Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Dario Dobranic, Eric Schreiber

Projekt Aufbau XML/XSLT-basierter Web-Server

Studienarbeit SA-2000.31
Sommersemster 2000

Betreuer: Dr. Erik Wilde

Verantwortlicher:
Prof. Dr. Bernhard Plattner

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Inhalt

1. Vorwort

Mit dieser Semesterarbeiten wurde in erster Linie beabsichtigt, eine neue und funktionsfähige Lösung für die bis anhin existierende Kursverwaltung zu implementieren. Dabei sollte das Problem der Dateninkonsistenz des bisherigen Systems behoben werden. Der Aspekt der Forschung hatte dabei eine eher untergeordnete Bedeutung.

Mit dieser Zielstellung lag die Priorität in der Implementation, wofür wir auch den Grossteil des Arbeitsaufwandes einsetzten. So beabsichtigen wir mit diesem Dokument hauptsächlich, die von uns implementierten Systemkomponenten ausreichend zu beschreiben, um neben der Gewährleistung eines Überblicks und der Vermittlung von Ideen zum Ausbau, eine rasche Einarbeitung in unser Projekt zu ermöglichen.

Das Schwergewicht unseres Projektes lag also auf der Erarbeitung und Implementation einer Lösung für die Kursverwaltung. Lösungen für unsere Problemstellungen sind solche, wie sie heutzutage im e-Commerce Bereich oft eingesetzt werden. Nach einigen Abklärungen und Erwägungen haben wir die Kombination einer MySQL-Datenbank und PHP-Modulen gewählt. Der Aufbau dieser Datenbank und die Überführung der Daten in diese, werden in den Kapiteln 3 und 4 beschrieben, die Verwendung von PHP im Kapitel 7.
Kapitel 5 befasst sich mit der Verwendung von XML in unserem Projekt. Die Motivation XML zu verwenden lag v.a. in dessen Aktualität und weil es unser System flexibel und ausbaufähig macht. In diesem Teil des Projektes konnten neue, auf dem Markt noch weitgehend unbekannte Mittel mit Erfolg integriert und eingesetzt werden, was unserer Semesterarbeit nicht zuletzt einen Forschungsakzent verpasste.
Kapitel 6 befasst sich mit dem administrativen Teil des neuen Systems, welcher mit dem altbewährten Microsoft Produkt MS Access erstellt wurde. Da eine Fortsetzung dieses Teils erforderlich ist, wurde hier besonders auf eine detaillierte Dokumentation der implementierten Komponenten geachtet um eine erfolgreiche Einarbeitung zu ermöglichen. Trotzdem empfehlen wir hierfür auf einschlägige MS Access Literatur nicht zu verzichten.
Das letzte Kapitel 8 schliesslich soll einen Überblick darüber geben welche Komponen-ten des bestehenden Systems noch eines Ausbaus oder einer Verbesserung bedürfen und Ideen und Konzepte hierfür zum Ausdruck bringen.

2. Das Projekt

2.1 Bisheriges System der Kursverwaltung

Wie aus Abbildung 1 hervorgeht stand bei der bisherigen Kursverwaltung das Sekretariat im Zentrum. Dieses bildete die Schnittstelle zwischen Interessenten oder Kursteilnehmern und den Referenten. Wollte sich ein Interessent anmelden, so konnte er dies direkt über das Sekretariat tun, oder er konnte ein Web-Formular ausfüllen, welches beim Bestätigen ein Email an das Sekretariat sendete. In beiden Fällen musste letzteres die Informationen des neuen Teilnehmers in die zur Verfügung stehende FileMaker–Datenbank eintragen. Ebenso ergab sich natürlich bei jeglicher Änderung seitens des Teilnehmers ein Arbeitsaufwand für das Sekretariat.

Daneben benutzte das Sekretariat zudem ein DTP-Programm (FrameMaker), mit dessen Hilfe die Kursinformationen gedruckt werden konnten. Die Informationen über die bestehenden Kurse existierten somit, gleichzeitig in unterschiedlicher Form, als HTML-Dokument im Internet, als auch im FrameMaker-Format in druckbarer Form. Selbstverständlich hatte dies den Nachteil, dass bei einer Änderung der Kursinformation durch den Dozenten, dieser nicht nur das Dokument im Web zu modifizieren hatte, sondern auch das Sekretariat informierten musste, damit die entsprechenden Einträge im FrameMaker vorgenommen werden konnten.

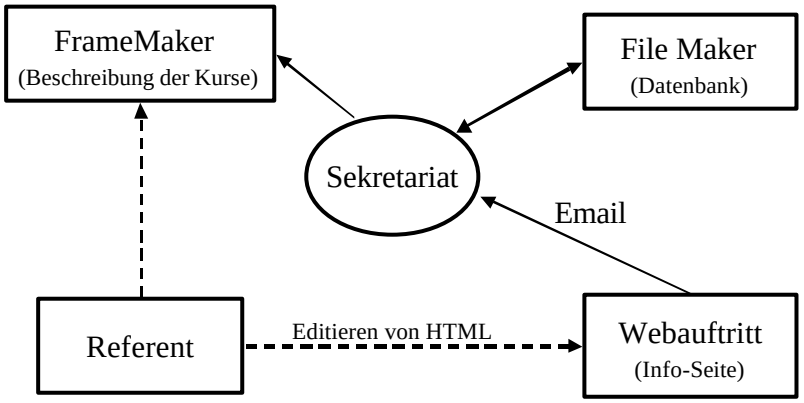


Abbildung 1: Bisheriges System der Kursverwaltung

2.2 Neues Konzept der Datenverwaltung

Unser Konzept sah vor, mit einer zentralen Datenbank zu arbeiten, auf welche die verschiedenen Benutzer und Benutzergruppen zugreifen können (siehe Abbildung 2). Diese Datenintegrität haben wir mit einer MySQL-Datenbank realisiert, welche mit Daten aus der alten FileMaker Datenbank gefüttert, sowie mit Kursinformationen ergänzt worden ist.
Neu hat nun ein Interessent die Möglichkeit via Web seine persönlichen Daten in der Datenbank selbst abzulegen und gegebenenfalls auch zu verändern, ohne dass das Sekretariat in diesen Prozess involviert werden muss. Letzteres hat aber weiterhin die Möglichkeit mit dem MS Access Front-end die Daten einzusehen und zu bearbeiten.

Weiter hat der Dozent die Möglichkeit, seine Kursinformationen direkt zu bearbeiten oder neue Kursinformationen zu kreïeren. Dies geschieht ebenfalls auf einer Web-Seite, welche nur für den jeweiligen Dozenten oder Kursleiter zugänglich ist.
Die Kursinformationen sollen dabei im Format XML in der Datenbank abgespeichert werden.
Nun kann aus diesem XML-Dokument mittels einer Transformation, welche vom Servlet Cocoon ausgeführt wird (mehr dazu im Kapitel 5)-ein HTML-Dokument dynamisch erzeugt werden und so der Inhalt dem Interessenten per Web präsentiert werden.
Als Ersatz für die im alten System verwendeten FrameMaker Dokumente in druckbarer Form bietet es sich hier an aus den XML-Daten PDF-Dokumente zu erstellen. Wie in Kapitel 7.3 beschrieben sollte dieses Konzept in einer 2. Phase realisiert werden.

Das neue Konzept bietet also sowohl dem Interessenten / Kursteilnehmer als auch dem Referenten die Möglichkeit seine Daten bzw. im Fall des Referenten die Informationen seines Kurses selbst zu verwalten. Das Sekretariat muss damit von Seite des Teilnehmers oder des Dozenten nur noch in Spezialfällen in Anspruch genommen werden und wird damit deutlich entlastet.

