Access SQL

Ergänzende Einführung in eine Windows-Datenbank

Prof. Dr. Manfred Sommer
Wirtschaftsinformatik I
Sommersemester 1992

Der folgende Text ergänzt das im praktischen Teil von Wirtschaftsinformatik I als Basislektüre verwendete Buch von André Klahold: WindowBase. Quick & Easy. Vaterstetten 1992. IWT-Verlag. Ein Buch direkt zu Access SQL ist m.W. nicht auf dem Markt und wird wohl auch nicht mehr erscheinen, da WindowBase das allerdings weitgehend identische "Nachfolgeprodukt" ist. Auf Unterschiede wird hier hingewiesen.

Für Hinweise auf Fehler und Unklarheiten bin ich dankbar!

1. Access SQL starten

Anders als WindowBase (Klahold 1992: 16) muß bei Access SQL zuvor der lokale Datenbankserver DBWINDOW als backend durch Doppelklick auf die gleich-namige Ikone in der Programmgruppe Access SQL gestartet werden. Anschließend wird das frontend durch Doppelklick auf die Ikone ASQL gestartet. Wenn man Access SQL zwischenzeitlich verlassen hat, braucht DBWINDOW nicht vor einer Rückkehr in Access SQL erneut gestartet werden, sofern die Taste DBWindows nicht, z.B. durch die Rückkehr zur DOS-Ebene beendet wurde. Das Fehlen der Taste "Datenbank anlegen" im Eingangsfenster "Access SQL DB-Verbindung" weist bereits darauf hin, daß das Anlegen einer neuen Datenbank in Access SQL etwas anders erfolgt. Den ersten Kontakt mit Access SQL gewinnt man aber besser durch die mitgelieferte Datenbank DEMO. Diese sollte zu Beginn als einzige im rechten Listenfeld stehen. Sie kann durch einfaches Anklikken direkt in das Eingabefeld "Datenbank" übernommen werden. Als Anwendernamen und Passwort muß ebenfalls DEMO eingegeben werden. Jetzt erst wird die Taste "ok" aktiv.

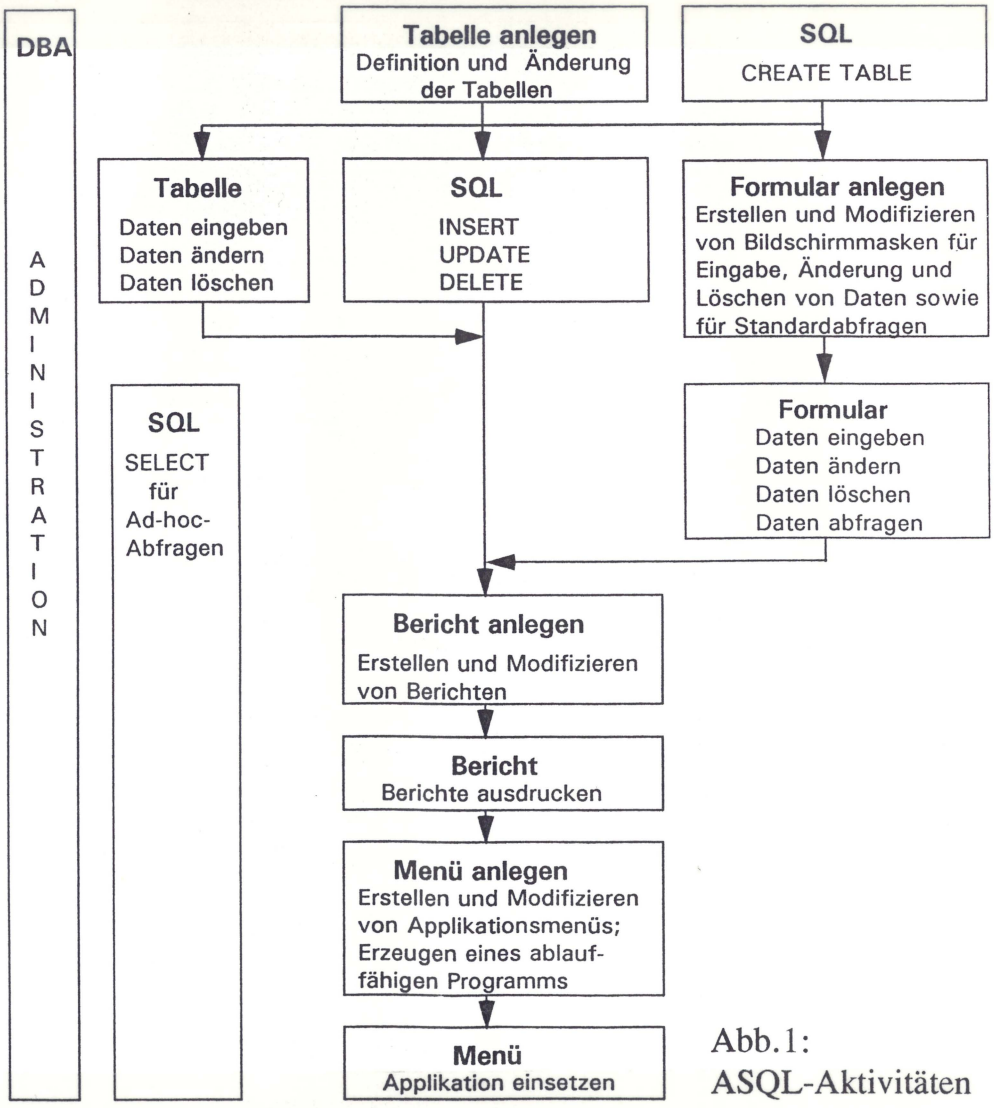

Abb.1:
ASQL-Aktivitäten

Nachdem sie gedrückt wurde erscheint das Access SQL-Hauptmenü. Klicken Sie das Vollbildfeld an. In der Titelleiste wird eine der zehn möglichen Aktivitäten von Access SQL aufgeführt, die durch Anklicken des Menüfeldes Aktivitäten sichtbar werden. Die beim Programmstart aktuelle Aktivität ist die voreingestellte Aktivität. Diese kann über die Option Voreingestellte Aktivität im Menü Einstellungen geändert werden. Eine Kurzbeschreibung der Aktivitäten findet sich bei Klahold 1992: 247. Anschaulicher ist eine vorgehensorientierte Gliederung der Access SQL-Aktivitäten wie in Abb.1.

2. Das Access SQL-Hilfesystem

Nachzulesen bei Klahold 1992: 18f. Am besten gleich ausprobieren.

3. Ein erster Überblick über die Access SQL-Aktivitäten an Hand der DEMO-Datenbank

Dies wird gemeinsam in den PC-Praktika abgehandelt.

4. Anlegen einer neuen Datenbank

Immer dann, wenn eine datenbankgestützte DV-Lösung entwickelt werden soll, deren Datenstruktur keine Verknüpfungen zur Datenstruktur bereits vorhandener Access SQL-Datenbanken aufweist, wird eine neue Datenbank angelegt. Diese Datenbank muß einen eindeutigen Namen gem. der DOS-Dateinamenskonvention haben. Access SQL vergibt automatisch die Namenserweiterung .DBS. Eine neue Datenbank muß in Access SQL - anders als in Windowbase (vergl. Klahold 1992: 27) - auf der DOS-Ebene aus dem Verzeichnis C:\SQLBASE heraus erzeugt werden:

```
C:\SQLBASE>INIT datenbankname
```

Die Beispieldatenbank "Galerie" wird also angelegt mit:

```
C:\SQLBASE>INIT galerie
```

Bitte genau diesen Datenbanknamen nehmen. Access SQL legt ein Unterverzeichnis C:\SQLBASE\GALERIE an und stellt dort die Datei GALERIE.DBS als Kopie der leeren Datenbankhülse START.DBS bereit. Hieraus werden zwei eindeutige Randbedingungen ersichtlich:

a) Der Datenbankname und der Name des Datenbankverzeichnisses müssen übereinstimmen

b) Das Datenbankverzeichnis (bei uns GALERIE bzw. DEMO) muß ein Unterverzeichnis des sog. Datenbankstammverzeichnisses SQLBASE sein

Nach Rückkehr zu WINDOWS und Access SQL enthält die Auswahlliste nun neben DEMO auch die Datenbank GALERIE.

