Datenbanken

Gliederung

Einführung

Mit dem Thema "Datenbanken" sprechen wir neben Textverarbeitung und Tabellenkalkulation einen dritten großen Komplex der Computeranwendung an. Auch dieses Thema rechnet man zu der sog. Standardsoftware auf Personalcomputern. Im Gegensatz zu den anderen beiden bisher besprochenen Komponenten der Standardsoftware und dabei insbesondere im Gegensatz zur Tabellenkalkulation haben Datenbankanwendungen nicht erst seit Einführung der Personalcomputer eine weite Verbreitung gefunden. Schon lange werden solche Programme überall dort eingesetzt, wo größere geordnete Datenbestände zu unterhalten sind (etwa Banken, Bibliotheken etc.).

Zwei grundlegende Aspekte sind es, die für den Einsatz einer Datenbank bei Vorliegen solcher größerer geordneter Datenbestände sprechen:

1. Abfrage der detaillierten Information über ein bestimmtes Objekt oder eine Gruppe von Objekten (Auflistung aller oder ausgewählter Merkmale für ein oder mehrere Objekte);
2. Übersicht über Verteilungen von Merkmalen im statistischen Sinne (Beispiel Naturschutzdatenbank: "Wie viele der gespeicherten Naturdenkmale sind Bäume?" oder: " Wie viele liegen in der Gemeinde X?")

Die erste Art der Information lässt sich auch auf konventionellem Weg mit einem Karteikartensystem noch vergleichsweise effizient realisieren, allerdings nur eingeschränkt auf das eine Ordnungskriterium der Kartei, also etwa die alphabetische Reihenfolge der Objektnamen. Anders sieht dies aber schon bei der zweiten Art aus, wo es darum geht, einen Überblick zu gewinnen. Hier sind EDV-Lösungen den konventionellen, bei denen man mühsam alle Karteikarten durcharbeiten muss, überlegen.

In der Frühzeit der Datenverarbeitung stand das programmgesteuerte Bearbeiten von Daten im Vordergrund. Heute wird dagegen die Bedeutung der permanenten Datenspeicherung immer größer. In Abb. 1 und Abb. 2 sind die Unterschiede zwischen diesen beiden Vorgehensweisen der Datenverarbeitung dargestellt.

