Installation einer Datenbank am Astronomischen Institut der Universität Bern

Projektbericht

Informatikprojekt

am Institut für Informatik und angewandte Mathematik
der
Universität Bern

von Simon Fankhauser

Betreut durch:

Prof. Dr. Oscar Nierstrasz
Dipl. Inf. Jean-Guy Schneider

Januar 1998

Zusammenfassung

Das astronomische Institut der Universität Bern (AIUB) beschäftigt sich seit Jahren mit Satellitengeodäsie(Erdvermessung mittels Satelliten). In diesem Zusammenhang wurde ein Software System zur Auswertung dieser Satellitendaten entwickelt, welches weltweit verkauft wird.
Der Vertrieb dieser Software bedingt einigen administrativen Aufwand. Darum wurden alle wichtigen Kunden- und Software Daten in einer (Papier) Kartei zusammengetragen. Dabei wurde immer versucht, den einzelnen Kunden individuell gerecht zu werden. Dies hat dazu geführt, dass die einzelnen Kundensätze unterschiedlich und zum Teil recht komplex geworden sind.
Das Astronomische Institut hat sich nun entschlossen, diese Daten auf einem elektronischen Medium zu speichern und zu verarbeiten. Das Ziel ist einerseits eine bessere Übersicht und andererseits soll dieses System die Administration stark erleichtern.

Das Ziel dieser Projektarbeit ist die Installation einer Datenbank, welche diesen Anforderungen möglichst gerecht wird, sowie das Übertragen der Daten von der Papierkartei in diese Datenbank.

Inhaltsverzeichnis

1. Einleitung

1.1 Einführung

Seit Jahren beschäftigt sich das Astronomische Institut mit der Auswertung von GPS-Daten (GPS = Global Positioning System). In diesem Rahmen wurde eine Software geschrieben, welche die immer immenser werdende GPS-Datenflut behandeln und auswerten kann. Diese Software, im folgenden Berner GPS Software (BGPS) genannt, ist seit 1986 käuflich und wurde bis jetzt an ca. 200 verschiedene Institutionen verkauft. Interessenten sind meist Universitäten oder Fachhochschulen sowie staatliche und private Institutionen welche an geodynamischen Problemstellungen interessiert sind.

1.2 Situationsbeschreibung

Wie wird man Kunde der Berner GPS Software?

Grundsätzlich unterscheiden wir zwischen Forschungslizenzen (Universitäten, Fachhochschulen,..) und den Kommerziellen Lizenzen (Landesvermessungsämter, Vermessungsbüros,...). Der Verkauf und damit auch die ganze Administration von Forschungslizenzen wird direkt an unserem Institut (AIUB) erledigt. Der Verkauf von kommerziellen Lizenzen wird durch eine auswärtige Partnerfirma unterstützt.

1) Forschungslizenz

Normalerweise wünscht ein Interessent zuerst ganz allgemeine, oder manchmal auch spezielle Informationen über unsere Software. Diese werden ihm dann, auf Wunsch zusammen mit einer offiziellen Offerte, zugeschickt. Entschliesst er sich zum Erwerb der Software, so schickt er uns ein entsprechendes, zusammen mit der offiziellen Offerte verschicktes Formular (sog. “Order Form”) ausgefüllt zurück. Aufgrund der Information aus dem Order Form wird Ihm dann ein Vertrag, welcher den Gebrauch der Software regelt, zugeschickt. Sobald der Interessent diesen Vertrag unterzeichnet an uns zurückgeschickt hat, wird Ihm die Software zusammen mit der Rechnung zugesandt.
Die Information des “Order Forms” bildet auch gleich die Grundlage für den Eintrag in die Datenbank.

2) Kommerzielle Lizenz

Ersehen wir aus einer Anfrage, dass ein Kunde die Software kommerziell nutzen will, so wird er an unsere Partnerfirma verwiesen. Diese kümmert sich dann um die Offerte, den Vertrag, die Zustellung der Software und der Rechnung (nicht aber um den Software Support).

Für beide Lizenzen gilt folgendes:

1. Der Preis, und somit natürlich auch die Offerte sind sowohl abhängig von der Lizenz (Forschung oder kommerziell), als teilweise auch von der Software Version (PC-DOS, UNIX/LINUX, VAX/VMS).

2. Sobald eine Institution Kunde unserer Software geworden ist, steht Ihr das folgende Material und der folgende “Software Support” zu:
    - Software Source Code, ca. 400’000 Zeilen Code
    - Dokumentation
    - Einführungskurs für 1 oder mehrere Personen hier in Bern
    - “Online” Support via Post, Fax und e-mail
    - Eintrag in die Bernese Software Mailbox Liste
    - meist gratis upgrade sobald eine neue Version bereit ist

3. Der Support wird ausschliesslich durch unser Institut erledigt.

4. Der Einführungskurs dauert normalerweise eine Woche und wird durch das AIUB organisiert.

2. Anforderungen an die Datenbank

Durch den Vertrieb der BGPS hat sich der administrative Aufwand vervielfacht. Im Durchschnitt muss jeder Kunde von der ersten Information, bis zum korrekten Bedienen der Software etwa 10 Mal kontaktiert werden. Die Datenbank soll diesen Sachverhalt möglichst gut unterstützen.

Es werden folgende Anforderungen an die Datenbank gestellt:

2.1 Allgemeine Anforderungen

2.1.1 Kapazität und Geschwindigkeit

Da diese Datenbank in den nächsten 5 bis 10 Jahren sicher nicht über die Datenmenge von ca. 10’000 Personen und 1000 Kunden wachsen wird, werden geschwindigkeitssteigernde Massnahmen nicht unbedingt nötig sein. Im Moment beträgt die Datenmenge ca. 700 Personen und 200 Kunden. Es darf mit einer Jährlichen Zuwachsrate von ca. 150-200 Personen und 50 Institutionen gerechnet werden.

2.1.2 Die Umgebung

Diejenigen Personen, welche die Datenbank voraussichtlich brauchen werden, sind Sekretariat sowie die Verantwortlichen für die Administration und den Support. Diese Personen arbeiten alle auf den Betriebssystemen Windows 95 und Windows NT und somit wird der allergrösste Teil der brieflichen Korrespondenz mit Word für Windows erledigt, welches zusammen mit dem Office Paket erhältlich ist.
Aus dieser Situation ergeben sich darum die folgenden Anforderungen an die Datenbankumgebung:

1. Die Datenbank muss mit den Betriebssystemen Windows 95 und Windows NT kompatibel sein
2. Der Datentransfer von der Datenbank ins Word für Windows soll voll unterstützt werden.
3. Die Datenbank wird auf einem Server installiert, damit von verschiedenen Arbeitsplätzen aus zugegriffen werden kann.

2.1.3 Das Interface zu Word für Windows

Das Interface besteht aus Datenbankabfragen, welche neue Tabellen erzeugen, deren Daten dann in Word für Windows Vorlagen eingeführt werden können. Das heisst insbesondere, dass diese Tabellen von Word für Windows (z. Bsp. mit dem Mail merge tool) gelesen und richtig interpretiert werden müssen.
Es werden folgende Abfragen vorgesehen:

- Flag_label: Alle personenbezogene Information. Diese Abfrage wird gebraucht, um Kontaktpersonen von Kunden mit einem als Variable eingegebenen Flag zusammenzufassen. (Bsp. Alle Kontaktpersonen von Instituten, welche eine Dokumentation erhalten). Diese

Daten können in verschiedene Word für Windows Mail merge Dateien hineingezogen werden, z. Bsp. um Adressen Etiketten, oder Bestellformulare mit a priori Information zu füllen...

- Course_information: Diese Abfrage stellt alle personenbezogene Information bezüglich eines Einführungskurses in einer Tabelle zusammen. Dies sind unter anderem: Personendaten wie Institut, Adresse..., Kursanmeldung, Hotelreservation, Ankunfts- und Abreisedatum usw. Auch diese Tabelle wird in verschiedene Word für Windows Mail merge Dokumente eingelesen, um Teilnehmerlisten, Hotelreservationslisten, Hotelreservationsbestätigungen, Adressetiketten, Teilnehmerdiplome, usw. zu erzeugen.
- Contact_persons: Personendaten, wie Institut, Adresse... von allen Kontaktpersonen. Auch diese Information wird in Word für Windows Mail merge Dokumente eingelesen und dient dazu, Mitteilungen an alle unsere Kunden zu schicken.

2.1.4 Das Benützer Interface

Es soll möglich sein, alle notwendigen Operationen in einer grafischen Oberfläche durchzuführen. Damit sollen allen Benützern tiefere “Datenbankstudien” erlassen werden. Diese Oberfläche wird also folgende Aufgaben zu lösen haben:

(a) Es soll möglich sein, Daten übersichtlich darzustellen. Dazu ist nötig, dass zwischen den einzelnen Datensätzen navigiert werden kann (z. Bsp. erster, letzter, nächster... Datensatz).
(b) Es soll möglich sein, Abfragen direkt von der Oberfläche aus zu starten
(c) Es soll möglich sein, Daten direkt von der Oberfläche aus interaktiv zu ändern. Dazu gehört einerseits das Ändern eines einzelnen Datenfeldes (Attributs), als auch das Löschen, Einfügen, Kopieren eines ganzen Datensatzes.

Diese Oberfläche soll sowohl mit Maus als auch per Tastatur (short cuts) bedient werden können.