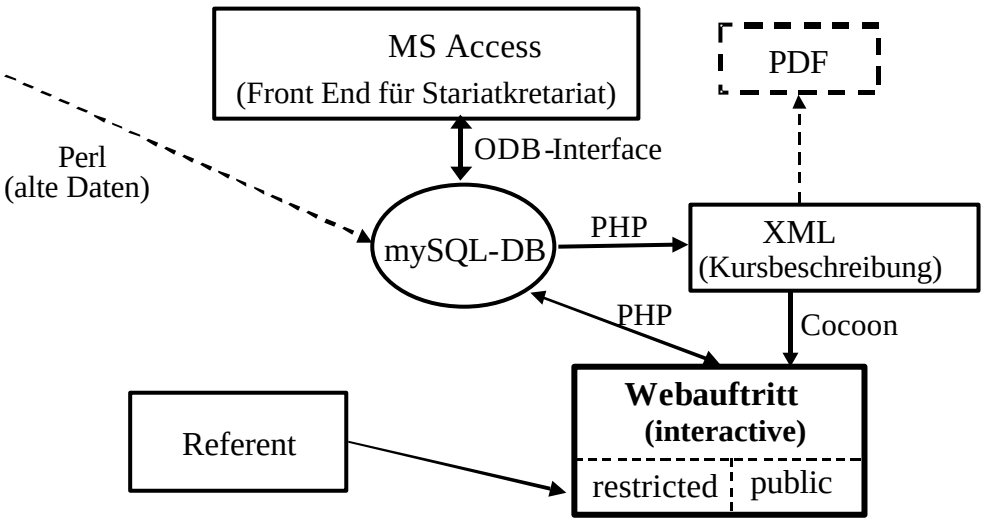


Abbildung 2: Neues Konzept der Datenverwaltung

3. Aufbau der MySQL-Datenbank

3.1 Grundsätzliche Überlegungen

Von Anfang an war es klar, dass wir für die Datenverwaltung eine relationale Datenbank benötigen. Da wir kein Geld ausgeben wollten, hat sich unsere Wahl auf die zwei im Moment populärsten Datenbanken für e-Commerce-Lösungen beschränkt: PostgreSQL und MySQL. Letztere war aus Gründen der Performance, Vorteilen beim Beschaffen von Perl- und PHP-Modulen und Treibern und nicht zuletzt wegen der einfacheren Entwicklung und Administration besser geeignet für unser Projekt.
Der Nachteil einer MySQL-Datenbank ist das Fehlen von Triggerfunktionen und virtuellen Felder, die für die Konsistenz der Daten benutzt werden. Eine solche Datenbank enthält somit selber keinerlei Logik, die die Abhängigkeiten der Daten kontrolliert und gegebenenfalls korrigiert. Aus diesem Grunde wird oft eine Schicht zwischen der Datenbank selber und den Applikationen, die die Daten benötigen, eingebaut, welche solche Aufgaben erledigt. Da in unserem Falle diese Abhängigkeiten nur in geringem Masse vorkommen, haben wir auf diese Zwischenschicht verzichtet. Der Entwickler von Applikationen muss sich diesen Abhängigkeiten jeder Zeit bewusst sein und aufgrund solcher Überlegungen seine Applikation aufbauen.
Für den strukturellen Aufbau unserer Datenbank haben wir das Tools "Dezign"[1] von Heraut gekauft. Mit diesem Tool kann man eine Datenbank grafisch beschreiben und alle Tabellen und die Beziehungen zwischen ihnen definieren. In den Tabellen lassen sich einzelne Spalten mit verschiedenen Datentypen erstellen. Das Tool erzeugt dann SQL-Befehle, welche im MySQL-Shell ausgeführt werden können.
Um die Daten während der Entwicklunszeit für uns besser sichtbar zu machen, haben wir das Tool PhpMyAdmin[2] installiert. Mit diesem Tool kann man praktisch all das machen, was auch im MySQL-Shell durchführbar ist, nur dass die Ausgabe als HTML-Seite schöner und übersichtlicher ist - sofern die Daten in den einzelnen Feldern nicht zu gross sind.

3.2 Struktur der Datenbank

Unsere Datenbank besteht aus acht Tabellen, die man vom Dateninhalt her in drei Gruppen unterteilen kann. Die erste Gruppe verwaltet alle Daten zu einer Person (z.B. einem Kunde oder Referenten) . Das sind die Tabellen *Person*, *Telefon* und *Adresse*. Die zweite Gruppe, die aus den Tabellen *Kurs*, *Kursdatum* und *Referent* zusammengesetzt ist, verwaltet alle Daten zu einem Kurs. Für die Verwaltung der Anmeldungen zu einem bestimmten Kurs werden die Tabellen *Teilnahme*, *Rechnung* und wieder die Tabelle *Adresse* gebraucht. Es ist denkbar, dass noch eine weitere Tabelle dazukommt, die die Authentifizierung der User im Web bzw. die Sessions verwalten könnte.
Abbildung 3 zeigt die Struktur der Datenbank mit den Beziehungen zwischen den Tabellen. Für die genaue Definition der einzelnen Felder schaue man sich diese am besten mit dem Dezign-Tool an.

[1] Erhältlich unter http://www.heraut.demon.nl
[2] Erhältlich unter http://www.phpwizard.net

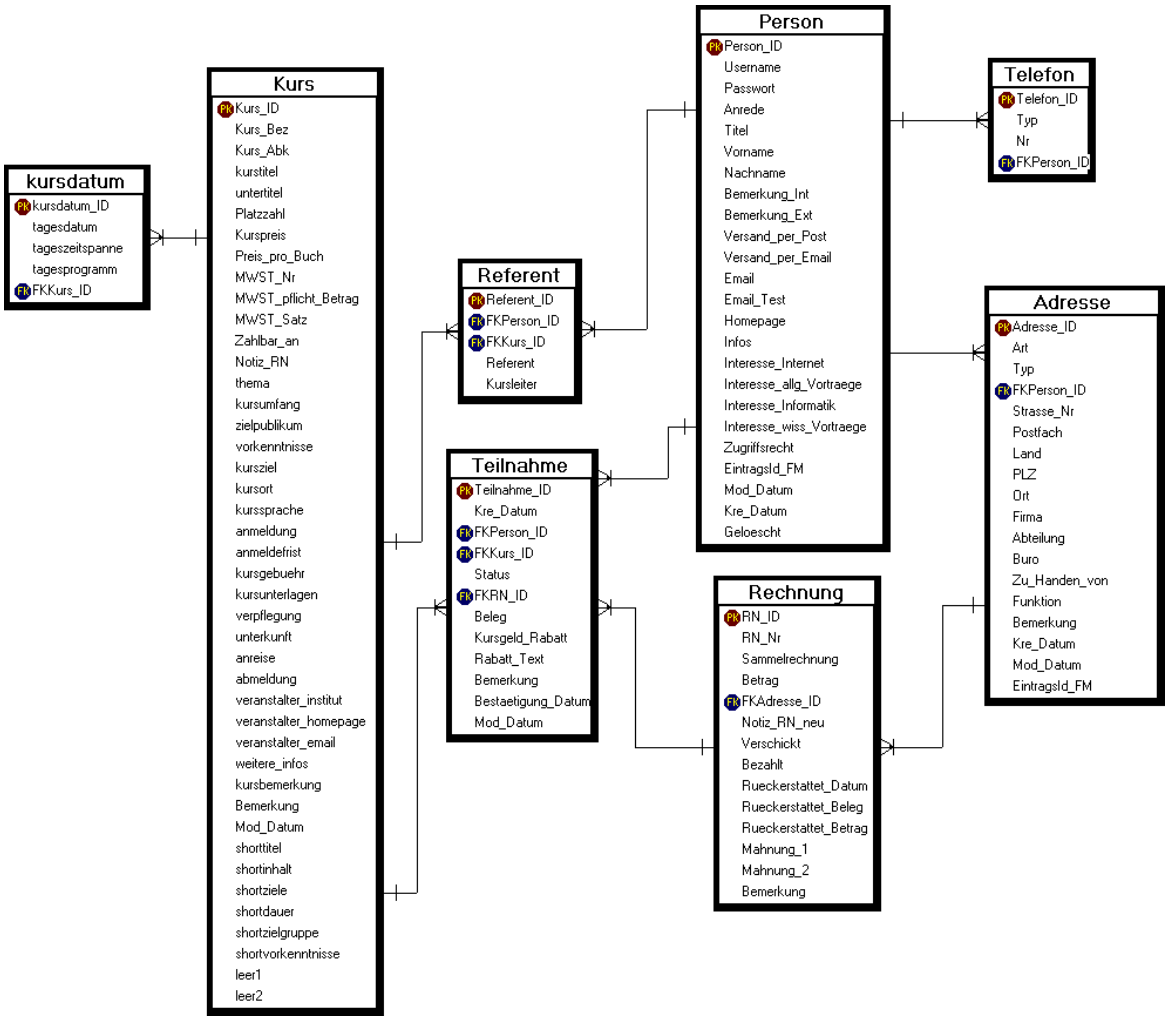


Abbildung 3: Struktur der Datenbank

3.2.1 Tabelle “Person“

Diese Tabelle beinhaltet die Personalien eines Kunden, Referenten oder einer beliebigen Person. Zudem werden hier Ergebnisse von Umfragen gespeichert - z.B. ob die betref-fende Person vom Sekretariat weiterhin Informationen erhalten will oder nicht (Feld *Infos*).

Speziell zu erwähnende Felder:

Username Dieser Eintrag muss eindeutig sein, da er für die Authentifizierung der betreffenden Person im Web benötigt wird.

Zugriffsrecht Falls das Zugriffsrecht auf Ja gesetzt ist, darf die Person via Web alle Kurse editieren. Es handelt sich dann in der Regel um einen Referenten oder Kursleiter.

3.2.2 Tabelle “Telefon“

Die Tabelle *Telefon* enthält alle Telefonnummern, die zu einem Eintrag, bzw. einem Kunden aus der Tabelle *Person* gehören. Eine Telefonnummer ist vom Typ G - Geschäft, P - Privat, N - Natel oder F - Fax.

3.2.3 Tabelle “Adresse”

In der Tabelle *Adresse* finden wir entweder eine Postadresse, die zu einer Person gehört oder eine Rechnungsadresse, die zu einer oder mehreren Rechnungen gehört. Ein Kursteilnehmer darf nach seiner Anmeldung seine Postadresse jederzeit selbst verändern bzw. aktualisieren. Eine Rechnungsadresse (muss bei der Anmeldung angegeben werden) kann später nur noch vom Sekretariat modifiziert werden. Damit soll das Löschen der Rechnungsadresse verhindert, bzw. einem Missbrauch vorgebeugt werden. Falls eine Postadresse auch als Rechnungsadresse verwendet wird, muss diese kopiert und somit doppelt eingetragen werden. Jede Rechnungs-adresse hat im Feld FKPerson-ID den Wert 0 (null), da sie nicht einer Person zugewiesen wird, sondern zu einer oder mehreren Rechnungen.

Speziell zu erwähnende Felder:

Art — Dieses Feld besagt, ob es sich um eine Firmen- oder Privatadresse handelt. Diese Angabe wird für die Erstellung der Brief- oder Zustelladresse benötigt, da eine Firmenadresse anders als als eine Privatadresse zusammensetzt wird.

Typ — Identifiziert eine Adresse als Post- oder Rechnungsadresse

*Zu_Handen_*von — Dieses Feld wird bei Rechnungsadressen benötigt, da diese nicht eine Person referenzieren, sondern zu einer Rechnung gehören. Hier kann man auch einen Eintrag für die Postadresse erstellen, wenn man sich wünscht, dass z.B. die Sekretärin zuerst die Unterlagen erhält. Ein möglicher Eintrag wäre z.B. Frau Rosa Müller

FKPerson_ID — Für eine Postadresse besitzt dieses Feld den gleichen Wert, wie das *Person_ID*-Feld (Primary Key) in der Tabelle *Person*. Bei einer Rechnungsadresse hat es den Wert 0 (null).