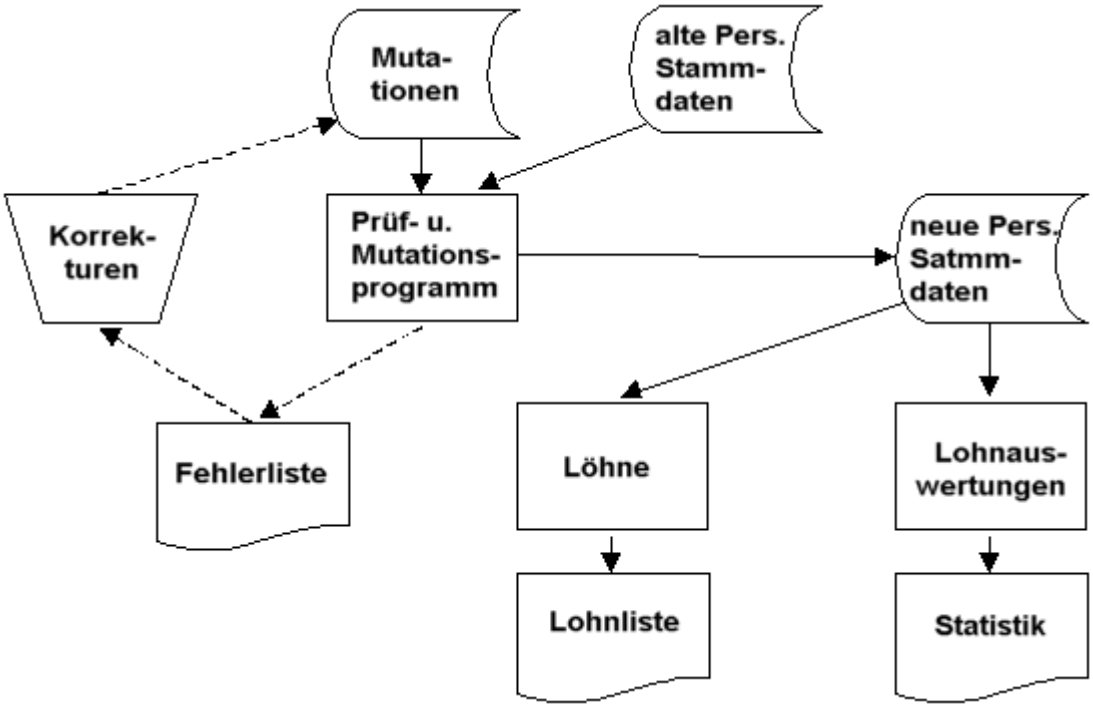

Abb. 1: Lohnabrechnung mit sequentieller Datei-Verarbeitung

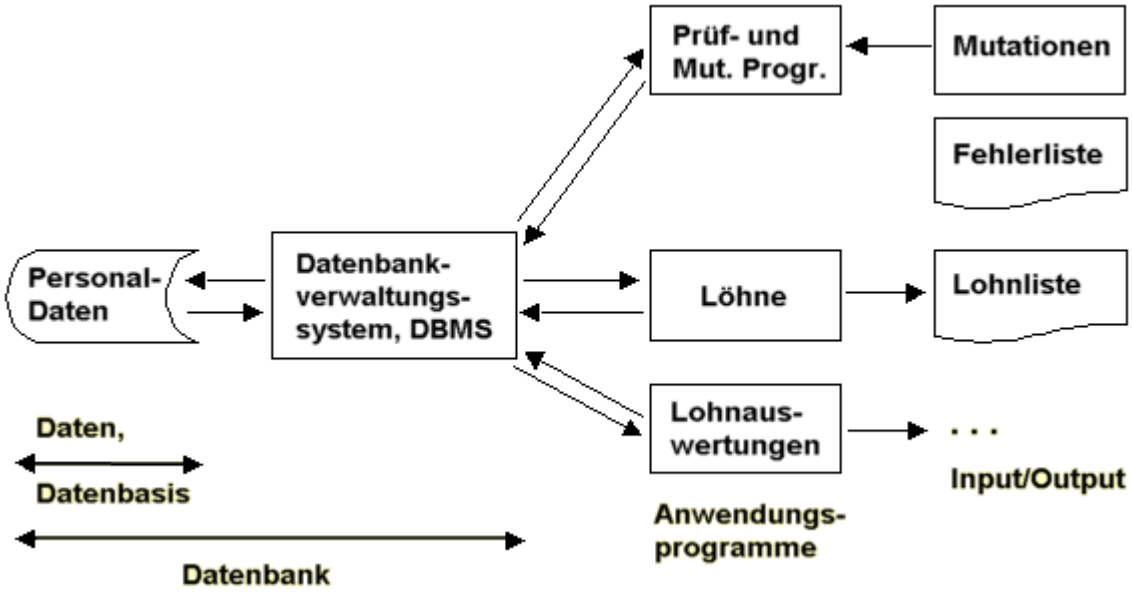

Abb. 2: Lohnabrechnung mit Einsatz einer Datenbank

Das der Datenbankanwendung zugrunde liegende Prinzip kann dabei wie folgt umrissen werden: Mehrere Benutzer wollen ganz oder teilweise auf die gleichen Daten zugreifen. Wenn nun ein sog. "Datenverwaltungssystem" (engl.: DBMS = data base management system) einen auf Dauer angelegten "Datenbestand" (engl.: data base) organisiert, schützt und verschiedenen Benutzern (gleichbedeutend mit Anwendern) zugänglich macht, so fasst man diese beiden Komponenten (Datenverwaltung und Daten) unter dem Begriff "Datenbank" zusammen. Die Datenverwaltung steht zwischen den Daten und den Benutzern und regelt den Zugang (Abb. 3).

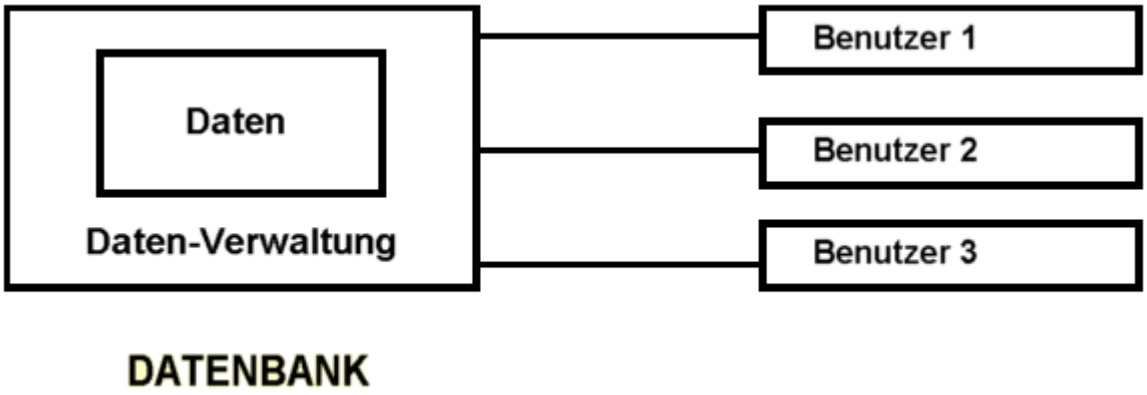

Abb. 3: Grundstruktur einer Datenbank

Folgende Vorteile ergeben sich aus der Anwendung einer Datenbank:

- Die Datenbankorganisation macht es möglich, dass sich nicht jeder Anwender neu mit der inneren Struktur des Datenbestandes herumschlagen muss.
- Sie soll weiterhin verhindern, dass jeder Benutzer unkontrolliert an die Datenbestände gelangen kann und damit die Integrität der Daten gefährdet.
- Sie ermöglicht, jeweils eine anwenderspezifische Datensicht zu definieren, d. h. der Benutzer sieht nicht alle vorhandenen Daten, sondern nur die, die er sehen will bzw. die man ihn sehen lassen will.
- Anwenderprogramme, die auf die Daten zugreifen wollen, sind datenunabhängig, d. h. Änderungen in der internen Datenorganisation machen keine Neufassung von Anwenderprogrammen notwendig.

Allerdings sind mit der Datenbankanwendung auch gewisse Nachteile verbunden. Insbesondere ist damit ein erhöhter Verwaltungsaufwand verbunden und auch die Abhängigkeit von zentralen Funktionen und Entscheidungen. Dies gilt aber mehr für Großdatenbanken. Was wir in dieser Lehrveranstaltung praktisch kennen lernen wollen, sind dagegen zunächst einmal Kleinstanwendungen auf dem PC, für die diese Aspekte nicht gelten.

Logische Datenmodelle

Es existieren verschiedene Modelle in der Datenbankwelt, von denen die 4 wichtigsten hier kurz erwähnt werden sollen. Zur Verdeutlichung soll beispielhaft eine Datensammlung über Klimamessungen herangezogen werden.

Hierarchisches Modell

Ein klassisches Datenmodell ist das Modell der **hierarchischen Gliederung**:

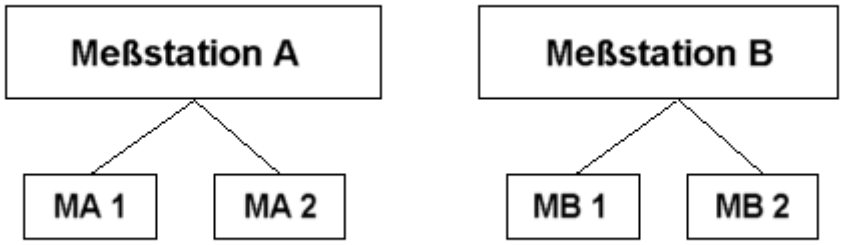

Abb. 4: Prinzip des hierarchischen Datenbankmodells

Solche Hierarchien können natürlich auch mehrstufig sein. Mit Hilfe des hierarchischen Modells kann man sehr gut sog. "1 zu m"-Beziehungen darstellen, d. h. zu jeder Messstation gehören mehrere (m) Messungen, zu jeder Messung aber genau 1 Messstelle. Ein aktueller Einsatzbereich sind Verzeichnisdienste in Netzwerken.

Netzwerkmodell

Ein zweites, ebenfalls weit verbreitetes Modell in der Datenbankwelt ist das sog. **Netzwerkmodell** (heute hauptsächlich auf Großrechnern eingesetzt):

