionifizierung
4. Datenbank-Regel

Es existieren auch 2 interessante Projekte zur Beschreibung der graphischer Benutzerschnittstelle als XML-Dokumente. Einer davon stammt als Nebenprodukt der Entwicklung von Open Source Netscape (Mozilla). Es ist XPToolkit ("<http://www.mozilla.org/xpfe>¹), die Benutzerschnittstelle können plattformunabhängig als XML-Dokumente (XUL XML UserInterface Language). Ein anderer Projekt in mit dem graphischen Editor für die Entwicklung von graphischen Benutzerschnittstellen GLADE <http://glade.pn.org> verbunden. Es ist ein Teil des GNOME Projekt, freies Desktop Environment für LINUX. Die Idee bei

der beiden Projekten ist, dass das Aussehen der graphischen Benutzerschnittstelle nicht festkodiert im Programm ist, sondern immer als ein XML-Dokument vorliegt. Das könnte eine bessere und flexible Konfigurierbarkeit der Schnittstelle durch den Benutzer erlauben, ohne das Programm neu kompilieren zu müssen. Die beide Projekte sind um so mehr wertvoll, dass die komplette Quelltexte frei sind und die verfügbare Komponenten auch bei fremden Projekten verwendet werden können. Die Entwicklung von Formular-Editor und Formular-Server könnte davon profitieren.

Eine andere Richtung für die Weiterentwicklung des Systems wäre die Unterstützung von weiteren Abstraktionen und Semantik in Schema-Editor und anschließend beim Formular-Server. Hier sind zwei Fälle vorstellbar: 1. das System bietet eine Schnittstelle auf die Zusatzfunktionen der DBMS wie Versionierung, neue Datentypen, DB-Regel (Trigger) 2. Das System implementiert selbst neue Funktionalitäten wie Versionierung, Authentifizierung, Integrität Regel, kaskadierendes Löschen. Für den konzeptionelles Modell wäre die Unterstützung von weiteren Abstraktionen denkbar: abgeleitete Attributen, abgeleitete Beziehungen, abhängige Beziehungen (z.B ein Mitarbeiter kann nur an dem Projekt Teilnehmer wenn er die speziellen Qualifikationen hat)., Darstellung und Manipulation von Busstrukturen, die als Tabellen abgelegt sind.

Datenbankanbindung

Zurzeit werden folgende Datenbank unterstützt.

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

Die angebotene Funktionalität entspricht etwa SQL-92 (SQL2) Standard.

Xdoby basiert auf XOSQL-Addapter, der die spezifische Tcl-Schnittstellen verallgemeinert.

WWW-Verweise für Entwickler

1. TCL-Tk Home³ beste Einstiegseite zum Thema Tcl allgemein
2. XOTcl⁴ Homepage des Objektorientierten Tcl-Erweiterung, die in Xdoby benutzt wurde
3. Tix⁵ weitere GUI-Widget Homepage
4. Tclobc⁶ ODBC Schnittstelle für Tcl
5. Mysqtcl⁷ Mysql Schnittstelle für Tcl
6. tDom⁸ Dom Implementierung für Tcl (wird von Xdoby intensiv benutzt)
7. Mysql⁹ freie relationale Datenbank Homepage
8. Postgresql¹⁰ freie postrelationale Datenbank

Fußnoten

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>

Kapitel 5. Produktiver Einsatz von Xdobry

Lizenz

Dieses Software unterliegt der GPL Lizenz. Der genaue Wortlaut der GPL Lizenz auf Englisch ist im Hauptverzeichnis in LICENCE abgelegt.

Wichtig: Xdobry ist freie Software. Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergeben und/oder modifizieren, entweder gemäß Version 2 der Lizenz oder (nach Ihrer Option) jeder späteren Version. Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGEND EINE GARANTIE, sogar ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK. Details finden Sie in der GNU General Public License.

Einsatz von Xdobry

Dieses Software ist noch in der Entwicklung. Das bedeutet jedoch nicht, dass es nicht einsatzfähig ist. Xdobry unterstützt viele Funktionalitäten, die für Produktiveinsatz und Manipulation der sensible Daten notwendig wären (Transaktionen, Sperren, Komplexe-Validierung). Ich kann mir aber vorstellen, dass Xdobry für ad hoc Programmierung und Prototyping einsatzfähig ist.

Vorteile gegenüber bestehenden Systemen

Es gibt viele ähnliche Systeme, die mit der Funktionalität dieses weitgehend übertreffen, trotzdem ist es wert ein paar Besonderheiten der Software aufzuzählen, die das Produkt auszeichnen.