2.1.5 Die Resourcen

Die Datenbank soll auf einem Windows NT Server laufen (PC- Pentium). Die Periferie besteht ebenfalls aus PC-Pentium Maschinen. Alle diese Maschinen sind mit mindestens 2 GB Festplattenspeicher ausgerüstet. Es kann folgende Abschätzung für den voraussichtlichen Speicherplatz gemacht werden:
Annahme: Jeder Kunde und jede Person benötigt im Schnitt ca. 1Kb Speicher (= etwa 20 Einträge à 50 Bytes). Bei einer maximalen Grösse von ca. 1000 Kunden und 10’000 Personen ergibt das einen Speicherplatzbedarf von ca. 11 Mb Dazu muss noch der Platzbedarf für die DLL und die Oberfläche (sog. Overhead) mit ca. 4 Mb berücksichtigt werden. Dies führt dann zu einem maximalen Festplattenbedarf im “Vollzustand” von ca. 15 Mb.

2.1.6 Die Dokumentation

Die Benützerdokumentation wird alle für die Bedienung, die Korrektur (oder Reparatur) und die Erweiterung nötigen Informationen enthalten. Die Beschreibung der benützten Programmiersprache, der Makrosprache, sowie der Datenbankabfragesprache (z. Bsp. SQL) ist in der einschlägigen Literatur nachzusehen. Die Benützerdokumentation wird folgende Kapitel enthalten:

1. Einleitung
2. Beschreibung Datenstruktur
3. Beschreibung der Oberfläche
4. Beispiele
5. Interaktion mit Word für Windows

2.1.7 Die Sicherheit

Da ein Datenverlust gravierende Folgen haben kann, wird darauf wert gelegt, dass die Datenbank über einen Wiederherstellungsmechanismus verfügt. Dies ist z. Bsp. durch die Verwendung und Verwaltung von sogenannten Logfiles möglich, welche von den meisten Datenbankherstellern berücksichtigt werden.
Zudem wird von der Datenbank regelmässig und automatisch eine Sicherheitskopie erstellt werden müssen. Diese soll durch das Windows NT Backupsystem übernommen werden, worum sich der Systemadministrator kümmern wird.

2.1.8 Spätere mögliche Erweiterungen

Die Datenbank unterstützt im Moment nur den Verkauf, Support, usw. bezüglich der BGPS. Es zeichnet sich aber bereits jetzt ab, dass ein zweiter Forschungszweig unseres Institutes, welcher auch Software entwickelt, an dieser Datenbank interessiert sein wird. Dies gilt es bei der Installation der Datenbank und insbesondere bei der Aufteilung der Daten in Tabellen und Beziehungen zu berücksichtigen.

2.2 Problemspezifische Anforderungen

2.2.1 Die Daten

Die Datenbank soll alle Software-, Personen- und Kunden-relevanten Daten enthalten, welche sowohl für die Administration, sowie auch für den Softwaresupport nötig sind.
Im Folgenden sind diese Daten aufgelistet:
Kundendaten:

- Institut, Universität, Stadt, Land, Kontaktsprache, Kundentyp (kommerziell, Universität).
- Wann und welche Typen von Offerte hat er erhalten, und (wann) hat er darauf geantwortet?
- Wann und welche Typen von Vertrag hat er erhalten, und (wann) hat er sie unterschrieben zurückgeschickt?
- Wer hat den Vertrag unterschrieben?

- Wann und welche Typen von Rechnung hat er erhalten, und (wann) hat er sie beglichen?
- Wann hat er welche Versionen der Software erhalten?
- Hat dieser Kunde schon Leute an den Einführungskurs geschickt, wenn ja, welche?
- Hat dieser Kunde die neuen Dokumentationen erhalten, hat er einen gratis Upgrade der Software zugute, darf er zusätzliche Leute an den Einführungskurs schicken....?
- Wer ist Kontaktperson (evtl. mehrere)?
- Gibt es über diesen Kunden wichtige Informationen, welche in nichtstrukturierter Form in der Datenbank abgespeichert werden sollen (Memo)?

Personendaten:
- Anrede, Vorname, Name, Adresse, Postleitzahl, Ort, Land Telefon, Telefax, E-mail.
- Gehört diese Person zu einem Kunden, wenn ja, zu welchem?
- Wie gehört die Person zu einem Institut: als Kontaktperson, verantwortliche Person (unterschreibt Vertrag), wünscht sie einen Eintrag in die BGPS-Mailliste...?
- An welchem Kurs war die Person?
- Gibt es über diese Person wichtige Informationen, welche in nichtstrukturierter Form in der Datenbank abgespeichert werden sollen (Memo)?

Softwaredaten:
- Wann wurde welche Softwareversion hergestellt?
- Auf welchen Platformen läuft die Software?

Kursdaten:
- Wann findet der Einführungskurs statt (von, bis)?
- Von welchem Grad ist er (Basic, Advanced)?
- Welche Leute besuchen ihn?
- Sind die Einladungen an alle interessierten Institutionen verschickt worden?
- (Wann) wurden die Personen angemeldet?
- Wann kommt die Person in Bern an, wann reist sie wieder zurück?
- In welchem Hotel wurde ein Zimmer reserviert, welchen Zimmertyp (Einzel, Doppel), von wann bis wann und für welche Person?

Hoteldaten:
- Namen, Adresse, Postleitzahl, Ort, Land, Telefon, Telefax

2.2.2 Die Operationen

Die Datenbank soll alle administrativen Schritte, welche zum Erwerb und beim Support durchlaufen werden müssen, unterstützen. Dies sind stark zusammengefasst folgende:
1. Neueintragen, ändern, löschen von allen Daten
2. Suchen nach Personen und Kunden.
3. Komplette Kunden und Personendaten übersichtlich darstellen.
4. Daten wie Adressen, Kursinformationen in Tabellen bereitstellen, welche von Word für Windows gelesen werden können.

3. Grobdesign

In diesem Abschnitt soll ein Überblick über die groben Vorstellungen der verschiedenen Datenbankteile gegeben werden. So werden wir z. Bsp. ein kommerzielles Produkt auswählen, einige Überlegungen zur Oberfläche vornehmen, so wie eine Grobstruktur des Menüsystems entwickeln. Die detaillierten Planungsschritte werden dann im nächsten Abschnitt dargelegt.

3.1 Selektion der Datenbank

Ausgehend von den Benützeranforderungen kommen verschiedene Produkte in Frage. Die hauptsächlichen Bedingungen an die Datenbank sind folgende:
- "Angenehme" und übersichtliche Oberfläche, mit Maus und Tastatur bedienbar.
- Muss alle Daten verwalten können.
- Interface zu Word für Windows
- Muss auf Windows NT laufen.

Es wäre also im Prinzip sogar möglich, von Grund auf eine neue Datenbank zu entwickeln.

Aus der Sicht der Softwareentwickler gibt es aber noch einige zusätzliche Überlegungen:
- Die Kompatibilität zu Microsoft Word wird am besten durch eine Datenbank derselben Firma gewährleistet.
- Um mit einer gewissen Sicherheit auch bei zukünftigen Versionen von Microsoft Word kompatibel zu sein, sollte ein einigermassen bewährtes Produkt, mit einer gewissen "Überlebenschance auf dem Markt" gewählt werden.
- Dieses Produkt sollte Entwicklungshilfen beinhalten, um den Programmierer möglichst stark von der Routinearbeit zu entlasten. So kann er seine Zeit für Spezialwünsche einsetzen.

Es wurden mit 2 verschiedenen Produkten Erfahrungen gesammelt:
Als am astronomischen Institut 1992 zum ersten Mal ein Versuch gestartet wurde, eine Datenbank zu installieren, wählte man Dbase. Es hat sich dann gezeigt, dass dieses Produkt relativ schwerfällig zu bedienen war und zudem auch die Zusammenarbeit mit Word für Windows nur mässig befriedigend ausfiel. Dies führte dazu, dass diese Datenbank nie benützt wurde.
Im Jahre 1996 wurde dann ein zweiter Versuch unternommen, mit Access von Microsoft, welches die oben erwähnten Bedingungen erfüllt. Die Testläufe mit Access fielen recht gut aus und so entschied man sich dann definitiv für Access.

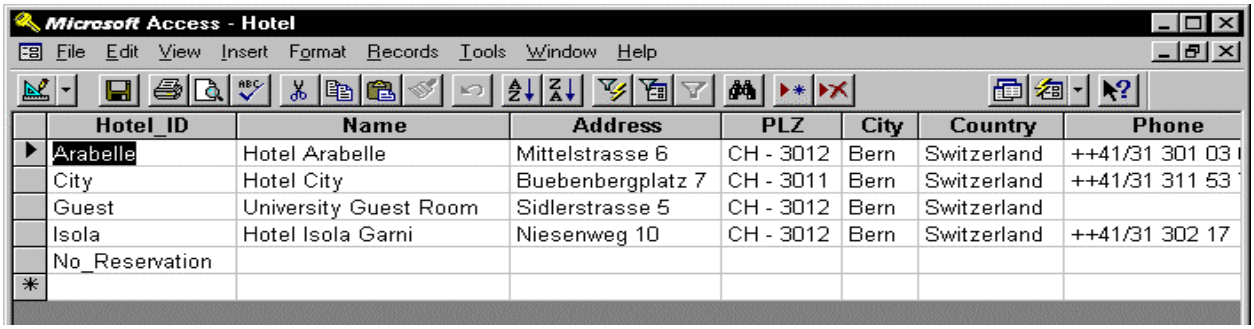
3.2 Definition der Oberfläche

In diesem Abschnitt geben wir einen kurzen Überblick über die Möglichkeiten der Datendarstellung in Access. In einem 2. Teil definieren wir dann die Struktur der Oberfläche und geben einen Eindruck, wie diese Oberfläche aussehen wird.

3.2.1 Datendarstellung in Access

Access bietet die Möglichkeit, auf zwei verschiedene Darstellungsformen von Tabellen zuzugreifen, nämlich die “Data sheet view” und die “Form view”.

Die “Data sheet view” zeigt die Daten einer Tabelle/Relation in Tabellenform. Dies heisst, dass im Prinzip alle Datensätze zusammen aufgelistet werden. Die darstellerischen Freiheitsgrade sind aufs Minimale beschränkt und erlauben nur ein Anpassen der Zellenbreite. Ein Beispiel dieser Darstellung zeigt Darstellung 1.



