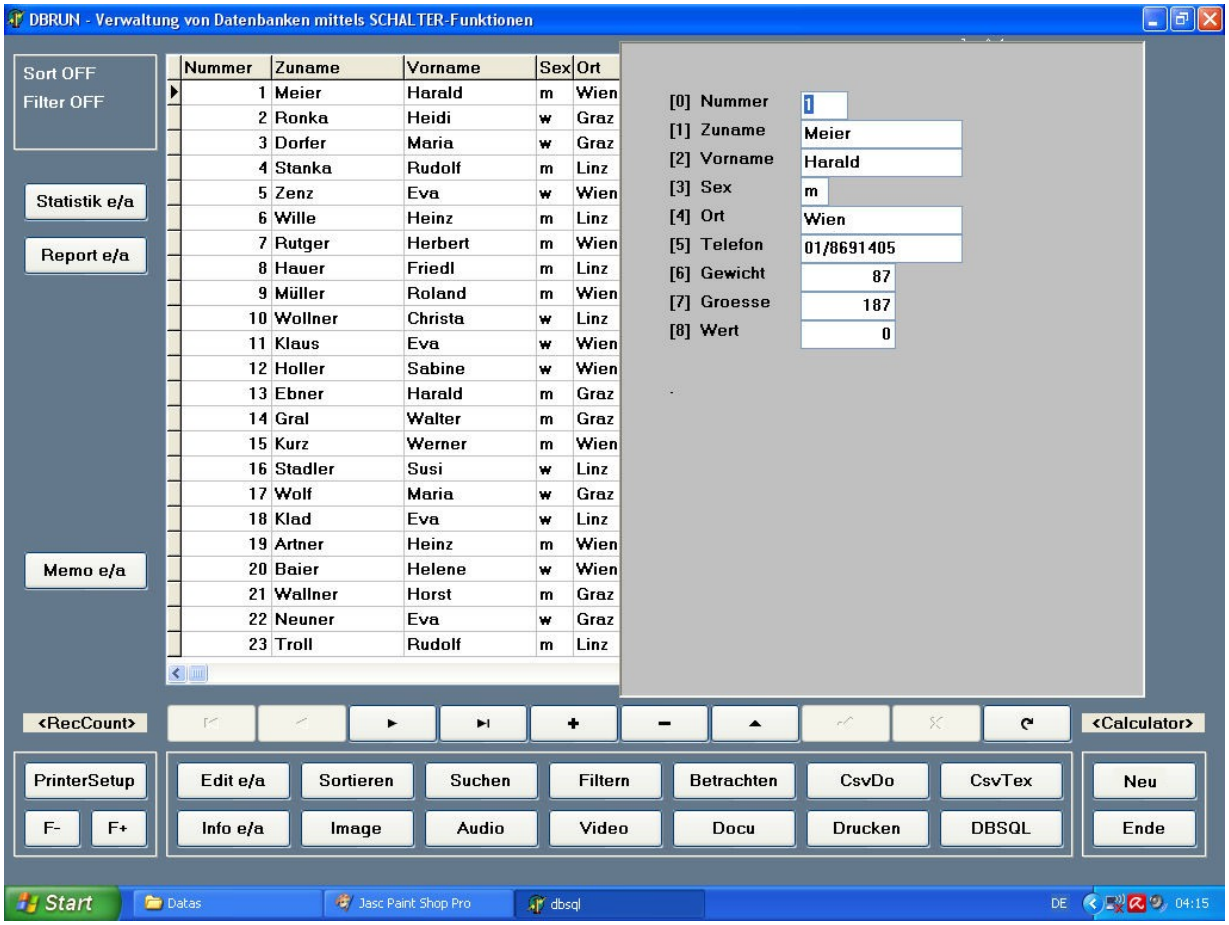
D B H E L P

Datenbank-Theorie
und
Beschreibung von DBSQL



Dieses Skriptum soll der Einführung in die Datenbankverwaltung und in die SQL-Sprache dienen. Der Autor verfasste dazu das Programm DBSQL, mit dem der Anwender verschiedene SQL-Befehle zur Bearbeitung von Datenbanken eingeben kann. Diese werden dann interpretiert und ausgeführt. Dadurch können Schüler auf einfachem Niveau mit Datenbanken arbeiten und werden so mit deren Aufbau und Verwaltung vertraut. Zunächst sollen einige Grundbegriffe über Datenbanken erklärt werden. Im Anschluss daran werden wichtige Befehle der SQL-Sprache beschrieben. Das Skriptum gliedert sich in folgende sechs Abschnitte:

I. Der Aufbau von Datenbanken - 03 –

II. Die Normalisierung von Tabellen - 07 –

III. Die Datenbanksprache SQL - 13 -
(demonstriert am Programm DBSQL)

IV. Das Datenbankprojekt „FIRMA“ - 23 –

V. Der Multifunktions-Editor CSVTEX - 26 –

VI. Kurzbeschreibung des Systems - 37 –

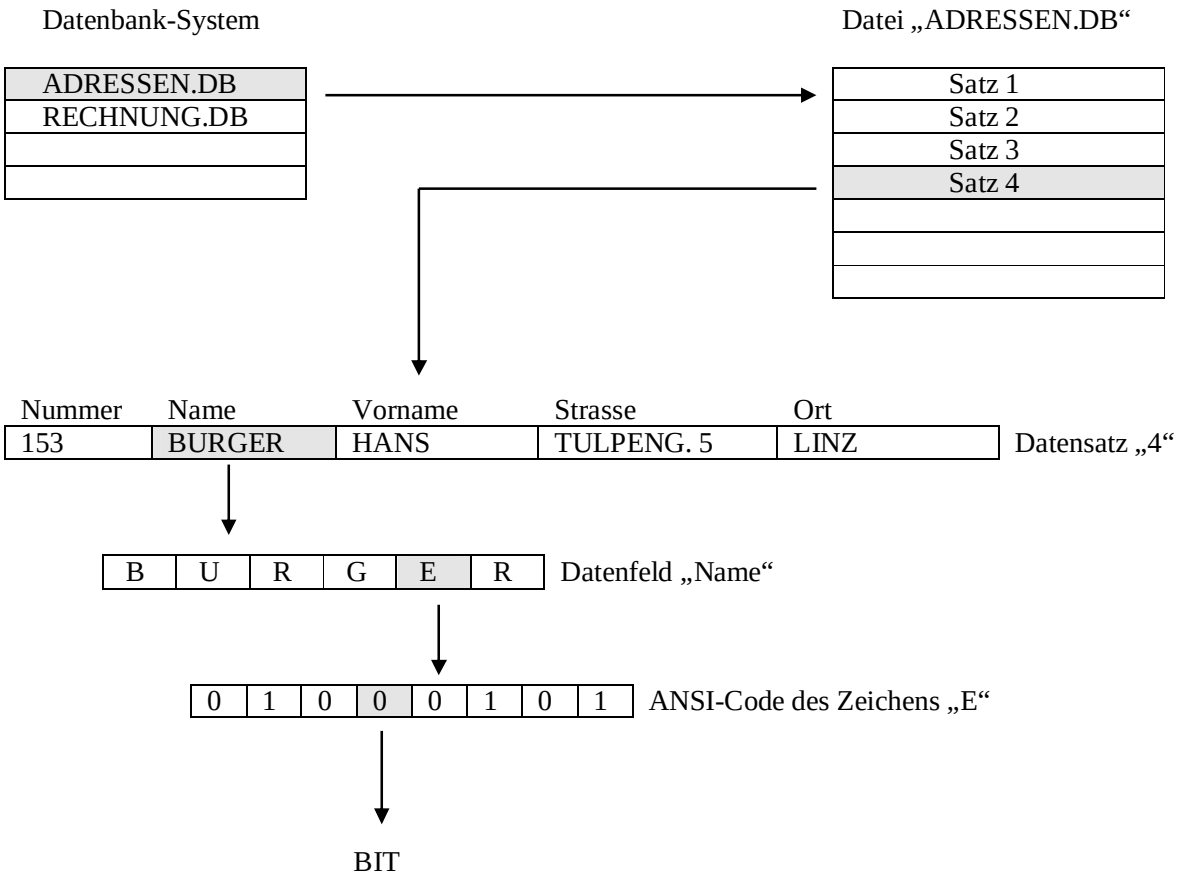
I. Der Aufbau von Datenbanken

Eine **Datenbank** kann aus mehreren Dateien (Files) mit einer besonderen inneren Struktur bestehen. Eine **Datei** enthält gleichartig aufgebaute Datensätze (Records). Ein **Datensatz** besteht aus einer bestimmten Anzahl von verschiedenen Datenfeldern (Fields). Die **Datenfelder** enthalten die **Bytes** der Daten. Der Datentyp (Text, Zahl, usw.) richtet sich nach der Art der Interpretation der Datenbytes.

Die Definition der einzelnen Datenfelder erfolgt beim Anlegen einer Datei durch Angabe bestimmter Feldattribute:

Feldname
Feldtyp (Text, Numerisch, Logisch, Datum usw.)
Feldlänge

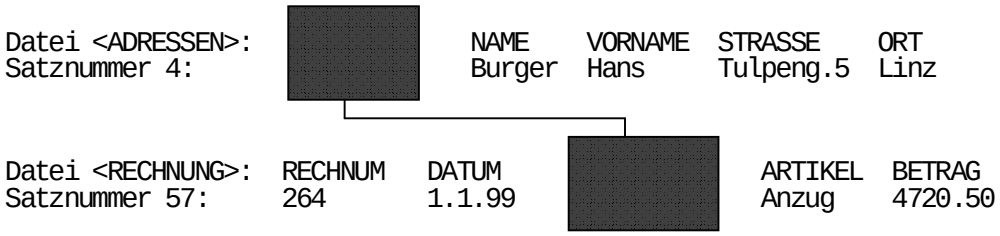
Die Struktur eines Datensatzes - und damit auch die Struktur der ganzen Datei - ist durch die Definition seiner Felder gegeben. Intern kann eine Datenbankdatei in einen Kopfteil (Vorspann, Header) und in einen Datenteil aufgegliedert werden. Ersterer enthält alle wichtigen Strukturinformationen (z.B. die Feld-definitionen). Zweiterer enthält die eigentlichen Datensätze.