Eine neu angelegte Datenbank ist nur dem Systemadministrator mit dem Anwendernamen SYSADM und dem Passwort SYSADM zugänglich (natürlich kann er sein Passwort ändern, doch das kommt später), weil das Anlegen einer Datenbank nun einmal ihm vorbehalten bleibt, damit nicht jeder Benutzer beliebige und beliebig viele Datenbanken anlegen kann. Jede neue Datenbank belegt vom

Start weg 450 KB auf der Festplatte, bevor eine einzige Anwendertabelle erzeugt ist! Mit "SYSADM" als Anwendername und Passwort kommen Sie also in das Access SQL-Hauptmenü für die Datenbank "GALERIE". In der Aktivität TABELLE (Titelleiste beachten!) kann man sich in der Option ÖFFNEN des Menüpunktes TABELLE davon überzeugen, daß die Datenbank noch keine Tabelle enthält. Die Auswahlleiste des Fensters "Tabelle öffnen" enthält nämlich nur sogenannte Systemtabellen mit den qualifizierten Tabellennamen SYSADM.tabellenname und SYSSQL.tabellenname.

5. Aufnahme eines neuen Anwenders

In der Aktivität DB-ADMINISTRATION, Menüpunkt ADMINISTRATOR, Option ANWENDER HINZUFÜGEN können sie sich unter Ihrem Anwendernamen (max. 8 Zeichen) und Passwort (max. 8 Zeichen) als weiteren Anwender neben dem SYSADM eintragen. Da unser Übungsspiel keine Datenschutzbelange berührt, empfiehlt sich der Verzicht auf ein exotisches Passwort. Falls Sie Ihrem Gedächtnis mißtrauen, begnügen Sie sich doch mit "GALERIE" als Anwendernamen und Passwort. Durch Anklicken des Kontrollfeldes "DB-Administrator (DBA)" können Sie dem neuen Anwender DBA-Rechte verleihen - auch dazu später Näheres. Es ist aber für den Fortgang des Kurses nicht notwendig. Nach Betätigung der Taste "Hinzufügen" könnten Sie jetzt - immer noch in Ihrer Eigenschaft als SYSADM! - weitere Anwender hinzufügen. im wirklichen EDV-Leben macht das auch bei Einzelplatzdatenbanken wie der unseren dann Sinn, wenn mehrere Anwender mit ihr arbeiten sollen. Im Augenblick macht es aber keinen Sinn, also "Abbrechen". (Vergleichen Sie Klahold 1992: 26 zu den Unterschieden in Windowbase). Verlassen sie die Datenbank in Ihrer Eigenschaft als SYSADM über SCHLIEßEN im Steuerungsfeld und kehren sie unter ihrem soeben neu vergebenen Anwendernamen und Passwort in die Datenbank zurück.

6. Anlegen von Tabellen

Nach dem Anlegen einer leeren Datenbank und vor der Eingabe einzelner Datensätze müssen die Tabellen angelegt werden, in denen diese Datensätze gespeichert werden sollen. Ohne diese nur der Struktur nach definierten, ansonsten aber ebenfalls leeren Tabellen geht gar nichts, denn relationale Datenbanken bestehen aus "nothing but tables" (Chris Date). Sogar die Beschreibung der Datenbank selbst durch sog. Metadaten erfolgt in Tabellen, nämlich in den bereits erwähnten Systemtabellen. Tabellen sind also das gemeinsame Organisationsprinzip für die Nutzerdaten und die Metadaten.

Welche Tabellen anzulegen sind, sagt einem weder Access SQL noch sonst ein Datenbanksystem. Diese Kernaufgabe des Datenbankentwurfs ist bei komplexeren Problemstellungen als dem stark didaktisierten Galeriebeispiel keineswegs trivial und wird in der populären Datenbankliteratur leider häufig verharmlost oder - vermutlich um den Einsteiger in die schöne neue Welt der Datenbanken nicht zu verschrecken - gar nicht erst erwähnt. So wird auch bei Klahold (1992: 20 22) zwar die Redundanzfreiheit als zentraler Vorzug relationaler Datenbanken erwähnt, nicht jedoch, daß dieses Ziel bei realistischen DB-Entwürfen ohne Normalisierung nicht erreichbar ist. Auch die SPI-Handbücher zu Access SQL schweigen sich hierzu aus. Wenigstens einen Hinweis auf ein gutes Buch z.B. aus der SQL-Seminarreihe von SPI hätte man dem Access SQL-Benutzer zumuten können: R.G. Flatscher: Design relationaler Datenbanken (SQL-Seminar 1). Vaterstetten 1990.

Anstelle der geheimen Botschaft der Anbieter von Datenbanksoftware "Jede Datenbank ist besser als keine!" sollten Sie sich einprägen: "Eine Datenbankanwendung wird nicht besser als ihre Tabellenstruktur!". Theoretischer ausgedrückt: ein mißratenes konzeptionelles Schema ist eine

schwere Hypothek für die externen Schemata. Wie man zu "guten" Tabellen kommt, wird in der Vorlesung behandelt. Wir übernehmen hier die Tabellen des Galeriebeispiels (Klahold 1992: 33).

Beim Anlegen von Tabellen, die natürlich einen Tabellennamen bekommen, werden die Spalten einer Tabelle festgelegt. Die Zeilen einer Tabelle bleiben hiervon völlig unberührt, da diese sich auf konkrete Datensätze beziehen und somit die Eingabe, das Ändern und Löschen von Daten betreffen. Die Spalten bezeichnen die Eigenschaften bzw. Attribute desjenigen Datenobjekts, das in der Tabelle dargestellt wird. Der Spaltenname sollte deshalb diese Eigenschaft genau bezeichnen. Dies wird dadurch erleichtert, daß die Spaltennamen bis zu 30 Zeichen lang sein können (keine Umlaute, kein Leerzeichen, statt dessen aber ein Unterstrich möglich, erstes Zeichen muß ein Buchstabe sein). Verbalakrobatik wie bei den DOS-Dateinnamen kann also weitgehend vermieden werden.

6.1 Tabellen mit der Aktivität TABELLE ANLEGEN kreieren

Der Wechsel in die Aktivität TABELLE ANLEGEN zeigt sofort unterhalb der Menüleiste an, worum es hier geht. Neben dem Namen einer Spalte ist der Datentyp anzugeben und u.U. festzulegen, daß in dieser Spalte bei Datenangabe zwingend ein Wert einzugeben ist; schließlich ist bei numerischen Daten die Stellenzahl und eventuell die Zahl der Nachkommastellen zu bestimmen. Nachdem Sie über TABELLE und ANLEGEN den Tabellennamen "Kunden" eingegeben haben, erscheint dieser jetzt (statt "ohne Namen") in der Titelleiste.

Jetzt wird über das Menüfeld SPALTEN und die Option ANLEGEN... das Dialogfenster "Spaltenattribute" geöffnet. (Abb. 2; vgl. Klahold 1992: 34 zum entsprechenden WindowBase-Fenster und beachten Sie die Unterschiede). Als erste Spalte wird die "Kundennummer" vom Datentyp SMALLINT festgelegt. Bei diesem Datentyp erübrigt sich die Länge sowie die Zahl der Stellen und Nachkommastellen. Schließen Sie die Spaltendefinition mit "ok" ab. Sekunden später fällt Ihnen siedend heiß ein, daß die Kundennummer den Primärschlüssel dieser Tabelle bildet. Die Eingabepflichtigkeit dieser Spalte sollte deshalb unbedingt durch das Kontrollfeld "NOT NULL" gesichert werden, was Access SQL im Gegensatz zu WindowBase unterstützt.

Solange eine Tabelle noch nicht abgespeichert ist, kann jede Spalte problemlos geändert werden - nach dem Speichern wird es heikler. Eine Spalte wird durch Anklicken im "Tabelle anlegen"-Fenster markiert und kann dann über SPALTEN und MODIFIZIEREN... geändert werden. Das Fenster "Spaltenattribute" zur Spalte "Kundennummer" wird erneut geöffnet. Jetzt kann die Eingabepflicht nachgetragen werden.