3.2.4 Tabellen “Kurs“ und “kursdatum”

Alle Daten, die zu einem Kurs gehören werden in den Tabellen *Kurs* und *kursdatum* abgelegt. Felder aus der Tabelle *Kurs* die mit einem kleinen Buchstaben anfangen, werden für die Beschreibung vom jeweiligen Kurs benötigt. Die Einträge sind oft im XML-Format. Die restlichen Felder - mit grossem Anfangsbuchstaben - sind für administrative Zwecke bestimmt (Kurspreis u.ä.).
In der Tabelle *kursdatum* befinden sich die Tagesprogramme zum jeweiligen Kurs. Für einen dreitägigen Kurs zum Beispiel, gibt es also genau drei Einträge.

Speziell zu erwähnende Felder:

Platzzahl — Dieses Feld muss beim Erstellen eines Kurses unbedingt ausgefüllt werden. Anhand dieser Zahl wird die Anzahl verfügbarer Plätze eines aktuellen Kurses berechnet, was dann dem Interessenten im Web angezeigt wird.

3.2.5 Tabelle “Referent“

Zu jedem Kurs gibt es mehrere Referenten (Referenten oder Kursleiter). Eine Person kann aber auch an mehreren Kursen unterrichten. Solche „n-zu-n Beziehungen“ stellen bei relationalen Datenbanken ein Problem dar. Es wird umgangen, indem eine Hilfstabelle eingefügt wird, die die problematische Beziehung auf zwei 1-zu-n Beziehungen reduziert. Genau das haben wir bei den Tabellen *Kurs* und *Person* mit der Hilfstabelle *Referent* gemacht.

3.2.6 Tabelle “Teilnahme“

In diese zentrale Tabelle wird jeweils ein Eintrag gemacht, wenn sich eine Person für einen Kurs anmeldet. Zur jeweiligen Anmeldung wird zum Zeitpunkt der Erstellung das Feld *Status* auf "angemeldet" oder "auf der Warteliste" gesetzt. Nachdem die angemeldete Person den Kurs besucht hat (oder auch nicht), muss der Status angepasst werden.

Speziell zu erwähnende Felder:

Kursgeld_Rabatt Gibt die Höhe des Rabatts in Franken an. Dieser Betrag soll später, beim Erstellen der Rechnung, vom Kurspreis abgezogen werden.

3.2.7 Tabelle “Rechnung“

Die Angaben zu einer Rechnung werden in der gleichnamigen Tabelle gespeichert. Das Feld *FKAdresse_ID*, das eine Rechnungsadresse referenziert, muss schon beim Erstellen einer Teilnahme ausgefüllt werden. Die Einträge in die Felder *RN_Nr*, *Betrag* und *Verschickt* werden beim Erstellen resp. Verschicken der Rechnung angegeben.

Speziell zu erwähnende Felder:

Sammelrechnung Falls es sich um den Spezialfall „Sammelrechnung“ handelt, kann dies mit diesem Feld mit einem „Ja“ angegeben werden.

4. Import der Daten

4.1 Idee

Nach langem Testen wie man die alten Daten aus den FileMaker-Dateien in die neue Datenbank transferiert, hat sich folgende Methode als die sicherste erwiesen: Die Daten werden zuerst im FileMaker vorbereitet und die Datumsfelder werden in ein richtiges Format gebracht. Danach werden sie in mehrere Zwischendateien im TAB-Format exportiert. In der neuen Datenbank werden drei Hilfstabellen für den Datenimport vorbereitet. Die Daten aus den Zwichendateien werden dann in diese Tabellen eingelesen. Schliesslich werden die Daten aus den Hilfstabellen mit Perl-Skripten auf die eigentliche Datenbank verteilt.

4.2 Anleitung

Um die alte Daten erfolgreich in die neue Datenbank zu importieren, sind die unten aufgelisteten Arbeitsschritte der Reihe nach durchzuführen. Alle FileMaker Felder die ein Datumsformat aufweisen, müssen auf die Form '2000-08-31' umgeformt werden. Der eigentliche Datenexport erfolgt im Tab-Format mit Windows(ANSI)-Zeichensatz und formatiert wie im aktuellen Layout.

Arbeitsschritte:

1. Erzeuge eine neue Datenbank mit dem Namen "filemaker".

2. Erzeuge in dieser Datenbank drei Tabellen "kursdaten", "adressen" und "kurs". Verwende dafür die Dateien *kursdaten.sql, adressen.sql* und *kurs.sql*, die die benötigten SQL-Befehle enthalten.

3. Lese die Datei *Kursdaten.fp3* in FileMaker ein.
   Exportiere folgende Felder unter Beachtung der Reihenfolge: Id, Titel, Untertitel, Kursgeld, PreisProBuch, Zahlbar_an, Kurslokal, Adresse, Kursdatum.
   Die erstellte Datei muss *kursdaten.TAB* heissen.

4. Führe den folgenden Befehl im MySQL-Shell aus:
   ./MySQLimport --host=tik.ethz.ch -u admin -p -P=3306 filemaker kursdaten.TAB

5. Überschreibe im Perl-Skript *kursdaten.pl* die Variablen $dbold und $dbnew und führe es aus.

6. Bearbeite die Einträge in der Spalte *tagesdatum* und ändere den Datentyp auf Datum.

7. Lese die Datei *Adresse.fp3* in FileMaker ein.
   Bei den Feldern *e-mail-Adresse-getestet, Mod.Datum* und *Kre.Datum* muss das Datumsformat angepasst werden.
   Wähle ungelöschte Einträge zum Exportieren aus.
   Exportiere folgende Felder unter Einhaltung der Reihenfolge: EintragsId, Anrede, Titel, Vorname, Nachname, Bemerkung, VersandPerPost, VersandPerEmail, E_Mail, e-mail-Adresse-getestet, Interesse_Informatik, Interesse_Internet, Interesse_wiss_Vorträge, Interesse_allg_Vorträge, Kre.Datum, Mod.Datum, Telefon P, Telefon G, Telefax, Telex, Eintragstyp, Strasse Nr, Land, PLZ, Ort, Firma, Abteilung, Buero, Funktion, Zu_Handen_Adresse.

Die erstellte Datei muss *kursdaten.TAB* heissen.

Funktion
Zu_Handen_Adresse.

Die erstellte Datei muss *adressen.TAB* heissen.

8. Führe den folgenden Befehl im mySQL-Shell aus
   ./mysqlimport –host=tik.ethz.ch -u admin -p -P=3306 filemaker adressen.TAB

9. Im Perl-Skript *adresse.pl*: Setzte den richtigen Wert für die Variablen *$dbold, $dbnew* und *$heute* und lasse es ausführen.

10. Erzeuge einen Ordner mit dem gleichen Namen wie die Kursbezeichnung vom jeweiligen Kurs.

11. Lese die Kursdatei (z.B. *XML0800.fp3*) in FileMaker ein.
    Bei den Feldern *Anmeldung erhalten*, *Bestätigung, Rechnungsdatum* und *Rueckerstatt.Datum* muss das Datumsformat angepasst werden.
    Exportiere folgende Felder unter Beachtung der Reihenfolge:
    Kurs
    Teilnehmer
    Anmeldung erhalten
    Beleg
    Status
    Bestätigung
    Bemerkungen
    RG No
    ID Sammelrechnung
    Rechnung an
    Rechnungsdatum
    Total
    Bezahlt
    Notiz fuer Rechnung
    Rueckerst. Datum
    Rueckerst. Beleg
    Rueckerst. Betrag.
    Die erstellte Datei muss *kurs.TAB* heissen und im Kursordner abgelegt werden.

12. Fange wieder beim Schritt 10 an bis alle Kursdateien exportiert sind.

13. Im Perl-Skript *kurs.pl*: Setze den richtigen Wert für die Variablen *$dbold, $dbnew* und *$heute* und lasse es ausführen (./kurs.pl > raport.txt).

5. Kursdaten im XML-Format

5.1 Idee und Konzept

Im XML-Format abgelegte Daten werden, im Gegensatz zu anderen Formaten inhaltsorientiert beschrieben und nicht wie zum Beispiel in HTML ausgabe-, beziehungsweise formorientiert. Diese grundlegende Eigenschaft ermöglicht es uns aus einem inhaltlich strukturierten XML-Dokument, mittels einer Transformation, verschieden Dokumente in anderen Formaten herzustellen. Damit wird der bereits im Vorwort angedeutet grosse Vorteil von XML ersichtlich: Eine erhöhte Flexibilität und Ausbaufähigkeit unseres Systems.
Wie aus Abbildung 4 hervorgeht wird das XML-Dokument direkt mit PHP aus Daten der MySQL-Datenbank generiert. Dieser Vorgang kann nur durch den Dozenten in einem geschützten Bereich ausgelöst werden. Dabei werden die XML-Tags direkt aus der Datenbank gelesen und in ein speziell benanntes Ausgabefile geschrieben. Durch die Vorgabe der Hierarchie im XML-File, mittels einer Documenten Type Definition (DTD), sind alle somit erzeugten Dokumente gleich strukturiert und können von einem Parser und einer daran anschliessenden Extended Stylesheet Language Transformation (XSLT) gelesen werden. Genau eine solche Transformation führt das Programm Cocoon durch.