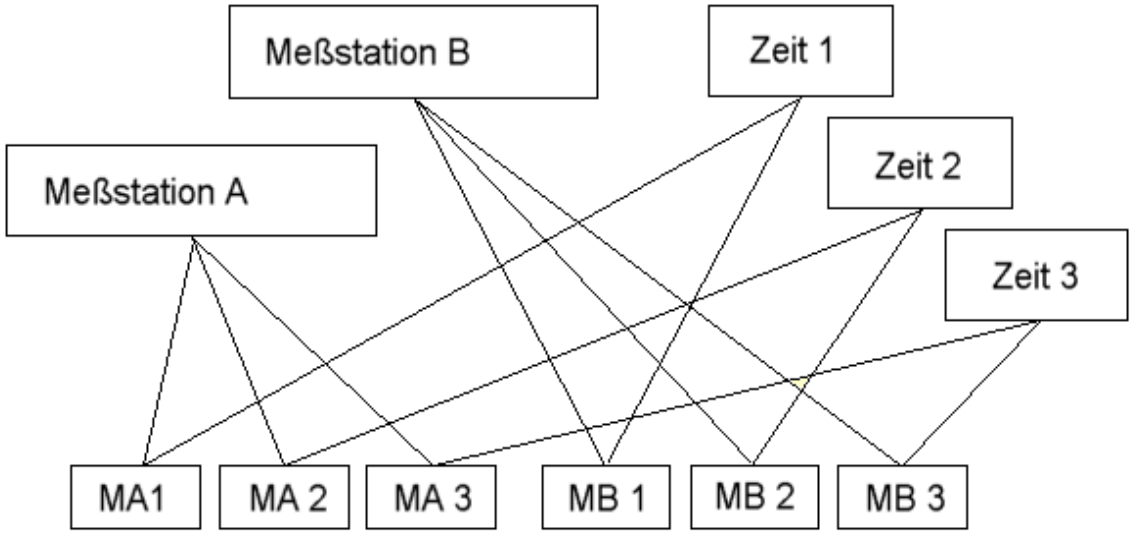

Abb. 5: Prinzip des Netzwerk-Datenbankmodells

Objektorientiertes Modell

Ein weiteres Modell ist das Objektorientierte Datenbankmodell. Datenbanken, die dieses Modell verwenden, heißen **Objektdatenbanken** oder „Objektorientierte“ Datenbanken. Sie legen die Daten nicht in Datensätzen, sondern als *Objekte* an. Diese Objekte erhalten Eigenschaften (z. B. die Farbe rot bei einem Auto) und können über diese Eigenschaften aus der Datenbank abgerufen werden. Damit sind Vorteile, aber auch Nachteile verbunden: entsprechend der Ablage als Objekte gibt es keine besonderen Beziehungen zwischen Datentabellen, was die Handhabung vereinfacht. Da die Objektdatenbankmanagementsysteme (ODBMS) selbst mit objektorientierten Sprachen programmiert wurden, kommen sie besser mit Objekten als mit Tabellen zurecht. Ein Nachteil ist, dass bei bestimmten Datenbankoperationen erhöhte Rechenleistung gefordert ist und es zu einer Verlangsamung der Programmabläufe kommen kann.

Objektdatenbanken sind bislang in der Praxis noch unterrepräsentiert. Als eines der wenigen, aber bekannten Beispiele sei hier der ZOPE-Application Server[1] genannt, der auf einer Objektdatenbank basiert. ZOPE ist für Webanwendungen gedacht und wird für eine große Zahl von Webseiten verwendet.

Relationenmodell

Das heute am weitesten verbreitete und auf PC-Datenbanken praktisch einzige realisierte Datenmodell, mit dem wir uns auch im Folgenden weiter beschäftigen werden, ist das **Relationenmodell** (Tabellenmodell). Im Zentrum steht dabei die Tabelle als Datenmodell. In einer Tabelle stehen je Zeile ein Datensatz, je Tabelle kann es unterschiedliche Anzahlen von Feldern, den sog. Attributen, geben (Tabelle 1).

Tab. 1: Prinzip des relationalen Datenbankmodells

Messwert	Messstation	Zeit
17,3	A	7.00
19,1	A	13.00
21,4	A	19.00
18,7	B	7.00
22,0	B	13.00
16,9	B	19.00

Für das Relationenmodell ist typisch, dass nur Daten, aber keine Beziehungen zwischen diesen explizit dargestellt werden, wie dies etwa beim Netzwerkmodell der Fall ist. Daraus resultiert eine einfachere Speicherung, allerdings auch ein Mehraufwand für den Computer bei Abfragen. Die Beziehungen werden nämlich durch Vergleiche von Werten in verschiedenen Tabellen *während der Abfrage* rekonstruiert.

Ein Beispiel könnte eine Datenbankanwendung einer Firma sein, die aus einer Kundendatei und einer Rechnungsdatei besteht. Die Verknüpfung beider Dateien bei der Rechnungserstellung erfolgt über die in beiden Dateien vorkommende Kundennummer. Dadurch braucht die Kundenadresse nur in der Kundendatei, nicht aber zusätzlich in der Rechnungsdatei gespeichert werden. Bei der heutzutage zur Verfügung stehenden Computerleistung fällt dieser Mehraufwand jedoch nicht mehr ins Gewicht.

[1] http://www.zope.org/, http://www.zope.de/

Datenbankbegriffe

Im Folgenden sollen einige grundlegende Datenbankbegriffe kurz erläutert werden.

Schlüssel

Wesentliche Eigenschaft von Datenbankanwendungen ist die Möglichkeit, gezielt auf die gesuchten Informationen zusteuern zu können. Nehmen wir das Beispiel eines Telefonbuches: Es würde wenig nützen, wenn die Einträge in der Reihenfolge ihrer Eingabe, also des Anmeldedatums des jeweiligen Telefonanschlusses abgedruckt wären. Man müsste dann das ganze Buch durchsuchen, um einen bestimmten Teilnehmer zu finden. Doch genau in dieser Art und Weise sind in der Regel in einer Datenbank die Datensätze gespeichert, einer nach dem anderen gemäß dem Zeitpunkt der Eingabe. Um gezielt Informationen suchen zu können, braucht es also noch eine Hilfskonstruktion, nämlich Sortier- und Suchroutinen.

Im Falle des Telefonbuches ist es bekanntlich sinnvoll, die Einträge in alphabetischer Reihenfolge sortiert vorliegen zu haben, bei Einträgen mit gleichem Familiennamen zusätzlich in alphabetischer Reihenfolge der Vornamen. Manchmal wäre es auch nützlich, wenn man zusätzliche Telefonbücher hätte, in denen die Einträge beispielsweise nach Straßen sortiert wären, etwa dann, wenn man den genauen Namen eines Teilnehmers nicht weiß, wohl aber die Adresse kennt.

All dies ist mit einer Datenbank (z. B. einem digitalen Telefonbuch) realisierbar. Man definiert dazu bestimmte Felder (z. B. NAME) als sog. Schlüsselfelder. Zum grundlegenden Befehlsschatz jedes Datenbankprogramms gehört es, die Einträge gemäß den definierten Schlüsselfeldern sortiert anzuzeigen. Man spricht in diesem Zusammenhang von *Sortierschlüsseln*[2].

Sortierschlüssel müssen nicht eindeutig sein, der Name "Meier" im Feld "NAME" darf also durchaus mehrfach vorkommen. Wenn ein Schlüssel eindeutig ist, d. h. jeder Wert für den Schlüssel nur einmal vorkommt, spricht man von einem Identifikationsschlüssel (z. B.: Jede Kundennummer kommt nur einmal vor, während ein Name häufig mehrfach vorkommt).

Schlüsselfelder, die man definiert, um bestimmte Ausschnitte der Datenbank isolieren zu können, nennt man *Suchschlüssel*. Mit dem Suchschlüssel "Name, Wohnort" und den Werten "Meier, München" können alle Personen, die diese Merkmale erfüllen, aus der Datenbank herausgesucht werden.