Die Datentypen von Access SQL und WindowBase (Klahold 1992: 28 - 30) unterscheiden sich, weil WindowBase im Gegensatz zur Access SQL (und im Gegensatz zur Behauptung von Klahold 1992: 147) nicht den Datenbankserver SQLBase von Gupta Technologies verwendet. Die Datentypen können wie in Abb.3 gegenübergestellt werden.

Alle übrigen Spalten der Tabelle "Kunden" sind vom Typ CHAR. (Vgl. Klahold 1992: 33). Nur der Nachname (Spalte "Name") soll eingabepflichtig sein, weil er das Minimum an Informationen über einen Kunden darstellt. Legen Sie diese Spalten ebenfalls an und speichern Sie die Tabelle über TABELLE.SPEICHERN ab. Falls Sie eine Spalte anzulegen vergessen haben oder sonstige Änderungen an einer bereits existierenden Tabelle vornehmen wollen, geht dies über TABELLE.MODIFIZIEREN. Beachten Sie im Menü TABELLE die für Dokumentationszwecke interessante Möglichkeit, die Tabellendifinition über ARBEITSBEREICH DRUCKEN... auf Papier festzuhalten.

Abb.2: Fenster "Spaltenattribute"

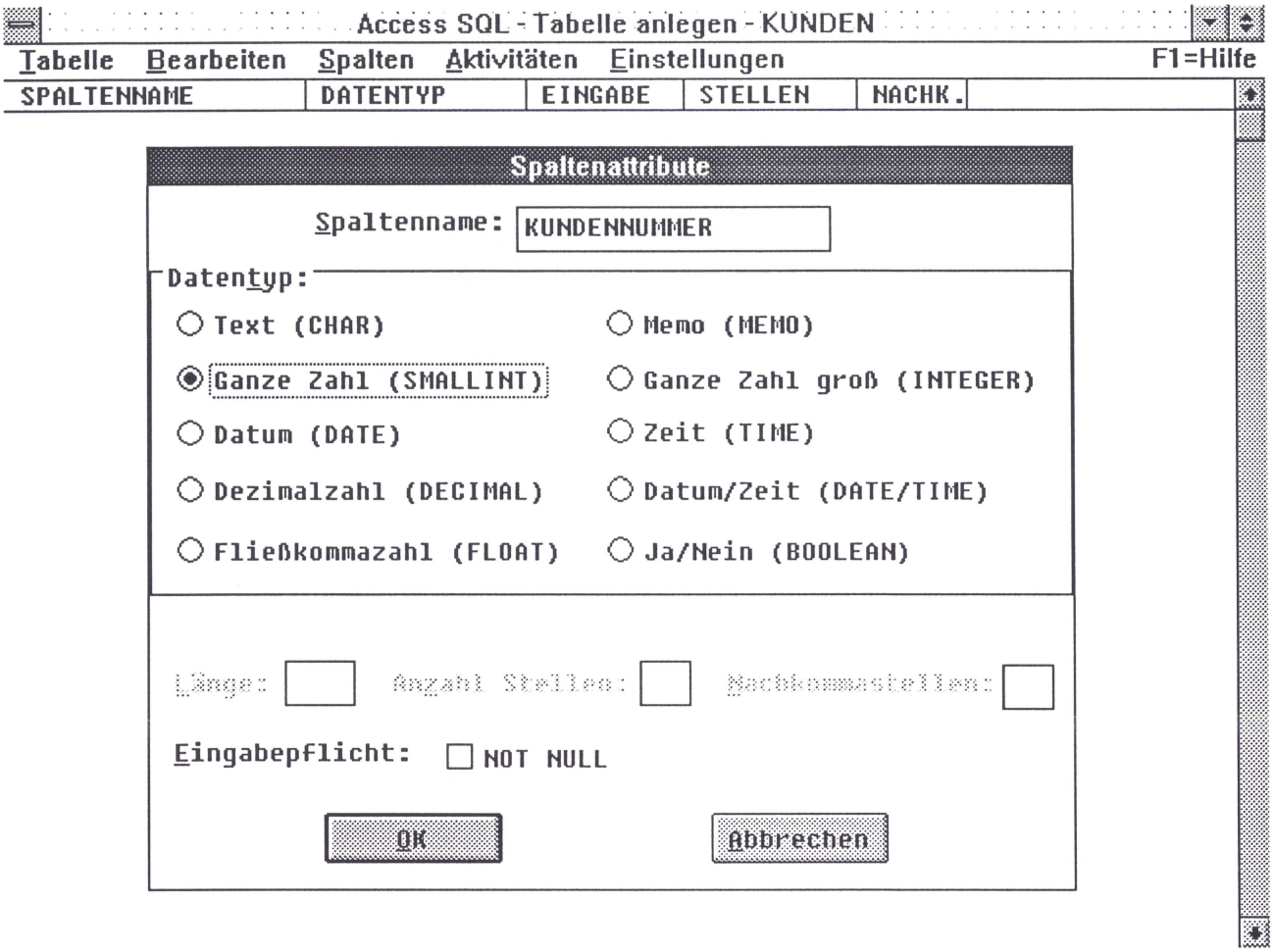

Abb.3: Gegenüberstellung der Datentypen in Access SQL und WindowBase

Datentyp	**Access SQL**	**WindowBase**
Text	CHAR (max.254 Bytes) MEMO (max. 32.768 Bytes)	CHAR, VARCHAR -
numerisch	SMALLINT (ganze Zahlen, -32.768 bis 32.767) INTEGER (Festkommazahlen, -2.147.483.784 bis 2.147.483.747) DECIMAL (max. 17 Stellen insg., davon zwischen 0 und 17 Nachkommastellen) FLOAT (Gleitkommazahlen, -9,99E99 bis 9,99E99) -	SMALLINT INTEGER MONEY (lt. Klahold 16 Vor- und 4 Nachkommastellen) FLOAT BINARY; VARBINARY (Hexadezimalzahlen)
logisch	BOLLEAN (TRUE/FALSE/NULL)	BIT
Datum/Zeit	DATE TIME DATE/TIME	DATE TIME DATE/TIME

6.2 Tabellen mit der Aktivität SQL kreieren

Natürlich würde man normalerweise die Tabelle "Ware" analog anlegen, wobei in Access SQL statt des Datentyps MONEY für den Einkaufs- und Verkaufswert der Datentyp DECIMAL mit 2 Nachkommastellen zu verwenden ist. Wir wollen es uns aber übungshalber ein wenig schwerer machen, um zu verstehen, was hinter der Windows-Kulisse geschieht. Tatsächlich wird der im vorangegangenen Abschnitt auf der Benutzeroberfläche geführte Dialog nämlich in einen SQL-Befehl CREATE TABLE übersetzt, der mit TABELLE.SPEICHERN ausgeführt wird. Wir erzeugen die Tabelle "Ware" jetzt mit einem solchen CREATE TABLE-Befehl in der Aktivität SQL. Diese Aktivität wird in der Praxis vorwiegend für Ad-hoc-Abfragen eingesetzt, gestattet aber grundsätzlich die Ausführung aller SQL-Befehle und eignet sich deshalb gut zum Erlernen von SQL.