1. DB-Formulare können durch die Formular-Links (siehe Abschnitt namens *Navigieren mit Formularen* in Kapitel 2) zum Navigieren in einer Datenbank benutzt werden. Die Formular-Link werden von System automatisch erzeugt und behandelt.
2. Es ist sehr klein. Das Programm umfasst nur 12.000 Zeilen. Es kann locker auf einer Diskette gebracht werden
3. Es wurden bereits existierende Komponenten mit Programmiersprache TCL zusammengeklebt.
4. Quelltext ist zugänglich
5. Es kann leicht um spezielle Eigenschaften erweitert werden
6. Es benutzt konsequent XML-Format

Der Hauptvorteil ist allerdings der neuartige Entwurf von DB-Formularen. Die Grundlage für die Formulare stellen nicht die Tabellen selbst sondern das konzeptionelle Modell. Das konzeptionelle Modell wird durch Reverse Engineering aus Datenbank Schema (Data Dictionary) gewonnen. Die Formulare haben so bessere Qualität und bedürfen weniger Anpassungen als bei herkömmlichen Systemen. Die Fremdschlüssel werden als Listen dargestellt. Es wird die Vererbung direkt unterstützt. Eingebettete Formulare sind möglich. Das Entwurf und Benutzung von Formularen wurde getrennt (Es wird fast immer von

unterschiedlichen Leuten gemacht), was den Ressourcenverbrauch klein macht. Mehr davon in der zugrundeliegenden Diplomarbeit¹.

Kontakt mit Autor

Homepage: <http://www.xdobry.de> Email: mail@xdobry.de³

Fußnoten

1. <http://www.xdobry.de/artur/diplomarbeit.html>
2. <http://www.xdobry.de>
3. <mailto:mail@xdobry.de>

Kapitel 6. Unterstützung

Das Programm ist frei und es ist ein Geschenk. Wenn Sie Problem damit haben oder es funktioniert nicht scheuen Sie dennoch nicht mich davon zu benachrichtigen. Ich verspreche keine 100-prozentige Garantie der Antwort aber auf jede Fehlermeldung werde ich mich freuen. Das erlaubt das Programm zu verbessern. Zuerst besuchen sie das Homepage des Programms, vielleicht ist der Fehler bereit bekannt und in neuer Version behoben. Formulieren Sie die Fehlerbeschreibung so, dass die Reproduktion des Fehlers möglich ist. (Kopieren sie die Fehlermeldung) Sie können auch ihre Datenbankstruktur (als SQL-Dump) schicken, wenn die Daten nicht vertraulich sind. Ein ER-Diagramm oder Spezifikation der Daten wäre auch sinnvoll. Unterstützung nur durch Email.

Kapitel 7. Xdoby aus Quelldateien installieren

Es wird empfohlen die gepackte und vorkompilierte Starpacks zu verwenden. Diese benötigen keine besondere Kompilierung oder Installation. Zur Zeit werden die Starpacks für Linux x86 und Windows angeboten.

Das Xdoby System besteht aus verschiedenen Tcl-Skripten und braucht deswegen keine Kompilierung. Es benutzt allerdings viele Tcl-Erweiterung (Bibliotheken), die in C oder C++ programmiert wurden. Dazu gehören Tk, Tix die zu jeder Standard Linux-Distribution gehören, also bereits auf Ihren Rechner installiert sein sollten oder von der Linux-Distribution nachinstalliert werden können. Die Windows-Benutzer können auf eine kostenlose Active-State Tcl-Distribution zurückgreifen.

Weitere Tcl-Erweiterungen müssen wahrscheinlich separat installiert werden:

1. XOTcl (objektorientierte Erweiterung des Tcl)
2. tDom - ist eine DOM (Document Object Modell) Schnittstelle für Bearbeitung von XML-Dokumenten.
3. mysqltcl - ist eine Tcl Schnittstelle zu MySql Datenbank.
4. pgtcl - ist eine Tcl Schnittstelle zu Postgresql Datenbank. Gehört zu auch zu neuesten RedHat Distribution.

Die Installation von obigen Erweiterungen kann problematisch sein, da es einige Erfahrung bei Kompilierung abverlangt.

Xdoby analysiert beim Start welche DB-Schnittstellen vorhanden sind und bietet nur diese zur Auswahl. Für die Überprüfung wird Tcl-Kommando **package require** benutzt. Folgende packages werden gesucht:

- mysqltcl für MySql Datenbanksystem
- sqlite für Sqlite Datenbanksystem
- Pgtcl für Postgres
- ORATcl für Oracle
- tclodbc für ODBC
- tclodbc für MS Access (nur Windows)
- tclodbc für MS SQL-Server (nur Windows)