Primärschlüssel

Der Primärschlüssel enthält das oder die Felder, das oder die zur eindeutigen Bestimmung eines Datensatzes verwendet werden. Der Wert des Primärschlüsselfeldes oder die Kombination von Werten, wenn mehrere Felder im Primärschlüssel aufgenommen werden, muss für jeden Datensatz *ein*eindeutig, also unverwechselbar sein. Wenn Datenbank-Tabellen miteinander verknüpft werden sollen, ist es unbedingt nötig, einen Primärschlüssel anzulegen.

In PC-Datenbanksoftware wird der Primärschlüssel oft mit einem Schlüsselsymbol markiert: 

Fremdschlüssel

Ein Fremdschlüssel ist eine Entsprechung zwischen Spalten in einer Tabelle und den Primärschlüsselspalten in einer anderen Tabelle.[3]

---

[2] Hinweis: "Sortierschlüssel" hat nichts mit dem umgangssprachlichen Begriff "heraussortieren" zu tun. Hier geht es vielmehr um die Herstellung einer bestimmten Reihenfolge der Datensätze

[3] Vgl. Microsoft Access-Hilfe sowie das Beispiel in Abb. 8.

Referenzielle Integrität

In komplexen Datenbanken werden verschiedene Tabellen über Beziehungen („Referenzen“) miteinander verknüpft. Dadurch kann es aber zu Problemen kommen, z. B. wenn zu einem Fremdschlüsselwert der Datensatz in der zugeordneten Tabelle fehlt. Ein Beispiel: in einer Datenbank werden die Daten der Kunden und die von ihnen getätigten Bestellungen in zwei verschiedenen Tabellen geführt. Wenn aus irgendeinem Grund ein Kunde gelöscht wird, obwohl noch eine Bestellung von ihm offen ist, sind die Bezüge zwischen den Tabellen nicht mehr schlüssig – also: die *Referenzen* sind nicht mehr *integer*.

Um diese Integrität zu gewährleisten, müssen bestimmte Regeln erfüllt sein:

- Wenn ein Kunde gelöscht wird, müssen auch die von ihm getätigten Bestellungen gelöscht werden („kaskadiertes Löschen“)
- Wenn ein Kunde eine neue Kundennummer bekommt, müssen in allen darauf Bezug nehmenden Tabellen die Fremdschlüssel (Kundennummern) angepasst werden („kaskadiertes Ändern“)

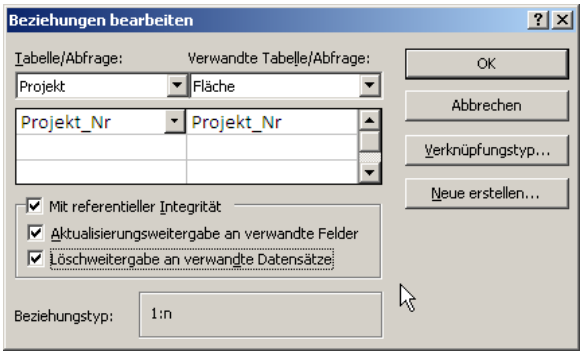

Abb. 6: Einstelldialog für eine Tabellenverknüpfung – Access 2010

PC-Datenbanksysteme können z. T. so eingestellt werden, dass sie die Einhaltung der *Referenziellen Integrität* überwachen bzw. kaskadiertes Löschen und kaskadiertes Ändern automatisch durchführen („Löschweitergabe“ und „Aktualisierungsweitergabe“ in Abb. 6).

Normalisierung

„Der Sinn der Normalisierung besteht darin, Redundanzen (gleiche, mehrmals vorhandene Information) zu verringern und dadurch verursachte Anomalien (z. B. einander widersprechende Dateninhalte) zu verhindern, um so die Wartung einer Datenbank zu vereinfachen sowie die Konsistenz der Daten zu gewährleisten.“[4]

Um Redundanzen zu verringern und Anomalien zu vermeiden, müssen Daten in mehrere Tabellen aufgeteilt werden. Dabei wird in Stufen vorgegangen, wobei die sog. „Normalformen“ erreicht werden sollen:

1. Normalform (1NF)

- In den Attributen einer Tabelle stehen nur „atomare“ Werte, also keine Zusammensetzungen wie „Herbert Niemand“, „85354 Freising“ o. ä.

2. Normalform (2NF)

- Die Tabelle steht in der ersten Normalform und
- Die Werte aller Nichtschlüssel-Attribute sind voll funktional[5] vom Primärschlüssel abhängig.

3. Normalform (3NF)

- Die Tabelle steht in der zweiten Normalform und
- Es gibt keine Abhängigkeiten irgendwelcher Nichtschlüssel-Attribute von anderen Nichtschlüssel-Attributen in der gleichen Tabelle.

---

[4] http://de.wikipedia.org/wiki/Normalisierung_(Datenbank)

[5] Man spricht von funktionaler Abhängigkeit, wenn einzelne Attribute einer Tabelle andere Attribute eindeutig bestimmen.

1. Normalform

Ein einfaches Beispiel: Die Angabe einer Adresse wie „Dombibliothek Freising, Domberg 40, 85354 Freising“ muss für die Verwendung in einer Datenbank in mehrere Felder aufgeteilt werden:

Feldname	Nummer	Bibliothek	Straße	Nr	PLZ	Ort
Datensatz	1	Dombibliothek Freising[6]	Domberg	40	85354	Freising

Hier sind bereits alle Daten in atomare Werte aufgeteilt, die Tabelle befindet sich also in der 1. Normalform.

2. Normalform

Kommen zu einer Tabelle wie im Beispiel oben weitere Felder hinzu – wie z. B. beim Standort (fiktiver) Bücher aus der Dombibliothek – so enthält die entstehende Tabelle einige Daten mehrfach (die letzten fünf Felder):

Nr	Autor	Vorname	Jahr	Titel	Bibliothek	Straße	Nr	PLZ	Ort
1	Pastorius	Markus	1557	Panem et circenses	Dombibliothek Freising	Domberg	40	85354	Freising
2	Precht	Manfred	2005	Mathematik für Nichtmathematiker	HSWT-Bibliothek Weihenstephan	Am Hofgarten	4	85354	Freising
3	Engel	Engelbrecht	1789	Engel und Erzengel	Dombibliothek Freising	Domberg	40	85354	Freising

Außerdem wird deutlich, dass die Attribute nicht nur vom Primärschlüssel – der eingefügten Datensatznummer, gekennzeichnet durch – abhängen: Die Felder „Bibliothek“, „PLZ“ und „Ort“ hängen nicht *nur* vom Primärschlüssel ab. Um das aufzulösen, müssen daraus mehrere Tabellen entstehen. Jede dieser Tabellen erhält einen Primärschlüssel, über den sie mit der anderen Tabelle verknüpft werden kann:

Zitate

Nummer	Autor	Vorname	Jahr	Titel	Standort
1	Pastorius	Markus	1557	Panem et circenses	1
2	Precht	Manfred	2005	Mathematik für Nichtmathematiker	2
3	Engel	Engelbrecht	1789	Engel und Erzengel	1

Bibliotheken

Standort	Bibliothek	Straße	Nr	Ort
1	Dombibliothek Freising	Domberg	40	1
2	HSWT-Bibliothek Weihenstephan	Am Hofgarten	4	1
3	Universitätsbibliothek München	Geschwister-Scholl-Platz	1	2

Orte

Ort	PLZ	Ort
1	85354	Freising
2	80539	München

Die Tabelle wurde in drei Tabellen aufgeteilt: Zitate, Bibliotheken und Orte. Innerhalb jeder Tabelle ist jedes Attribut vom Primärschlüssel abhängig. Somit ist die zweite Normalform erreicht.

[6] Wir belassen hier „Dombibliothek Freising“ in einem Feld, da es sich um einen feststehenden Begriff handelt.

3. Normalform

Offenbar bestehen nun weitere Abhängigkeiten: Ort und Postleitzahl sind voneinander abhängig. Beides sind Nichtschlüssel-Attribute. Wir müssen diese Tabelle noch ändern, um sie in die dritte Normalform zu bringen. Es macht hier Sinn, die Postleitzahl zum Primärschlüssel zu machen und die Tabellen entsprechend umzustellen:

Zitate

Nummer	Autor	Vorname	Jahr	Titel	Standort
1	Pastorius	Markus	1557	Panem et circenses	1

…

Bibliotheken

Standort	Bibliothek	Straße	Nr	PLZ
1	Dombibliothek Freising	Domberg	40	85354
2	HSWT-Bibliothek Weihenstephan	Am Hofgarten	4	85354
3	Universitätsbibliothek München	Geschwister-Scholl-Platz	1	80539

Orte

PLZ	Ort
85354	Freising
80539	München

Neben der 3NF gibt es noch weitere Normalformen. Beim Datenbankentwurf kann es aus Gründen der Performance aber durchaus sinnvoll sein, auf Normalisierungsschritte zu verzichten. Für eine ausführlichere Darstellung der ersten drei Normalformen siehe auch das Übungs-Skript zum Datenbankentwurf.

Entity Relationship Model (ERM)

Das Entity Relationship Model („Gegenstands-Beziehungs-Modell“) stellt in einer Grafik den fertigen Datenbankentwurf dar (Abb. 10). Dabei werden die involvierten Tabellen, deren Primärschlüsselfelder und die Beziehungen zwischen den Tabellen angezeigt.

Datenbankmanipulationen

Unter dem Begriff Datenbankmanipulation werden beliebige Operationen auf einer Datenbank zusammengefasst. Dies sind unter anderem:

- Als zu allererst notwendige Operation die Eingabe der Daten, ohne die alles weitere keinen Sinn macht.
- Als wesentliche Operation für den Benutzer einer Datenbank die gezielte *Abfrage* oder Recherche (engl.: query). Diese erlaubt, einen bestimmten Ausschnitt einer Datenbank abzugrenzen und den abgegrenzten Inhalt in geeigneter Form herauszulesen.
- Eine weitere Operation, die irgendwann im Laufe der Existenz einer Datenbank einmal notwendig wird: die *Mutation* oder Nachführung, denn irgendwann ändern sich die in der Datenbank abgebildeten Tatbestände, so dass eine Korrektur ansteht.
- Die *Transaktion* als sehr komplexe Datenbankoperation. Darunter versteht man eine Konsistenz erhaltende Operation auf einer Datenbank, also Freiheit von Widersprüchen innerhalb der Datenbank.

Ein Beispiel zur Erläuterung: Eine Bank erhält den Auftrag, einen bestimmten Betrag von einem Konto auf ein anderes zu überweisen. Dazu sind zwei Datenbankoperationen nötig. Erstens muss der Betrag vom Konto des Auftraggebers abgebucht und zweitens muss er dem Konto des Überweisungsempfängers gutgeschrieben werden. Ein für solche Aufgaben ausgelegtes Datenbankprogramm muss sicherstellen, dass tatsächlich beide Operationen durchgeführt werden. Sollte bei der zweiten

Operation irgendetwas schief gehen, muss auch die erste rückgängig gemacht werden. Es ist einleuchtend, dass die Realisierung derart komplexer Operationen nicht ganz trivial ist.

Datenbankprogrammierung

Eine Datenbank (insbesondere eine solche auf PC-Ebene wie z. B. Access oder Base) kann unmittelbar mit den vom Datenbankprogramm zur Verfügung gestellten Befehlen bearbeitet werden. Dies erfolgt auf die gleiche Weise, wie Sie dies bereits von der Textverarbeitung und der Tabellenkalkulation her kennen.

Darüber hinaus besteht jedoch die Möglichkeit, unter Verwendung der Datenbankbefehle für einen bestimmten Anwendungszweck eine maßgeschneiderte Lösung zu programmieren. Typischerweise sieht das so aus, dass dem Benutzer zunächst in Form eines Bildschirmmenüs die zur Verfügung stehenden Möglichkeiten angezeigt werden. Er kann aus diesem Menü wählen und bekommt dann entweder ein nachgeordnetes Menü für die weitere Spezifizierung der Wünsche gezeigt oder er wird in einen Dialog verwickelt, um Detailfragen der gewünschten Anwendung zu klären. Schließlich wird die gewünschte Anwendung ausgeführt und das Menü erneut angezeigt. Programmiersprachen, die in solch ein Datenbankprogramm eingebettet sind, werden häufig als Programmiersprachen der vierten Generation („4GL“)[7] bezeichnet. Die Leistungsfähigkeit dieser Sprachen beruht im Wesentlichen auf zwei Gesichtspunkten:

1. Möglichkeit der direkten Verwendung von Datenbankbefehlen
2. Möglichkeit meist bequemer Programmierung von Bildschirmdialogen

Gegenüber konventionellen höheren Programmiersprachen wie Pascal, C und dessen Nachfolger C++, Java oder anderen[8] ist man damit in der Lage, ausgesprochen schnell Programmieraufgaben zu lösen, wie sie für Datenbankanwendungen typisch sind.

SQL (Structured Query Language)

Die 4GL-Sprache „SQL“, die „Structured Query Language“, ist die in relationalen Datenbanken gebräuchliche Datenabfrage- und -manipulationssprache. Mit ihr lassen sich Tabellen anlegen oder löschen, Daten abfragen und Daten verändern. Da die Beziehungen zwischen den Tabellen bei relationalen Datenbanken nicht in den Tabellen festgelegt sind (s. o.), werden die Beziehungen ebenfalls in SQL beschrieben.