Nach dem Wechsel in die Aktivität SQL erscheint ein horizontal zweigeteilter Bildschirm mit einem oberen Eingabebereich für die auszuführenden SQL-Befehle und einem unteren Ausgabebereich für Antworten des Datenbanksystems. Wir geben bewußt einen fehlerhaften Befehl zum Anlegen der Tabelle "Ware" (Klahold 1992: 33) ein:

```
CREATE TABLE ware
       (werk              CHAR (40)        NOT NULL
        name              CHAR (20),
        vorname           CHAR (20),
        stil              CHAR (20),
        art               CHAR (20),
        kaufdatum         DATE,
        verkaufsdatum     DATE,
        einkaufswert      MONEY,
        verkaufswert      MONEY,
        kunde             SMALLINT);
```

Befehle werden ausgeführt, indem der Cursor hinter das Semikolon gesetzt wird und ABFRAGE (mißverständlich, weil es hier gar nicht um eine Abfrage geht) und AUSFÜHREN angeklickt werden. Die wie leider häufig nicht sehr aufschlußreiche Fehlermeldung "Rechte Klammer fehlt" im Ausgabebereich kommt dadurch zustande, daß hinter NOT NULL ein Komma fehlt, mit dem die erste und zweite Spalte zu trennen sind. Das Komma läßt sich leicht einfügen. Aber auch dann kann der Befehl noch nicht ausgeführt und die Tabelle nicht angelegt werden, weil der Datentyp MONEY zwar in WindowBase, nicht aber in Access SQL definiert ist. Statt dessen soll hier eine sechsstellige Dezimalzahl mit zwei Nachkommastellen verwendet werden. Eine gute Gelegenheit, um sich mit dem Hilfesystem nach dem richtigen Datentyp zu erkundigen, bevor Sie jetzt weiterlesen.

```
CREATE TABLE ware
       (werk              CHAR (40)        NOT NULL,
        name              CHAR (20),
        vorname           CHAR (20),
        stil              CHAR (20),
        art               CHAR (20),
        kaufdatum         DATE,
        verkaufsdatum     DATE,
        einkaufswert      DECIMAL (6,2),
        verkaufswert      DECIMAL (6,2),
        kunde             SMALLINT);
```

Dieses Mal gibt es keine Fehlermeldung. Bevor wir wieder in die Aktivität TABELLE ANLEGEN wechseln, um uns das Ergebnis anzusehen, empfiehlt es sich, den CREATE TABLE-Befehl mit DATEI, SPEICHERN UNTER... z.B. in der Datei WARE_C (_C soll an CREATE erinnern) zu sichern,

für die automatisch die Namenserweiterung SQL vergeben wird. Wie nützlich es sein kann, längere SQL-Befehle mit ihrer gewöhnungsbedürftigen Syntax in einer Befehlsdatei auf Abruf parat zu haben, werden wir gleich sehen.

Allgemein lautet die Syntax des CREATE TABLE-Befehls mit folgenden Backus-Naur-Symbolen:

Wiederholungssymbol:	{ }
(0 oder mehrmals)	z.B. FROM tabelle {,tabelle}
Optionalsymbol:	[ ]
(kann entfallen)	z.B. SELECT [ALL]
Aternativensymbol:	|
(eines der folgenden)	z.B. ALL |DISTINCT

```
CREATE TABLE tabellenname
        (spaltendefinition {, spaltendefinition})

spaltendefinition ::= spaltenname datentyp [NOT NULL]

datentyp ::= CHAR (länge)
        |VARCHAR (länge)
        |LONG VARCHAR
        |DECIMAL [(stellen, nachkommastellen)]
        |FLOAT
        |INTEGER
        |SMALLINT
        |DATE
        |TIME
        |DATETIME
        |BOOLEAN
```

Sehen wir uns jetzt in der Aktivität TABELLE anlegen über TABELLE.MODIFIZIEREN die Struktur der Tabelle "Ware" an. Der Blick auf die Spalten für den Einkaufs- und den Verkaufswert läßt den Verdacht voreiliger kaufmännischer Bescheidenheit aufkommen: bei insgesamt sechs Stellen und zwei Nachkommastellen verbleiben nur vier Stellen vor dem Komma, was den An- und Verkauf von Kunstgegenständen über 9999,99 ausschließt. Auch wenn unsere Galerie den Handel mit van Gogh's realistischerweise nicht in Erwägung zieht, muß es ja nicht unbedingt bei Heidelandschaften von Hobbymalern bleiben. Sechs Stellen vor dem Komma sollten es sein können, also DECIMAL (8,2).

Versuchen wir gleich eine Tabellenänderung: Markieren der Spalte "einkaufswert", SPALTE.MODIFIZIEREN führt wieder in das Fenster "Spaltenattribute". Die Anzahl der Stellen ist jedoch grau, also nicht aktivierbar. Die Änderung eines Datentyps einer vorhandenen Spalte ist nach dem SQL-Standard nämlich nicht möglich und nicht etwa eine Schwäche von Access SQL! Da unsere Tabelle "Ware" noch keine Datensätze enthält, ist es am einfachsten, diese neu anzulegen. Änderungen an gefüllten Tabellen sind kritischer, weil Datenverluste drohen, und erfordern deshalb eine kompliziertere Vorgehensweise:

- Anlegen einer neuen Spalte in der vorhandenen Tabelle mit dem neuen Datentyp.
- Aktualisieren der neuen Spalten mit den Werten der alten Spalte.
- Löschen der alten Spalte.

Das wird später geübt.

Um die Tabelle komplett neu anzulegen wechseln wir wieder die Aktivität SQL und öffnen die Datei WARE_C.SQL. Im CREATE TABLE-Befehl kann nun in DECIMAL (6,2) die 6 durch eine 8 ersetzt werden. Der Versuch, diesen korrigierte Befehl auszuführen, muß natürlich fehlschlagen. Das relationale Datenbankmanagmentsystem (RDBMS) erinnert uns daran, daß eine gleichnamige Tabelle bereits in dieser Datenbank existiert. Die alte, falsche Warentabelle muß zuvor gelöscht werden. Dies geht jetzt am schnellsten, indem wir vor dem CREATE TABLE-Befehl schreiben:

```
DROP TABLE ware;
```

Das Semikolon, daß den DROP TABELE- vom CREATE TABLE-Befehl syntaktisch trennt, darf nicht vergessen werden. Man kann die beiden Befehle gemeinsam ausführen. Wir üben das schrittweise Vorgehen, indem zuerst der DROP-Befehl markiert und ausgeführt wird. Danach wird nur der CREATE-Befehl markiert und ausgeführt.

Dieser DROP TABLE-Befehl zum Entfernen von Tabellen aus der Datenbank versteckt sich hinter folgendem Benutzerdialog. In der Aktivität DB-ADMINISTRATION kann man über den Menüpunkt ANWENDER ebenfalls Tabellen entfernen. Man kann das jetzt noch schadlos ausprobieren, da die leere Tabelle ja - wie gehabt - über die Befehlsdatei WARE.SQL sekundenschnell wieder neu erzeugt werden kann.

Mit den wesentlichen Grundlagen des Anlegen, Ändern und Löschens von Tabellen sind Sie nunmehr vertraut. Wir können uns jetzt dem Eingeben, Ändern und Löschen von Datensätzen in Tabellen zu wenden, was sich zwar ähnlich anhört wie das Amlegen, Ändern und Löschen von Tabellen, aber etwas fundamental anders ist.

7. Datensätze einfügen

Wie in der Abb. 1 bereits gezeigt, kann die Dateneingabe - ebenso wie auch das Ändern und Löschen von Datensätzen - über drei Aktivitäten erfolgen:

- Aktivität TABELLE
- Aktivität SQL
- Aktivität FORMULAR, was voraussetzt, daß ein solches bereits mit der Aktivität FORMULAR ANLEGEN erstellt wurde.

Zunächst werden nur die beiden erstgenannten Möglichkeiten besprochen, da das Anlegen von Formularen ein Thema für sich ist (vgl. xxx).

7.1 Datensätze mit der Aktivität Tabelle eingeben

Nach dem Wechsel in die Aktivität TABELLE wird mit TABELLE ÖFFNEN ein Dialogfenster aufgemacht, das sämtliche Tabellen anzeigt, die bearbeitet werden können. Die qualifizierten Tabellennamen in der Auswahlleiste zeigen, daß außer diversen Systemtabellen z.Zt. zwei Tabellen der Galeriedatenbank bearbeitet werden können. Die Systemtabellen unterliegen dabei bestimmten Einschränkungen, die jetzt noch nicht interessieren. Wir öffnen die Warentabelle, deren Attribute nun

Spaltenweise angezeigt werden und die natürlich noch leer ist. Über BEFEHLE.EINFÜGEN soll jetzt der erste Datensatz mit der internen Nr. 1 (Klahold 1992: 44) eingegeben werden.