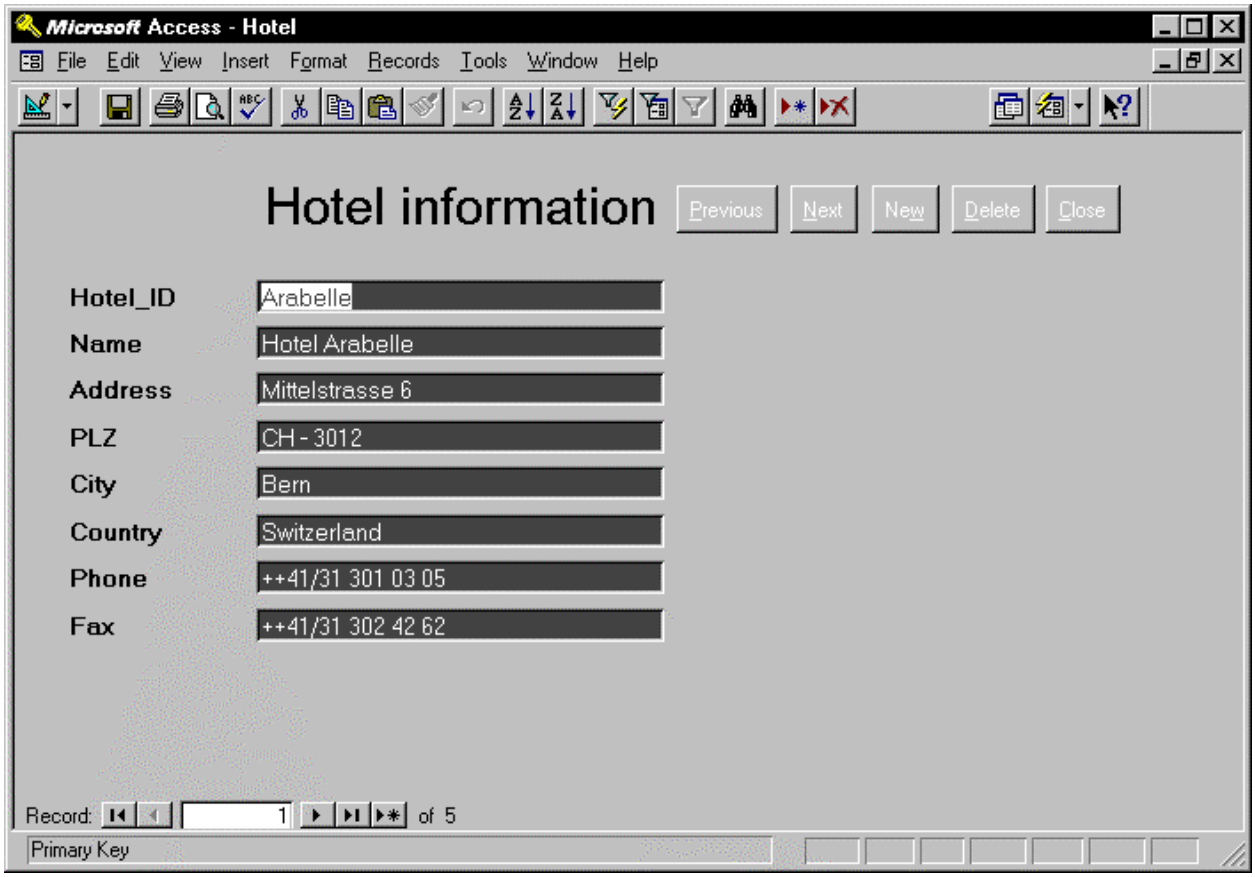
Hotel_ID	Name	Address	PLZ	City	Country	Phone
Arabelle	Hotel Arabelle	Mittelstrasse 6	CH - 3012	Bern	Switzerland	++41/31 301 03
City	Hotel City	Buebenbergplatz 7	CH - 3011	Bern	Switzerland	++41/31 311 53
Guest	University Guest Room	Sidlerstrasse 5	CH - 3012	Bern	Switzerland	
Isola	Hotel Isola Garni	Niesenweg 10	CH - 3012	Bern	Switzerland	++41/31 302 17
No_Reservation						

Darstellung 1:
Beispiel einer “Data sheet view”

Die zweite Möglichkeit, Daten auf dem Bildschirm zu präsentieren ist die “Form view”. Diese Ansicht der Daten ist mit einem Formular vergleichbar. Im Prinzip wird nur ein Datensatz dargestellt, dafür sind verschiedene darstellerische Varianten möglich:

- Farben
- Felder einzeln und beliebig im Fenster positionierbar.
- Einfügen von Buttons

Darstellung 2 zeigt dieselbe Tabelle wie Darstellung 1 in “Form view”.



Darstellung 2
zeigt das Formular “Hotel information”. Die Eingabefelder sind dunkelgrau gekennzeichnet mit weisser Schrift. Die Felder oben rechts, beschriftet mit “Previous”, “Next”, “New”, “Delete” und “Close” bezeichnen Bereiche, welche sowohl mit der Maus als auch per Tastatur aktiviert werden können (per Maus: anklicken, per Tastatur durch den short cut “alt” gleichzeitig mit dem unterstrichenen Character: z. Bsp. Alt P für “Previous”). Mit diesen Tasten lassen sich z. Bsp. folgende Aktivitäten ausführen:

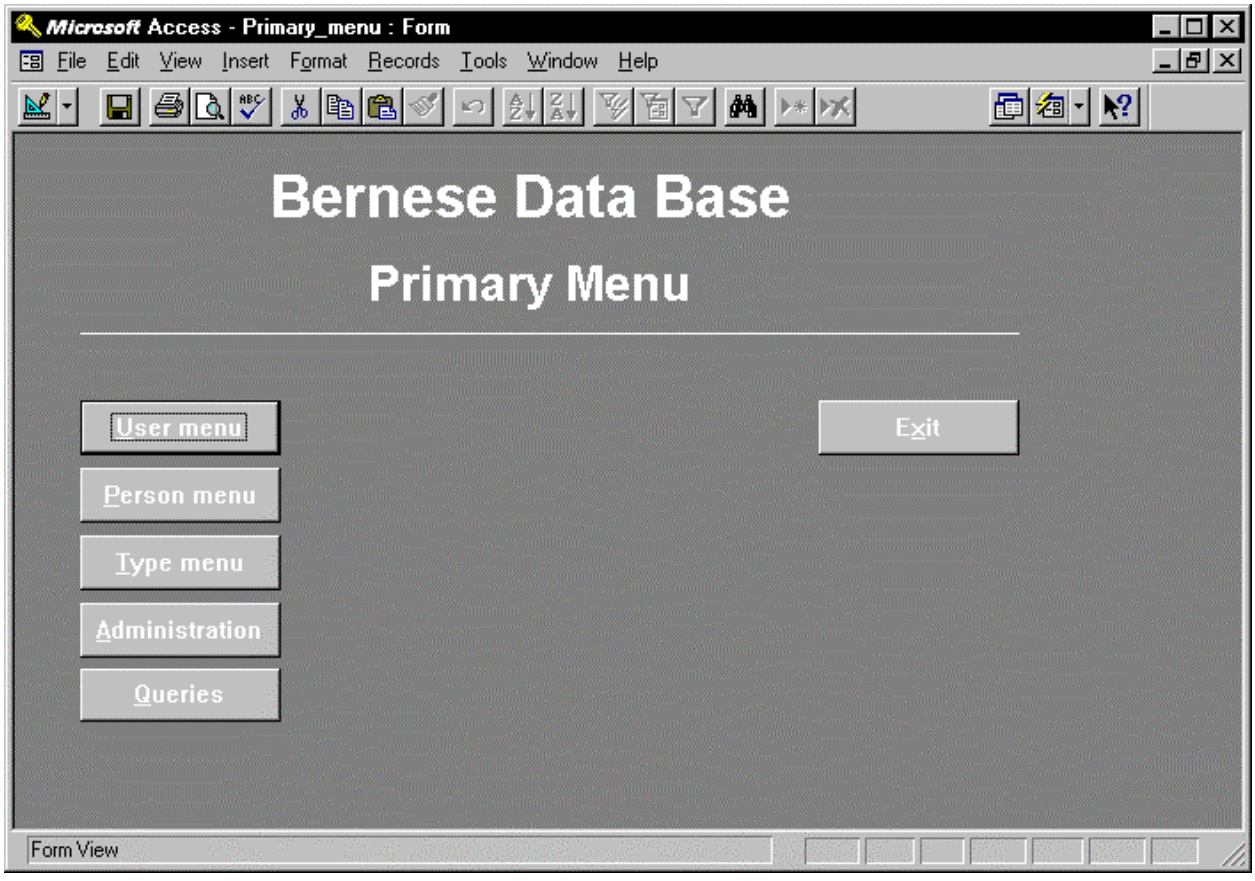
“Previous”: Zeige den vorhergehenden Datensatz.
“Next”: Zeige den nachfolgenden Datensatz.
“New”: Kreiere wenn noch nicht vorhanden diese Tabelle und füge einen neuen Datensatz ein.
“Delete”: Lösche den aktuellen Datensatz.
“Close”: Schliesse diese Ansicht.

Das Ändern von Daten geschieht direkt durch Ändern der Angaben in den entsprechenden Feldern.

