1. Relationale Datenbankentwicklung mit ACCESS

Warum sollte man eine Datenbank anstelle einer Tabellenkalkulation entwickeln?

Beispiel: Angenommen, es sollen für Grundstücke jeweils der *Ort* und der oder die *Eigentümer* gespeichert werden. Das könnte man in einer Tabellenkalkulation mit einer Liste lösen:

	A	B	C	D	E
1	**Ort**	**Grundstück**	**Eigentümer1**	**Eigentümer2**	**Eigentümer3**
2	Frankfurt a.M.	Paulsplatz 20	H. Mayer, Berliner Str. 1, Hamburg	Anja Mayer, Berliner Str. 1, Hamburg	
3	Frankfurt	Waldstr. 1a	Horst Mayer, Berliner Straße 1		Anja Mayer
4	Hauptstr. 1	Dresden	Fischer, Rolf, Schulstr. 17, 10001 Berlin	- keine -	- keine -

So eine Liste macht aber einige Probleme:

- Liegt das zweite Grundstück auch in Frankfurt am Main, oder in Frankfurt an der Oder?
- Sind H. Mayer und Horst Mayer ein und dieselbe Person?
- In welcher Stadt wohnt Horst Mayer?
- Warum wurde Anja Mayer einmal als 'Eigentümer3', und nicht als 'Eigentümer2' eingetragen?
- Was, wenn es mehr als drei Eigentümer gibt?
- Falls alle Mayers die gleiche Anschrift haben, wird bei einer Anschriftenänderung auch überall die neue Anschrift eingegeben?
- Im dritten Datensatz sind 'Ort' und 'Grundstück' vertauscht, Vor- und Nachname anders sortiert, eine PLZ eingetragen und in Zellen, die leer bleiben sollten, findet sich der Eintrag '– keine –'

Hier liegen die Vorteile eines Datenbankprogramms wie **Access**: Uneinheitliche Schreibweisen, unvollständige Daten und Duplikate werden vermieden. In Excel ginge das − wenn überhaupt − nur mit viel Programmieraufwand.

---

2. ACCESS-Objekte

Eine Datenbank besteht aus verschiedenen Objekten. **Access** gliedert diese wie folgt:

1. Tabellen
2. Abfragen
3. Formulare
4. Berichte
5. Makros
6. Module

Datenbankobjekte bauen ca. in dieser Reihenfolge aufeinander auf und wurden bis Access 2003 auch so im **Datenbankfenster** dargestellt:

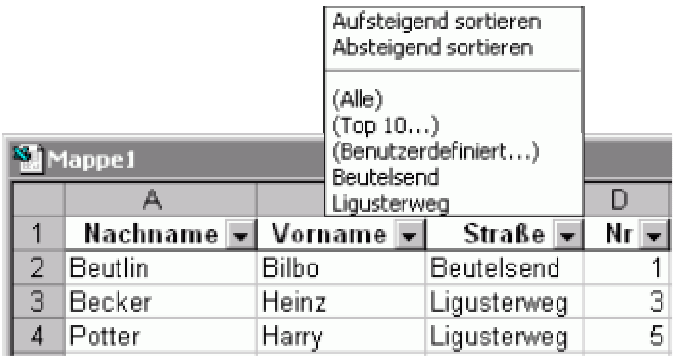
1. In **Tabellen** werden die eigentlichen Daten gespeichert. Man kann sich das fürs Erste vorstellen wie mehrere Tabellenblätter einer Tabellenkalkulation, die jeweils Listen enthalten.
2. In **Abfragen** werden diese Daten dann ausgewertet, also z.B. gefiltert, zusammengeführt und damit gerechnet. Tabellen und Abfragen sind eigentlich nur für den Datenbankentwickler gedacht. Die Daten sehen dort noch aus wie in einer Tabellenkalkulation.
3. Der Anwender sollte dagegen nur **Formulare** sehen. Dort erfolgt, optisch eventuell hübsch aufbreitet, die Datenerfassung und Auswertung.
4. **Berichte** dienen dem Ausdruck auf Papier.
5. **Makros** sowie
6. **Module** dienen der Automatisierung und Programmierung der Datenbank.

Jede Datenbank braucht zuerst eine gründliche Planung - auch die neueren Versionen von **Access**. Dabei sollte man stets das **mdb**-Dateiformat (statt **accdb)** nutzen.

---

3. Eignung von Access

Bevor man ein Datenbankprojekt realisiert, sollte man prüfen, ob **Access** überhaupt dafür geeignet ist.



Vielleicht reicht ja bereits eine Tabellenkalkulation. Die Möglichkeiten, die z.B. **Excel** mit 'Autofilter' (siehe Bild oben) und 'Spezialfilter' bietet, sind vielen Anwendern nicht bekannt.