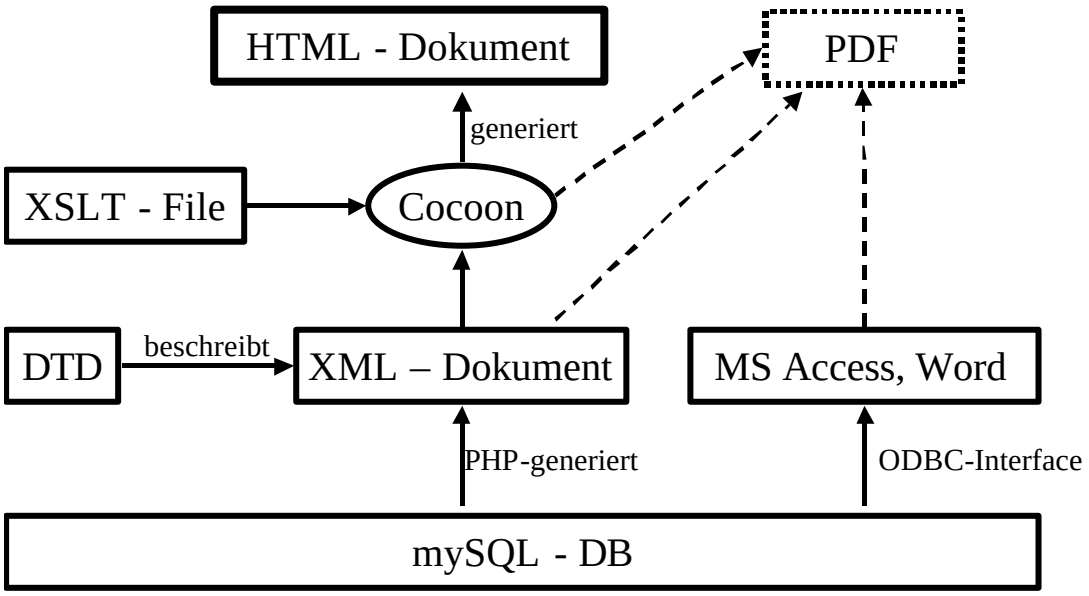


Abbildung 4: Das XML-Konzept

5.2 XSL-Transformation mit Cocoon

Cocoon ist eine Ansammlung von Java-Klassen, welche als Servlets direkt im Webserver – in unserem Fall Apache - eingebettet sind. Neben einem Parser und einem Validator besitzt Cocoon als beinahe wichtigsten Bestandteil die Transformationsengine. Aufgrund einem im XML-Dokument vorhandenen Element liefert diese Engine dann das

gewünschte Format (XHTML, HTML, WML, PDF) auf den gewünschten Ausgabekanal. Für jedes Ausgabeformat ist wiederum ein Dokument (im bereits angedeuteten XSL-Format) nötig, in welchem neben grundliegenden Layout-Angelegenheiten eine eindeutige Zuweisung zwischen den hierarchischen XML-Elementen und der gewünschten ausgabeorientierten Sprache gemacht werden muss.
Cocoon selber wird aufgrund der Dateinamen-Erweiterung ohne spezielle Befehle automatisch aktiviert und fällt anschliessend an die Transformation wieder in den Standby-Modus zurück.
Beim heutigen Stand generiert Cocoon aus dem XML-File, mittels des implementierten XSL-Files, die im Web aufgerufenen HTML-Seite. Überlegungen zur Generation weiterer Formate finden sich im Kap. 8.3.

6. Das MS Access Front-end

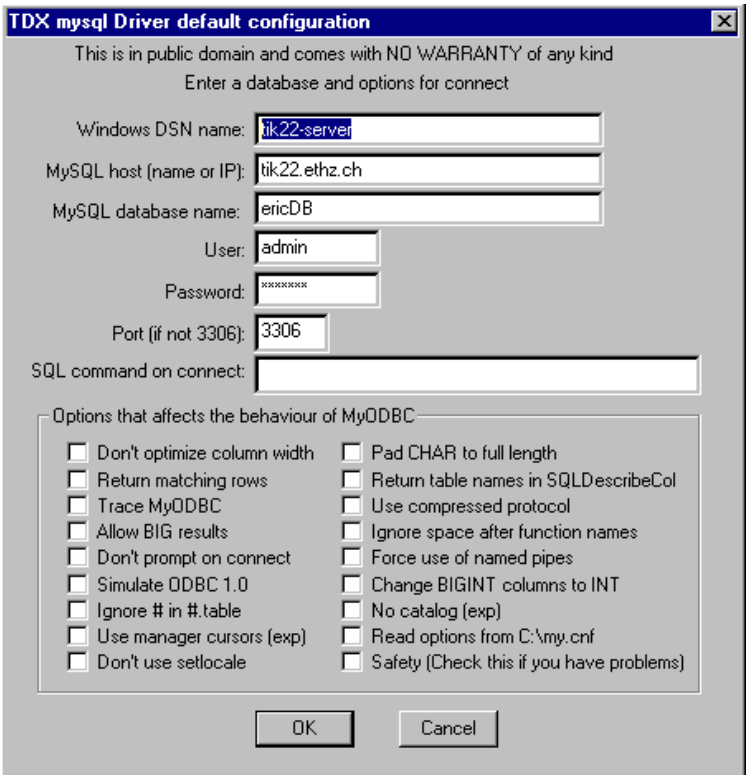
6.1 Installation

Zur Verwendung von MS Access als Frontend für die MySQL – Datenbank benötigt man zunächst einen **MySQL-Treiber**[3] für Windows, der eine ODBC-Verbindung zu einer beliebigen MySQL-Datenbank herstellt. Wurde der Treiber richtig konfiguriert, können die Daten der MySQL Datenbank via Verknüpfung in der MS Access-Datenbank, wie in einer lokalen Datenbank eingesehen und bearbeitet werden.

Wurde der Treiber installiert muss noch die ODBC Datenquelle folgendermassen konfiguriert werden:

Kontrollfenster ? Data Sources (ODBC)

- Das Fenster *ODBC Data Source Administrator* hat sich geöffnet.
- Auf der ersten Seite (*User DSN*) *Add...* wählen
- Das Fenster *Create New Data Source* hat sich geöffnet.
- Den (installierten) MySQL-Treiber auswählen (in unserem Bsp. tik22-server)
- Das nun erscheinende Formular sollte analog folgendem Beispiel mit der Datenbank „ericDB“ ausgefüllt werden:



Jetzt lassen sich die MySQL-Tabellen in MS Access folgendermassen verknüpfen:

[3]Der MySQL-Treiber für Windows ist unter http://www.mysql.com/downloads/api-myodbc.html erhältlich

Bsp. MS Access 2000:

File ? Get External Data ? Link Tables
Das Fenster *Link* hat sich geöffnet. Dort muss nun unter *Files of Type* die Option *ODBC-Databases* gewählt werden.
Das Fenster *Select Data Source* hat sich geöffnet. Hier muss nun unter *Machine Data Source* der oben definierte MySQL-Treiber gewählt werden (Bsp: *tik22-server*)
Das Fenster *Link Tables* hat sich geöffnet. Die gewählten Tabellen können nun verlinkt werden.

6.2 Tabellen und deren Beziehungen

Bei MS Access unterscheidet man 3 Schichten:
Die Schicht der **Tabellen**, die der Abfragen und jene der Formulare und Berichte.

Auf Ebene der ersten Schicht gab es bei uns nicht viel zu tun, da es bei verknüpften Tabellen wenig Sinn macht die Eigenschaften der Felder noch einmal zu definieren (nachdem das bereits in MySQL gemacht wurde).
Bei verknüpften Datenbanken erlaubt MS Access nachträglich noch einige wenige Einstellungsänderungen vorzunehmen. Auf das entsprechende Fenster kommt man, wenn man eine Tabelle in der Entwurfsansicht öffnet.
Je nachdem, welche Eigenschaft eines Tabellenfeldes man fokusiert, wechselt die Schriftfarbe der Beschreibung zwischen blau und rot (rechts unten). Bei blau lassen sich die Einstellungen ändern, so z.B. das Format oder der Name, bei rot jedoch nicht. Bsp. Feldgrösse.
Fälle, wo es vorteilhaft sein kann, das Format anders als in MySQL zu definieren treten dann auf, wenn das Format bei MS Access mehr Einschränkungen unterworfen ist als bei der MySQL Datenbank. Beispielsweise könnte bei einem Feld, wo MySQL das Format „Float" verlangt bei MS Access ein Währungsformat eingesetzt werden. Es lassen sich aber auch auf der Ebene der Formulare sogenannte Eingabemasken erstellen, die das Gleiche bewirken, aber benutzerfreundlicher sind.

Beim Verknüpfen der Tabellen werden zwar die meisten MySQL-Informationen übernommen, nicht jedoch die **Beziehungen** zwischen ihnen. Diese lassen sich in einem speziellen Fenster global definieren. Weil es sich hier um verknüpfte Tabellen handelt, lässt sich bei diesen Beziehungen keine Referentielle Integrität[4] garantieren, was den Nachteil hat, das MS Access darauf verzichtet „gefährliche Manipulationen" zu verhindern.
Weiter lassen sich bei den Beziehungen die sogenanten „join properties" definieren, wo man drei Möglichkeiten hat: Bei der ersten Wahl werden bei Abfragen nur diejenigen Einträge aufgelistet, wo jeweils ein Fremdschlüssel und dessen zugehöriger Primärschlüssel existieren. Meistens wurde diese Option gewählt doch gibt es einige Fälle, wo es von Vorteil ist, ALLE Einträge einer Tabelle mit Primärschlüssel und dazu alle verknüpften Einträge der zweiten Tabelle aufgelistet zu bekommen. Dies war z.B. in der Abfrage *qry/allereferenten* der Fall, wo es – in einer ersten Testphase – Referenteneinträge gegeben hat, die noch keinem Kurs zugewiesen worden waren. Wäre eine Abfrage über die zwei Tabellen gemacht worden - mit der ersten Option gewählt -

[4]Bei *Referentieller Integrität* (Konsistenz der DB) wird bei MS Access mittels einer Reihe von Regeln garantiert, dass die zwischen Einträgen in verwandten Tabellen bestehende Beziehung gültig bleibt. Ein versehehentliches Löschen verwandter Dateneinträge wird somit verhindert.

wäre es nicht garantiert und auch nicht wahrscheinlich gewesen, dass alle Einträge der Tabelle *Referent* aufgelistet worden wären.

Wurden die **Beziehungen global** definiert werden diese als Voreinstellungen beim Erstellen von neuen Abfragen übernommen. Weil es aber nötig war, die Beziehungen bei praktisch jeder Abfrage nochmals zu überdenken, haben wir darauf verzichtet die Beziehungen global zu definieren.

Schliesslich ist noch das **Startformular** zu erwähnen welches beim Starten von MS Access als Popup-Menu erscheint. Dies wird unter „Tools“ bei der Wahl „Startup“ eingestellt.