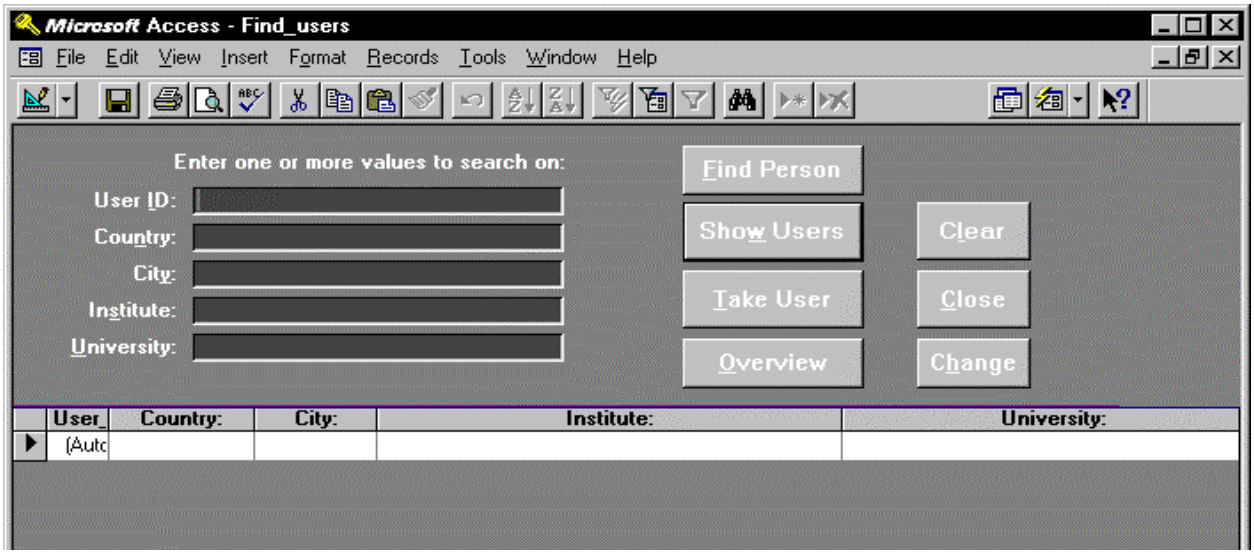
3.2.2 Struktur der Oberfläche

Die Aufgabe der Oberfläche ist es, den Benützer menüartig durch die Datenbank zu führen, Daten darzustellen und abzuändern. Darum müssen für alle in der Datenbank vorgesehenen Operationen entsprechende Formulare entwickelt werden.
Die Darstellungen 2, 3 und 4 zeigen Beispiele dieser Formulare, welche je nach Aufgabe wiefolgt klassifiziert werden können:

- Formulare welche den Benützer an die richtige Stelle der Datenbank manövrieren, sog. Menüformulare. Darstellung 3 zeigt das Hauptmenü-Formular
- Formulare welche dem Benützer eine effiziente Suche nach Daten und eine übersichtliche Darstellung der Daten erlauben, sog. Operationsformulare. Darstellung 4 Zeigt das “Search_User” Formular.
- Formulare welche die Datenbearbeitung erlauben, sog. Bearbeitungsformulare (siehe Darstellung 2).



Darstellung 3
zeigt das Hauptformular, welches beim Aufstarten der Datenbank automatisch gestartet wird. Es zeigt die verschiedenen Bereiche der Datenbank. Durch Anklicken mit der Maus oder mit “alt zusammen mit dem unterstrichenen Character” lassen sich diese Teile auswählen. In diesem Fall sind das “User Menü” , “Personen Menü”, “Type Menü”, “Administration” und “Queries”.



Darstellung 4
zeigt das Find_user Formular. Dieses Formular ermöglicht die Suche nach Kunden, welche die in den linken Feldern angegebenen Bedingungen erfüllen. Das Feld “Show Users” listet dann alle gefundenen Kunden im unteren Teil des Formulars auf. Diese Daten können dann mit den auf der rechten Seite lokalisierten Feldern dargestellt oder verarbeitet werden.

3.3 Übersicht über das Menüsystem

Das Menüsystem soll im Prinzip baumartig aufgebaut werden. Dennoch sollte es Möglich sein, durch eine begrenzte Anzahl Ausnahmen einige praktische Verbindungen herzustellen. Ausgehend von einem “Primary Menü” sollten in Anbetracht der gestellten Anforderungen in etwa folgende Untermenüs zu finden sein:

- Menüzweig, der sich mit den Personendaten befasst
- Menüzweig, der sich mit den Kunden (=Institutionen) befasst
- Menüzweig, der die Neueinfügung von Versions-, Offerten-, Vertrags-, Rechnungstypen unterstützt
- Menüzweig, der die Administration des Einführungskurses unterstützt
- Suchmöglichkeiten

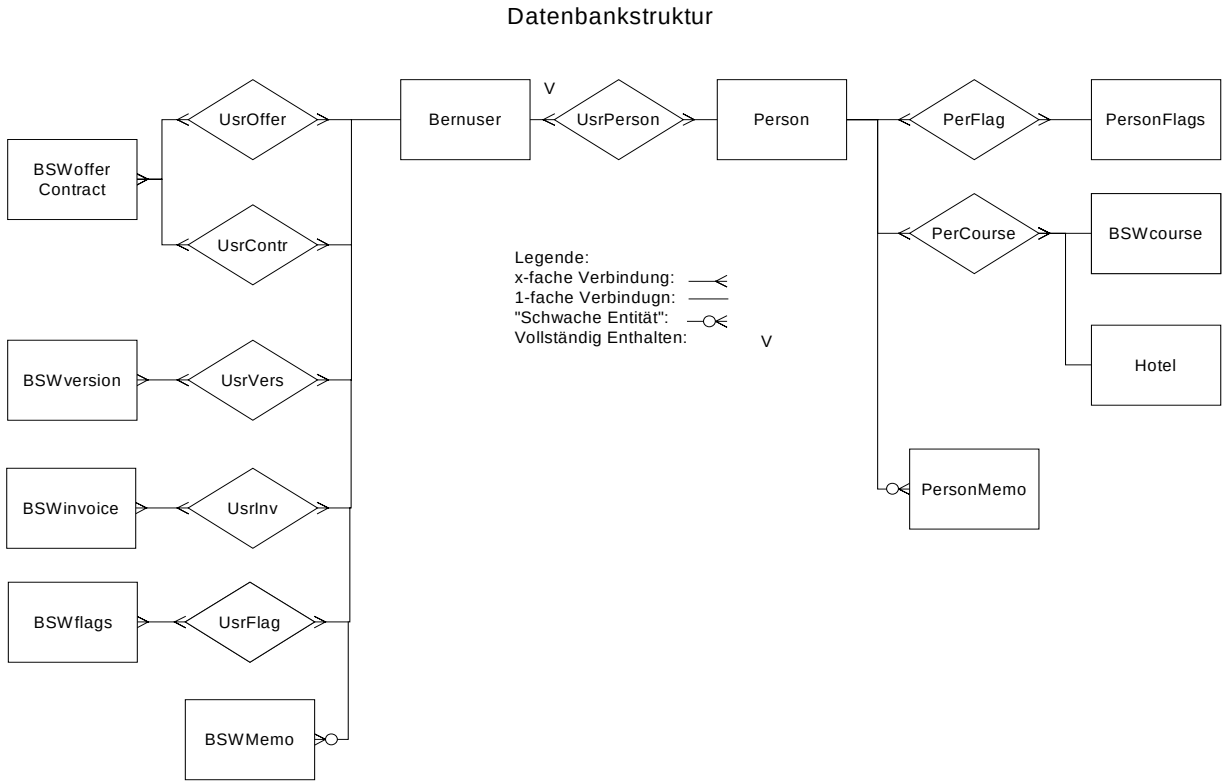
In Kapitel 4 (Feindesign) wird dann das ganze Menüsystem noch detailliert entwickelt.

4. Feindesign

In diesem Abschnitt entwickeln wir die eigentliche Struktur der Datenbank soweit, dass anschliessend mit der Codierung begonnen werden kann. Es gilt also, die Datenstruktur die Deklaration der Tabellen, die Oberfläche sowie die Struktur des Menüsystems detailliert zu entwickeln .

4.1 Organisation der Daten in Tabellen und Relationen

Um den in Kapitel 2 aufgelisteten Anforderungen gerecht zu werden wurden die Daten so in Tabellen und Relationen aufgeteilt, wie sie in Grafik 1 und Tabelle 1 dargestellt sind.
Grafik 1 zeigt das Tabellen- und Relationendiagramm (ohne die Attribute)[1]. In diesem Diagramm ist ersichtlich, wie die einzelnen Tabellen und Relationen zusammengesetzt sind. Die Struktur der Datenbank kann wiefolgt beschrieben werden:
Die Software- und Kundenspezifischen Daten wurden klar von den Personenbezogenen Daten getrennt. Bindeglied ist die Relation UsrPerson, welche jedem Institut mindestens eine Person zuweist. Diese Aufteilung ermöglicht eine eventuelle spätere Erweiterung der Datenbank um ein weiteres Produkt, wie z. Bsp. eine weitere Softwareentwicklung.



[1] Die Darstellung in Grafik 1 entspricht nicht der üblichen Notation. Doch leider war diese mit dem Program „ABC Flow-charter“ nicht erzeugbar.

Die Software spezifischen Daten (Zweig von Bernuser) sind folgendermassen unterteilt:

(a) BSWofferContract: Diese Tabelle enthält alle möglichen Offerten und Vertragstypen. Da gilt, dass für jeden Offertentyp genau ein Vertragstyp besteht, wurden diese in derselben Tabelle zusammengefasst. Die Relationen UsrOffer und UsrContr weisen den Kunden 0..n Offerten und Verträge zu. Da nicht für alle Offerten ein Vertrag abgeschlossen wird, mussten für die Relationen Offerte und Vertrag getrennt werden.

(b) BSWversion: Diese Tabelle enthält alle Bezeichnungen für die verschiedenen Software Versionen (z. Bsp. V34UNIX= Version 3.4 für UNIX Platform). Die Relation UsrVers ordnet dann jedem Kunden 0..n Software Versionen zu. Ein Kunde welcher zum Beispiel nie auf eine Offerte geantwortet hat, ist in unserer Datenbank eingetragen, hat aber nie einen Vertrag unterschrieben oder eine Software Version erhalten.