Die jeweils im Zugriff befindliche Zelle wird durch eine im Uhrzeigersinn wandernde Strichmarkierung hervorgehoben. Zur nächsten Zelle gelangt man mit der Tabulator- oder der Eingabetaste, zur vorherigen Zelle mit dem Rückwärtstabulator oder der ESC-Taste. Spalten ohne Eingabepflicht können übersprungen werden, hier z.B. das Verkaufsdatum, der Verkaufswert und der Kunde. Beim Einkaufswert ist zu beachten, daß natürlich nur zwei Nachkommastellen angezeigt und später auch in die Datenbank eingetragen werden. Warum später? Weil der Datensatz vor dem eigentlichen Einfügen in die Datenbank momentan nur am Bildschirm vorbereitet wird.

Wenn Sie bis zur Spalte "Kunde" vorgedrungen sind, richten Sie bitte Ihre Aufmerksamkeit auf die Menüzeile und betätigen dann erneut die Eingabe- oder die Tabulatortaste. Wahrscheinlich wird Ihnen mehr auffallen, daß die Eingabezelle nun von der letzten Spalte des ersten Datensatzes wieder zur ersten Spalte des zweiten Datensatzes "nach vorne" gesprungen ist, als daß Sie das in der Menüzeile jetzt grau aufscheinende COMMIT! bemerkt hätten. Grundsätzlich erfolgt eine Transaktion, also z.B. das Einfügen von einem oder mehreren Datensätzen in die Datenbank erst durch ein COMMIT. Der entsprechende Hinweis erscheint aber erst dann grau (= inaktiv) in der Menüzeile, wenn ein kompletter Datensatz am Bildschirm editiert wurde, die Eingabezelle also schon wieder am Anfang des nächsten Datensatzes steht. Das COMMIT! erscheint noch grau, weil der Eingabemodus noch nicht verlassen wurde. Dies erfolgt erst durch gleichzeitiges Drücken der Strg- und der Pause-Taste. Die blinkende Eingabezelle verschwindet und COMMIT! wird jetzt schwarz (= aktiv). Der soeben auf den Bildschirm gebrachte Datensatz befindet sich aber immer noch nicht in der Datenbank. Das Schwarzwerden von COMMIT! zeigt nur die Möglichkeit des Datenbankeintrags an. Von der Möglichkeit zur Wirklichkeit kann man nun auf zwei Arten gelangen: entweder durch Anklicken von COMMIT! oder über BEFEHLE, COMMIT. Die zweite Variante weist darauf hin, daß im Menü BEFEHLE unter COMMIT auch ROLLBACK steht, was den Nichteintrag in die Datenbank erlaubt, weil ROLLBACK alle Operationen seit dem letzten COMMIT wieder rückgängig macht.

Innnerhalb einer Transaktion können auch mehrere Datensätze auf einmal eingegeben werden. Tun Sie dies mit dem zweiten und dritten Datensatz. Je mehr Datensätze man auf einmal - also ohne zwischenzeitliches COMMIT - einzugeben versucht, desto verheerender würden sich

- ein versehentlicher ausgelöstes ROLLBACK,
- ein versehentliches Beenden von Access SQL ohne COMMIT,
- ein Absturz von Access SQL oder Windows
- oder ein Hardwarefehler

auswirken: alle Eingabemühe wäre umsonst gewesen. Insbesondere bei arbeitsaufwendigen Dateneingaben lohnt sich also ein häufigeres COMMIT. Geben Sie nun die weiteren Datensätze bis einschließlich des achten ("working day") ein (Anhang 1) und experimentieren Sie mit COMMIT und ROLLBACK. Schließen Sie die Tabelle auch einmal und öffnen Sie diese dann erneut, um zu sehen, welche Datensätze tatsächlich in der Datenbank gelandet sind.

Mit dem neunten und zehnten Datensatz sei noch etwas anderes demonstriert. Beim zehnten Datensatz macht Sie die Werksbezeichnung "8762" stutzig, zumal in Abb. 4.4 (Klahold 1992: 45) "beauty & wiz" als Werkstitel angegeben ist. Sie wollen lieber noch einmal rückfragen, ob das stimmt, und versuchen deshalb, den Datensatz vorläufig mit einem leerem Werk-Feld einzugeben. Es wird Ihnen nicht gelingen, diesen Datensatz zuspeichern, da das Attribut "Werk" als eingabepflichtig (NOT NULL) definiert wurde. Geben Sie also "8762" ein mit anschließendem COMMIT. Nach Eingabe des neunten Datensatzes stellen Sie fest, daß die Datensätze Nr. 9 und Nr. 10 gegenüber der

Buchvorlage vertauscht sind. Das ist aber völlig unproblematisch, da die Reihenfolge der Zeilen (= Datensätze) in relationalen Datenbanken ebenso beliebig ist wie die der Spalten (= Attribute).

7.2 Datensätze mit der Aktivität SQL eingeben

Die Dateneingabe über den SQL-Befehl INSERT ist so unkomfortabel, daß man sie unbedingt gemacht haben muß, um zu wissen, warum Datenbankensysteme, die nur eine befehlsorientierte Benutzerschnittstelle bieten würden, unakzeptabel wären. Andererseits wird aus dem SQL-Modus deutlicher als aus dem Tabellenmodus, was eigentlich passiert.

Die Syntax des INSERT-Befehls zum Einfügen eines einzelnen Datensatzes lautet:

```
INSERT INTO tabellenname
        [(spaltenname {,spaltenname})]
        VALUES (wertlsystemschlüsselwort {,wertlsystemschlüsselwort})
```

Wenn man auf die Spaltenliste verzichtet, müssen die Werte hinter VALUES vollständig und in der Reihenfolge aufgeführt werden, in der sie in der Tabelle stehen. Spaltennamen sowie Zellenwerte sind untereinander durch Kommata zu trennen, weshalb ein Dezimalkomma durch einen Dezimalpunkt ersetzt werden muß. Leerbleibende Zellen sind mit NULL anzugeben. Sollen nur einzelne Spalten gefüllt werden, ist es einfacher, deren Namen hinter dem Tabellennamen zu spezifizieren. Ihre Werte sind dann hinter VALUES in derselben Reihenfolge aufzuführen; diese kann von der Spaltenreihenfolge in der Tabelle abweichen. Überlegen Sie einmal, mit welchem INSERT-Befehl die Eingabe des folgenden Datensatzes gelingt:

```
Werk:          Heideröschen
Name:          Feyerabend
Vorname:       Otto
Stil:          edelkitsch
Art:           Bild
Kaufdatum:     1.4.91
Einkaufswert:  345,50
```

Sehen Sie sich einen ähnlichen INSERT-Befehl für die Tabelle "Kunden" an (Klahold 1992: 82). Bevor Sie sich die folgende Lösung ansehen, machen Sie mindestens drei eigene Versuche und wahrscheinlich dreimal Bekanntschaft mit "aufschlußreichen" Fehlermeldungen. So geht es ohne Spaltenliste:

```
INSERT INTO ware
       VALUES ('Heideröschen', 'Feyerabend', 'Otto', 'edelkitsch', 'Bild',
       '1991-4-1', NULL, 345.50, NULL, NULL);
```

Speichern Sie diesen Befehl in einer Datei WARE_I (I für INSERT) ab und sehen Sie jetzt in der Aktivität TABELLE nach, ob der Datensatz korrekt eingefügt wurde. Natürlich geht es auch mit Spaltenliste:

```
INSERT INTO ware
       (name, vorname, werk, stil, art, kaufdatum, einkaufswert)
       VALUES ('Feyerabend', 'Otto', 'Heideröschen', 'edelkitsch', 'Bild',
       1991-4-1', 345.50);
```

Beachten Sie die andere Spaltenreihenfolge und den Verzicht auf NULL-Werte. Wenn Sie auch mit diesem Befehl in der Aktivität SQL Erfolg haben, werden Sie in der Aktivität TABELLE feststellen, daß der Datensatz jetzt zweimal als Nr. 11 und Nr. 12 auftaucht: ein sehr leicht erkennbarer Fall von Redundanz, der durch Löschen eines der beiden Datensätze gleich behoben wird (vgl. 8.1). SQL kann derartige Doppeleinträge aber auch präventiv verhindern (vgl. xxx).