Typische Anweisungen beginnen mit den Schlüsselwörtern „CREATE TABLE“ zum Anlegen einer Tabelle, „SELECT“ um Daten abzufragen, „INSERT“ um Daten einzufügen, „DELETE“ um Daten zu löschen u.v.m. Man sieht daran bereits den Charakter von SQL: es gibt keine komplizierten Konstruktionen, die Variablendeklarationen benötigen, Prozeduren verwenden oder mit Programmcode wie „begin … end“, „if …then“ oder „for x=1 to n“ etc. arbeiten. Die SQL-Anweisungen bestehen aus Elementen der normalen Sprache und sind dadurch leichter zu verstehen.[9]

Abfragen

Sehen wir uns ein paar Beispiele an. Aus einer Datenbank eines Reisebüros sollen Daten von Mitarbeitern abgefragt werden. Das Grundstatement ist „SELECT“, also „wähle aus“. Nun muss man wissen, um welche Tabelle der Datenbank es geht – diese wird ausgewählt mit „FROM“ –, und welche

---

[7] Zu Beispielen siehe http://de.wikipedia.org/wiki/Fourth_generation_language

[8] http://de.wikipedia.org/wiki/Höhere_Programmiersprache

[9] Das heißt aber nicht, dass SQL-Programme nicht auch sehr komplex sein können!

Felder benötigt werden. Sagen wir, Nachname, Vorname und eMail-Adresse sind gesucht, diese stehen in der Tabelle „Mitarbeiter“. Außerdem sollen nur die Mitarbeiterinnen abgefragt werden, diese Bedingung beschreiben wir mit „WHERE“. Am Ende der Anweisung steht immer ein Semikolon! Also:

```
SELECT Nachname, Vorname, email
FROM Mitarbeiter
WHERE Geschlecht = „w“;
```

wähle Nachname. Vorname und eMail-Adresse
und zwar aus der Tabelle „Mitarbeiter“
wenn im Feld „Geschlecht“ ein „w“ (für „weiblich“) steht.

Das Ergebnis einer solchen Abfrage ist dann wieder eine Tabelle, die nur die drei Felder „Nachname“, „Vorname“ und „email“ sowie die Daten aus den Datensätzen der weiblichen Mitarbeiter enthält.

Wildcards

Um *alle* Felder einer Tabelle zu erhalten, muss man nicht alle Felder in die SQL-Anweisung hinein schreiben, es genügt, stattdessen ein „*****“ zu verwenden:

```
SELECT *
FROM Mitarbeiter
WHERE Geschlecht = „w“;
```

wähle alle Felder
aus der Tabelle „Mitarbeiter“
wenn im Feld „Geschlecht“ ein „w“ steht.

Verknüpfungen

Nun stehen die Daten ja nicht alle in derselben Tabelle (siehe „Normalisierung“, S. 7), wir müssen also in der Lage sein, verschiedene Tabellen miteinander zu verknüpfen. Das nötige SQL-Schlüsselwort ist „JOIN“. Von dieser Verknüpfung gibt es verschiedene Typen. Hier soll uns die Verknüpfung interessieren, die jene Datensätze zweier Tabelle zusammenführt, die in einem bestimmten Feld den selben Wert haben. Dies ist der sog. „INNER JOIN“. Zu den Angaben für die Abfrage aus einer einzelnen Tabelle müssen wir jetzt noch wissen, welche weitere Tabelle verknüpft werden soll und welche Felder die Beziehung definieren. Hier ist das die in beiden Tabellen vorkommende Reiseveranstalter-Nummer (RVNr):

```
SELECT ReiseNr, Ziel, Preis, Veranstalter
FROM Reisen
INNER JOIN Reiseveranstalter
ON Reisen.RVNr = Reiseveranstalter.RVNr

WHERE Preis > 1000;
```

wähle ReiseNr, Ziel, Preis und Veranstalter
aus der Tabelle „Reisen“
verknüpfe sie mit der Tabelle „Reiseveranstalter“
über die Felder RVNr der Tabelle „Reisen“ und RVNr der Tabelle „Reiseveranstalter“
wenn der Preis über 1000 liegt

Je nach Implementierung von SQL in den Datenbankprogrammen kann es aber zu unterschiedlichen Schreibweisen der Anweisungen kommen. Während MS-Access z. B. das Semikolon am Ende der Anweisungen strikt benötigt, sieht das OpenOffice Pendant Base über ein fehlendes Semikolon hinweg. Dagegen müssen in Base die Tabellennamen hinter FROM in Anführungszeichen gesetzt wer-

den, in Access aber nicht. Beachten Sie daher die den Programmen beigefügten Hilfen, um die jeweils korrekte Schreibweise zu finden.

Wichtig ist in jedem Fall, die Bezeichnungen für Tabellen und Felder zu kennen – SQL unterscheidet Groß- und Kleinschreibung und findet Felder und Tabellen etc. nur, wenn die Namen in den Anweisungen *genau* mit den Namen in der Tabellen und Felder etc. übereinstimmen!

Ähnlich wie die SELECT-Anweisung arbeiten auch die anderen Anweisungen für die Datenänderung oder
-löschung. Wer sich intensiver damit beschäftigen möchte, findet erste Unterstützung in den Hilfen, die den Programmen beigefügt sind. Ein SQL-Tutorial findet sich unter

http://www.sql-und-xml.de/sql-tutorial/

Datenbankentwurf

Dem Entwurf einer Datenbank liegt immer die Notwendigkeit zugrunde, eine Menge meist komplexer Daten in eine in der EDV verarbeitbare Form zu bringen. Dazu müssen die realen Verhältnisse modellhaft in Einzeldaten und deren Beziehungen zueinander abgebildet werden. Wir müssen also zuerst einmal feststellen, welche Objekte (Entitäten) es gibt, zu denen wir Daten vorliegen haben und welche Beziehungen zwischen diesen Objekten bestehen. Dann muss geklärt werden, wie wir die Daten in Tabellen einer relationalen Datenbank ablegen können.

Entitätstypen identifizieren

Objekte oder Entitäten können sein:

- Individuen (Mitarbeiter, Kunden …)
- Reale Objekte (Reisen, Kurse, Bücher …)
- Abstrakte Konzepte (Abteilungen …)
- Ereignisse (Reisebuchung, Kursbesuch, Bezahlung …)

Entitäten sind z. B. der *Mitarbeiter Maier* der Firma X oder deren *Abteilung „Controlling“*. Wenn es mehrere Objekte des gleichen Typs gibt (gleichartige Informationsobjekte), kann man diese als *„Entitätstypen*“ zusammenfassen, z. B alle Mitarbeiter einer Firma oder die verschiedenen Abteilungen der Firma.