Schematischer Aufbau einer Datenbank.

In **Textdateien** sind die einzelnen Textzeilen verschieden lang, d.h. sie enthalten verschieden viele Zeichen. Der Zugriff auf den Text erfolgt **sequentiell**, d.h. eine Zeile nach der anderen. In **Datenbankdateien** haben alle Datensätze die gleiche Struktur und sind daher gleich lang. Dadurch kann von jedem einzelnen Datensatz die Adresse auf dem externen Datenträger berechnet werden, sodass ein **direkter Zugriff** schnell und sicher möglich ist (Random Access, wahlfreier Zugriff).

Logische Beziehungen (Relationen) zwischen den Datensätzen zweier verschiedener Dateien werden über den Inhalt der Datenfelder selbst hergestellt; z.B. kann eine Kundennummer die Relation zwischen einer Adressendatei und einer Rechnungsdatei herstellen, wenn die gleiche Nummer in beiden Tabellen vorkommt. Durch die Verwendung verschiedener, solcherart synchronisierter Dateien wird es möglich, eine Redundanz der gespeicherten Information zu vermeiden.



Ein Datenbank-System besteht aus mehreren Dateien, welche über entsprechende, gleichartig und eindeutig angelegte Schlüsselfelder miteinander verknüpft werden können. Dadurch ergibt sich dann folgende praktische Möglichkeit: Angenommen man befindet sich beispielsweise in der Datei <RECHNUNG> und editiert hier den 57. Datensatz. Dieser enthält in einem Schlüsselfeld die Kundennummer 153. Will man nun nähere Informationen über den entsprechenden Kunden einholen, so wechselt man in die Datei <ADRESSEN> und gelangt dort (ohne zusätzliches Suchen) über das gleiche Schlüsselfeld automatisch zum 4. Datensatz, der den Kunden mit der Nummer 153 beschreibt.

Ein Beispiel soll die Vorteile eines Datenbank-Systems aufzeigen:

VERSION 1 (Einzelversion):

Eine Firma verwendet nur eine Datei <RECHNUNG>, um Rechnungen für alle Warenaufträge zu speichern und zu drucken. Die Datei enthält 10 Felder pro Datensatz:

Auftragsnummer - Auftragsdatum - Kundennummer - Kundenname - Kundenadresse - Artikelnummer - Artikelbeschreibung - Anzahl - Stückpreis - Gesamtpreis

Bei 10 Aufträgen müssen daher 100 Feldwerte eingetragen werden. Diese Aufträge sollen so beschaffen sein, dass sie sich auf 3 verschiedene Kunden und auf 4 verschiedene Artikel beziehen.

VERSION 2 (Systemversion):

Die Firma verwendet zur Auftragsverwaltung ein Datenbank-System <VERBUND>, welches aus den drei Dateien <KUNDEN>, <AUFTRAG> und <ARTIKEL> besteht. Diese enthalten je ein gemeinsames Schlüsselfeld in ihren Datensätzen, wodurch sie miteinander verknüpft werden.

Die Datei <KUNDEN> enthält 3 Felder pro Datensatz:
*{**Kundennummer** - Kundenname - Kundenadresse}*

Die Datei <AUFTRAG> enthält 6 Felder pro Datensatz:
*{Auftragsnummer - Auftragsdatum - **Kundennummer** - **Artikelnummer** - Anzahl - Gesamtpreis}*

Die Datei <ARTIKEL> enthält 3 Felder pro Datensatz:
*{**Artikelnummer** - Artikelbeschreibung - Stückpreis}*

Bei 10 Aufträgen, welche 3 Kunden und 4 Artikel umfassen, müssen 10 Datensätze in die Datei <AUFTRAG>, 3 Sätze in die Datei <KUNDEN> und 4 Sätze in die Datei <ARTIKEL> eingegeben werden. Das sind dann insgesamt 10*6 + 3*3 + 4*3 = 81 Feldeinträge. Dadurch werden bei der System-Version 19% der Eintragungen eingespart. Betreffen die 10 Aufträge nur Kunden und Artikel, die bereits in den entsprechenden Dateien registriert sind, so müssen nur die Feldwerte in der Datei <AUFTRAG> eingegeben werden. Das sind 60 Eintragungen, was einer Einsparung von 40% entspricht.

Die Vorteile eines Datenbanksystems mit mehreren Dateien sind also:

(a) Einsparung von überflüssiger Information (Redundanz).
(b) Verringerung des Arbeitsaufwandes.
(c) Erleichterung des Änderungsdienstes. (Adressenänderungen von Kunden sind beispielsweise nur in der Datei <KUNDEN> durchzuführen. Die zwei anderen Dateien bleiben davon unberührt.)

Grundlegende Datenbank-Funktionen

Die Verwaltung einer Datenbank erfolgt entweder durch eine Menüsteürung oder durch die Eingabe von Befehlen. Die wichtigsten Routinen einer Datenbankverwaltung sind:

- *Anlegen einer neuen Datei (Namensvergabe, Felddefinition).*
- *Öffnen / Schließen einer bestehenden Datei.*
- *Eingeben von Datensätzen (Physisches Anfügen).*
- *Ausgabe von Datensätzen (Bildschirm/Drucker).*
- *Unter **Datenprojektion** versteht man die Begrenzung der Ausgabe und Eingabe der Daten auf bestimmte Felder (View).*
- *Ändern von Datensätzen (Editieren).*
- *Löschen von Datensätzen.*
- *Sortieren der Datensätze - entsprechend den Inhalten eines bestimmten Feldes.*
- *Durchsuchen von Datensätzen nach bestimmten Suchbegriffen.*
- *Unter **Datenselektion** versteht man das Auswählen und Filtern von Datensätzen nach einem bestimmten Suchkriterium.*
- *Einfache Auswertungsfunktionen (Anzahl, Summe, Mittelwert).*
- *Änderung der Dateistruktur (Neue Felder anlegen, bestehende Felder löschen).*
- *Import und Export von Daten aus Fremd-Dateien.*
- *Verknüpfung von verschiedenen Dateien über gemeinsame Schlüsselfelder.*
- *Dateimanipulationen (Löschen, Kopieren, usw.)*

Die Verwendung von Indexdateien

Die wichtigsten Datenbankoperationen sind **Suchen** und **Sortieren**. Die Datensätze einer Datei können zunächst immer nur nach einem bestimmten Datenfeld (**Schlüsselfeld**) sortiert werden. Soll die Datei einmal nach dem einen und dann nach dem anderen Schlüsselfeld sortiert werden, dann ist es zweckmäßig, so genannte **Indexdateien** anzulegen. Auch wenn häufig in bestimmten Schlüsselfeldern nach Begriffen gesucht wird, erweisen sich Indexdateien als sehr sinnvoll.

Eine Indexdatei ist gewissermaßen eine Teilkopie ihrer Stammdatei. Die Sätze der Indexdatei bestehen aus zwei Feldern, dem **Schlüsselfeld** und dem **Zeigerfeld**. Das Schlüsselfeld enthält den Inhalt des herausgegriffenen Datenfeldes, das Zeigerfeld enthält die entsprechende Nummer des Datensatzes in der Stammdatei.

Die Indexdatei wird nun - soweit wie möglich - im Hauptspeicher nach dem Schlüsselfeld sortiert, während die wesentlich längeren Datensätze der Stammdatei unbewegt auf dem Festspeicher verbleiben. Das Gleiche gilt auch für das Durchsuchen einer Datei. Der Zugriff auf die Datensätze der Stammdatei erfolgt sodann entsprechend der Reihenfolge der Satznummern in der Indexdatei.

Es können mehrere Indexdateien angelegt und auf der Diskette abgespeichert werden. Der Zugriff auf die Stammdatei kann jedoch immer nur über **eine** Indexdatei erfolgen (Hauptindexdatei). Das nachfolgende Beispiel soll das Konzept der Indexdateien veranschaulichen.

Stammdatei <ADRESSEN> *(unsortiert)*

SATZNUMMER (der Stammdatei)	KUNDNUM	NAME	VORNAME	STRASSE	ORT
1	203	Haber	Helmut	Blüteng. 7	Graz
2	97	Zenz	Maria	Hauptpl. 1	Linz
3	195	Meier	Willi	Uferstr. 8	Linz
4	153	Burger	Hans	Tulpeng. 5	Linz
5	241	Adam	Eva	Baumg. 9	Graz

Indexdatei <ADRNAM> *(sortiert nach „NAME“)*

SATZNUMMER (der Indexdatei)	SCHLÜSSELFELD (Name)	ZEIGERFELD (Satznummer in der Stammdatei)
1	Adam	5
2	Burger	4
3	Haber	1
4	Meier	3
5	Zenz	2

Anfügen eines neuen Datensatzes in der Stammdatei

6	301	Gober	Herbert	Freipl. 2	Linz

Indexdatei <ADRNAM> nach dem alphabetisch geordneten Einfügen

SATZNUMMER	SCHLÜSSELFELD	SATZNUMMER (Zeigerfeld)	
1	Adam	5	
2	Burger	4	
3	Gober	6	
4	Haber	1	nach hinten
5	Meier	3	verschobene
6	Zenz	2	Datensätze

Das Sortieren und Durchsuchen der Indexdatei erfolgt größtenteils im Hauptspeicher und wird meistens mithilfe sehr schneller Verfahren durchgeführt (z.B. mittels „Binärbaum“). Die Stammdatei bleibt dabei unbewegt auf dem Festspeicher.

Beim **Suchen** (Datenselektion) wird eine Abfrage nach Suchbegriffen in Datenfeldern formuliert, welche in einer bestimmten Form (Abfragesprache) erfolgen muss und als Suchkriterium bezeichnet wird, z.B. könnte ein Suchkriterium ***(Name < „C“) und (Ort = „Linz“)*** lauten. Damit werden alle Linzer, deren Namen mit „**A**“ oder „**B**“ beginnen, aus der Datei <ADRESSEN> herausgefiltert.