8. Datensätze löschen

Was in die Datenbank hineingelangt, muß auch wieder aus ihr entfernt werden können. Beispiele:

- Ein Kunstwerk wird der Galerie telefonisch angeboten und sogleich in der Datenbank vorgemerkt, der Ankauf kommt dann aber doch nicht zustande.
- Ein Kunde muß nicht gleich sterben, um zum Löschkandidaten zu avancieren. Es reicht die Geschäftsregel, daß, wer 5 Jahre nichts mehr gekauft hat, auf den obligatorischen Neujahrsgruß verzichten muß, keine Einladungen mehr zu Ausstellungseröffnungen erhält etc., kurzum: als Datensatz zu löschen ist.

In der Praxis wird ein vorsichtigerer Galerist den Kunden vielleicht ebenfalls aus der Tabelle "Kunden" löschen, ihn zuvor aber in eine strukturidentische Tabelle "Altkunden" auslagern. Falls sich der Kunde nach 6 Jahren überraschend doch wieder meldet, könnte es sich gut machen, ihn - nach kurzem Blick in die Altkundentabelle - für seinen erlesenen Geschmack zu loben, den er mit dem seinerzeitigen Erwerb der "Heideröschen" von Otto Feyerabend nachdrücklich unter Beweis gestellt habe.

Gelöscht werden kann wiederum in den drei Aktivitäten TABELLE, SQL und FORMULAR.

8.1 Datensätze mit der Aktivität TABELLE löschen

Daß eine Tabelle in der gleichnamigen Aktivität geöffnet werden muß, bevor Ihre Daten manipuliert werden können, wird ab jetzt als bekannt vorausgesetzt. In der Tabelle "Ware" soll der redundante Datensatz gelöscht werden. LÖSCHEN im Menüpunkt BEFEHLE ist aber grau und damit inaktiv, solange die zu löschende Zeile nicht markiert wurde. Dies erreicht man am leichtesten durch Anklicken der Datensatznummer, hier also der Nr. 11 oder 12. LÖSCHEN in BEFEHLE wird dann aktiv. Das System antwortet mit einer Sicherheitsrückfrage, ob man die Zeile(n) wirklich löschen möchte; Zeile(n) deshalb, weil auch mehrere direkt untereinander stehende Zeilen als Bereich markiert und gemeinsam gelöscht werden können. Nach Bestätigung der Löschabsicht erscheint wieder das COMMIT! in der Menüzeile. Noch ist also in der Datenbank nicht gelöscht worden, sondern nur auf dem Bildschirm, wovon man sich mit BEFEHLE.ROLLBACK überzeugen kann.

Markierte Zeilen können auch über BEARBEITEN.ZEILE(N) LÖSCHEN aus der Datenbank entfernt werden wie auch über BEARBEITEN.ZEILEN(N) AUSSCHNEIDEN. Bei der letzteren Vorgehensweise wird der ausgeschnittene Datensatz bzw. deren mehrere in üblicher Windows-Manier in die Zwischenablage verlagert, aus der sie von anderen Windows-Programmen - nicht aber von Access SQL selbst! - wieder eingefügt werden können.

Eine zum Löschen vorgesehene Zeile kann zwar am einfachsten durch Anklicken ihrer Datensatznummer markiert werden, es geht aber auch durch Markieren von der ersten bis zur letzten Zelle in der entsprechenden Zeile, also von "Heideröschen" bis zur leeren Kundenzelle. Es reicht nicht aus, nur bis "345.00" in der Einkaufswertzelle zu markieren, weil auch die NULL-Werte dieses

Datensatzes für den Verkaufswert und den Kunden gelöscht werden müssen. Das Löschen kann sich also nicht auf einzelne Zellen eines Datensatzes beschränken: ganz oder gar nicht. Ein "teilweises Löschen" ist nämlich eine spezielle Form des Aktualisierens (vgl. 9.). Auch der Versuch, das Kunstwerk "Heideröschen" nicht als Edelkitsch zu klassifizieren und die entsprechende Zelle in Ermangelung eines besseren, aber immer noch zutreffenden Begriffs einfach zu leeren, schlägt mit BEFEHLE.LÖSCHEN fehl.

Und schließlich funktioniert es auch nicht, eine Spalte durch Anklicken des Spaltennamens zu markieren und anschließend zu löschen. Dies wäre keine Datenmanipulation mehr, wie sie mit Aktivität TABELLE möglich ist, sondern ein Eingriff in die Tabellenstruktur, der nur in der Aktivität TABELLE ANLEGEN möglich ist.

Löschen Sie nun den redundanten Datensatz mit COMMIT, um sich davon zu überzeugen, daß es wirklich funktioniert.

8.2 Datensätze mit der Aktivität SQL löschen

Was mit einer modernen Benutzeroberfläche einfach geht, läßt sich natürlich auch umständlich traditionell befehlorientiert bewerkstelligen - wie immer in der Aktivität SQL. Die Syntax des Zeilenlöschbefehls lautet:

```
DELETE FROM tabellenname
        [WHERE auswahlbedingung]
```

Der Verzicht auf die optionale WHERE-Klausel würde das Löschen sämtlicher Datensätze bewirken. Damit würde die Tabelle als leeres Gerippe jedoch noch weiterexistieren, anders als beim DROP TABLE-Befehl (vgl. xxx), der nicht nur sämtliche Datensätze einer Tabelle, sondern auch die Tabelle selbst löscht. Mit der WHERE-Klausel, deren Auswahlbedingung auch eine sog. Unterabfrage sein kann, ist jedoch ein gezieltes Löschen von einzelnen Datensätzen bzw. auch von Datensatzgruppen möglich.

```
DELETE FROM ware
       WHERE werk = 'Heideröschen'
```

hat denselben Effekt wie die Übung im vorangegangenen Abschnitt. Dies kann gefahrlos ausprobiert werden, das der Datensatz mit Hilfe der Datei WARE_I.SQL leicht wieder eingefügt werden kann. Will sich die Galerie aber nicht nur von den "Heideröschen", sondern von der gesamten Gattung des Edelkitsches trennen, würde man die Datenbank bitten:

```
DELETE FROM ware
       WHERE stil = 'edelkitsch'
```

Die Mächtigkeit des Löschbefehls liegt in der sehr komplex formulierbaren WHERE-Klausel, die im Rahmen des SELECT-Befehls zu behandeln ist. Schon jetzt ist aber einsehbar, daß eine falsch formulierte WHERE-Klausel verheerende Folgen für die Datenbank haben kann, da in der Aktivität SQL gelöscht wird ohne daß man vorher sieht, welche Datensätze davon betroffen sein werden. Man sollte deshalb besser in der Aktivität TABELLE eine Abfrage definieren, sich das Selektionsergebnis ansehen, es gegebenenfalls komplett markieren und dann löschen.

9. Datensätze aktualisieren

Die dritte Form der Datenmanipulation bildet das Aktualisieren, das auch als Updaten oder Datenpflege bezeichnet wird. Wenn in einem Datensatz ein Attributwert neu eingefügt werden soll, in dessen Zelle bisher nichts stand (NULL), dann handelt es sich ebenfalls um Aktualisieren und nicht um Einfügen, weil letzteres sich stets auf einem neuen Datensatz bezieht.

9.1 Datensätze mit Aktivität TABELLE aktualisieren

Da meistens nicht sämtliche Datensätze auf einmal aktualisiert werden, sondern nur eine häufig sehr kleine Teilmenge - oft sogar nur ein einzelner Datensatz - wird man in der Praxis vor allem bei Tabellen mit vielen Datensätze diese nicht durchblättern wollen, sondern sie gezielt mit einer SQL-Abfrage heraussuchen. Vor der eigentlichen Aktualisierung muß allerdings festgelegt werden, was aktualisiert werden soll. Dies geht in der Akivität TABELLE entweder durch den eben angesprochenen SELECT-Befehl über ABFRAGE.DEFINIEREN oder auch über einfaches Markieren. Deshalb ist BEFEHLE.AKTUALISIEREN (ebenso wie LÖSCHEN) inaktiv, bevor der Aktualisierungsbereich markiert wurde. Während jedoch beim Löschen nur Zeilen markiert werden können, gibt es beim Aktualisieren folgende Markierungsmögkichkeiten:

- ein einziges Feld,
- eine oder mehrere zusammenhängende Zeilen,
- eine oder mehrere zusammenhängende Spalten,
- ein beliebiger, rechteckiger Bereich oder
- alle Zellen.