(c) BSWinvoice: Diese Tabelle enthält alle Rechnungstypen. Diese sind nicht immer mit der Offerte identisch. Es gibt zum Beispiel Kunden, welche den Betrag für die Software von verschiedenen Konten zahlen müssen und darum auch 3 entsprechende Rechnungen benötigen.

(d) BSWFlags: In dieser Tabelle sind alle möglichen Flagtypen zusammengefasst. Jeder Kunde hat 0..n dieser Flags. Diese Flags können z. Bsp. dazu benützt werden, um alle Kunden zu kennzeichnen, welche die letzte Version unserer Software Dokumentation zugeschickt erhalten haben.

(e) BSWMemo: Diese Tabelle enthält kundenrelevante Information, welche nicht direkt in diese Datenbankstruktur integriert werden muss, wie z. Bsp. warum hat ein Kunde auf eine Offerte nicht reagiert, oder welche Zwischenfirma hat den Softwarekauf organisiert. Diese Tabelle ist eine sog. “Week entity”, dass heisst, dass jeder Eintrag dieser Tabelle genau einem Kunden zugeordnet ist.

Die **personenbezogenen Daten** sind wiefolgt unterteilt:

(a) PersonFlags: Diese Tabelle enthält, analog zu BSWflags, alle möglichen personenbezogenen Flags. Die Relation PerFlag ordnet entsprechend einer Person 0..n Flags zu. Diese werden zum Beispiel gebraucht, um eine Person als Kontaktperson zu kennzeichnen.

(b) BSWcourse: Diese Tabelle fasst alle Information über die Einführungskurse zusammen, wie zum Beispiel Datum des Beginns und des Endes.

(c) Hotel: In dieser Tabelle sind alle möglichen Hotels zusammengefasst, in welchen wir unsere Kunden unterbringen. Die Relation PerCourse ordnet einer Person einen Einführungskurs und ein Hotel zu.

(d) PersonMemo: In dieser Tabelle sind alle personenrelevanten Informationen enthalten, welche wichtig genug sind um in der Datenbank eingetragen zu werden, aber zu speziell als dass es sich lohnen würde eine weitere Struktur zu generieren. Dieses Memo wird gebraucht, um z. Bsp. zu vermerken, dass eine Person an ein anderes Institut gewechselt hat.

In der Tabelle 1 sind diese Tabellen und Relationen noch einmal zusammengefasst und um die Attribute erweitert. Eine ganz präzise Definition der Datenstruktur wird im nächsten Abschnitt gezeigt.

Tabellenbezeichnung	**Funktion (Tabelle, Beziehung (Relation))**	**Enthaltene Daten (Attribute)**
Bernuser	Tabelle	Identifikationsnummer, Institut, Universität/Institution, Stadt, Land, Kontaktsprache, Typ; Typ= Universität oder Kommerziell
BSWOfferContract	Tabelle	Typ, Preis, Beschreibung
BSWVersion	Tabelle	Typ, Beschreibung
BSWInvoice	Tabelle	Typ, Preis, Beschreibung
BSWFlags	Tabelle	Typ, Beschreibung
Person	Tabelle	Identifikationsnummer, Anrede, Titel, Vorname, Name, Institut, Universität, Adresse, Postleitzahl, Ort, Land, Telefon, Telefax, E-mail
PersonFlags	Tabelle	Typ, Beschreibung
BSWCourse	Tabelle	Bezeichnung, Beginn, Ende, Beschreibung
Hotel	Tabelle	Bezeichnung, Name, Adresse, Postleitzahl, Ort, Land, Telefon, Telefax
UsrOffer: Bernuser ↔ BSWofferContract	Beziehung	Benützeridentifikation, Offertentyp, Sendedatum, Bestelldatum
UsrContr: Bernuser ↔ BSWOfferContract	Beziehung	Benützeridentifikation, Vertragstyp, Sendedatum, Rücksendedatum
UsrVers: Bernuser ↔ BSWVersion	Beziehung	Benützeridentifikation, Software Typ, Sendedatum
UsrInv: Bernuser ↔ BSWInvoice	Beziehung	Benützeridentifikation, Rechnungstyp, Sendedatum, Bezahldatum
UsrFlag Bernuser ↔ BSWflags	Beziehung	Benützeridentifikation, Flagtyp, Datum
BSWMemo Bernuser ↔ BSWMemo (week entity)	Beziehung	Memoidentifikation, Benützeridentifikation, Datum, Memoinformation
UsrPerson: Bernuser ↔ Person	Beziehung	Benützeridentifikation, Personenidentifikation
PerFlag: Person ↔ PersonFlags	Beziehung	Personenidentifikation, Flagtyp
PerCourse: Person ↔ BSWcourse, Hotel	Beziehung	Personenidentifikation, Kursbezeichnung, Datum der Einladung, Datum der Anmeldung, Hotelbezeichnung, Ankunftsdatum, Abreisedatum, Hotelzimmertyp (=einzel, doppel))
PersonMemo:	Beziehung	Memoidentifikation,

Person ↔ PersonMemo (week entity)		Personenidentifikation, Datum, Memoinformation

Tabelle 1:
Daten, welche die Datenbank für alle Benützer enthalten soll, aufgelistet nach Tabellen und Beziehungen.

4.2 Definition der Tabellen und Relationen

In diesem Abschnitt werden die Tabellen und Relationen exakt definiert. Diese Definition beinhaltet Namen, Attribute und deren Deklaration und Schlüsselattribut der Tabelle/Relation.

Tabellen:

Name	Attribut	Deklaration	Schlüssel
Bernuser	User_ID	Number	X
	Country	String 30	
	City	String 30	
	Institute	String 50	
	University	String 50	
	Language	String 1	
	Customer_of	String 10	

Name	Attribut	Deklaration	Schlüssel
BSWOfCo	OC_ID	String 10	X
	Price	Fr. xx,xxx.00	
	Description	Memo	

Name	Attribut	Deklaration	Schlüssel
BSWvers	SW_type	String 10	X
	Description	Memo	

Name	Attribut	Deklaration	Schlüssel
BSWinvoice	Invoice_typ	String 10	X
	Price	Fr. xx,xxx.00	
	Description	Memo	

Name	Attribut	Deklaration	Schlüssel
BSWflags	Flag_typ	String 10	X
	Description	Memo	

Name	Attribut	Deklaration	Schlüssel
Person	Person_ID	Number	X
	Calling	String 5	
	Title	String 15	
	Given_name	String 20	
	Name	String 30	
	Institute	String 50	
	University	String 50	
	Street_POBox	String 30	
	Zip-Code1	String 15	
	City	String 30	
	Zip-Code2	String 20	
	Country	String 30	
	Country_code	String 15	
	Phone	String 30	
	Fax	String 30	
	E-mail	String 40	

Name	Attribut	Deklaration	Schlüssel
Pflags	Flag_typ	String 20	X
	Description	Memo	

Name	Attribut	Deklaration	Schlüssel
BSWcourse	Course_ID	String 10	X
	Beginning_at	Date/Time	
	Ending_at	Date/time	
	Description	Memo	

Name	Attribut	Deklaration	Schlüssel
Hotel	Hotel_ID	String 50	X
	Name	String 50	
	Address	String 50	
	PLZ	String 15	
	City	String 50	
	Country	String 50	
	Phone	String 50	
	Fax	String 50	

Relationen:

Name	Attribut	Deklaration	Schlüssel
UsrOffer	User_ID	Number	X
	OC_ID	String 10	X
	Send_offer	Date/Time	X
	Rec_offer	Date/Time	

Name	Attribut	Deklaration	Schlüssel
UsrContr	User_ID	Number	X
	OC_ID	String 10	X
	Send_contr	Date/Time	X
	Rec_contr	Date/Time	

Name	Attribut	Deklaration	Schlüssel
UsrInv	User_ID	Number	X
	Invoice_typ	String 10	X
	Send_inv	Date/Time	X
	Rec_inv	Date/Time	

Name	Attribut	Deklaration	Schlüssel
UsrFlag	User_ID	Number	X
	Flag_type	String 10	X
	Flag_date	Date/Time	

Name	Attribut	Deklaration	Schlüssel
UsrPerson	User_ID	Number	X
	Person_ID	Number	X

Name	Attribut	Deklaration	Schlüssel
PerFlag	Person_ID	Number	X
	Flag_typ	String 20	X

Name	Attribut	Deklaration	Schlüssel
PerCourse	Person_ID	Number	X
	Course_ID	String 10	X
	Send_invit	Date/Time	
	Rec_invit	Date/Time	
	Hotel_name	String 20	
	Arrival	Date/Time	
	Departure	Date/Time	
	Room	String 50	

Name	Attribut	Deklaration	Schlüssel
BSWMemo	Memo_ID	Number	X
	User_ID	Number	
	Date	Date/Time	
	Memo	Memo	

Name	Attribut	Deklaration	Schlüssel
PerMemo	Memo_ID	Number	X
	Person_ID	Number	

	Date	Date/Time	
	Memo	Memo	

4.3 Das Menüsystem

Das Menüsystem ist im Prinzip baumartig aufgebaut, mit einigen kleinen praktischen Ausnahmen. Grafik 2 zeigt die Übersicht.