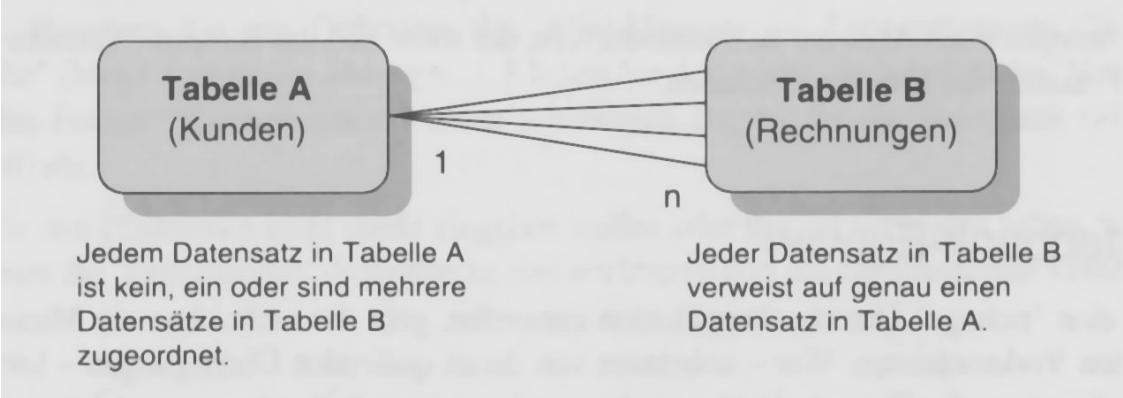
II. Die Normalisierung von Tabellen

Wir wollen annehmen, dass wir in einer Firma die beiden Tabellen "Kunden" und "Rechnungen" haben. Die Beziehungen (Relationen) zwischen diesen Tabellen könnten rein theoretisch wie folgt aussehen:

(a) Jedem Kunden ist genau eine Rechnung zugeordnet ([1 : 1] - Relation).
(b) Jedem Kunden ist keine, eine oder sind mehrere Rechnungen zugeordnet ([1 : n] - Relation).

Obwohl das relationale Modell noch weitere Typen kennt, verwenden wir unter Paradox in der Regel nur diese zwei Relationen. Davon ist die [1 : n] - Beziehung die eindeutig dominierende, da sich Tabellen mit [1 : 1] - Relation auch zu einer einzigen zusammenfassen lassen und nur dann einen Sinn ergeben, wenn nebensächliche Informationen ausgelagert werden sollen.

Ein Beispiel für eine [1 : n] - Relation:



Da in jedem relationalen Modell nur die Daten, nicht aber die Beziehungen zwischen diesen Daten abgespeichert werden, sind zusätzliche Maßnahmen zur Herstellung der Verknüpfungen zwischen den Tabellen erforderlich. Eine zentrale Rolle spielt dabei der sogenannte Identifikationsschlüssel oder kurz Schlüssel. Ein solcher Schlüssel (manchmal auch als Hauptschlüssel, Primärschlüssel oder Hauptindex bezeichnet) kann einer Tabelle als Attribut (Feld) zugeordnet sein und muss folgende Bedingungen erfüllen:

(a) Jeder Datensatz muss durch den Schlüssel eindeutig identifizierbar sein.
(b) Der Schlüssel darf durch Datenbankzugriffe nicht überschrieben werden.

Exakt sind diese Forderungen meist nur mit einem künstlichen Schlüssel erfüllbar, z.B. einer laufenden Nummer. Warum ? Attribute (Felder) wie Name, Ort, ... in einer Kunden-Tabelle eignen sich z.B. nicht immer, da sie der ersten Forderung widersprechen, denn es gibt mit Sicherheit mehrere Kunden, die Müller heißen oder aus Berlin stammen.

Allerdings sind auch zusammengesetzte Schlüssel möglich, die zwei oder noch mehr Attribute zusammenfassen. Aber leider ist auch hier die Eindeutigkeit nicht 100%-ig gegeben, da z.B. die Wahrscheinlichkeit, dass es nur einen Müller aus Berlin gibt, leider alles andere als Null ist.

Nun zur zweiten der oben genannten Forderungen. Was würde passieren, wenn Sie den Inhalt eines Schlüssels (z.B. Kunden-Nummer in der Kunden-Tabelle) nachträglich ändern ? Ein Chaos würde entstehen, denn die in der Rechnungen-Tabelle erfassten Kunden-Nummern würden nicht mehr stimmen. Also darf an den Kunden-Nummern nicht herumgedoktert werden. Auch die Nummer eines gelöschten Kunden bleibt für die weitere Verwendung tabu.

Hinweis: Als Indentifikationsschlüssel sollten grundsätzlich eindeutige künstliche Schlüssel verwendet werden, die keine weiteren Informationen beinhalten. In Paradox-Datenbanken verwenden Sie für einen solchen Schlüssel am besten ein Zählerfeld, das durch die Datenbank automatisch inkrementiert wird.

Die optimale Aufteilung einer Datenbank in mehrere Tabellen ist ein schrittweiser Prozess, der auch als Normalisierung bezeichnet wird. Wir wollen diesen Vorgang am Beispiel einer Datenbank einer Firma "Visual Data AG", erläutern. Ziel ist dabei das optimierte Organisieren und Abspeichern von Rechnungsdaten.

Ausgangstabelle

Wir notieren zunächst einmal aus dem Stegreif eine erste Version einer Tabelle mit dem Namen Rechnungen, in die wir alle benötigten Informationen hineinpacken (Rechnungsdatum, Rechnungsbetrag, Kundennummer, Kundenname, Kundenort, Artikelnummer, Artikelname).

Rechnungen

ReNr	ReDatum	ReBetrag	KuNr	KuName	KuOrt	ArtNr	ArtName
1	12.09.95	1.500,-	2	Müller	Berlin	2,4,11	Delphi 3.0, VB 5.0, Paradox 7.0
2	15.10.95	950,-	5	Schultze	München	5,4	Word 7.0, VB 5.0
3	17.11.95	1025,-	1	Mayer	Hamburg	2,5	Delphi 3.0, Word 7.0

Aus Übersichtlichkeitsgründen bleiben in unserem Beispiel diese Informationen auf das absolute Minimum reduziert (Artikelpreis und -anzahl fehlen zum Beispiel, könnten aber später problemlos ergänzt werden). Nach näherem Hinsehen sticht uns bereits ein gravierender Mangel ins Auge:

In den Feldern ArtNr und ArtName sind mehrfache Merkmalswerte eingetragen. Wenn ein Kunde viele Artikel kauft, passen diese möglicherweise nicht mehr alle in das entsprechende Feld. Wir müssen die Tabelle umstrukturieren, um die Erste Normalform zu erreichen:

Erste Normalform

Eine Tabelle hat dann die Erste Normalform (NF1), wenn sie nur einfache Merkmalswerte enthält. Durch einfaches Umgruppieren der Daten erreichen wir die NF1:

Rechnungen

ReNr	ReDatum	ReBetrag	KuNr	KuName	KuOrt	ArtNr	ArtName
1	12.09.95	1.500,-	2	Müller	Berlin	2	Delphi 3.0,
1	12.09.95	1.500,-	2	Müller	Berlin	4	VB 5.0
1	12.09.95	1.500,-	2	Müller	Berlin	11	Paradox 7.0
2	15.10.95	950,-	5	Schultze	München	5	Word 7.0
2	15.10.95	950,-	5	Schultze	München	4	VB 5.0
3	17.11.95	1025,-	1	Mayer	Hamburg	2	Delphi 3.0
3	17.11.95	1025,-	1	Mayer	Hamburg	5	Word 7.0

Mit diesem Anblick sollten wir uns aber keinesfalls zufriedengeben, da die gleichen Daten (Adresse des Kunden, Artikelname) mehrfach abgespeichert sind, es liegen also viele überflüssige Informationen vor, man spricht von Redundanz. Abgesehen von der Speicherplatzverschwendung stellen Sie sich bitte vor, ein Kunde wechselt seinen Wohnort. Sie müssten dann möglicherweise sehr viele Datensätze ändern. Wir müssen also etwas gegen diese lästige Redundanz unternehmen.

Zweite Normalform

Eine Tabelle weist dann die Zweite Normalform (NF2) auf, wenn sie sich in der NF1 befindet und wenn jedes Merkmal (außer dem Schlüssel) unmittelbar vom Schlüssel abhängt.

Wie wir sehen, sind z.B. die Merkmale KuNr, KuName, KuOrt, ArtNr und ArtName vom Wert des Schlüssels (ReNr) unabhängig, sie widersprechen also der NF2. Sie glauben es nicht? Dann überzeugen Sie sich bitte selbst davon, dass es zu jeder Rechnungsnummer nur ein bestimmtes Rechnungsdatum gibt, der Inhalt der übrigen (unabhängigen) Felder aber beliebig sein könnte und mit der Rechnungsnummer nicht unmittelbar gekoppelt ist.

Offenbar genügt eine einzige Tabelle nicht mehr, um die Forderungen der NF2 zu erfüllen. Wie wir im Folgenden sehen, erzwingt die NF2 die Aufteilung unserer Ausgangstabelle in mehrere Einzeltabellen, die bestimmten "Sachgebieten" entsprechen müssen:

Kunden

KuNr	KuName	KuOrt
1	Mayer	Hamburg
2	Müller	Berlin
3	Schultze	München

Rechnungen

ReNr	KuNr	ReDatum	ReBetrag
1	2	12.09.95	1.500,-
2	5	15.10.95	950,-
3	1	17.11.95	1.025,-

Artikel

ReNr	ArtNr	ArtName
1	2	Delphi 3.0
1	4	VB 5.0
1	11	Paradox 7.0
2	5	Word 7.0
2	4	VB 5.0
3	2	Delphi 3.0
3	5	Word 7.0

Lange können wir uns aber auch an diesen drei Tabellen nicht erfreuen, zumindest die Artikel-Tabelle beunruhigt uns zunehmend. Nach längerem Hinsehen entdecken wir nämlich wieder untrügliche Spuren von Redundanz: Der gleiche Artikelname taucht mehrfach auf ! Der Einwand "Ach lassen wir das doch so stehen ....." kann schnell vom Tisch gewischt werden: Haben Sie vielleicht Lust, jede Menge Einträge nachträglich zu korrigieren, wenn sich z.B. später die Bezeichnung "Delphi 3.0" in "Delphi 5.0" ändern sollte ? Also knöpfen wir uns wohl oder übel noch einmal die Artikel-Tabelle vor (die Kunden- und die Rechnungen-Tabelle bleiben unverändert) .