6.3 Abfragen

Mit dem Design-Tool von MS Access können Abfragen (Queries) in der Entwurfsansicht relativ schnell und unkompliziert erstellt werden. Der Gebrauch des Wizards scheint hier weniger Sinn zu machen.
Beim **Erstellen einer Abfrage** werden zuerst Tabellen (oder andere Abfragen) als Grundlage für die Abfrage ausgewählt. Wie bereits oben beschrieben müssen dann die Beziehungen zwischen den Tabellen (auch Abfragen werden dann als Tabellen betrachtet) definiert werden.
Nachdem die Felder, welche die Abfrage zeigen sollen, ausgewählt worden sind, kann auf die Sicht „Datasheet“ gewechselt werden, wo das Ergebnis der Abfrage gezeigt wird. Die dritte Ansicht „SQL view“ zeigt die Syntax der Abfrage in SQL.

Neben der Möglichkeit Tabellenfelder auszuwählen können auch selbst Felder – sogenannte **ungebundene Felder** – definiert werden. Beispielsweise wird in der Abfrage *qry/rechnung* ein Ausdruck erstellt, der aus den Feldern *Anrede*, *Vorname* und *Nachname* einen String zusammensetzt. Hier wurde auf das Feld *titel* verzichtet weil dieser Eintrag oft leer ist. Ist ein Feld leer, zeigt der aus mehreren Feldern verkettete String nichts an.

Die Ergebnisse der erzeugten Abfragen werden in Formularen oder Berichten dargestellt. Bei deren Eigenschaften (Form ? Design View ? Properties ? Data ? Record Source) kann nun entweder eine Verknüpfung auf eine bestimmte Abfrage angegeben werden (wird bei der Wizard-Erstellung eines Formulars, das auf einer existierender Abfrage beruht automatisch gemacht) oder es kann in diese Zeile direkt der SQL-Code hineinkopiert werden. Auf diese Weise lassen sich **Abfragen in Formulare oder Berichte integrieren**. Der Vorteil hiervon ist, dass eine Abfrage eindeutig einem Formular untergeordnet ist und damit weniger Abfrageeinträge existieren. Die Nachteile sind, dass man wissen muss, dass man bei Unklarheiten die Formulare auf ihre Datenherkunft überprüfen muss und dass man die Abfrage kein zweites Mal verwenden kann. Hingegen ist es kein Nachteil die Abfrage scheinbar nur als SQL-Code zur Verfügung zu haben: Clickt man nämlich zuerst auf den SQL-Code und dann auf das Kästchen rechts davon (Symbolisiert mit drei Punkten) sieht man die Abfrage wie derum in der Entwurfsansicht.

Da es unser Ziel war, mit dem MS Access Front-end sämtliche Daten der MySQL-Datenbank auf eine übersichtliche Weise darzustellen und gleichzeitig deren Manipulation zuzulassen, war es entscheidend den Hauptformularen (im Gegensatz zu den meisten Unterformularen) ein Abfrageergebnis zur Verfügung stellen zu können, welches eine **Datenmanipulation** zulässt. Es scheint unklar, welche und wieviele Tabellen bei einer solchen Abfrage toleriert werden. Beispielsweise erlaubt eine Abfrage über die Tabellen *kursdatum*, *Kurs* und *Referent* keine Datenmanipulationen, während es diejenige über *Kurs, Teilnahme, Person* und *Rechnung* tut. Eine mögliche Erklärung hierfür wäre die Regel, dass bei einer solchen Abfrage neben einer (oder mehreren) Tabelle mit Primärschlüssel nicht mehr als eine (untergeordnete) Tabelle mit

Fremdschlüssel zugelassen ist. Untersuchungen haben gezeigt dass solche Probleme nur bei verlinkten, nicht aber bei importierten Tabellen existieren.

Will man eine Abfrage nach Einträgen mit bestimmten Eigenschaften erstellen, wird dies in der Zeile „**Conditions**“ (in der Abfrage-Entwurfsansicht) definiert. Hier kann auch ein Pfad zu einem Feldinhalt (meist ein Index) definiert werden, womit es möglich ist eine Abfrage basierend auf den Inhalt eines Formularfeldes zu machen. Beispielsweise beruht die Abfrage *qry/kurs* auf den Primärschlüssel im Formular *Kurse overview* und erzeugt so als Ergebnis nur eine Zeile mit den Details des gewählten Kurses.

Auf diese Weise lassen sich auch **Abfragen für Unterformulare** erstellen, welche Daten liefern, die sich auf diejenige Daten des übergeordneten Formulars beziehen. Ein gutes Beispiel dafür ist das Formular *Subform Telefon*, das auf einer Abfrage beruht, welche als Condition den Personen-Primärschlüssel im Formular *Person* referenziert. Wird nun zuerst das Formular *Person* geöffnet und eine bestimmte Person und damit eine bestimmte *Person_ID* ausgewählt, wird die beim Öffnen des Formulars *Subform Telefon* gestartete Abfrage diesen Wert (der *Person_ID*) als Condition verwenden und als Ausgabe die entsprechenden Telefon-Einträge anzeigen.
Sämtliche Unterformulare (mit Namen *Subform* ....) beruhen auf solchen (versteckten) Abfragen.

In unserer Lösung haben alle Abfragen die Bezeichnung qry/.... ausser Zwei: *sortPerson* und *extractKursbezeichnungen*. Hierbei handelt es sich um knapp gehaltene Abfragen, welche als Grundlage für Listen oder Kombinationsboxen in Formularen dienen und dabei die Daten in sortierter Weise auflisten.

Speziell zu erwähnen ist noch die Abfrage *qry/kursteilnehmerliste*. Hier haben wir es mit einer Art „hierarchische Abfrage“ zu tun, wo zu einem Kurseintrag sämtliche Informationen - in den verlinkten Tabellen - gesucht werden. Da es in der jetzigen Experimentierphse in den Tabellen Fehleinträge oder „Löcher“ geben kann, wird so verhindert, dass bei der Abfrage eine leere Zeile herauskommt. Weiter ist bei dieser Abfrage speziell, dass die Tabelle *Person* zweimal auftritt. Der Grund ist, dass einmal über die verlinkte Zwischentabelle *Referent* die Personendaten zum Referenten und das andere Mal über die dazwischenliegende Tabelle *Teilnahme* die am Kurs teilnehmenden Personen gesucht werden.

6.4 Formulare und Berichte

Im Gegensatz zu den Abfragen macht es sich bei den Formularen bezahlt, mit dem Wizard zu arbeiten. Auch wenn man das vom Wizard erstellte Formular noch einmal gänzlich überarbeiten muss, hat man v.a. bei grösseren Formularen Zeit gespart, weil nicht jedes Feld einzeln ins Formular eingesetzt werden muss. Unsere Formulare beruhen alle auf Abfragen. Im Wizard wurde das „Spalten-Layout“ und das Standart-Design gewählt.
Wie bereits im letzten Kapitel angedeutet, kann ein Formular ein Unterformular mit Daten aufrufen, welche mit denjenigen im aufrufenden Formular in einer bestimmten Beziehung stehen. Um dies zu erreichen haben wir grundsätzliche zwei verschiedene Wege benutzt. Die eine Möglichkeit, die bereits beschrieben wurde, basiert darauf, dass sich bereits die Abfrage, welcher ein Unterformular obliegt, auf ein Feld im Basisformular bezieht. Eine zweite relativ einfache Möglichkeit besteht darin mit dem Wizard der Befehlsschaltflächen (vgl. Beschreibung weiter unten) - per Klick - ein Formular öffnen zu lassen, bei dem ein oder mehrere Einträge herausgefiltert werden.

6.4.1 Arbeiten mit der Toolbox

Zu einem Formular in der Entwurfsansicht erscheint immer auch ein Fenster zur Bearbeitung (Toolbox). Mit dessen Hilfe kann mit oder ohne Wizard eine Vielzahl von Feldern, Listen und Knöpfen eingebaut werden, auf die hier besser anhand unserer verwendeten Beispiele als in der Theorie eingegangen werden soll.

Während man bei der Wahl des Bezeichnungsfeldes einfache Labels erstellen kann, erlaubt das zweite Werkzeug das Einfügen von **Textfeldern**, die sich auf eines jener Felder beziehen, welche aus der, vom Formular referenzierten, Abfrage stammen. Die gleichen Textfelder wurden auch schon beim Erstellen des Formulars durch den Wizard eingefügt.

Will man **Umschaltknöpfe oder -flächen** (Toggle Button, Option Button oder Check Box) ausserhalb einer Optionsgruppe (Option Group) erstellen (dazu ist mehr in der Literatur zu finden) hat man drei Möglichkeiten, die sich nur im Layout unterscheiden. Für unsere verlinkte Datenbank ist leider keiner der drei Varianten brauchbar, da ihre Logik vorschreibt für Ja -1 und für Nein 0 zu verwenden. Eine andere Möglichkeit ist die, Umschaltknöpfe in Optionsgruppen unterzubringen. Man kann damit zwar beliebige Werte in beliebigen Feldern abspeichern doch eben nur Werte und keine Strings – so wie „Ja“ oder „Nein“.

Eine Alternative war schlisselich der Einsatz von **Kombinationsfeldern** (Combo Box) wobei mit Hilfe des Wizards auf einfache Weise ein Ja / Nein – Feld erzeugt werden kann und der Inhalt (ein String) im gewünschten Feld abgespeichert wird. Der Einsatz eines Listenfeldes würde, weil hier keine zusätzlichen Daten eingegeben werden können, der Problemstellung besser entsprechen, doch haben wir uns aus Gründen des Designs für die Kombinationsfelder entschieden, da damit keine funktionellen Nachteile in Kauf genommen werden müssen. Kombinationsfelder sind auf vielen Formularen zum Einsatz gekommen u.a. auch weil die aufgelisteten Daten nicht nur direkt eingegeben werden können, sondern auch weil die Datenherkunft eine Abfragen oder Tabelle sein kann. Neben dem häufigen Einsatz bei den „Ja / Nein – Feldern“ oder bei Länderlisten wurden sie deshalb auch verwendet um beispielsweise die Datensätze der Personen (Nachname, Vorname und Personen-ID) oder die Kursbezeichnungen in Listenform darzustellen. Mit Hilfe des Wizards kann zudem noch eine Visual Basic Prozedur erstellt werden, welche im Formular den Eintrag sucht, der in dem Kombinationsfeld ausgwählt wurde.