Im **Primary Menü** hat der Benützer die Wahl ob er Kunden-, Personen- oder Typen-Daten bearbeiten muss, oder ob Administration oder Abfragen gemacht werden müssen (siehe Darstellung 3). Im Abschnitt **User Menü** hat der Benützer die Möglichkeit sämtliche kundenrelevanten Daten darzustellen oder zu bearbeiten. Dieses Menü bringt als nächstes das Search_user Formular auf den Bildschirm (Siehe Darstellung 4). Dieses Formular unterstützt alle nötigen Aktionen. Der Abschnitt **Person Menü** ist entsprechend aufgebaut, für die Bearbeitung von personenbezogenen Daten. Der Abschnitt **Typ Menü** erlaubt die Bearbeitung der verschiedenen Kurs-, Offerten-, Vertrags-, Rechnungs-, Software- und Flagtypen. Diese können eingefügt, gelöscht und geändert werden. Im Abschnitt **Administration** wird in der ersten Phase nur der Unterabschnitt **Course** zu finden sein, der wiederum in **Participants** und **Hotel** unterteilt ist. Dieser Abschnitt erlaubt das Eintragen von Personen in den Einführungskurs, sowie der für die Hotelreservation nötigen Informationen wie Ankunftszeit, Abfahrtszeit, Hotel, Zimmergrösse... Der Abschnitt **Queries** soll es erlauben, bestimmte häufig gebrauchte Abfragen zu starten. In einer ersten Phase wird er nur die beiden Abfragen **Contact Person** und **NO_Contact Person** enthalten, welche alle Kontaktpersonen resp. alle Kunden ohne Kontaktpersonen auflisten.

Grafik 2
Zeigt die Übersicht über das Menüsystem

4.4 Interface zu Word für Windows

Die in Abschnitt 2.1.3 erwähnten Abfragen erzeugen folgende Tabellen:

Name	Attribut	Aus Tabelle/Relation
Flag_label	Calling	Person
	Title	Person
	Given_name	Person
	Name	Person
	Institute	Person
	University	Person
	Street_POBox	Person
	ZipCode1	Person
	City	Person
	Country	Person
	Country_code	Person
	ZipCode2	Person
	Flag_type	UsrFlag

Die Tabelle **Flag_label** kann z. Bsp. mit folgendem SQL Kommando erzeugt werden:

```
SELECT DISTINCTROW Person.Calling, Person.Title, Person.Given_name,
Person.Name, Person.Institute, Person.University, Person.Street_POBox, Person.[Zip-
code1], Person.City, Person.Country, Person.Country_code, Person.[Zip-code2],
User_flags.Flag_type
FROM (Bernuser INNER JOIN User_flags ON Bernuser.User_ID = User_flags.User_ID)
INNER JOIN ((Person INNER JOIN UserPerson ON Person.Person_ID =
UserPerson.Person_ID) INNER JOIN PersonFlag ON Person.Person_ID =
PersonFlag.Person_ID) ON Bernuser.User_ID = UserPerson.User_ID
WHERE (((User_flags.Flag_type)=[Enter_flag]) AND
((PersonFlag.Flag_typ)="contact"));
```

Name	Attribut	Aus Tabelle/Relation
Course_information	Person_ID	Person
	Calling	Person
	Title	Person
	Given_name	Person
	Name	Person
	Institute	Person
	University	Person
	Street_POBox	Person
	ZipCode1	Person
	City	Person
	ZipCode2	Person
	Country	Person
	Country_code	Person
	Phone	Person

Fax	Person
E-mail	Person
Course_ID	PerCourse
Send_invit	PerCourse
Rec_invit	PerCourse
Hotel_ID	Hotel
Name	Hotel
Address	Hotel
PLZ	Hotel
City	Hotel
Country	Hotel
Phone	Hotel
Fax	Hotel
Arrival	PerCourse
Departure	PerCourse
Room	PerCourse
Customer_of	Bernuser
User_Id	Bernuser

Die Tabelle **Course_information** kann z. Bsp. mit folgendem SQL Kommando erzeugt werden:

```
SELECT DISTINCTROW PersonCourse.Person_ID, Person.Calling, Person.Title,
Person.Given_name, Person.Name, Person.Institute, Person.University,
Person.Street_POBox, Person.[Zip-code1], Person.City, Person.[Zip-code2],
Person.Country, Person.Country_code, Person.Phone, Person.Fax, Person.[E-mail],
PersonCourse.Course_ID, PersonCourse.Send_invit, PersonCourse.Rec_invit,
Hotel.Hotel_ID, Hotel.name, Hotel.Address, Hotel.PLZ, Hotel.City, Hotel.Country,
Hotel.Phone, Hotel.Fax, PersonCourse.Arrival, PersonCourse.Departure,
PersonCourse.Room, Bernuser.Customer_of, Bernuser.User_ID
FROM Bernuser INNER JOIN ((Person INNER JOIN UserPerson ON Person.Person_ID
= UserPerson.Person_ID) INNER JOIN (Hotel INNER JOIN PersonCourse ON
Hotel.Hotel_ID = PersonCourse.Hotel_name) ON Person.Person_ID =
PersonCourse.Person_ID) ON Bernuser.User_ID = UserPerson.User_ID
WHERE (((PersonCourse.Course_ID)=[ID of the Course]))
ORDER BY Person.Name, Person.Given_name, Person.Country;
```

Name	Attribut	Aus Tabelle/Relation
Contact_persons	Person_ID	Person
	Calling	Person
	Title	Person
	Given_name	Person
	Name	Person
	Institute	Person
	University	Person
	Street_POBox	Person
	ZipCode1	Person
	City	Person
	ZipCode2	Person
	Country	Person
	Country_code	Person
	Phone	Person
	Fax	Person
	User_ID	Bernuser

Die Tabelle **Contact_persons** kann z. Bsp. mit folgendem SQL Kommando erzeugt werden:

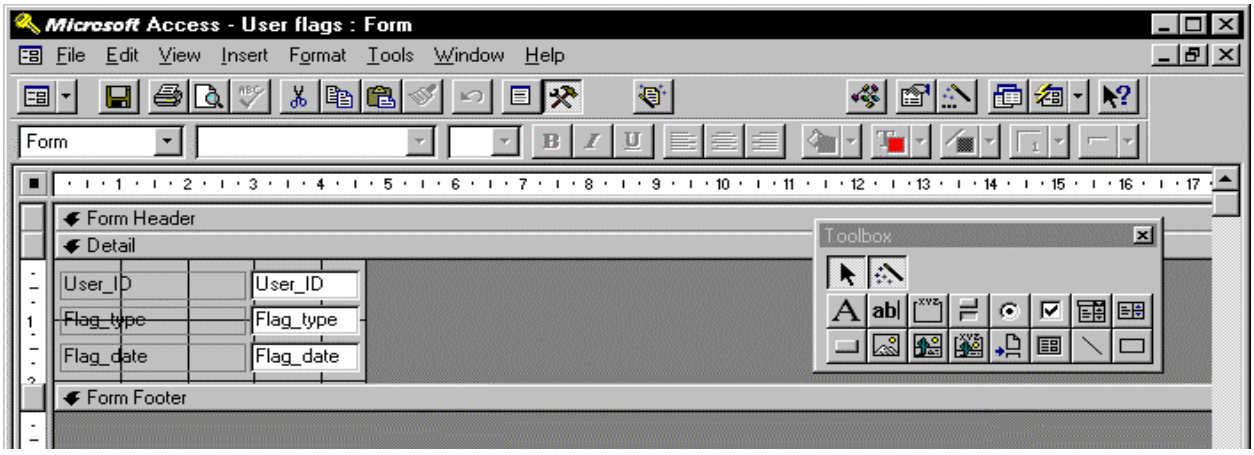
```
SELECT DISTINCTROW Person.Person_ID, Person.Calling, Person.Title,
Person.Given_name, Person.Name, Person.Institute, Person.University,
Person.Street_POBox, Person.[Zip-code1], Person.City, Person.[Zip-code2],
Person.Country, Person.Country_code, Person.Phone, Person.Fax, Person.[E-mail],
Bernuser.User_ID
FROM Bernuser INNER JOIN ((Person INNER JOIN UserPerson ON Person.Person_ID
= UserPerson.Person_ID) INNER JOIN PersonFlag ON Person.Person_ID =
PersonFlag.Person_ID) ON Bernuser.User_ID = UserPerson.User_ID
WHERE (((PersonFlag.Flag_typ)="contact"));
```

5. Implementation

5.1 Einige Bemerkungen zu den Entwicklungsmöglichkeiten in Access

Im Lieferumfang von Access ist eine ausführliche Entwicklungsumgebung inbegriffen, welche die Erzeugung von Tabellen/Relationen, Abfragen, Formularen und mausaktiven Feldern stark unterstützt. Zudem bietet diese Umgebung zwei weitere Möglichkeiten an, welche zur Automatisation und zur Erzeugung komplexerer Abfragen sehr nützlich sind: Die Makrosprache und Visual BASIC.
Wo immer es möglich sein wird, wird man versuchen, den automatischen Generierungsmechanismen den Vorzug zu lassen und erst dort, wo deren Möglichkeiten überschritten werden, von der Makrosprache Gebrauch zu machen. Dort wo dann die Makrosprache nicht mehr genügt, wird Visual BASIC eingesetzt werden. Dieses Vorgehen hat den Vorteil, dass bei einer späteren Erweiterung ebenso vorgegangen werden kann und somit ein gewisser, in erster Linie durch Access bestimmter Standard eingehalten wird. Die folgenden Beispiele geben eine Idee, wozu diese Entwicklungstools zu brauchen sind. Eine vollständige Beschreibung dieser Entwicklungstools sowie der Makrosprache und Visual BASIC kann z. Bsp. dem Handbuch und “Dem grossen Buch zur ACCESS 95 Programmierung”, entnommen werden.