Üben Sie jetzt das Markieren und Aktualisieren. Achten Sie darauf, wann Ihnen das Datenbanksystem die Möglichkeit zum COMMIT gibt, denn ohne COMMIT werden die Änderungen nicht abgespeichert. Sorgen Sie dafür, daß die Daten der Tabelle "Ware" am Ende Ihrer Übungen wie im Anhang 2 aussieht.

Insbesondere beim Aktualisieren breiterer Tabellen mit vielen Spalten bietet Access SQL über eine Anpassung der Spaltenbreite auf dem Bildschirm (nicht in der Datenbank!) und über eine vertikale Bildschirmteilung eine übersichtlichere Darstellung der wichtigsten Informationen. Näheres finden Sie bei Klahold 1992: 44 - 46.

9.2 Datensätze mit der Aktivität SQL aktualisieren

Nehmen wir einmal an, in der Spalte "einkaufswert" seien die Preise bisher incl. Mehrwertsteuer aufgeführt. Aus welchem Grund auch immer sollen diese Preise in Nettopreise geändert, also um 14 % reduziert reduziert werden. Überlegen Sie einmal kurz, wie dies in der Aktivität TABELLE zu bewerkstelligen wäre. Man müßte die entsprechende Spalte markieren und Zelle für Zelle die von Hand ausgerechneten Nettopreise eingeben. Das Bespiel zeigt, daß Änderungen, die sich auf viele Datensätze beziehen und denen dieselbe mathematische Formel zugrunde liegt (hier die Multiplikation mit dem Faktor 0.86), mit der Aktivität TABELLE nur erheblich umständlicher zu erledigen sind als mit dem SQL-Befehl UPDATE, dessen Syntax so lautet:

```
UPDATE tabellenname
SET wertzuweisung {, wertzuweisung}
        [WHERE auswahlbedingung]

wertzuweisung ::= spaltenname = skalarausdruck|NULL
```

Folgende Fälle sind besonders relevant: Aktualisieren

- eines einzelnen Wertes,
- mehrerer Werte in einer Zeile,
- aller Werte in einer Spalte oder
- mehrerer Werte in einer oder mehreren Spalten.

Je ein Beispiel möge dies illustrieren. Wechseln Sie zuvor in die Aktivität SQL. In der ersten Übung soll der Stil des Kunstwerkes "Heideröschen" nunmehr als volkstümlich ausgewiesen werden. Die Änderung betrifft also ein Attribut eines Datensatzes. Jede Änderung, die sich nur auf einen Teil der Datensätze bezieht, erfordert die bereits mehrfach angesprochene WHERE-Klausel, die in ihrer Einfachheit intuitiv einleuchtet:

```
UPDATE ware
SET stil = 'volkstümlich'
       WHERE werk = 'Heideröschen;
SELECT * FROM ware;
ROLLBACK;
```

Mit den unmittelbar anschließend SELECT- und ROLLBACK-Befehlen (Semikolons beachten!) erreichen wir, daß das UPDATE-Ergebnis im unteren Ausgabefenster angezeigt wird und die Änderung beim Verlassen der Aktivität SQL nicht in die Datenbank eingetragen wird. Sie sind also für Übungszwecke sehr nützlich. Alle drei SQL-Befehle können im oberen Eingabefenster markiert und dann auf einmal ausgeführt werden.

Im zweiten Beispiel verdoppelt sich mit der Stilbezeichnung zugleich der Einkaufspreis:

```
UPDATE ware
SET stil = 'volkstümlich', einkaufswert = einkaufswert * 2
       WHERE werk = 'Heideröschen';
SELECT * FROM ware;
ROLLBACK;
```

Im dritten Beispiel sollen alle Werte einer Spalte verändert, hier der Einkaufswert um die Mehrwertsteuer reduziert werden. Die WHERE-Klausel ist jetzt überflüssig und sogar gefährlich, weil sie die Gefahr impliziert, daß die Änderung nur in einem Teil der Datensätze durchgeführt wird, der nämlich von der WHERE-Klausel selektiert wird.

```
UPDATE ware
SET einkaufswert = einkaufswert * 0.86;
SELECT * FROM ware;
ROLLBACK;
```

Achten Sie auf den Dezimalpunkt bei 0.86, da SQL hinter einem Komma einen weiteren Spaltennamen erwarten würde.

Sollen sich die Änderungen auf mehrere Spalten in mehreren, aber nicht auf alle Datensätzen beziehen, wird wieder eine WHERE-Klausel benötigt. Nehmen wir an, unser Galerist will sich zukünftig nur noch auf Bilder spezialisieren. Er veranstaltet am 31. Mai 1992 eine Sonderaktion, mit der es ihm gelingt, sich von allen Skulpturen genau zum Einkaufpreis wieder zu trennen. Mit dem folgenden SQL-Statement wäre die Datenbank dann wieder "up to date":

```
UPDATE ware
SET verkaufsdatum = 1992-05-31, verkaufswert = einkaufswert
       WHERE art = 'Skulptur';
```

10. Datenbankabfrage mit SQL

Die Abfrage steht im Mittelpunkt des praktischen Interesses an Datenbanken. Schließlich sind die Datenmanipulationen INSERT, UPDATE und DELETE kein Selbstzweck sondern dazu da, eine geeignete und verläßliche Datenbasis für vielfältige Abfragemöglichkeiten bereit zu stellen. Das Ergebnis einer Abfrage mit dem SELECT-Befehl ist wiederum eine Tabelle. Das EVA-Prinzip stellt sich hier also so dar:

Basistabelle(n) →	SELECT →	Ergebnistabelle
(E)	(V)	(A)

Die Ausgabe der Ergebnistabelle kann in der schmucklos tabellarischen Form erfolgen, wie sie dem reinen SELECT-Befehl entspricht. Für die Testphase von SELECT-Befehlen oder für Ad-hoc-Abfragen mag das ausreichen. Der Endbenutzer erwartet zu Recht, daß ihm die Ergebnistabellen am Bildschirm oder auf Papier ergonomisch vernünftig präsentiert werden. Datenbanksysteme stellen hierfür als Werkzeuge Bildschirmmaskengeneratoren (z.B. die Aktivität FORMULAR ANLEGEN in Access SQL oder SQL*Forms in ORACLE) und Reportgenerator (z.B. die Aktivität BERICHT ANLEGEN in Access SQL oder SQL*ReportWriter in ORACLE) zur Verfügung. Die Darstellung kann datensatzweise oder in Listenform erfolgen.

Es dürfte deutlich geworden sein, daß jeder Bildschirmmaske und jedem Report eine Datenbankabfrage logisch vorausgehen muß. Ich halte es deshalb für fragwürdig, den Datenbankneuling vorschnell auf das Erstellen von Formularen und Berichten hin zu orientieren und ihm die Beschäftigung mit der Abfragesprache - zunächst - zu ersparen. Der sog. parametrische Endbenutzer einer Datenbank, die z.B. mit Access SQL erstellt wurde, muß in der Tat nichts von SQL verstehen - aber auch nichts vom Erstellen von Formularen und Berichten, sondern muß deren Benutzung beherrschen. Umgekehrt müssen Datenbankentwickler die u.a. Formulare und Berichte entwerfen, wissen, wie die in diesen Masken und Reports darzustellenden Daten aus den Basistabellen der Datenbank extrahiert werden können - und das wiederum geht nicht ohne SQL. Deshalb sollte in der Datenbankdidaktik gelten: LOGIK vor KOSMETIK.

Programmtechnisch schlägt sich die große Bedeutung von Abfragen in Access SQL nicht etwa darin nieder, daß es eine eigene Aktivität ABFRAGE gibt, sondern in ihrer Querschnittsfunktion: Abfragen können nicht nur in der Aktivität SQL, sondern auch in den Aktivitäten TABELLE, FORMULAR ANLEGEN, BERICHT ANLEGEN definiert und in den Aktivitäten FORMULAR und BERICHT zumindest geändert werden.