Ein erwähnenswertes und oft verwendetes Werkzeug ist auch jenes der **Unterformular**-erstellung (Subreport). Mit diesem Werkzeug lassen sich erstellte (Unter-) Formulare in andere (übergeordnete) Formulare einbetten. Auf Wunsch kann beim Erstellen mit dem Wizard eine Beziehung zwischen den Formularen angegeben werden, worauf in unserer Lösung jedoch stets verzichtet worden ist, weil diese Beziehung oder Abhängigkeit bereits beim Erstellen der Abfrage, auf die das Unterformular beruht, definiert worden ist.
Ein Unterformular sollte zuerst alleine funktionieren, die richtigen Daten anzeigen und im Layout angepasst werden, bevor es eingebettet wird.

Erstellt man **Befehlsschaltflächen** (Command Buttons) mit dem Wizard, wird eine kleine Visual Basic Prozedur erstellt, die per Klick Vorgänge ausführt. Häufig zum Einsatz gekommen ist die Prozedur, welche ein bestimmtes Formular (oder einen Bericht) öffnet. Hier kann nun ähnlich wie bei den Unterformularen eine Beziehung zwischen den Textfeldern des aufrufenden und denjenigen des aufzurufenden Formulares angegeben werden. Wird dies gemacht - wie z.B. beim Aufruf des Formulars *Referent* durch die Befehlsschaltfläche *Details* auf der Registerfläche *Details* im Formular *Person* – so wird das aufzurufende Formular geöffnet und durch den

automatisch zum Einsatz kommenden Filter werden nur die benötigten Daten angezeigt. Dies hat den grossen Vorteil, dass ein Formular von verschiedenen anderen Formularen heraus aufgerufen werden und trotzdem unterschiedliche Einträge auflisten kann. Deutlich wird dies beim Formular *Teilnahme* welches einmal vom Formular *Person* aus und einmal vom Formular *Kurse overview* aus aufgerufen wird und dabei z.T. die gleichen Daten hervorbringt.
Nach einigen Performance-Untersuchungen bei Verwendung dieses Systems bei Anbindung an die MySQL-Datenbank über ein 56KB Modem ist überraschenderweise deutlich geworden, dass das Öffnen eines Formulars, dessen Einträge gefiltert wurden weniger lange dauert als wenn die gleichen Daten über die dem Formular zugrunde liegende Abfrage geholt werden.

Neben dieser Formularoperation sind bei uns auch die Eintrags-Operationen (Record Operation) häufig zum Einsatz gekommen. Damit können die Daten eines Formulars und damit bestimmte Daten der Datenbank gelöscht werden, es kann zum letzten Eintrag gesprungen werden oder es können – wie häufig verwendet – die zuletzt gemachten Änderungen im Formular rückgängig gemacht werden.

Bleiben noch das Register-Element und die graphischen Mittel wie Linien und Quadrate zu erwähnen, welche alle in verschiedenen Formularen zum Einsatz gekommen sind.

6.4.2 Das Eigenschaftsfenster

Wird bei einem Formular in der Entwurfsansicht das Eigenschaftsfenster (Properties) geöffnet, bieten sich einem dort ungeahnte Möglichkeiten: Das kleine Fenster enthält nicht nur alle wichtigen Eigenschaften zum Formular, sondern passt sich jeweils auch anderen Objekten an. Clickt man bei geöffnetem Eigenschaftsfenster z.B. auf ein Feld oder ein Kombinationsfeld im Formular, werden die entsprechenden Eigenschaften gezeigt.

Bei Formularen:
Das Layout des Formulars lässt sich auf der ersten Registerseite einstellen. Darauf soll hier nicht genauer eingegangen werden. Auf der zweiten Registerseite *Data* gibt es in der ersten Zeile den Link auf die Abfrage. Hier ist es auch, wo direkt der SQL-Code eingefügt werden kann. In beiden Fällen gelangt man durch klicken des Kästchens rechts dieser Zeile (Symbolisiert mit drei Punkten) auf die Entwurfsseite der Abfrage.

Bei Textboxen:
Hier ist vor allem die Wahl *Locked* – Ja / Nein auf der zweiten Registerseite *Data* zu erwähnen weil hier bestimmte Felder vor Manipulationen geschützt werden können.
Manipulierbare Felder, welche mit blauer Schrift gefüllt werden können haben in diesem Feld ein Nein, wohingegen jene die wir auf schwarze Schrift eingestellt haben gesperrt werden.

Bei Kombinationsfeldern
Bei diesen Feldern ist – ebenfalls auf der Registerseite *Data* - die Zeile *Row Source* relevant. Hier können nachträglich noch Änderungen an der Auswahl vorgenommen werden. Beispielsweise könnte auf diese Weise das Kombinationsfeld der Länder im Formular *Rechnung* (auf der Registerseite *Rechnungsadresse*) noch nachträglich erweitert werden.
Auf der gleichen Registerseite befindet sich auch die Einstellung *Default Value*. Hier haben wir bei allen Kombinationsfeldern, welche ihre Werte aus einer festen Datenliste auslesen (wie in der gerade erwähnten Länderliste), die Default-Datenherkunft derselben Variable zugewiesen, unter der auch der Inhalt des Kombinationsfeldes abgespeichert werden soll. Bei dem einfachen Ja / Nein – Feld *Zugriffsrecht* im Formular *Person* beispielsweise bedeutet dies, dass sowohl der Default Wert, als auch die Abspeicherung von Ja / Nein das Feld *Zugriffsrecht* referenziert.

Bei Befehlsschaltflächen

Auf der Dritten und hierfür entscheidenden Registerseite *Event* kann nachgesehen werden bei welchem Ereignis eine Prozedur, ein Makro oder die Auswertung eines Ausdruckes erfolgen soll. Wir haben fast ausschliesslich Prozeduren beim Ereignis *On Click* verwendet. Tritt das Ereignis auf wird die Prozedur, welche im Visual Basic Editor eingesehen, bearbeitet und getestet werden kann gestartet. Auf die verwendeten Visual Basic Programmstücke soll an späterer Stelle noch eingegangen werden um hier zwei für Formulare relevante Bemerkungen Platz zu lassen:

1) Die Visual Basic Prozedur hat den gleichen Namen wie die Schaltfläche. Will man einen aussagekräftigen Namen verwenden, sollte man dies beim Erstellen (per Wizard) tun, will man ihn nicht später an zwei oder mehr Stellen (Im Formular und im Visual Basic Code) ändern.
2) Löscht man eine Befehlsschaltfläche, wird die zugehörige Visual Basic Prozedur nicht automatisch mitgelöscht. Dies ist zwar nicht gefährlich, doch ist es unsauber wenn unter zig Formularen noch viele Zeilen verlassenen und vergessenen Codes liegenbleiben.

6.4.3 Informationen zu den erstellten Formularen

Die einzelnen Formulare und Berichte sind gemäss folgendem Schema (Abbildung 5) miteinander verlinkt:

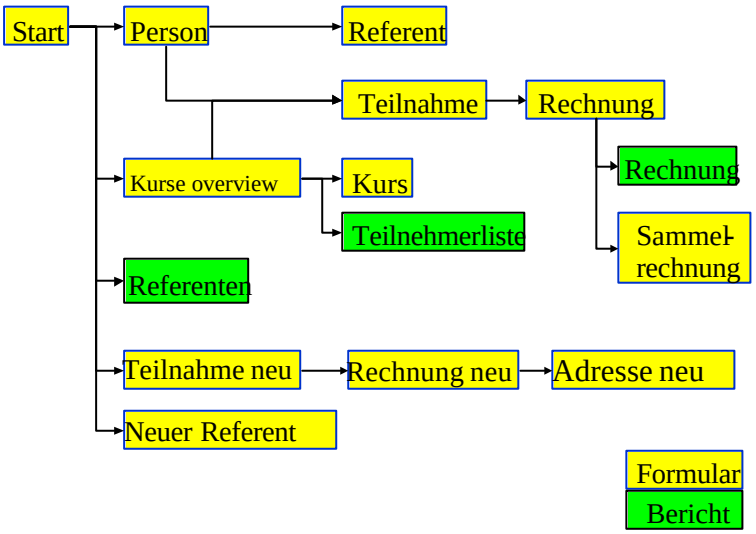


Abbildung 5: Formulare und Berichte in MS Access

Es folgen noch einige Bemerkungen zu ausgewählten Formularen:

- ***Start***
  Dieses Formular besteht praktisch nur aus Befehlsschaltflächen.
  Alle Ereignisse führen zum Öffnen eines Formulars oder eines Berichts. Filter werden beim Starten der nächsten Formulare nicht verwendet
- ***Person***
  In dieses Formular sind verschiedene Unterformulare integriert. Mit Ausnahme des Formulars *Subform Postadresse*, bei dem übrigens nicht sofort ersichtlich ist, dass es sich

um ein Unterformular handelt, referenzieren sämtliche Abfragen der Unterformulare das Feld *Person_ID* auf dem Hauptformular *Person.*
Die Daten auf denen das Formular *Person* basiert und aus der Tabelle *Person* stammen werden fast ausschliesslich auf den ersten zwei Registerseiten gezeigt. Die Einträge sind nach *Namen* und *Vornamen* alphabetisch sortiert, was ermöglicht, das Kombinationsfeld sinvoll einzusetzen. Aus dem gleichen Grund ist es nicht gleichzeitig möglich auch die Personen-ID sortiert darzustellen, doch erlaubt das Kombinationfeld, welches diese IDs (unsortiert) auflistet, zumindest das Suchen eines Eintrages nach einer bestimmten ID.
Eine spezielle Visual Basic Prozedur wird beim Antreffen eines Eintrages, bei dem die Person gelöscht wurde, gestartet. Dazu mehr im nächsten Kapitel.