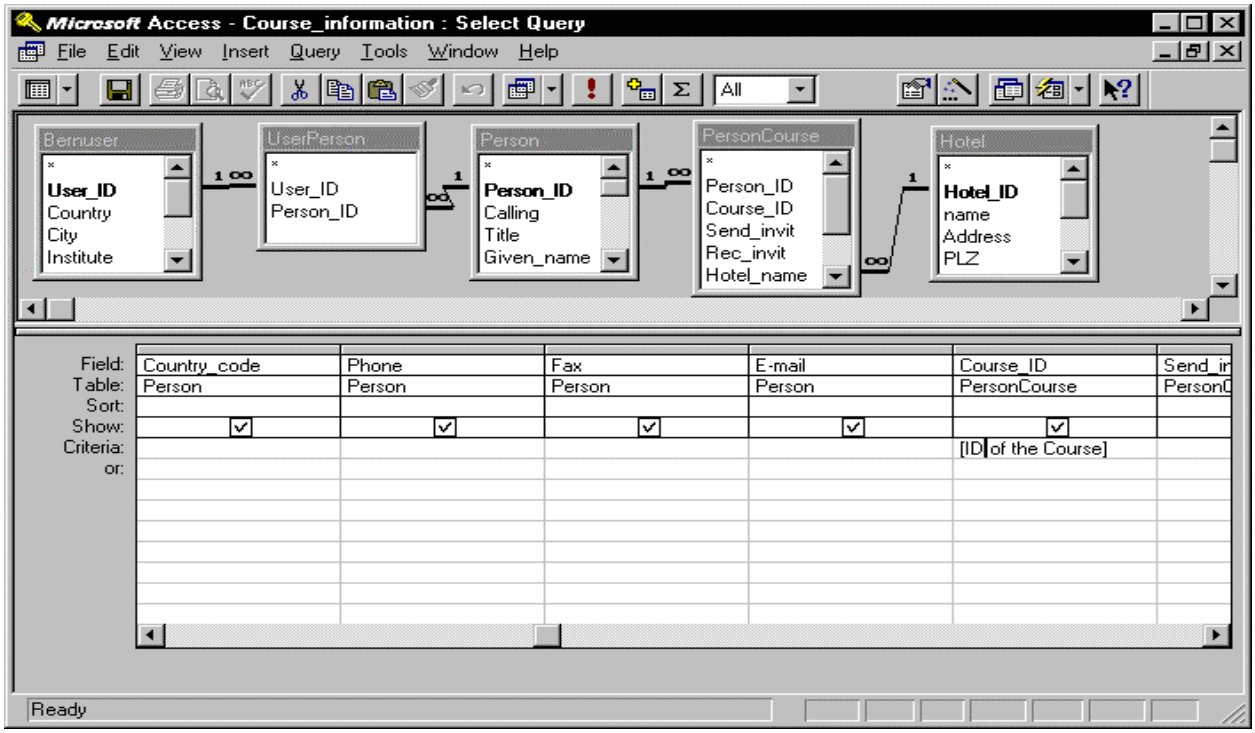
Beispiel 1: Die Erzeugung eines Formulars:
Ein Formular kann aufgrund einer Tabelle/Relation erzeugt werden. Dieses vom Rechner erzeugte Formular weist dann die wesentlichen Eingabefelder auf. Diese müssen anschliessend noch richtig geordnet und um die benutzerdefinierten Dialogelemente (Knöpfe…) ergänzt werden. Je nach Wunsch müssen auch die Eigenschaften dieser Felder und Fenster angepasst werden. Darstellung 5 zeigt ein automatisch erzeugtes Formular.



Darstellung 5
zeigt ein automatisch erzeugtes Formular (links). Auf der rechten Seite sind die zur Verfügung stehenden Anpassungshilfen zu sehen.

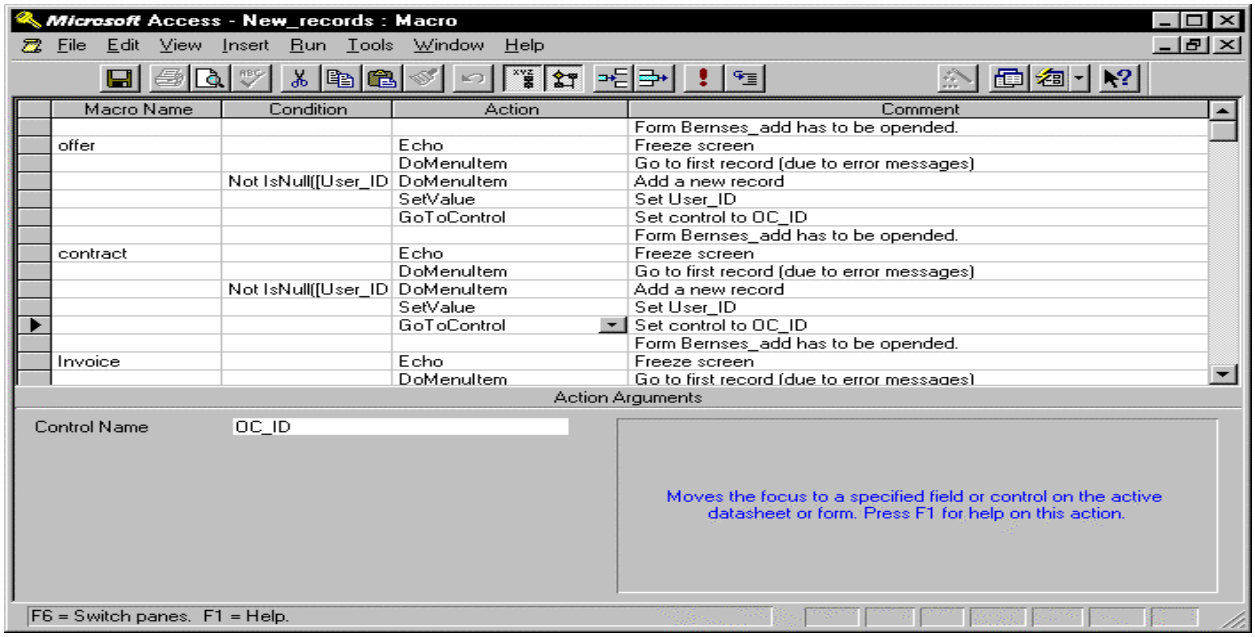
Beispiel 2: Erzeugung eines Queries

Im “Query-tool” können direkt diejenigen Tabellen ausgewählt werden, über welche die Abfrage gemacht werden soll. Anschliessend werden dann die anzuzeigenden Attribute ausgewählt. All dies geschieht grafisch. Als Resultat wird dann eine Abfrage in SQL angelegt, welche immer noch von Hand angepasst werden kann. Darstellung 6 zeigt als Beispiel das Fenster, welches für die Erzeugung des **Course_information** Queries gebraucht worden ist.



Darstellung 6
zeigt die Anordnung auf dem Fenster, welches für die Erzeugung des **Course_information** Queries gebraucht wurde. Im oberen Teil sind die Tabellen mit den Verbindungen zu sehen. Im unteren Teil sind die Attribute mit den entsprechenden Abfragekriterien aufgelistet. Mittels Knopfdruck kann das SQL Kommando angezeigt und angepasst werden.

Beispiel 3: Makro
Werden für eine Operation, mehrere in der Datenbank über Menübefehle abrufbare Aktionen gestartet, so können diese auch als Makro abgespeichert werden. Dies ist speziell dann nützlich, wenn immer wiederkehrende Aktionsfolgen gestartet werden müssen. Als simples Beispiel soll ein neuer Eintrag in einer Tabelle gemacht werden. Ist in einer Tabelle noch kein Eintrag, so generiert Access bereits den Schlüssel (z.B. Person_ID) des ersten Eintrages, dies entspricht der ersten Zeile in der Tabelle. Bei diesem ersten Eintrag muss also die Information lediglich vervollständigt werden, während bei allen folgenden wirklich eine neue Zeile in der Tabelle eingefügt wird. Da dies eine immer wiederkehrende Aktionsfolge ist, wurde dafür ein Makro geschrieben. Darstellung 7 zeigt die entsprechende Makroumgebung.



Darstellung 7
Zeigt das Fenster für die Generierung eines Makros.

Dies sind lediglich 3 Beispiele welche einen kleinen Einblick geben, wie mit Access gearbeitet werden kann. Nachschlagemöglichkeiten wurden am Beginn dieses Kapitels schon erwähnt.

5.2 Die eigentliche Implementation

Im Prinzip wurde die ganze Datenbank von Grund auf neu installiert. Einzige Ausnahme war die Tabelle mit den Personendaten (Namen, Adresse, ...) welche teilweise schon auf Dbase vorhanden war. Diese Informationen wurden importiert und nachträglich angepasst.

In einem ersten Schritt wurden die beiden Tabellen, Person und Bernuser definiert und mit einigen Einträgen gefüllt. Anschliessend wurden diese Tabellen durch die Relation UsrPerson miteinander verknüpft.

Anschliessend an diesen ersten Kern der Datenbank wurden dann die entsprechenden Formulare angefertigt. Dieses kleine System wurde dann getestet um einen ersten Eindruck vom “handling” von Access zu gewinnen.

Nach und nach wurden dann auch die weiteren Tabellen, Relationen und Formulare implementiert und jeweils getestet. Nachdem dieses System dann einigermassen korrekt lief, wurden die Testdaten gelöscht und durch die echten Daten ersetzt.
In diesem Schritt waren dann noch einige Strukturänderungen nötig, da sich beim genaueren Durchsehen der “Papierdaten” noch weitere Probleme zeigten. Dadurch mussten verschiedene Tabellen und Formulare ersetzt werden, welche dann aus der Datenbank entfernt werden mussten. Das Problem dabei war, dass diese Tabellen häufig auch schon in verschiedenen Abfragen vorkamen: Hat man diese nicht mehr benötigten Tabellen nämlich einfach

(unvorsichtigerweise) gelöscht, so haben meist verschiedene Abfragen nicht mehr funktioniert. Darum war auch die “Säuberung” heikler als anfänglich angenommen.