Wenn eine oder mehrere der folgenden Bedingungen zutreffen, ist eine Tabellenkalkulation allerdings nicht mehr zu empfehlen:

- Ein einzelnes Tabellenblatt genügt nicht (mehr), um den komplexen Sachverhalt zusammenhängend darzustellen. Bei der Verteilung der Daten auf mehrere Tabellenblätter sind Informationen aber nicht mehr vollständig darstellbar.
- Häufige Verwendung von Matrixfunktionen, z.B. S`Verweis()`.
- Die Tabelle ist sehr umfangreich und unübersichtlich geworden.
- Der Tabelle müssen regelmäßig (z.B. monatlich) neue Spalten angefügt werden.
- Es sind Tausende von Datensätzen (Zeilen) zu erwarten
- Die Anwendung soll vielen Anwendern zur Verfügung stehen. Die Anmeldeprozeduren von Excel sind dafür aber nicht ausreichend bzw. komfortabel genug.
- Für eine vernünftige Benutzerführung sollen komplexe Formulare eingesetzt werden.
- Die Anwendung benötigt Auswertungen, die auch 'stilistisch gut' ausgedruckt werden sollen.

Nach einer sehr persönlichen Richtlinie ist Excel geeignet, solange man alle Daten eines Tabellenblattes auf dem Bildschirm sieht. Nur Listen sollte man nach unten, aber nicht zur Seite scrollen müssen.

Access sind natürlich auch Grenzen gesetzt. Wer das erste Mal mit einer Datenbankanwendung zu tun hat, neigt unter Umständen dazu, **Access** zu überschätzen.

Allerdings wird **Access** auch von professionellen Datenbankprogrammierern immer wieder *unter*schätzt. Übersehen wird nämlich oft, dass Access eine Lücke schließt, die zwischen Tabellenkalkulationen und ganz großen Datenbankanwendungen oder Unternehmensinformationssystemen wie S**AP R/3** liegt.

Unter bestimmten Voraussetzungen sollte **Access** bestenfalls noch als Frontend eingesetzt werden, z.B.:

- Normalerweise ist stets mehr als eine bestimmte Anzahl Benutzer *gleichzeitig* angemeldet. Je komplexer die Anwendung und je mehr Netzwerktraffic, desto kleiner sollte die Anzahl gleichzeitiger Zugriffe sein; auch eine allereinfachste Anwendung sollte nicht mehr als ca. 50 gleichzeitige Benutzer haben.
- Es ist sehr wahrscheinlich, dass mindestens eine der Tabellen größer als 2GB wird.

Auch dann kann **Access** aber durchaus noch als Frontend, also Benutzeroberfläche, sinnvoll sein.

Die folgenden Punkte geben Hinweise, wann der Einsatz von **Access** ideal ist:

- **Access** ist beim Benutzer schon vorhanden.
- Die Anwendung soll auf einem einzelnen Rechner oder einem sehr kleinen Netzwerk laufen.
- Es werden maximal nur einige (hundert)tausend Datensätze anfallen.

---

4. Namenskonventionen

(vgl. **Visual Basic**: *cmd* = command, *txt* = text, *img* = image usw.)

Beim Erstellen einer Datenbank legt man zumeist viele Objekte an, die alle benannt werden müssen. Dabei sollte man einfache, 'sprechende' Namen finden.

Ein Name sollte nicht ÄÖÜß und keine Leerzeichen enthalten (sonst muss man ihn fast überall in [eckigen Klammern] schreiben).

Mehrere Worte schreibt man einfach 'am Stück', mit Großbuchstaben als optische Trennung (z.B. *KundeTelefonNr*).

Oft muss man rund um eine Sache mehrere Dinge benennen. Dann ist ein einheitliches Namensschema praktisch, so dass man Zusammengehöriges erkennt, aber auch eine eindeutige Unterscheidbarkeit gegeben ist. Eine Namenskonvention hilft auch, eine Anwendung zu verstehen, die man nicht selbst geschrieben hat.

Hierzu hat sich in der Praxis ein System aus Präfixen etabliert: Die so genannte *ungarische Notation*.

Natürlich ist dieses System nicht verbindlich und wird in der Praxis oft nur teilweise umgesetzt. Hier sind auch nur die wichtigsten Punkte dargestellt. Und es spricht auch nichts gegen ein eigenes Namenssystem, wobei allerdings die Zusammenarbeit mehrerer Datenbankentwickler nominelle Einheitlichkeit erfordert.

Hier die gängigen Namenskonventionen für **Datenbankobjekte**, **Formulare** und **Feldnamen**:

Datenbankobjekte		
Objekt	**Präfix**	**Beispiel**
Tabellen	tab	tabKunden
Abfragen	qry	qryBestelldetails
Formulare	frm	frmKunden
Berichte	rpt	rptKatalog
Module	mod	modFunktionen
Klassenmodule	cls	clsDateiDialog

Formulare: Objekte		
Objekt	**Präfix**	**Beispiel**
Bezeichnungsfeld	lbl	lblName
Textfeld	txt	txtName
Optionsfeld	opt	optAuswahlAlle
Kontrollkästchen	chk	chkBearbeitet
Kombinationsfeld	cbo	cboAuswahl
Listenfeld	lst	lstAuswahl
Befehlsschaltfläche	cmd	cmdOK

Tabellen: Feldnamen		
Objekt	**Präfix**	**Beispiel**
AutoWert	ID	IDObjekt
Boolean	bln	blnEingabeOK
Byte	byt	bytAbteilung
Integer	int	intLagerbestand
Long Int.	lng	lngKunde
Double	dbl	dblKurs
Dezimal	dec	decRabatt
Datum/Zeit	dat	datBestellung
Text	txt	txtNachname
Währung	cur	curPreis