Nehmen wir als Beispiel die Datenbank eines Reisebüros. Hier gibt es die Entitäten

- Mitarbeiter
- Kunden
- Reisen
- Reisebuchungen

Jeder dieser Entitätstypen wird in einer eigenen Tabelle modelliert, die eine bestimmte Anzahl von Attributen (Spalten oder Felder) hat. Diese Attribute dienen dazu, die Eigenschaften der Objekte näher zu beschreiben. Zum Aufbau einer Tabelle müssen folgende Schritte erfolgen:

- Attribute festlegen (Feldnamen)
- Datentyp festlegen (Zahl, Text, Datum …)
- Datengrößen festlegen (Feldlängen)
    - ggf. Anzahl Dezimalstellen
- Gültigkeitsregeln festlegen:
    - z. B. Geschlecht: „m“ oder „w“
    - z. B. Postleitzahl < 99999
- Primarschlüssel bestimmen:
    - nicht leer
    - keine Duplikate
    - je Tabelle genau *ein* Primärschlüssel, der allerdings aus mehreren Attributen bestehen kann

Erst dann können in die Tabelle Datensätze eingegeben werden. Allerdings sollten vorher noch die Beziehungen der Daten untereinander und die u. U. nötige Aufteilung von Tabellen in mehrere Tabellen geklärt werden!

Normalisierung

Die Normalisierung ist beim Aufbau einer Datenbank wohl der komplizierteste Schritt (siehe „Normalisierung“, S. 7). Dennoch ist dieser Schritt unverzichtbar, um keine Inkonsistenzen zu riskieren und nicht dieselben Daten mehrfach zu speichern, was die Datenbank unnötig aufblähen kann. Wenn z. B. in einer Tabelle der Studierenden der vhb[10] in jedem Datensatz die Adresse der Heimatuniversität eingetragen wird, kann man sich leicht vorstellen, wie umfangreich eine solche Tabelle werden kann. Das braucht Speicherplatz und verschlingt bei Abfragen oder Änderungen viel Rechenpower.

Erst wenn der Normalisierungs-Schritt abgeschlossen ist, können die Beziehungen zwischen den Daten und damit zwischen den verschiedenen Tabellen genauer betrachtet werden.

Beziehungen zwischen Entitätstypen

Um die verschiedenen Typen möglicher Beziehungen zwischen den Entitäten zu klären, soll ein Datenbankentwurf für ein Reisebüro als Beispiel dienen (Abb. 7). Hier sind fünf Tabellen erstellt worden, mit den Entitätstypen Abteilung, Mitarbeiter, Reisen, Reiseveranstalter und Kunden.

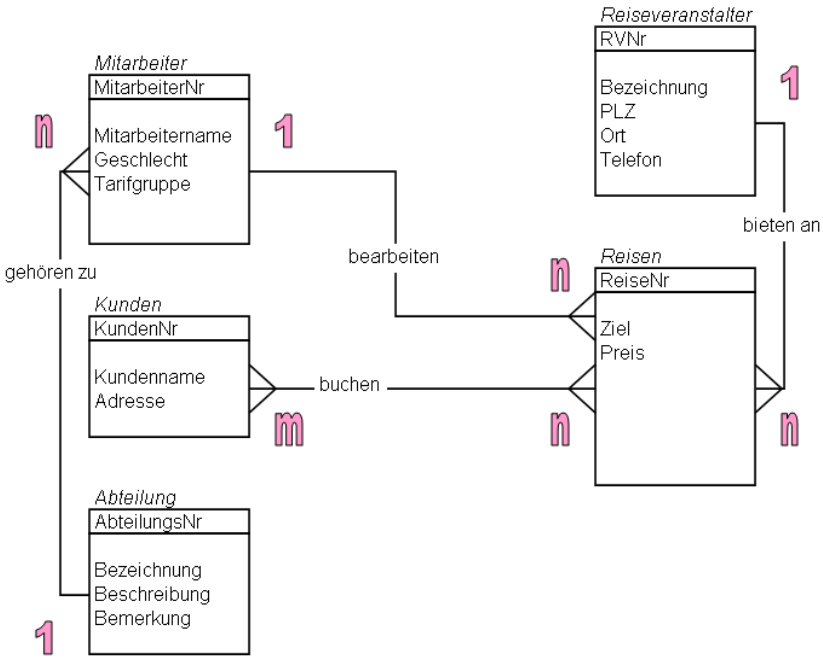

Abb. 7: Beispielszenario für 1:n- und m:n-Beziehungen zwischen verschiedenen Entitätstypen

Wir finden zwischen den Entitäten die beiden Beziehungstypen 1:n und m:n.[11] Die **1:n-Beziehung** bedeutet, dass zu *einem* Datensatz einer Tabelle *mehrere* Datensätze einer anderen Tabelle gehören: ein Mitarbeiter kann mehrere Reisen bearbeiten, üblicherweise wird jede Reise aber nur von einem Mitarbeiter bearbeitet; eine Abteilung hat mehrere Mitarbeiter, aber jeder Mitarbeiter gehört nur einer Abteilung an.

Die **m:n-Beziehung** dagegen verbindet *mehrere* Datensätze einer Tabelle mit *mehreren* Datensätzen einer anderen Tabelle: ein Kunde kann mehrere Reisen buchen, eine Reise kann aber auch von mehreren Kunden gebucht werden.

Das Ziel der Datenbankmodellierung ist die Auflösung aller noch vorhandenen m:n Beziehungen in 1:n Beziehungen.

---

[10] Virtuelle Hochschule Bayern

[11] Es gibt mehrere Möglichkeiten, diese Beziehungen grafisch darzustellen. Hier wurde im Ansatz die sog. Martin- oder Krähenfuß-Notation verwendet. http://de.wikipedia.org/wiki/Martin-Notation

Umsetzung einer 1:n-Beziehung

Die 1:n-Beziehung wird mit Hilfe des Fremdschlüssels erstellt. Angenommen es liegen zwei Tabellen – „Mitarbeiter“ und „Reisen“ – vor. In der Mitarbeiter-Tabelle gibt es ein Feld „MitarbeiterNr“, das als Primärschlüssel definiert ist, wie am Schlüsselsymbol zu erkennen (Abb. 8).

Die zweite Tabelle enthält die Daten zu den einzelnen Reisen. Hier ist ebenfalls ein Feld mit der Mitarbeiternummer enthalten, um festzuhalten, welcher Mitarbeiter die jeweilige Reise bearbeitet. Wird der Primärschlüssel der ersten Tabelle in einer zweiten Tabelle als Feld aufgenommen, bezeichnet man das Feld dort als Fremdschlüssel. Mit der Zuweisung des Fremdschlüssels zwischen den Feldern *Mitarbeiter.MitarbeiterNr* und *Reisen.MitarbeiterNr* wird jeder Reise der richtige Datensatz aus der Tabelle Mitarbeiter zugeordnet.