Dritte Normalform

Die Dritte Normalform (NF3) einer Tabelle liegt dann vor, wenn sie sich in der NF2 befindet und wenn ihre Merkmale (außer der Schlüssel) untereinander unabhängig sind, d.h., es dürfen keine transitiven Abhängigkeiten bestehen.

Dem Sinn dieser Definition kommt man erst nach längerem Grübeln auf die Spur. Was versteht man unter transitiven Abhängigkeiten ? Aber halt, zäumen wir doch besser das Pferd von hinten auf und betrachten wir erst einmal ein Gegenbeispiel an Hand der Tabelle Kunden, die offenbar über den Verdacht transitiver Abhängigkeiten völlig erhaben ist. In der Tat, der Kundenort ist nicht abhängig vom Kundennamen, denn der Kunde kann den Ort wechseln, oder es können mehrere Kunden im gleichen Ort wohnen. Etwas anders sieht es bei der Artikel-Tabelle aus. Der Name des Artikels ist fest an die Artikelnummer gekoppelt, mehrere Artikel dürfen nicht die gleiche Nummer haben, es besteht also eine transitive Abhängigkeit der Merkmale ArtNr und ArtName, die es zu beseitigen gilt.

Dazu splitten wir die ursprüngliche Artikel-Tabelle in zwei Tabellen (Rechnungsdaten und Artikel) auf:

Rechnungsdaten

ReNr	ArtNr
1	2
1	4
1	11
2	5
2	4
3	2
3	5

Eine Tabelle, wie die obige, bezeichnet man auch als Interselektionstabelle.

Artikel

ArtNr	ArtName
2	Delphi 3.0
4	VB 5.0
5	Word 7.0
11	Paradox 7.0

Nunmehr besteht unsere Datenbank aus insgesamt vier Tabellen (Kunden, Rechnungen, Artikel und Rechnungsdaten), die alle die Dritte Normalform (NF3) aufweisen, die Datenbasis ist quasi normalisiert.

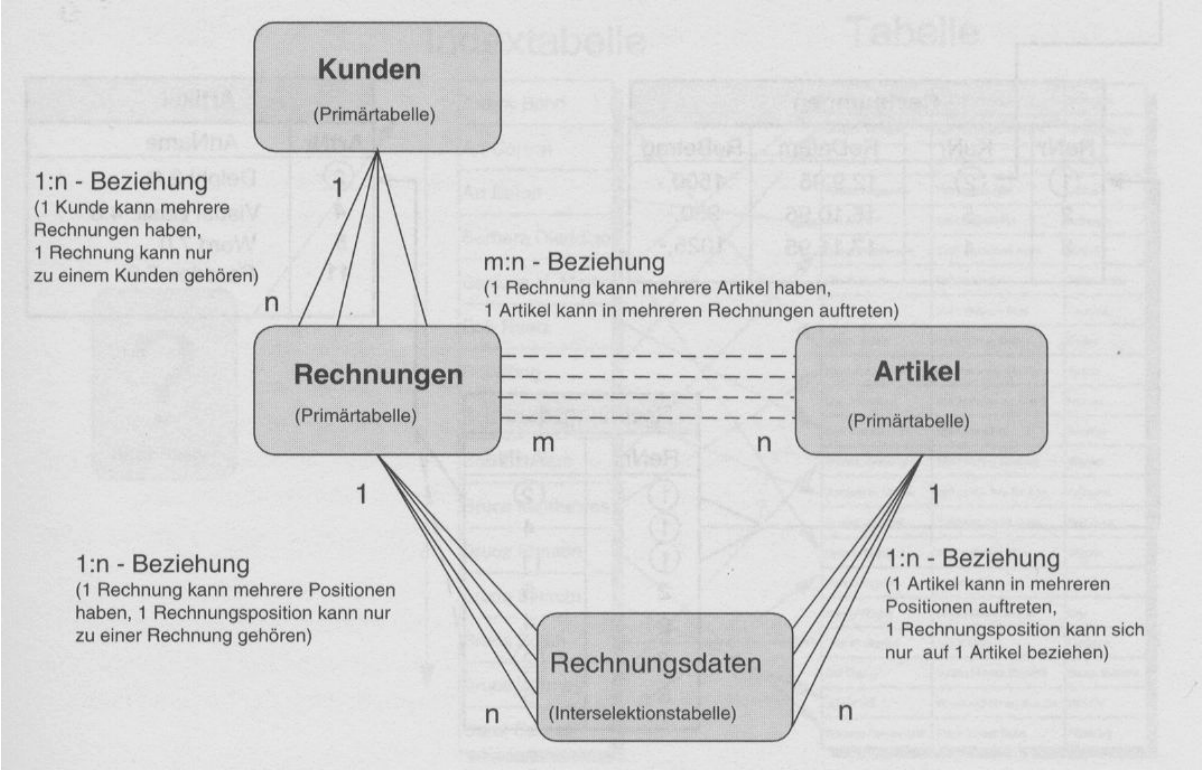
Zwar kennt die Theorie noch weitere Normalformen, aber dies ist eine Angelegenheit der Spezialliteratur. Für die überwiegende Mehrheit praktischer Einsatzfälle dürfte das Erreichen der NF3 ausreichend sein.

Hinweis: Hüten Sie sich vor einem "Normalisierungswahn", der zu einer unüberschaubaren Vielzahl kleiner Tabellen (und der dafür erforderlichen künstlichen Schlüssel) führen kann.

Einen kleinen Vorgeschmack auf mögliche Auswüchse vermittelt die Rechnungsdaten-Tabelle, die (quasi nur noch aus Schlüsselverweisen (Fremdschlüssel) besteht und ansonsten keine "echten" Felder enthält. Auch die Performance des Datenbanksystems (Antwortverhalten) leidet unter einer Übernormalisierung, die Fehleranfälligkeit wächst als Folge der Komplexität. Leider liefern auch die zu vielen Datenbanksystemen mitgelieferten "Experten" in der Regel eine übernormalisierte Tabellenaufteilung. Verzichten Sie deshalb besser auf derlei fragwürdige Dienste. Angestrebtes Ziel des Normalisierungsprozesses sollte stets ein optimaler Kompromiss zwischen Systemleistung und Redundanzfreiheit sein.

Verknüpfen von Tabellen

Durch die Normalisierung unserer Datenbank ist deren Struktur verlorengegangen, und die ursprünglichen Beziehungen zwischen den Daten existieren nicht mehr. Indem wir die Tabellen miteinander verknüpfen, wollen wir die alten Beziehungen restaurieren. Aber vorher ist es sinnvoll, sich noch einmal einen Gesamtüberblick über die grundlegenden Beziehungen zwischen allen Tabellen zu verschaffen:



Aus obigem Verknüpfungsdiagramm gewinnen wir folgende allgemeingültigen Erkenntnisse, die auch als Resultat der einzelnen Normalformen zu interpretieren sind:

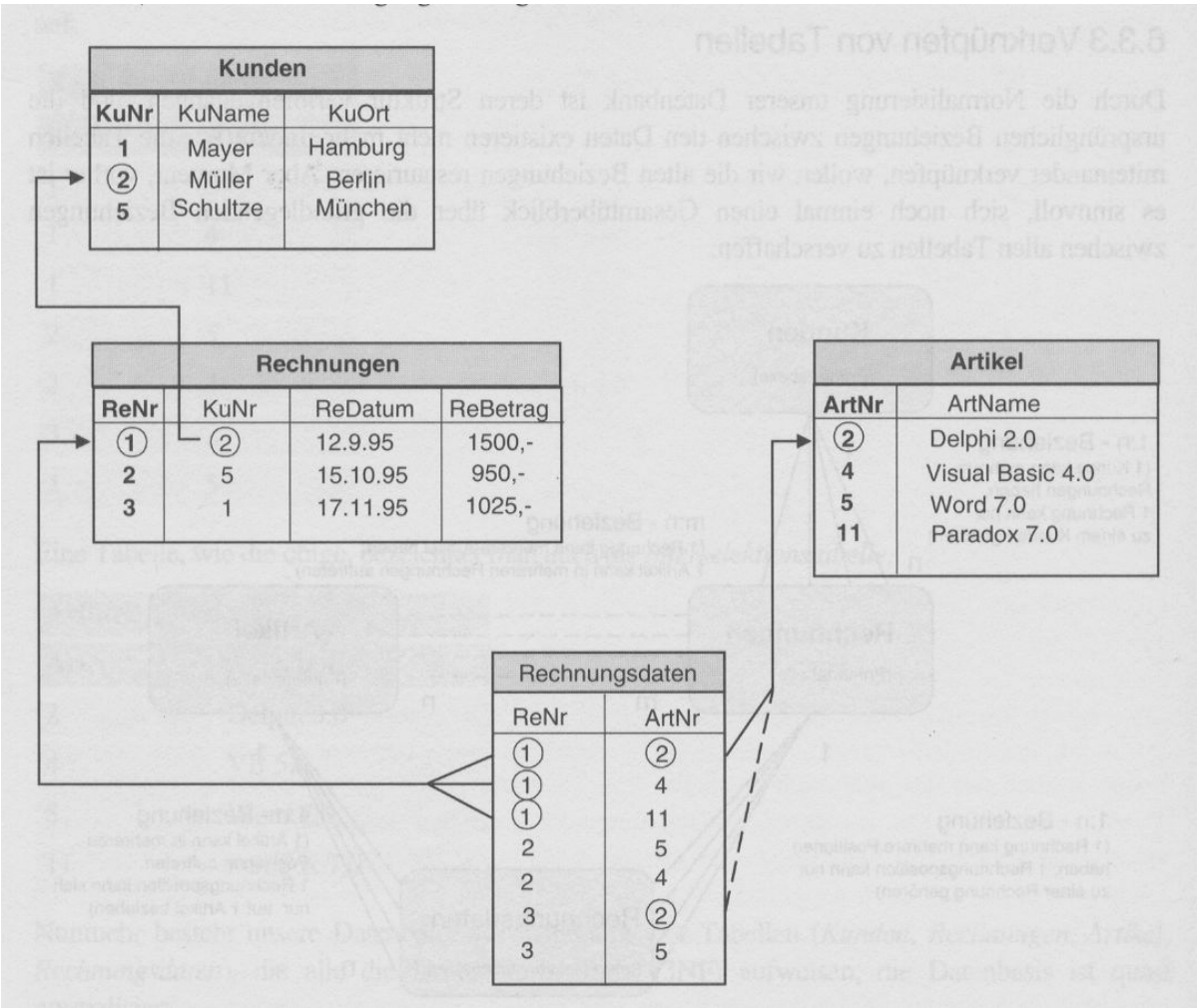
(a) Eine [m : n] - Beziehung, wie sie zwischen den Tabellen Rechnungen und Artikel besteht, kann so nicht nachgebildet werden. Sie muss mit Hilfe einer Interselektionstabelle (Rechnungsdaten) in zwei [1 : n] - Beziehungen aufgelöst werden. Dies ist unter anderem Ergebnis der NF1 und NF3.