- ***Kurse overview***
  Dieses Formular basiert wie das Formular *Person* auf einer sortierten Abfrage und stellt den sortierten Teil in einem Kombinationsfeld zur Suchhilfe dar. Sowohl beim Unterformular, als auch beim Starten des Formulars *Kurs* wird kein Filter verwendet, sondern die ensprechenden Abfragen referenzieren die ID des Kurses.
- ***Teilnahme***
  Weil dieses Formular mittels Filterung aus anderen Formularen heraus geöffnet wird, ist es auch möglich das Formular direkt (über das Datenbankfenster) zu starten und es werden sämtliche (ungefilterten) Einträge gezeigt.
  Von diesem Formular aus kann das Formular *Rechnung* geöffnet werden, welches ebenfalls mittels Filterung den zugehörigen Eintrag anzeigt.
- ***Rechnung***
  Wie das Formular *Teilnahme* kann auch dieses Formular direkt geöffnet werden um sämtliche Rechnungseinträge einzusehen. Das Textfeld *Sammelrechnung* – Ja / Nein - sollte solange das Konzept der Sammelrechnung nicht fertiggestellt ist (Vgl. Kap. 8.2) nicht auf „Ja“ gesetzt werden.
- ***Sammelrechnung***
  In diesem Formular werden die „Mitglieder“ einer Sammelrechnung aufgelistet. Bemerkung: Das Formular zur Erstellung einer Sammelrechnung wurde noch nicht implementiert. Vgl. Kapitel 8.2
- ***Neuer Referent***
  Im Gegensatz zum Erstellen einer neuen Teilnahme müssen beim Erstellen eines neuen Referenten nur Fremdschlüssel gesetzt werden, die auf Tabelleneinträge zeigen, die bereits existieren. Dies wird mit Kombinationsfeldern erledigt, welche nach der Auswahl den Fremdschlüssel automatisch auf den gewählten Eintrag setzen.
- ***Teilnahme neu***
  Bei diesem Formular wird zu Beginn auf einen leeren Eintrag im Formular gesprungen um somit die Eingabe neuer Informationen zu ermöglichen. Nachdem eine neue Person und ein neuer Kurs gesetzt worden sind und damit auch die entsprechenden Fremdschlüssel, fehlt nur noch der Fremdschlüsseleintrag der die Tabelle *Rechnung* referenziert. Ein schwerwiegenderes Problem, dass an dieser Stelle auftritt ist der, dass der Primärschlüssel noch immer fehlt. Das Einfügen einer Befehlsschaltfläche zum Speichern schafft keine Abhilfe hingegen führt das herumblättern in den Formulareinträgen zur Speicherung des Eintrages und Setzen des Primärschlüssels. Nun kann mit dem nächsten Formular *Rechnung neu* fortgefahren werden.
- ***Rechnung neu***
  Hier haben wir das gleiche Problem wie im letzten Formular. Wurde der Eintrag trotzdem gespeichert und die Rechungs-ID gesetzt, sorgt beim Verlassen des Formulars (Befehlsschaltfläche *weiter: Rechnungsadresse erstellen*) eine Visual Basic Routine dafür, dass der Fremdschlüssel *FKRN_ID* im Formular *Teilnahme neu* auf den zu referenzierenden Eintrag im Formular *Rechnung* zeigt. Das Analoge mit dem

Fremschlüssel *FKAdresse_ID* im Formular *Rechnung neu* geschieht nochmals beim Abschliessen des Formulars *Adresse neu.*

6.4.4 Visual Basic Prozeduren

Aus Zeitgründen und weil entsprechende Literatur fehlte sind Visual Basic Applikationen (VBA) nur in geringem Ausmass in unserer Lösung implementiert worden.
Will man eine bestimmte VBA einsehen, kann das Eigenschaftsfenster eines Formulars geöffnet werden wobei der Fokus auf einem VBA-enthaltenden Element liegen soll. Nun sollte auf der Registerseite Event ein Eintrag zu finden sein. Wird dieser angeklickt und anschliessend ebenfalls das mit drei Punkten gekenzeichnete Kästchen rechts daneben gelangt man direkt zur entsprechenden Prozedur im VBA - Editor.
Zur Erzeugung des Codes kamen drei Möglichkeiten zum Einsatz:

1. Erstellt man eine Befehlsschaltfläche und damit ein Ereignis per Wizard, erstellt dieser automatisch ein Programmstück, das in ein VBA-Objekt (eine Code-Seite) integriert ist, auf das sich das Formular, auf dem sich die Befehlsschaltfläche befindet, bezieht und in dem auch sämtliche anderen Prozeduren dieses Formulars zu finden sind.

2. Mit dem Makro-Editor lassen sich einfachere Ereignissabläufe erstellen. Ein erzeugtes Makro lässt sich in VBA – Code transformieren (Makro ? save as Module). Nun kann der erzeugte Code weiterverwendet werden und z.B. in ein vorbereitetes Programmstück, welches wie unter Absatz 1. beschrieben erstellt worden ist, übernommen werden.

3. Viele Programmbeispiele, so z.B. der Gebrauch der häufig verwendeten *message box (MsgBox),* lassen sich aus der VBA-Hilfe entnehmen

Am häufigsten kam ein Befehl zum Einsatz, der es ermöglicht ein Datenfeld auf das heutige Tagesdatum zu setzen. Die entsprechende Befehlsschaltfläche ist vor allem auf den Formularen *Teilnahmen* und *Rechnung* zum Einsatz gekommen.

Die Funktion *MsgBox* ist da zum Einsatz gekommen, wo es galt eine Warnmeldung anzuzeigen und eventuell eine begonnene Prozedur abzubrechen. Dies ist auf den Formularen *neuer Referent* und *Teilnahme neu* beim Klicken auf die Lösch-Befehlsschaltflächen implementiert worden. Es erscheint ebenfalls eine Warnmeldung wenn im Formular *Person* ein gelöschter Eintrag erscheint. Diese Prozedur wird ausnahmsweise beim Ereignis *On Current* im Formular *Person* gestartet.

Letztlich bleibt noch das Setzen der Fremdschlüssel bei den Formularen *Teilnahme neu*, *Rechnung neu* und *Adresse neu* zu erwähnen. Die entsprechenden Befehlszeilen wurden in Prozeduren integriert, die beim Klicken auf die „Weiter- Befehlsschaltflächen“ ausgeführte werden. Wird beim Klicken auf die Schaltfläche das Formular *Rechnung neu* verlassen, wird in der dann aufgerufenen Prozedur das Formular *Adresse neu* geöffnet und auch der Fremdschlüssel *FKRN_ID* im Formular *Teilnahme neu* auf den ensprechenden Eintrag im Formular *Rechnung neu* gesetzt. Logischerweise kann dies erst dann geschehen, wenn im Formular *Rechnung* der Primärschlüssel gesetzt wird. Deshalb wurde hier auch eine Warnmeldung hinzugefügt.

6.4.5 Bemerkungen zu den Berichten

Da Berichte ähnlich aufgebaut sind wie Formulare soll hier lediglich auf die bereits erstellten Berichte eingegangen werden. Wie die meisten Formulare wurden sie zuerst mit dem Wizard

erstellt und danach gründlich nachbearbeitet. Die Werkzeuge hierfür sind die gleichen wie bei Formularen.
Da einige Berichte aus Zeitgründen nicht fertiggestellt werden konnten, weisen wir für vorgesehene Weiterentwicklungen auf das Kapitel 7.2 hin.

Der Bericht ***Teilnehmerliste***, der aus dem Formular *Kurse overview* gestartet wird, listet alle Teilnehmer zu einem bestimmten Kurs auf. Die Daten für diesen Bericht stammen aus der Abfrage qry/kursteilnehmerliste.

Der Bericht ***Rechnung*** basiert auf der Abfrage *qry/rechnung_bericht*.
Die eigentliche Schwierigkeit bei der Rechnung besteht darin, eine Logik einzubauen, die zwischen Firmen- und Privatadresse unterscheidet. Im ersten Fall stammen alle Informationen für die Rechnungsadresse aus der Tabelle *Adresse*, während im zweiten Fall zusätzlich *Anrede, Titel, Vor- und Nachname* aus der Tabelle *Person* extrahiert werden müssen. Im vorliegenden Bericht wurde das Problem gelöst indem für den jeweiligen Adresstyp ein „Unterbericht" erstellt wurde (*Subrep Firmenrechnungsadresse* und *Subrep Privatrechnungsadresse*), der jeweils auf einer (SQL-integrierten) Abfrage beruht. Diese zwei Unterberichte wurden in den Bericht *Rechnung* integriert und dort übereinandergelegt. Da die beiden, den Unterberichten zugrunde liegenden Abfragen einander ausschliessen, d.h. es gibt nur jeweils EINE Ergebniszeile der Abfrage entweder bei der einen ODER der anderen Abfrage, erscheint im Basisformular Rechnung entweder der Unterbericht *Firmenrechnungsadresse* ODER *Privatrechnungsadresse.*

Der Versuchsbericht ***Referenten*** listet schliesslich sämtliche Referenten auf und stellt Informationen zu deren Funktion bei den verschiedenen Kursen zur Verfügung.