Die beiden “Suchtools” wurden dann erst zuletzt implementiert. Diese entpuppten sich dann mit zunehmendem Gebrauch der Datenbank als immer nützlicher, so dass sie mehr ins eigentliche Zentrum der Datenbank rutschten. Darum wurden diese Tools auch immer wieder mit neuen Möglichkeiten ausgestattet.

6. Erfahrungen

Es ging bei diesem Projekt darum, eine teilweise auf Papier und teilweise in Dbase bestehende Kundenkartei zu vereinheitlichen und in eine moderne, erweiterungsfähige und zu Word für Windows kompatible Datenbank zu übertragen. Dabei musste einerseits die ganze Datenbankstruktur entwickelt und installiert werden. Andererseits mussten aber auch die schon bestehenden Daten eingegeben werden. Dabei tauchten einige Probleme auf, welchen ich im Anfangsstadium der Projektarbeit viel zu wenig Beachtung schenkte. Diese Probleme sind vorwiegend problemspezifischer Natur und konnten darum in der Vorlesung nicht alle behandelt werden. Einige dieser Knacknüsse seien hier nun aufgelistet

Das Herauskristallisieren der Datenstruktur aus der schon bestehenden Datensammlung erwies sich als viel schwieriger als ursprünglich angenommen. Der Grund dafür war, dass eigentlich fast jeder Kunde in einer gewissen Weise individuell behandelt wurde. So gab es zum Beispiel Kunden, die anstatt einer Offerte für das ganze Software Paket drei Offerten für drei Software Teile haben wollten, damit sie die Anschaffungskosten auf drei verschiedene Kredite verteilen konnten. Dies führte unweigerlich zu weiteren Unterschieden der ganzen Verkaufsprozedur, wie zum Beispiel drei Rechnungen, das Abändern eines Vertrages und etliche mehr. Leider ist dies nur eine von vielen weiteren Spezialbehandlungen, welche überhaupt nötig waren, um einen neuen Kunden zu gewinnen.
Solche Spezialbehandlungen sind insofern ein Problem, als diese auch in einer gewissen Weise in der Datenstruktur der elektronischen Datenbank berücksichtigt werden. Dabei musste ich aber darauf achten, die zentrale Datenstruktur so wenig wie möglich zu belasten. In diesem Fall habe ich mich dann dazu entschieden, für diesen Kunden einen speziellen Offerten und Rechnungstyp zu einzuführen und bei der Rechnung nur ein Bezahldatum einzutragen. Weitere Einzelheiten zu diesem Kunden habe ich dann in Form einer unstrukturierten Information als MEMO beigefügt.

Ein weiteres Problem waren die zum Teil unvollständigen Datensätze. Häufig wurde zum Beispiel beim Offertendatum nur das Sendedatum, nicht aber das Datum eingetragen an welchem wir eine positive Zusage erhalten haben. Das Problem dieser Lücken (besonders bei sehr alten Datensätzen) ist nicht eigentlich die fehlende Information, sondern die Lücke selbst, da diese eine sinnvolle Abfrage von offenen Offerten sehr komplex machen. Darum habe ich mich entschlossen, als Antwortdatum den gleichen Wert wie das Sendedatum einzutragen. Dies macht zwar eine Abfrage nach der durchschnittlichen Zeit welche ein Kunde braucht, um sich zum Kauf zu entscheiden ziemlich zwecklos, doch habe ich diese Information als weit weniger wichtig eingeschätzt, als die Frage nach derzeit offenen Offerten.

Als erstaunlicherweise völlig unproblematisch erwies sich das Importieren einer alten Adressenliste von Dbase in Access. Allerdings ist zu bemerken, dass nur die Tabellen und nicht auch die Verknüpfungen und Relationen migriert wurden.

Eine sehr wichtige Erfahrung ist folgende:

Die ersten Grundzüge dieser Datenbank habe mir ich bereits während der Vorlesung “Datenbanken” zurechtgelegt, also mit sehr lückenhaftem Wissen. Im Laufe der Arbeit mit Datenbanken habe ich dann gemerkt, wie wichtig es ist die Grundzüge der Datenbankprogrammierung zu kennen. In meinem Fall musste ich nämlich mit zunehmendem Wachsen des Verständnisses immer wieder grundlegende Änderungen in Kauf nehmen, wodurch

ich natürlich auch ziemlich viel für den “Papierkorb” gearbeitet habe. Ich vermute, dass ich bei der nächsten Datenbank, die ich installieren werde, wesentlich mehr Gewicht auf die vorgängige Planung legen kann und werde. Dabei ist natürlich sehr wichtig, dass auch der Endbenützer möglichst genau weiss, was er am Ende eigentlich will. Obwohl dies eine Binsenwahrheit ist, habe ich im Gespräch mit anderen Leuten, welche eine (andere) Datenbank installiert haben wollten gemerkt, dass es ohne fundierte Kenntnisse einer Datenbank kaum möglich ist, sich eine einigermassen präzise Vorstellung eines Endergebnisses machen zu können.

Eine weitere erwähnenswerte Erfahrung ist, dass in einer zukünftigen Installation einer Datenbank die Suchtools von Anfang an mehr ins Zentrum gestellt werden. Es hat sich wirklich als sinnvoll erwiesen, gleich direkt vom Such- in den Bearbeitungsmodus wechseln zu können (z. Bsp. Suchen, ob ein Kunde schon vorhanden ist und gleich nachsehen, welche Personen diesem Institut zugeordnet sind und diese eventuell auch noch gleich anpassen...).

Eine Bemerkung zu Access:

Für diese vorgesehene Datenmenge erweist sich Access als ideal. Doch wird für grössere Datenmengen vermutlich die Verarbeitungszeit wesentlich grösser, da die Möglichkeiten eines speziell für grosse Datenbanken ausgelegten Systems wie z. Bsp. die interne Reorganisation der Daten nicht vorhanden sind.
Die Zusammenarbeit mit Word für Windows erweist sich als recht angenehm, ist aber äusserst resourcenintensiv (recht langsam).

Leider hat sich gezeigt, dass Access noch einige unpraktische Unschönheiten hat. So gibt es zum Beispiel keine Möglichkeit, ein auf dem Bildschirm farbiges Formular sinnvoll schwarz/weiss auszudrucken. Insbesondere kann die Hintergrundfarbe des Formulars für den Drucker nicht standardmässig auf “weiss” gesetzt werden. Darum wird z. Bsp. ein mit Hintergrund “grau” gesetztes Formular auf dem Drucker auch wirklich mit Hintergrund “grau” ausgedruckt (dies geschieht natürlich auch für Hintergrund “schwarz”, was dann dem verantwortlichen Druckmanager wegen des Tonerverbrauchs die Haare zu Berge stehen lässt). Ich hoffe, dass diese Möglichkeit in einer der nächsten Access-Versionen eingebaut sein wird.

Der momentane Status dieser Datenbank:

Im Moment (beim Einreichen des Projektberichtes) ist die Datenbank fester Bestandteil der Administration am AIUB. Alle wichtigen Daten werden laufend eingetragen und aktualisiert. Die ganze Administration und Organisation der Einführungskurse geschieht heute ausschliesslich mit Hilfe von Access und Word für Windows. Dies beinhaltet z. Bsp. Kunden einladen, Hotelreservationslisten und Adressenlisten drucken, usw.). Im Weiteren hat sich die Datenbank als äusserst effizient und angenehm in Fragestellungen wie “Übersicht über einen Kunden” oder “Ist dies wirklich ein Kunde” , erwiesen. Diese letzte Frage ist nicht ganz so trivial wie sie aussieht: z. Bsp. Institutionen können umziehen, Namen wechseln, Software Lizenzen können z. Bsp. auch transferiert werden...). Erstaunlicherweise hat sich aber gezeigt, dass z. Bsp. beim Schreiben einer Offerte eher mehr Zeit aufgewendet werden muss, um die Adresse mit Hilfe von Access in ein (einzelnes) Dokument einzufügen, als diese direkt ins entsprechende *.dot (Dokumentvorlage in Word für Windows) einzutragen.

Eine Abschliessende Erfahrung:

Um eine Datenbank auf eine Problemstellung masszuschneidern ist es unumgänglich nötig, diese Problemstellung vorgängig genauestens zu kennen oder abzuklären. Darum ist es auch ein "Muss" diese Problemstellung schematisieren zu können. Schon kleine "Spezialwünsche" und -behandlungen erhöhen die Komplexität der Datenbank empfindlich. Somit wird es immer nötig sein Kundenfreundlichkeit und sture, "kalkulierte" Übersicht gegeneinander Abzuwägen. Dies ist vor allem auch eine Frage der Anzahl Kunden. Bei uns mit bis Heute ca. 200 Institutionen und 600 Personen kennt man die einzelnen Kunden noch einigermassen. Bei zunehmender Anzahl Kunden rutschen diese aber mehr und mehr in die Anonymität ab, was dann aber auch die möglichen Spezialwünsche beträchtlich einschränken lässt. Andererseits ist es dann so, dass bei Grossystemen mehr und mehr sog. standardisierte Spezialwünsche eingebaut werden. So können z. Bsp. die Telecomkunden (also im Prinzip wir alle) aus mehreren verschiedenen möglichen Abrechnungsvorlagen auswählen (z .Bsp. summarisch, oder auflisten aller Telefonate, oder auflisten nur der Telefonate teuerer als xx Franken, usw.).
In diesem Sinne ist also diese Datenbank am AIUB als eine erste Kontaktaufnahme zu einem riesigen Anwendungsgebiet zu verstehen.

7. Literaturliste

Prof. O. Nierstrasz: “Datenbanken”, Vorlesungsskript WS 95/96, 1996

Korth, Silberschatz: “Databse System Concepts”, Second Edition, McGraw-Hill Computer Science Series, 1991

J. Bär, I. Bauder: “Das grosse Buch zur ACCESS 95 Programmierung”, Data Becker, 1996

R. Jennings: “Using Access 2 für Windows”, QUE, 1994