(b) Nur Primärtabellen, wie Kunden, Artikel und Rechnungen müssen einen (künstlichen) Schlüssel haben. Die Entstehung dieser Tabellen kann man als Ergebnis der NF2 betrachten.

(c) Fremdschlüssel (bzw. Sekundärschlüssel) verweisen auf den Primärschlüssel einer anderen Tabelle. In unserem Beispiel haben die Tabellen Rechnungen und Rechnungsdaten einen bzw. zwei Fremdschlüssel.

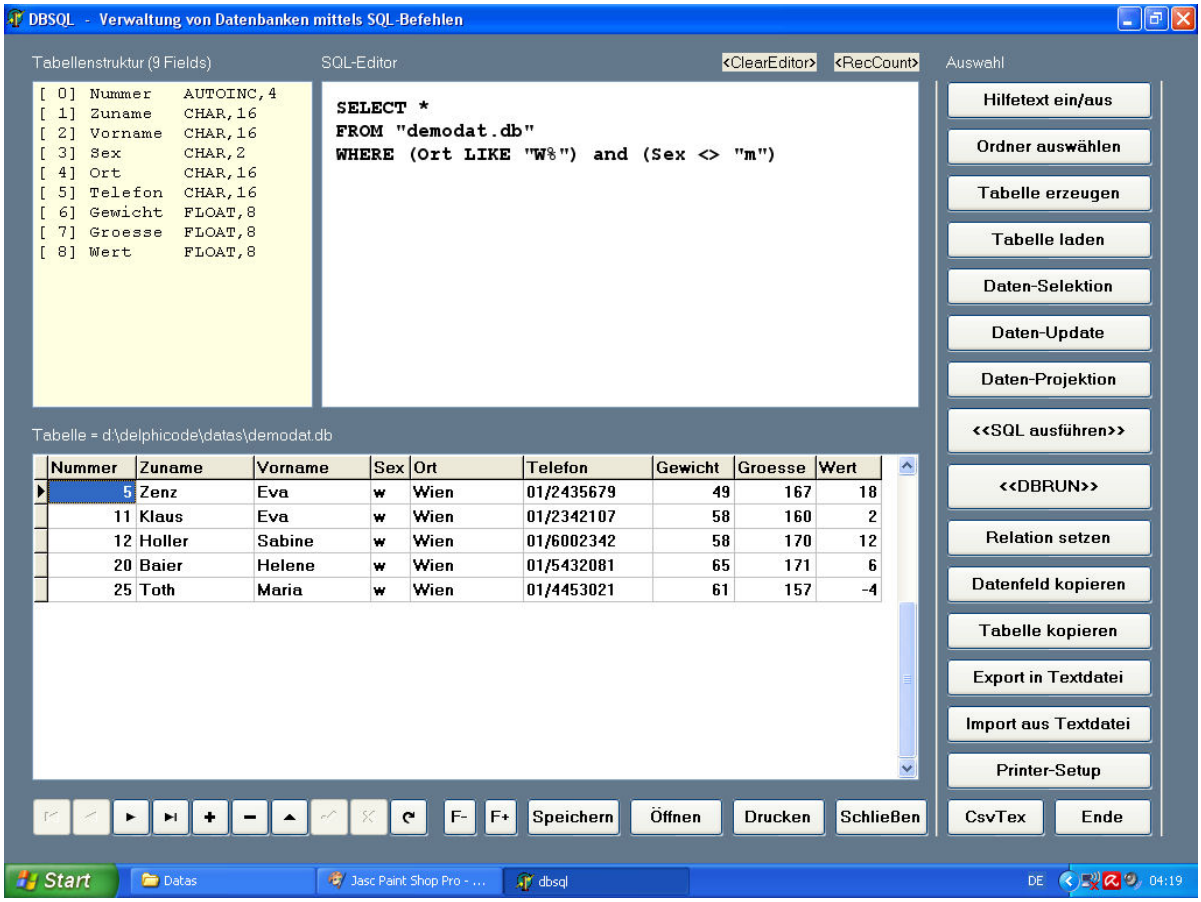
Als Verknüpfungsziele dienen die Schlüsselfelder der Tabellen, in unserem Fall sind dies die künstlichen Schlüssel ReNr, KuNr und ArtNr. Alle Felder mit Fremdschlüsseln (ReNr und ArtNr in Rechnungsdaten und KuNr in Rechnungen) sollten hingegen einen Sekundärindex zwecks Beschleunigung des Zugriffs erhalten.

Das folgende Diagramm gibt einen Überblick über das komplette Datenbanksystem mit seinen Tabellen und Relationen.



Da in der Datenbank nur die Tabellen abgespeichert werden, erfolgt die konkrete Umsetzung der abgebildeten Verknüpfungen erst in der jeweiligen Programm-Applikation über Datenzugriffselemente bzw. SQL-Anweisungen.

III. Die Datenbank-Sprache SQL
(demonstriert am Programm DBSQL)



Ähnlich wie es verschiedene Formate für Grafiken (JPG, BMP, usw.) und für Textdateien (TXT, RTF, usw.) gibt, so haben verschiedene Anbieter von Datenbanksystemen verschiedene Datenformate für ihre Applikationen entwickelt. Mittels SQL erfolgt die Verwaltung dieser unterschiedlichen Datenbanken mit genormten Befehlen und zwar unabhängig vom internen Format der darunter liegenden Datenbasis.

Unter **SQL** (Structured Query Language) versteht man im engeren Sinne eine genormte Abfrage-Sprache für Datenbanken. Sehr oft werden die SQL-Befehle mithilfe eines eigenen Übersetzungsprogrammes (Präcompiler) in die Originalsprache eines Datenbanksystems umgeformt. Ein SQL-Befehl entspricht dabei einem Unterprogramm-Aufruf aus einer eigenen Funktions-Bibliothek. Dadurch können mehrere Seiten Quellcode durch einen einzigen Befehl ersetzt werden, was die Kommunikation zwischen Benutzer und Datenbanksystem wesentlich vereinfacht. Man kann den Befehlsvorrat in drei Bereiche einteilen:

DDL	(Database Definition Language, Befehle zur Datenbankerzeugung)
DML	(Database Manipulation Language, d.h. Befehle zur Datenbankbearbeitung)
DSL	(Database Security Language, d.h. Befehle zum Datenbankschutz)

Relationale Datenbanken werden in der SQL-Sprache oft als Tabellen bezeichnet. Die einzelnen Datensätze sind dann die Zeilen und die Datenfelder die Spalten einer Tabelle. Durch beliebige Auswahl von Spalten wird eine bestimmte Sichtweise auf die Daten festgelegt (Projektion, View). Diese Sichtweise kann durch eine zusätzliche Datenselektion von bestimmten Datensätzen noch eingeschränkt und damit den individuellen Wünschen angepasst werden. **Projektionen** und **Selektionen** entsprechen horizontalen und vertikalen Dateneingrenzungen in der Tabelle.

Beschreibung und Bedienungsanleitung von DBSQL

Im Folgenden werden jene einfachen SQL-Befehle beschrieben, welche im Rahmen des Programmes DBSQL editiert und ausgeführt werden können. Das Programm DBSQL dient der Verwaltung von Datenbanken im so genannten Paradoxformat. Dabei sind die Dateien NAME.DB die Tabelle, NAME.PX die primäre Indexdatei und NAME.XG0 (NAME.YG0) sekundäre Indexdateien. Dateien und Programm sollten sich im gleichen Verzeichnis befinden.

Das Programm gliedert sich in zwei Teile (Module). Im ersten Modul **DBSQL** kann interaktiv mit Tabellen gearbeitet werden. Dabei wird durchgehend SQL (Standard Query Language) als Abfragesprache verwendet. Damit können neue Tabellen erzeugt, bestehende Tabellen verändert und editiert und Tabelleneinträge (Daten) selektioniert werden. Zur Erstellung der SQL-Befehle steht ein einfacher Editor zur Verfügung. Diese werden dann mit dem Schalter <SQL AUSFÜHREN> durchgeführt. Die im Editor eingegebenen SQL-Befehle können mit Hilfe von Textdateien gespeichert, geöffnet und gedruckt werden. Grundsätzlich kann nur ein einzelner SQL-Befehl im Editor ausgeführt werden. Es ist aber auch möglich, mehrere verschiedene SQL-Befehle hintereinander ablaufen zu lassen (Stapelverarbeitung). Dazu muss nach jedem SQL-Befehl eine leere Textzeile mit einem Strichpunkt am Anfang geschrieben werden.