7. Dynamische Webseiten mit PHP3

7.1 Allgemeines über PHP

Für den Aufbau von dynamischen Webseiten, haben wir PHP3 verwendet. Zur Zeit der Entwicklung war die PHP4-Version für Solaris noch nicht erhältlich. Es ist absehbar, dass beim definitiven Umziehen des Systems auf den neuen Server, die Skripte angepasst werden müssen, wenn auf PHP4 umgestiegen werden soll.
Das grösste Problem beim Erstellen von PHP-Seiten lag beim Fehlen einer richtigen Entwicklungsumgebung bzw. eines Debuggers. Sehr viel Zeit wurde für die Suche von Fehler aufgewendet. Die meisten Fehler lagen bei den SQL-Befehlen, die extrem "empfindlich" sind. Bei einem fehlerhaften SQL-Befehl kommt einfach ein leeres Array zurück und man weiss nicht, ob die Anfrage an die Datenbank misslungen war oder ob ein Fehler in der Anfrage selber vorlag. Am besten versucht man es mit ein Paar zusätzlichen Anfragen an die Datenbank.
Weitere Probleme lagen beim Erstellen von Formularen, da die dort gesetzte Variablen oft über mehrere PHP-Seiten durchgeschleust werden mussten. Bei solchen Fällen haben wir die kritische Variablen als Hidden-Fields im Formular selber versteckt. Sie werden dann zusammen mit den neu definerten Variablen mit der Post-Methode (HTML) weitergereicht.

7.2 Aufbau der PHP-Formulare

Unser Ziel beim Implementieren von Formularen war es, einen möglichst transparenten Code zu erstellen. Ein Formular ist immer Modular aufgebaut. Die ganze Abarbeitungslogik befindet sich in einer zentralen Datei (z.B. personalien.php3), die abhängig von den erhaltenen Daten verschiedene Module mit include-Anweisungen lädt. Am Anfang wird immer die Authentifizierung vom User überprüft. Falls diese misslingt, wird ein Login-Formular (z.B login.php3) angezeigt, sonst werden aus der Datenbank die letzten Daten angefordet und in einem Formular (z.B. personform.php3) angezeigt. Wenn der User die Daten aktuallisiert hat, werden diese mit einer Art Rückkopplung wieder an die zentrale Datei übergeben. Hier werden die Daten auf ihre Korrektheit überprüft und entweder wieder im gleichen Formular wie vorher mit einer Fehlermeldung angezeigt oder weitergegeben. In einer Schlussdatei (z.B. bericht.php3) werden beim Weitergeben die aktuellsten Daten in die Datenbank geschrieben, eine Email wird verschickt und eine Meldung wird ausgegeben. Damit ist die Verarbeitung dieses Datensets abgeschlossen.

8. Ausblick

8.1 Mögliche Erweiterungen mit PHP

Beim Einbinden von PHP-Seiten in unsere Seitenstruktur hat sich gezeigt, dass der Ansatz, bei dem die Userauthentifizierung mit den Cookies verwaltet wird, viele Probleme bereitet.Wenn der User durch mehrere Verzeichnisse geführt wird, müssen die Cookies immer enstprechend angepasst werden. Dazu kommt noch, dass die User, die keine Cookies akzeptieren, eine sehr umständliche Navigation durch die Seiten in Kauf nehmen müssen. Als mögliche Lösung bieten sich die sogenannten Sessions an. Diese Lösung wird zur Zeit des Schreibens von diesem Dokument implementiert.

8.2 Vorgesehene Weiterentwicklung des MS Access Front-ends

Zur Datenmanipulation und Zusammenstellung der Formulare:

- Zum Erstellen einer neuen Teilnahme sollte nach neuen Wegen gesucht werden. Die jetztige Vorgehensweise, den Benutzer die Formulare schrittweise ausfüllen zu lassen und dabei die Fremdschlüssel zu setzen hat sich als ungünstig erwiesen. Es könnte versucht werden, die Einträge über SQL-Befehle in VBA direkt in die Datenbank zu schreiben.
- Konzept der Sammelrechnung: Die Vorgehensweise zum Erstellen einer Sammelrechnung sollte noch einmal durchdacht werden. Sollte es möglich sein Teilnehmer aus verschiedenen Kursen in eine Rechnung zusammenzunehmen wird das Setzen der Fremdschlüssel kompliziert. Im Gegensatz zum Erstellen eines (richtig verlinkten) Sammelrechnungeintrags dürfte dessen Abfrage und Darstellung als Rechung weniger Schwierigkeiten bereiten.

Um eine reibungslose Kursadministration zu ermöglichen sollten noch einige zusätzliche Berichte erstellt werden. Eine Rechnung wurde zwar implementiert, doch ist das Layout, besonders das der Adresse, noch nicht zufriedenstellend. Weiter fehlen noch Berichte welche die aktuellen Kursteilnehmerinformationen in übersichtlicher Form darstellen.
Hierfür könnte eine Abfrage erstellt werden, welche die Teilnehmereinträge nach deren Status sortiert, so dass diese dann im Bericht gruppiert werden können. Beispielsweise erscheinen dann bei einer Teilnehmerliste zuerst alle angemeldeten Personen zu einem Kurs, dann diejenigen mit einer Reservation, dann jene mit Status „abklären“, usw.
Letztlich bedarf es noch Berichten zur Erstellung von Teilnehmerbestätigungen, Mahnungen (Erste und Zweite), Erinnerungsschreiben sowie solche zur Erfassung von Abrechnungsinformationen aufs Jahresende für die einzelnen Kurse.

Da in MS Word die Möglichkeit besteht via Mail Merge[5] Daten zu importieren besteht die Möglichkeit Rechnungen, Mahnungen, Teilnahmebestätigunen, etc. in Word zu erstellen. Solche in MS Word erzeugte Dokumente können auch nachträglich noch gut bearbeitet werden.
Am besten scheint eine Kombination beiden Möglichkeiten: Um einzelne Berichte wie eine einzelne Rechung zu drucken, scheint ein MS Access-Bericht am schnellsten und

[5] In MS Word: *Tools ? Mail Merge ?* Im neuen Fenster *Create ? Form Letters ? Active Window* und *Get Data ? Open Data Source* anklicken worauf ein Fenster erscheint, in dem die MS Access Datei ausgewählt werden kann. Wurde das getan erscheint ein Fenster mit sämtlichen Tabellen und Abfragen die nun in MS Word importierte werden können.

unkompliziertesten zu sein. Soll hingegen an jeden Teilnehmer eines Kurses eine Bestätigung oder eine Rechnung verschickt werden überzeugt die MS Word – Variante:
Es könnte mit einem Klick die ganze Serie von Briefen erstellte werden. Diese kann dann bearbeitet und daraufhin vollständig gedruckt werden.

8.3 Mögliche Erweiterungen mit XML

Ein von verschiedenen Seiten im Moment erforschtes Gebiet ist das Erstellen, bzw. das Transformieren von optisch ansprechenden Texten aus strukturierten Daten. Dafür bietet sich XML, als eine hierarchisch gegliederte Metasprache, und XSL als klar definierte und einfache Programmiersprache an. Der grösste Nachteil besteht darin, dass momentan noch kein graphisches Tool, das ähnlich wie MS Word oder DTP-Programme handzuhaben ist, auf dem Markt erhältlich ist. Mit solchen Tools wäre dann der Zusammenhang zwischen einem XML-Element und der entsprechenden grafischen Darstellung ersichtlich und es könnten auf unkomplizierte Weise das entsprechende XSL-File erstellt werden.
Wie bereits aus Abbildung 3 im Kapitel 5.1 hervorgegangen war, wollten wir aus den Kursdaten im XML-Format PDF-Dokumente erstellen. Solche PDF-Elemente zu erzeugen, scheint im Moment noch aufwendig zu sein, wird aber in nächster Zukunft – unter Annahme, dass es bald erschwingliche grafische Editoren geben wird – sicherlich einfacher möglich sein.
Mit der Flexibiltät, welche die Kombination von XML und XSL gewährt, ist es auch möglich, neben den bereits erwähnten Dokumenten (HTML und PDF), noch weitere Ausgabeformate wie PDA oder WML zu erzeugen.

Da unsere XML – Dokumente nich sonderlich gross waren, haben wir kaum Probleme mit der Performance von Cocoon. Letzteres ist aber sicherlich nicht der Weisheit letzter Schluss und es darf erwartet werden, dass die Performance, was die Geschwindigkeit angeht, in der nächsten Version Verbesserungen aufweist.

9. Schlussbemerkungen

Das Projekt der neuen Kursverwaltung hat sich als sehr umfangreiche und facettenreiche Arbeit entpuppt, die uns ermöglicht hat, verschiedene Tools kennenzulernen, welche heutzutage in vielen eCommerce-Projekten eingesetzt werden. Gerade die Modularität des Systems hat zu dieser Breite oder Vielseitigkeit geführt.
Da das Schwergewicht auf der Implementation gelegen hat, wurde mit vorliegendem Bericht vor allem beabsichtigt, die Systemkomponenten - allen voran das MS Access Front-End - eingehend zu dokumentieren. Dies hauptsächlich deshalb, da aus Mangel an Zeit und Resourcen, nicht allen Wünschen nachgekommen werden konnte. Auch hoffen und erwarten wir, dass, indem wir die Weiterführung des Projektes ermöglichen, das langfristige Ziel - die Ablösung des alten Kursverwaltungsystems – in naher Zukunft realisiert werden kann.
Letztlich war auch das Teamwork ein grosser Vorteil. Wir konnten nicht nur erfahren, was es heisst, ein grösseres Projekt auf die menschlichen Resourcen aufzuteilen, sondern auch das Wesentliche vom erarbeiteten Wissen der Kollegen übernehmen.

Literaturliste

Thema	Name	Author	Verlag	ISBN-Nr.
Access	Access 97 Exam Guide	(Microsoft)	www.quecorp.com	0-7897-1507-4
	Access 97	D. Ortmann	Hanser	3-446-19013-9
	Access 2000	Bauder, Bär	Hanser	3-446-21100-4
MySQL	MySQL & mSQL	R. Yarger, G. Reese, T. King	O'Reilly	3-89721-163-7
PHP	PHP – Dynamische Webauftritte professionell realisieren	E. Schmid, C. Cartus, R. Blume	Markt & Technik	3-8272-5524-4
	PHP – kurz & gut	R. Lerdorf	O'Reilly	3-89721-225-0
Web	JavaScript – kurz & gut	D. Flanagan	O'Reilly	3-89721-208-0
XML	World Wide Web	E. Wilde	Springer	3-540-64700-7
	XML – kurz & gut	R. Eckstein	O'Reilly	3-89721-219-6