Die Grundlagen des SELECT-Befehls und seiner Definition in Access SQL können Sie sich mit KLAHOLD 1992: 83 - 104 erarbeiten. Der theoretische Hintergrund, Vertiefungen und neuere Entwicklungen sind Gegenstand der Vorlesung. Ich werde mich hier auf einige Klarstellungen und

Fehlerkorrekturen beschränken. Ferner empfehle ich, in der Tabelle Kunde die Daten wie im Anhang 2 zu ergänzen. Leider enthält das Buch von KLAHOLD keine vollständige und widerspruchsfreie Dokumentation des Datenbestandes.

Sie sollten nicht nur die Abfragen im Buch nachvollziehen, sondern sich selbst weitere Abfragen überlegen und ausprobieren. Der Abfragegenerator, der mit ABFRAGE.DEFINIEREN... aufgerufen wird, erleichtert den Aufbau syntaktisch einwandfreier SELECT-Befehl, garantiert ihn aber nicht. Vor allen Dingen sollten Sie stets darauf achten, welcher SQL-Code im Abfragefenster und seinen Unterfenstern durch die Abfrageparameter-Drucktasten erzeugt wird.

Die SELECT-Syntax auf Seite 86 bei KLAHOLD ist mißverständlich. In Bachus-Naur-Form lautet sie:

```
SELECT [ALL | DISTINCT] spaltenname | ausdruck {,spaltenname | ausdruck}
        FROM tabellenname | viewname [alias] {,tabellenname | viewname [alias]}
        [WHERE auswahlbedingung]
        [GROUP BY spaltenname | ganzzahl {,spaltenname | ganzzahl}]
        [HAVING auswahlbedingung]
        [ORDER BY spaltenname | ganzzahl [ASC | DESC] {,spaltenname | ganzzahl
        [ASC | DESC]}]
```

Die einfache Abfrage auf Seite 89 bezieht sich auf die Tabelle "Ware" statt "Kunden".

Das Schlüsselwort DISTINCT wird durch den Pushbutton "Keine Wertwiederholung" erzeugt (Seite 90). Probieren Sie folgende Abfragen aus und beachten Sie die Unterschiede in den Ergebnistabellen:

```
SELECT name, stil
FROM ware;
SELECT DISTINCT name, stil
FROM ware;
```

Erweitern Sie die zweite Abfrage nacheinander um folgende Sortierkriterien:

a) `ORDER BY name`
b) `ORDER BY stil DESC`

Die Jahreszahl in Datumskonstanten darf nur die beiden letzten Stellen enthalten, also z.B. '20.11.91'D statt '20.11.1991'D (Seite 94).

Abfragen, die aus dem Abfragefenster des Abfragegenerators gespeichert werden, erhalten unabhängig von der jeweiligen Aktivität die Dateinamenserweiterung .QRY (für query). Abfragen, die aus dem Eingabebereich der Aktivität SQL gespeichert werden, erhalten das Suffix .SQL. Beide Dateitypen können völlig identisch sein, da es reine ASCII-Dateien sind. Deshalb können auch TXT-Dateien mit SELECT-Befehlen geladen werden, die außerhalb von Access SQL erstellt wurden, wenn man zwei Einschränkungen beachtet:

a) sie müssen den Syntaxregeln von Access SQL entsprechen
b) im Abfragegenerator kann anders als im Eingabebereich der Aktivität SQL nur ein SELECT-Befehl geladen werden, der natürlich Unterabfragen enthalten kann.

Um den Überblick über die von Acess SQL erzeugten externen Schemata zubehalten, empfiehlt sich eine Verzeichnisstruktur unterhalb des zugehörigen Datenbankverzeichnisses:

Verzeichnis:	Inhalt:	Dateien:
SQL	SQL-Befehle wie CREATE TABLE, DROP TABLE, INSERT, UPDATE, DELETE	.SQL
ABFRAGE	SELECT-Befehle	.QRY und .SQL
FORMULAR	Formulare	.FOR
BERICHTE	Berichte	.REP
MENUE	Menüsystem	.MNU
DBA	DBA-Befehle zur Rechte- und Indexverwaltung	.SQL

ANHANG 1

Tabelle WARE

WERK	NAME	VORNAME	STIL	ART	KAUF-DATUM	VERKAUFS-DATUM	EINKAUFS-WERT	VERKAUFS-WERT	KUNDE
Kaktus-rot/blau	Miller	Armin	modern	Bild	01.01.92		2.000,00		
Idolon	Quart	Susanne	impressionistisch	Bild	20.11.91		12.000,00		
Licht	Ast	Herbert	modern	Skulptur	02.03.91		2.500,00		
Landschaft A	Ast	Herbert	modern	Skulptur	02.03.91		7.000,00		
Schatten	Günzel	Sylvia	naiv	Bild	05.05.91		1.000,00		
Gipfel	Engel	Kerstin	modern	Bild	06.07.91		2.000,00		
Schwan auf Alster	Lim	Mark	naiv	Bild	22.05.91		4.300,00		
working day	Haubrich	Dirk	modern	Skulptur	08.08.91		700,00		
8762	Hansmann	Markus	expressionistisch	Skulptur	29.10.91		10.000,00		
beauty & wiz	Schmitz	Daniela	impressionistisch	Bild	02.03.91		5.000,00		
Heideröschen	Feyerabend	Otto	edelkitsch	Bild	01.04.91		345,50		

ANHANG 2

Tabelle WARE

WERK	NAME	VORNAME	STIL	ART	KAUF-DATUM	VERKAUFS-DATUM	EINKAUFS-WERT	VERKAUFS-WERT	KUNDE
Kaktus-rot/blau	Miller	Armin	modern	Bild	01.01.92	29.02.92	2000,00	3000,00	9101
Idolon	Quart	Susanne	impressionistisch	Bild	20.11.91	13.01.92	12000,00	18000,00	9103
Licht	Ast	Herbert	modern	Skulptur	02.03.91	30.06.91	2500,00	3750,00	9107
Landschaft A	Ast	Herbert	modern	Skulptur	02.03.91	30.06.91	7000,00	10500,00	9107
Schatten	Gamsbart	Sylke	naiv	Bild	05.05.91	10.10.91	1000,00	1500,00	9108
Gipfel	Vogel	Bettina	modern	Bild	06.07.91	06.12.91	2000,00	2800,00	9101
Schwan auf Alster	Laubig	Markus	naiv	Bild	22.05.91	28.02.92	4300,00	6000,00	9105
working day	Hempel	Thomas	modern	Skulptur	08.08.91	11.11.91	700,00	1000,00	9105
8762	Herkules	Matthias	expressionistisch	Skulptur	29.10.91		10000,00		
beauty & wiz	Schmidt	Dorothee	impressionistisch	Bild	02.03.91	03.02.92	5000,00	10000,00	9103
Heideröschen	Feyerabend	Otto	edelkitsch	Bild	01.04.91	24.12.91	345,50	500,00	9102
Heimat	Hasenclever	Henriette	impressionistisch	Skulptur	03.05.89		2500,00		
Hühnerhof	Brause	Friedolin	modern	Bild	02.05.91		4500,00		
Watzmann	Baumann	Beppo	edelkitsch	Skulptur	11.11.90		1300,00		
Karnaval in Rio	Gutierrez	Stefano	modern	Bild	31.05.90	06.06.90	4050,00	6100,00	9104

Tabelle KUNDEN

KUNDEN-NUMMER	ANREDE	NAME	VORNAME	STRASSE	PLZ	ORT	TELEFON
9101	Frau	Schmitz	Daniela	Lessingstr.4	1000	Berlin	030/1234567
9102	Frau	Günzel	Iris	Goethestr. 6	4000	Düsseldorf	0211/7654321
9103	Herr	Hansmann	Markus	Wittekindstr. 98	5000	Köln	0221/9876456
9104	Frau	Günzel	Sylvia	Hafenstr. 108	2000	Hamburg	040/3451238
9105	Herr	Lobpreis	Georg	St. Martinsweg 2	6000	Frankfurt	069/1357924
9106	Herr	Haubrich	Dirk	Parkallee 198	2800	Bremen	0421/981237
9107	Frau	Engel	Kerstin	Auf der Zeil 45	6000	Frankfurt	069/4587234
9108	Herr	Lim	Mark	Kurfürstendamm 3	1000	Berlin	030/2397309
9109	Herr	Sonnemann	Bernd	Hanauer Str. 23	6000	Frankfurt	069/2378242