Der zweite Programm-Modul **DBRUN** hingegen dient der menügesteuerten Tabellen-Verwaltung.

```
---------------------------------
(1) Erzeugung einer neuen Tabelle
---------------------------------

Eine Tabelle besteht aus gleichlangen Datensätzen
(records). Ein Datensatz besteht aus Datenfeldern
(fields), welche die Tabellenstruktur bilden.
Ein Beispiel zur Erzeugung einer neuen Tabelle:

CREATE TABLE "Tabelle"

  (Feld1 AUTOINC,
   Feld2 CHAR(10),
   Feld3 INTEGER,
   Feld4 FLOAT(8,2),
   Feld5 DATE,
   Feld6 BLOB(10,1),
   PRIMARY KEY(Feld1))

"Tabelle" ist der Tabellenname, "FeldXX" sind die
einzelnen Feldnamen gefolgt von ihren Feldtypen
mit ihren Feldlängen. Wichtige 7 Feldtypen sind:

CHAR(n)       Text-Feld mit der Länge n
INTEGER       Ganzahlen-Feld
FLOAT(n,d)    Gleitkommazahlen-Feld (z.B. n=8,d=2)
BOOLEAN       Logik-Feld (True=Wahr, False=Falsch)
DATE          Datums-Feld (TT.MM.JJJJ)
BLOB(n,d)     Binäres Daten-Feld (mit Feldbreite n und
              mit Feldtyp d, z.B. d = 1 als Memo-Feld)
AUTOINC       Satznummern-Feld
(wird bei jedem neuen Datensatz automatisch erhöht
 und kann nur gelesen werden. Mit der zusätzlichen
 Anweisung  PRIMARY KEY  sollte das Feld immer als
 Primär-Schlüssel verwendet werden).
```

----
(2) Datenbank-Ordner auswählen

Aktueller Ordner, Ordner der Demos, Ordner der Firma

----
(3) Bestehende Tabelle laden

SELECT Feld1,Feld2,..... { = Feldliste }
FROM "Tabelle"
*SELECT * FROM "Tabelle"* { = alle Felder }

----
(4) Daten-Selektion und Daten-Update

(4.1) Einfache Daten-Selektion
----

*SELECT * FROM "Tabelle"*
WHERE Selektions-Bedingung

Beispiele für Selektions-Bedingungen (Vergleiche).
Texte werden in doppelte Anführungszeichen gestellt.
(Auswahl der Feldnamen mit Mausklick in die Listbox).

(Feld1 > 160) AND (Feld1 < 180)	{Feld1 ist ein Zahlenfeld}
Feld2 = "Herbert"	{Feld2 ist ein Textfeld}
Upper(Feld2) <= "H"	{Feld2 ist ein Textfeld}
Feld2 LIKE "H%"	{% = beliebig}
Feld3 IN ("Wien","Graz","Linz")	{Feld3 ist ein Textfeld}
Feld4 IS (NOT) NULL	{NULL = unbelegt}
(EXTRACT(Year FROM Feld5) < 1945)	{Feld5 ist ein Datumsfeld}

(4.2) Datensätze löschen

DELETE FROM "Tabelle"
WHERE Trim(Upper(Feld2)) > "H"

Hier werden alle Datensätze mit Einträgen in Feld2,
die im Alphabet hinter *"H"* liegen, gelöscht. Die Funktion
Trim entfernt alle führenden und nachfolgenden Blanks,
Upper setzt auf Großschrift. ***Substring*** extrahiert Stringteile.
Ohne *WHERE* werden alle Datensätze der Tabelle gelöscht.

(4.3) Textfelder manipulieren

UPDATE "Tabelle"
SET Feld3 = Feld3 + Upper(Substring(Feld2 FROM 1 FOR 3))
WHERE ....

(4.4) Zahlenfelder berechnen (mit +,-,*,/)

UPDATE "Tabelle"
SET Feld3 = Feld1 + Feld2
WHERE ....

Hinweis zu den Zahlenfeldern: In Feldern vom Typ **FLOAT** wird in der Daten-Tabelle das Komma als Dezimalseparator verwendet. Im SQL-Editor hingegen wird standardmäßig der Punkt verwendet. Wird ein **FLOAT**-Feld mit Hilfe von **INTEGER**-Feldern berechnet, dann muss im Rechenausdruck ein Dezimalpunkt vorkommen .....

(4.5) Aggregat-Funktionen

SELECT COUNT(FeldXX) AS Anzahl FROM "Tabelle" WHERE ....
SELECT SUM(FeldXX) AS Summe FROM "Tabelle" WHERE ....
SELECT AVG(FeldXX) AS Mittel FROM "Tabelle" WHERE ....
SELECT MIN(FeldXX) AS Minimum FROM "Tabelle" WHERE ....

Diese Funktionen erlauben statistische Auswertungen von bestimmten Datenfeldern mit optionaler Selektion. Das Ergebnis steht im virtuellen Anzeigefeld hinter AS. Virtuelle Speicher-Felder können in reale Daten-Felder mit den **Kopierfunktionen (7) und (8)** kopiert werden.

(4.6) Gruppierungen

SELECT FeldXX, Count(FeldXX) AS Anzahl
FROM "Tabelle"
GROUP BY FeldXX

Hier werden gleichwertige Einträge von *FeldXX* zu Gruppen zusammengefasst und gezählt.

(4.7) Sortierung der Tabelle
----

*SELECT * FROM "Tabelle" ORDER BY FeldX1, FeldX2, ...*

Die Sortierung versetzt die Tabelle in den Anzeigemodus, sodass keine Datenedition möglich ist (read only). Sortieren mit *ORDER BY* läuft schneller, wenn vorher eine sekundäre Indexdatei erstellt wird, wobei der Name *IxField* als interner Bezeichner verwendet werden muss.

CREATE INDEX IxField ON "Tabelle" (FeldX1, FeldX2, ...)

(4.8) Daten an externe Tabelle anfügen

Als Quelle dient eine bestehende Tabelle und als Ziel eine andere bestehende Tabelle, wobei die Reihenfolge der Zielfelder (FeldYn) und Quellfelder (FeldXn) wichtig ist.

INSERT INTO "Zieltabelle"
(FeldY1, FeldY2, ...)
SELECT FeldX1, FeldX2, ...
FROM "Quelltabelle"
WHERE .....

INSERT INTO "Zieltabelle"
(FeldY1, FeldY2, ...)
VALUES (Wert1, Wert2, ...)

Hier wird ein neuer Datensatz in der Zieltabelle angefügt.

(4.9) Feld in Tabellenstruktur anfügen oder löschen
--

ALTER TABLE "Tabelle" ADD FeldXX CHAR(20)
ALTER TABLE "Tabelle" DROP FeldXX

DROP TABLE "Tabelle" (löscht die ganze Tabelle !)

(4.10) Die Verwendung von JOINS

Es können mehrere verschiedene Tabellen verbunden
werden (join), wenn sie ein gleichnamiges Schlüsselfeld
enthalten. Mit dem Schlüsselfeld wird zwischen den ver-
schiedenen Tabellen eine Beziehung (relation) erzeugt.
Zu beachten ist, dass jede Tabelle durch ein Kürzel
ersetzt werden kann (z.B *"Tabelle t"*). Ein Feld wird
dann mit *"Kürzel.Feldname"* bezeichnet (z.B. *"t.FeldXX"*).

SELECT r.ReNr, k.KuNr, k.KuName
FROM rechnung r, kunden k
WHERE r.KuNr = k.KuNr

Hier gibt es zwei Tabellen *"kunden"* und *"rechnung"*, die
über das Schlüsselfeld *"KuNr"* relationiert sind.

Das Ergebnis von SQL-Abfragen, die Joins, Groups oder
Aggregate enthalten, ist ein virtuelles Speicherbild,
welches nur angezeigt - aber nicht editiert werden kann.
Die **Kopierfunktionen (7) und (8)** kopieren auch die Inhalte
von virtuellen Ergebnisfeldern in reale Tabellenfelder.

----
(5) Daten-Projektion

Dabei kann die Tabellenansicht auf bestimmte Felder
eingeschränkt werden (Projektion, View).

----
(6) Relation setzen

Mit diesem Schalter ist es möglich, von der aktuellen
Tabelle in eine zweite Tabelle zu wechseln. Zuerst klickt
man in der aktuellen Tabelle in einem Datenfeld auf einen
bestimmten Wert. Wenn die zweite Tabelle in einem gleich-
namigen Datenfeld (Schlüsselfeld) den gleichen Feldwert
enthält, dann wird in der zweiten Tabelle der Satzzeiger
automatisch auf den entsprechenden Datensatz positioniert.
Dadurch können über Schlüsselfelder verschiedene Tabellen
relationiert werden (z.B. "demodat.db" und "bilanzdat.db").

----
(7) Datenfeld kopieren

Mit diesem Schalter werden die Einträge eines Feldes
(Quellfeld) der aktuellen Tabelle (Quelltabelle) in
ein Feld (Zielfeld) einer externen Tabelle (Zieltabelle)
kopiert. Quellfeld und Zielfeld müssen typenkompatibel
sein. Es werden nur so viele Datensätze kopiert, wie
bereits vorhanden sind.

----
(8) Tabelle kopieren

Hier wird von der aktuellen Tabelle (Quelle) entweder ihre Struktur OHNE Daten oder ihre Struktur MIT allen Daten in eine neu Tabelle (Ziel) kopiert.

----
(9) Import AUS einer Textdatei

Mit diesem Schalter werden sogenannte CSV-Textdateien in Paradox-Datentabellen umgewandelt.

----
(10) Export IN eine Textdatei

Mit diesem Schalter werden die Datensätze einer geladenen Tabelle zeilenweise in eine Textdatei exportiert. Als Feldbegrenzer wird der Strichpunkt ";" gesetzt. Die Namenserweiterung der Datei ist ".csv" (comma separated values). Solche CSV-Dateien können auch in EXCEL oder WORD verwendet werden.

----
(11) Der Navigator

Der NAVIGATOR verfügt über folgende Funktionen:

- --- zum ersten Datensatz
- --- einen Datensatz nach vor
- --- einen Datensatz nach hinten
- --- zum letzten Datensatz
- --- einen neun Datensatz anfügen (+)
- --- aktuellen Datensatz löschen (-)
- --- Datensatz bearbeiten
- --- Änderungen übernehmen (wichtig)
- --- Bearbeitung abbrechen
- --- Datensatz aktualisieren

----
(12) <<DBRUN>>

Damit wird der zweite Programm-Modul "**DBRUN**" aufgerufen.

Im ersten Programm-Modul "DBSQL" stehen noch folgende zusätzliche Schalter zur Verfügung:

<<Öffnen>>
Ermöglicht das Laden eines gespeicherten SQL-Befehls aus einer Textdatei.

<<Speichern>>
Ermöglicht die Speicherung eines SQL-Befehls in eine Textdatei.

Die nachfolgenden drei Beispiele sollen einen kurzen Überblick über SQL-Befehle geben, deren Ergebnisse in sogenannten virtuellen Datenfeldern angezeigt werden. Diese können nur gelesen, aber nicht editiert werden.