Mitarbeiter

MitarbeiterNr	Mitarbeitername	Geschlecht	Tarifgruppe
1	Eva Ewer	w	CD1
2	Horst Schmidt	m	CD2
3	Paula Winzig	w	CD1
4	Margot Picou	w	CD3
5	Sabine Rönnicke	w	CD2

Reisen

ReiseNr	Ziel	Preis	MitarbeiterNr
1	San Francisco	5000	1
2	New York	2500	1
3	Tokio	1250	2
4	Los Angeles	2000	5
5	San Francisco	1800	3
6	San Francisco	1900	1

Abb. 8: 1:n-Beziehung – Zuweisen eines Fremdschlüssels

Auflösung einer m:n-Beziehung

Die m:n-Beziehung lässt sich in einer Relationalen Datenbank nicht abbilden. Sie muss in zwei 1:n-Beziehungen aufgelöst werden, indem man eine zusätzliche Tabelle einschiebt. In unserem Fall ist das eine Tabelle mit den einzelnen Buchungen der Kunden. Jetzt finden wir *zwei 1:n-Beziehungen* wieder, die sich über die Zuweisung von Fremdschlüsseln realisieren lassen: ein Kunde kann mehrere Reisen buchen, zu jeder Buchung gehört aber nur ein Kunde. Jede Buchung bezieht sich nur auf eine Reise, jede Reise kann aber in mehreren Buchungen auftauchen (Abb. 9).

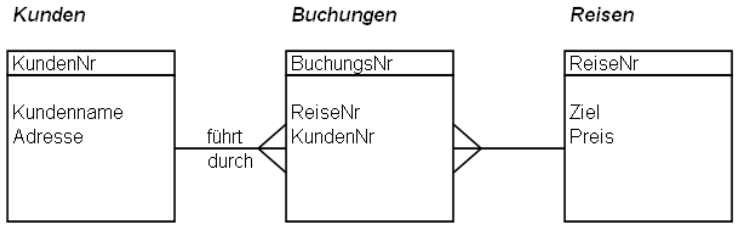

Abb. 9: Auflösung einer m:n-Beziehung über eine Zwischentabelle

Fertiger Datenbankentwurf: ERM

Datenbankprogramme für den PC wie MS-Access und Base aus dem OpenOffice-Paket enthalten einen Assistenten, mit dessen Hilfe die Beziehungen zwischen den Tabellen eingestellt und grafisch dargestellt werden können. Es entsteht das „Entity Relationship Model“ (ERM), das Modell der Objekt-Beziehungen einer Datenbank.

Dabei werden zuerst die nötigen Tabellen ausgewählt, diese werden dann als Listen der enthaltenen Felder angezeigt. Jetzt kann man mit der Maus die gewünschten Beziehungen zwischen Primärschlüsseln und Fremdschlüssel-Feldern herstellen. Die entstehenden 1:n-Beziehungen werden als Verbindungslinie zwischen diesen Feldern angezeigt, zusätzlich werden die „1-Seite“ und die „n-Seite“ angegeben (in OpenOffice Base „1“ und „n“, in MS Access als „1“ und „∞“; Abb. 10).

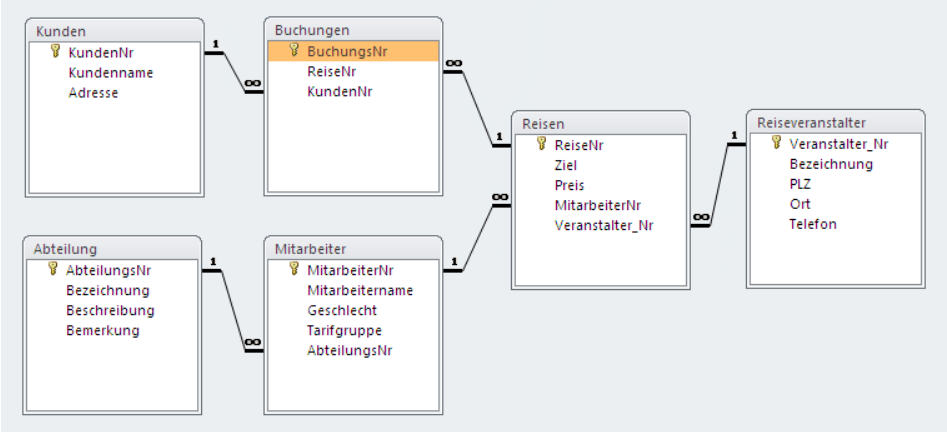

Abb. 10: Beispiel für einen fertigen Datenbankentwurf – Access 2010

Während der Anlage der Beziehungen können die Einstellungen zur Referentiellen Integrität, also kaskadiertes Löschen und kaskadiertes Ändern, erfolgen.

Fertigstellen der Datenbank

Tabellen

Nach Fertigstellung des Datenbankentwurfes können in die Tabellen nun Daten eingegeben werden. Dabei muss man gemäß der Referentiellen Integrität darauf achten, dass zu einem Datensatz alle nötigen Tabellen mitberücksichtigt werden, um keine ungültigen Bezüge zu erhalten.

Formulare

Eingaben und Ausgaben von Daten einer Datenbank werden durch Formulare erleichtert. Formulare erlauben, Daten einzelner Datensätze ansprechend formatiert anzuzeigen bzw. für eine Änderung bereitzustellen. Dabei können nicht nur Daten einer Tabelle, sondern gleichzeitig Daten anderer, verknüpfter Tabellen angezeigt werden. Auf diese Weise kann man Schablonen erstellen, die von vornherein alle nötigen Eingabefelder für einen Datensatz vorgeben, selbst wenn sie aus unterschiedlichen Tabellen stammen.

Abfragen

Die Daten einer Datenbank sind solange nutzlos, solange sie nur *in* der Datenbank liegen. Um nach bestimmten Kriterien Daten auszugeben, müssen Abfragen erstellt werden. Diese lassen sich – wie Tabellen und Formulare – abspeichern und nach Bedarf wieder aufrufen. Damit ist es möglich, Sichten auf die Daten für verschiedene Auswertungszwecke zu erstellen oder bestimmten Personen (nur) bestimmte Daten zugänglich zu machen. Fügt man den Abfragen Formulare bei, kann man auch hier die Ein- oder Ausgaben ansprechend gestalten.

Berichte

Abfragen können wie eine Tabelle in weiteren Abfragen genutzt werden, dienen aber auch dazu, Berichte zu erstellen. Berichte sind Zusammenstellungen von Daten aus der Datenbank, die für Dokumentationszwecke (Präsentationen, Sitzungen, schriftliche Berichte) ausgedruckt werden können.

Erst mit Abfragen und Formularen wird eine Datenbank zu einem handhabbaren Instrument der Datenverwaltung!

In den Übungen zu dieser Lehreinheit werden Sie sich mit verschiedenen hier dargestellten Teilaspekten praktisch auseinandersetzen.