[•] Statistiken

```
CREATE TABLE "statis"
(Number INTEGER,
Minimum FLOAT(8,2),
Maximum FLOAT(8,2),
Average FLOAT(8,2))
;
DELETE FROM "statis"
;
INSERT INTO "statis"
(Number,Minimum,Maximum,Average)
SELECT
COUNT(Gewicht) AS Number,
MIN(Gewicht) AS Minimum,
MAX(Gewicht) AS Maximum,
AVG(Gewicht) AS Average
FROM "demodat"
;
SELECT *
FROM "statis"
```

Die Statistikbefehle speichern ihre Ergebnisse in virtuellen Datenfeldern. In dem hier vorliegenden Befehlsstapel werden die virtuellen Ergebnisfelder in eine reale Datenbank-Tabelle "statis.db" abgespeichert.

[•] Gruppierungen

```
SELECT Ort, Count(Ort) AS Anzahl
FROM "demodat"
GROUP BY Ort
```

Aus der Tabelle "demodat.db" werden im Feld "Ort" nur alle verschiedenen Werte angezeigt. Dann werden alle Datensätze mit diesen Orts-Werten gezählt und die Anzahlen im virtuelle Feld "Anzahl" angezeigt.

[•] Verbindungen

```
SELECT d.Zuname, d.Sex, r.Gewinn, r.Verlust
FROM "demodat" d, "bilanzdat" r
WHERE d.Zuname = r.Zuname
```

Hier werden den zwei Datentabellen "demodat.db" und "bilanzdat.db" die zwei symbolischen Kürzel d und r zugeordnet. Sodann wird eine neue virtuelle Tabelle erzeugt, welche aus Datenfelder von beiden Tabellen besteht. Damit diese Tabellenverbindung (Join) auch funktioniert, müssen die beiden Tabellen über ein gemeinsames Schlüsselfeld verfügen ("Zuname").

Die virtuelle Ergebnistabelle des Joins kann mit Hilfe des Schalters **<Tabelle kopieren>** in eine reale neue Datenbank-Tabelle abgespeichert werden.

<<Drucken>>
Mit dem Schalter kann entweder die ganze Tabelle oder der aktuelle Datensatz gedruckt werden.

Ist im Editor ein Text markiert, dann werden die SQL-Befehle und die Tabellenstruktur gedruckt.

Ist der Hilfetext sichtbar, dann wird er ausgedruckt. Wenn ein Bereich markiert ist, dann wird nur dieser Bereich des Hilfetextes gedruckt.

<<CsvTex>>
Mit diesem Schalter wird das Programm "CSVTEX" aufgerufen. Eine ausführliche Beschreibung erfolgt auf Seite [26].

<<Schließen>>
Schließt eine geladene Tabelle.

Der zweite Programm-Modul "DBRUN"

In diesem Modul wird eine geladene Datentabelle nicht mit SQL-Befehlen verwaltet, sondern direkt mit Schaltern:

<<Edit e/a>>
Wechselt zwischen Tabellenansicht und Maskenansicht. Das Programm erstellt automatisch eine einfache Editiermaske für die verschiedenen Datenfelder des aktuellen Datensatzes.

<<Sortieren>>
Sortiert die Tabelle mittels sekundärer Indexdatei. Dabei müssen die Sortierfelder durch Beistriche getrennt eingegeben werden.

<<Suchen>>
Durchsucht die Tabelle. Dabei wird zuerst das Datenfeld, dann ein Beistrich und der Suchbegriff eingegeben.

<<Filtern>>
Filtert eine Datenmenge aus der Tabelle. Dabei wird ein Selektions-Kriterium eingegeben. Dieses enthält Feldnamen, Feldwerte, Vergleichsoperatoren (=, <, >) und auch logische Operatoren (not, and, or). Sind die Feldwerte Texte, dann müssen sie in einfache Anführungszeichen gestellt werden. In Texten ist auch das Jokerzeichen * erlaubt, und die Groß-/Kleinschreibung wird nicht berücksichtigt. Beispiel für ein Kriterium: (Zuname = ´s*´) and (Gewicht < 80)

<<Memo>>
Wechselt zwischen Tabellenansicht und einem Memofeld. Im integrierten RTF-Editor wird der Memotext editiert. Ein Seitenvorschub (¶) für den Textausdruck wird mit <Strg><Enter> erzeugt. Die Schriftart wird mit Hilfe des Schalters <MemoFont> eingestellt. Auch kann der Memotext nach einem Suchbegriff mit <MemoSeek> ab der aktuellen Cursorposition durchsucht werden. Mit <F3> wird die Suche fortgesetzt.

<<Image>>
Wird in der Tabellenstruktur ein Datenfeld mit dem Namen "Image" angelegt, dann können dort die Namen von mehreren JPG-Grafikdateien eingetragen werden. Diese müssen sich im gleichen Verzeichnis wie das Programm DBSQL befinden. So können Bildarchive erzeugt werden. Der Schalter wechselt zwischen Grafikdarstellung und Datenansicht. Mit dem Zusatzschalter <Stretch> wird die Grafik an die Bildschirmgröße angepasst.

<<Audio>>
Wird in der Tabellenstruktur ein Datenfeld mit dem Namen "Audio" angelegt, dann kann dort der Name einer Sounddatei (MP3,WMA,WAV) eingetragen werden. Diese muss sich in dem gleichen Verzeichnis wie das Programm DBSQL befinden. So können Musikarchive erzeugt werden. Mit dem Schalter wird die Musik ein- oder ausgeschaltet.

<<Video>>
Wird in der Tabellenstruktur ein Datenfeld mit dem Namen "Video" angelegt, dann kann dort der Name einer Videodatei (MPG,WMV,AVI) eingetragen werden. Diese muss sich in dem gleichen Verzeichnis wie das Programm DBSQL befinden. So können Videoarchive erzeugt werden. Mit dem Schalter wird das Video abgespielt.

<<Docu>>
Wird in der Tabellenstruktur ein Datenfeld mit dem Namen "Docu" angelegt, dann kann dort der Name einer Textdatei (TXT,RTF,DOC,PDF) eingetragen werden. Diese muss sich in dem gleichen Verzeichnis wie das Programm DBSQL befinden. So können Textarchive erzeugt werden. Mit dem Schalter wird der Text geöffnet.

<<Betrachten>>
Dabei kann die Tabellenansicht auf bestimmte Felder eingeschränkt werden (Projektion, View).

<<Drucken>>
Druckt je nach Ansicht den aktuellen Datensatz, eine gefilterte Datenmenge im Tabellenformat, einen geöffneten Memotext oder eine dargestellte Grafik.

<<Statistik e/a>>
Mit diesem Wechsel-Schalter werden statistische Auswertungen eines vorher angeklickten Datenfeldes durchgeführt.

<<CsvDo>>
Damit werden alle Datensätze einer geladenen Tabelle zeilenweise in eine Textdatei exportiert. Die einzelnen Felder sind durch Strichpunkte ";" getrennt. Die Namenserweiterung der Datei ist ".csv" (comma separated values).

<<CsvTex>>
Mit diesem Schalter wird das Programm "CSVTEX" aufgerufen. Eine ausführliche Beschreibung erfolgt auf Seite [26].

<<DBSQL>>
Kehrt zum ersten Programm-Modul "**DBSQL**" zurück.

<<Neu>>
Mit dem Schalter kann eine neue Datentabelle geladen werden. Zusatzfunktion: Klickt man vorher mit der Maus in der bereits geladenen Datentabelle in einem Datenfeld auf einen Feldwert und enthält die neue Datentabelle dieses Feld und diesen Wert, dann wird der Datenzeiger automatisch auf diesen Datensatz positioniert. Andernfalls zeigt er auf den ersten Datensatz der neuen Tabelle. Dadurch können über gemeinsame Schlüsselfelder verschiedene Tabellen relationiert werden (beispielsweise "demodat.db" und "bilanzdat.db").

Hinweis: Bei einem Doppelklick in eine Zelle der Tabelle wird der dort befindliche Text als Dateiname oder Internetlink interpretiert. "DBRUN" versucht dann Datei oder Link zu öffnen.

Der REPORT-Generator

Der Report-Generator wird mit dem Schalter **<<Report e/a>>** ein- oder ausgeschaltet. Der Report-Generator ist ein einfacher Editor mit dem Textfiles im RTF-Format angelegt, geöffnet, gespeichert oder gedruckt werden.

Im Text können für die Felder eines Datensatzes der Tabelle symbolische Feldvariable der Form **<Fxx>** geschrieben werden, wo xx die Nummern der Datenfelder sind. ACHTUNG: Im Report beginnen die Feldnummern immer bei 1 und nicht bei 0.

Mittels Schalter **<<Vorschau e/a>>** werden die Feldvariablen durch die Werte der einzelnen Felder des jeweiligen Datensatzes ersetzt. Dadurch können unterschiedlich gestaltete Reports erstellt und dann gedruckt werden. Eine nochmalige Schalter-Betätigung zeigt die Report-Vorlage im Editor an. Optional kann ein solcher Druckreport nicht nur für den aktuellen Datensatz sondern auch für alle Datensätze der Tabelle erzeugt werden. Auf diese Weise ist es möglich Serienbriefe, Formulare oder Drucktabellen zu erzeugen.

Um bei der Vorschau zu lange Wartezeiten zu vermeiden, sollten nur Serienbriefe für maximal 500 Datensätze in einem Durchlauf ausgeführt werden. Wenn die Tabelle aber mehr als 500 Datensätze enthält, kann durch entsprechende Datenfilterung die Tabelle geteilt werden.

Hinweis 1: Wenn anstelle der Feldvariablen <Fxx> die Namensvariablen <Nxx> in den Text geschrieben werden, dann werden bei der Vorschau dafür die Namen der Datenfelder eingesetzt.

Hinweis 2: Im Report kann auch EIN virtuelles Rechenfeld eingetragen werden. Dieses wird von zwei geschwungenen Klammern begrenzt, zwischen denen ein Rechenausdruck ohne Leerzeichen steht. Der Rechenausdruck kann Zahlen, Feldvariable <Fxx> von numerischen Datenfeldern und Rechenoperatoren enthalten. So berechnet {<F2>*<F1>/100} einen Prozentanteil, wobei <F1> der Grundwert und <F2> der Prozentsatz sind.

Hinweis 3: Spitze und geschwungene Klammern sind reservierte Zeichen, welche nur zur Kennzeichnung von Feldvariablen dienen.

IV. Das Datenbank-Projekt „FIRMA“

Das komplexe Datenbankprojekt soll die **Rechnungserstellung** und **Lagerverwaltung** in einer Firma simulieren. Es besteht zur Vermeidung von überflüssigen Informationen (Redundanz) aus vier verschiedenen Tabellen mit folgenden Datenfeldern:

```
"artikel":  (ArtNr),ArtName,ArtPreis,ArtAnzahl,
            ArtMin,ArtMax,ArtBestell
"kunden":   (KuNr),KuName,KuPlz,KuOrt,KuStrasse,KuTelefon
"rechnung": (ReNr),KuNr,ReDatum,ReBetrag,ReBezahlt,ReOffen
"redaten ":  ReNr,ArtNr,Menge
```

Die eingeklammerten Felder sind primäre Indexfelder.
"kunden" und "rechnung" verbindet der Schlüssel "KuNr".
"rechnung" und "redaten" verbindet der Schlüssel "ReNr".
"artikel" und "redaten" verbindet der Schlüssel "ArtNr".

Tabelle "artikel.db"

ArtNr	ArtName	ArtPreis	ArtAnzahl	ArtMin	ArtMax	ArtBestell
1	Wein	4,00	100	95	105	0
2	Bier	2,00	100	95	105	0
3	Mineral	1,00	100	95	105	0
4	Saft	3,00	100	95	105	0

Tabelle "kunden.db"

KuNr	KuName	KuPlz	KuOrt	KuStrasse	KuTelefon
1	Meier Harald	1010	Wien	Krugerstrasse	01/8567034
2	Zenz Rudolf	8010	Graz	Blumenweg	0316/3015687
3	Adam Helmut	4020	Linz	Hauptstrasse	0732/2185073

Tabelle "rechnung.db"

ReNr	***KuNr***	ReDatum	ReBetrag	ReBezahlt	ReOffen
1	2	01.01.00	0	0	0
2	3	01.01.00	0	0	0
3	1	01.01.00	0	0	0
4	1	02.01.00	0	0	0
5	2	02.01.00	0	0	0
6	2	03.01.00	0	0	0

Interselektionstabelle "redaten.db"

ReNr	***ArtNr***	Menge
1	2	3
1	3	2
1	1	2
2	4	1
3	1	3
3	4	4
4	2	2
4	3	1
5	1	3
6	3	2

Zur Projektverwaltung sind folgende zehn Funktionen verfügbar:

<F1> Tabelle "artikel" laden
<F2> Tabelle "kunden" laden
<F3> Tabelle "rechnung" laden
<F4> Tabelle "redaten" laden
<F5> Rechnungsbeträge aktualisieren
<F6> Offene Geldbeträge berechnen
<F7> Komplette Rechnung erstellen (nur als virtuelle Tabelle)
<F8> Artikelbestand aktualisieren
<F9> Artikelbestellungen ermitteln
<F10> Wiederherstellen des Originalzustandes der Tabellen

Im Mittelpunkt des Systems steht die Tabelle "rechnung" mit der Interselektionstabelle "redaten". Zu einer Rechnung, d.h. zu einem Datensatz aus "rechnung" gibt es für jeden Artikel in der Rechnung einen Datensatz in "redaten". Die Tabellen "rechnung" und "redaten" stehen in einer (1,n)-Relation. Eine neue Rechnung wird in neun Schritten erstellt, welche mit Hilfe der Funktionstasten <F1> bis <F9> ausgeführt werden.

F1: Mit <F1> Tabelle "artikel" laden und eventuell bearbeiten.

```
SELECT * FROM "artikel.db"
```

F2: Bei einem neun Kunden, müssen mit Hilfe von <F2> in der Tabelle "kunden" die Kundendaten eingetragen werden.

```
SELECT * FROM "kunden.db"
```

F3: Mit <F3> Tabelle "rechnung" laden, neuen Datensatz anlegen und die Felder [ReNr],[KuNr],[ReDatum] eintragen. Beispielsweise ein Datensatz für eine neue Rechnung mit der Nummer 7.

```
SELECT * FROM "rechnung.db"
```

F4: Mit <F4> Tabelle "redaten" laden. Hier muss für jeden verkauften Artikel ein neuer Datensatz angelegt und die Felder [ReNr],[ArtNr],[Menge] eingetragen werden. Beispielsweise neue Datensätze für jeden verkauften Artikel aus der Rechnung mit der Nummer 7.

```
SELECT * FROM "redaten.db"
```

F5: Mit <F5> wird der Rechnungsbetrag aktualisiert, d.h. mit den Einträgen in "rechnung" und "redaten" wird der neu Rechnungsbetrag ermittelt und in dem virtuellen Aggregatfeld [Wert] abgelegt. Mittels Update wird das Feld [Wert] automatisch in das Feld [ReBetrag] der Tabelle "rechnung" kopiert. Nun können die eventuell bezahlten Beträge in [ReBezahlt] eingegeben werden.

```
UPDATE "rechnung.db" r
SET r.ReBetrag =
(SELECT SUM(a.ArtPreis * d.Menge) AS Wert
FROM "artikel.db" a, "redaten.db" d
WHERE (a.ArtNr = d.ArtNr) AND (d.ReNr = r.ReNr)
GROUP BY d.ReNr)
;
SELECT * FROM "rechnung.db"
```

F6: Mit <F6> erfolgt ein Update des Feldes [ReOffen].

```
UPDATE "rechnung.db"
SET ReOffen = ReBetrag - ReBezahlt
;
SELECT * FROM "rechnung.db"
```

F7: Mit <F7> kann für den Kunden eine übersichtliche Rechnungstabelle erstellt und ausgedruckt werden. (Das Ergebnis ist eine virtuelle Tabelle, z.B. für Rechnung Nr.1). Diese kann mittels <Tabelle kopieren> in eine reale Tabelle "rechn01.db" kopiert werden. Von dieser Tabelle kann dann in <DBRUN> der Report "rechn01.rtf" geöffnet und ausgedruckt werden.

```
SELECT r.ReNr, r.ReDatum,
       k.KuName, k.KuPlz, k.KuOrt, k.KuStrasse,
       a.ArtNr, a.ArtName, a.ArtPreis,
       d.Menge, d.Menge * a.ArtPreis AS Summe,
       r.ReBetrag AS Gesamt
FROM   "kunden.db" k,
       "artikel.db" a,
       "rechnung.db" r,
       "redaten.db" d
WHERE  (k.KuNr = r.KuNr) AND
       (d.ReNr = r.ReNr) AND
       (d.ArtNr = a.ArtNr) AND
       (d.ReNr = 1)
```

F8: Artikelbestand in der Lagerverwaltung aktualisieren.

```
UPDATE "artikel.db" a
SET a.ArtAnzahl =
(SELECT a.ArtAnzahl - SUM(d.Menge) AS Wert
FROM "redaten.db" d
WHERE d.ArtNr = a.ArtNr
GROUP BY d.ArtNr)
;
SELECT * FROM "artikel.db"
```

F9: Artikelbestellungen in der Lagerverwaltung ermitteln.

```
UPDATE "artikel.db"
SET ArtBestell = ArtMax - ArtAnzahl
WHERE ArtAnzahl < ArtMin
;
SELECT * FROM "artikel.db"
```

Aktualisierte Tabelle "rechnung.db"

ReNr	KuNr	ReDatum	ReBetrag	ReBezahlt	ReOffen
1	2	01.01.00	16,00	0	0
2	3	01.01.00	3,00	0	0
3	1	01.01.00	24,00	0	0
4	1	02.01.00	5,00	0	0
5	2	02.01.00	12,00	0	0
6	2	03.01.00	2,00	0	0

V. Der Multifunktions-Editor CSVTEX

---

(1) Die Verwaltung von Tabellen

CSV-Listen (hier als Tabellen oder Datenbanken bezeichnet) können entweder aus EXCEL importiert oder in DBSQL selbst erzeugt werden. Der Texteditor kann auch verwendet werden, um mit Menü **<Serienbrief>** einfache Serienbriefe zu erzeugen oder mit Menü **<Mathematik>** die Listen statistisch auszuwerten.

Eine CSV-Liste (Tabelle, Datenbank) besteht aus Textzeilen (Datensätzen), in denen verschiedene Datenfelder durch das Separatorzeichen ";" voneinander getrennt sind. Die erste Zeile der Datenbank (Kopfzeile, header) muss die Feldnamen enthalten (ebenfalls durch ";" getrennt). Die erste Spalte (linke Randspalte) enthält meistens Namen oder Satznummern. Als Feldbegrenzer werden auch Tabulatoren "TAB" oder Kommas "," verwendet. (CSV = comma separated values). Mittels Menü **<Bearbeiten-Separator>** werden die Feldbegrenzer ausgetauscht.

Das Menü **<Datei>** enthält die wichtigsten Datei-Operationen, wie beispielsweise Öffnen (Laden), Drucken oder Speichern. Mit **<Ordner>** wird zwischen aktuellem Ordner und Demo-Ordner gewechselt. Dort stehen dann zur Demonstration verschiedene CSV-Listen und auch passende Serienbriefe zur Verfügung.

<CsvTool> enthält mächtige Werkzeuge zur Listen-Verarbeitung.

Mit dem Menü **<Datenbank aktualisieren>** wird eine einmal geladene Tabelle unsichtbar wiederhergestellt und dabei der Editor gelöscht. Mit dem Menü **<Datenbank anzeigen>** wird die Tabelle im Editor angezeigt. Eine Datenbank kann hier aus maximal 2000 Sätzen mit maximal 50 Feldern bestehen.

Ein Serienbrief ist ein normaler Text im RTF-Format, wo für die Datenfelder Platzhalter (Feldvariable) geschrieben werden. Diese haben die Form <Fxx>, wo xx die Nummer des gewünschten Datenfeldes ist. Ein Serienbrief kann auch als Formular oder als einfache Liste gestaltet werden.

<Serienbrief anlegen> ... Erstellen eines neuen Serienbriefs
<Serienbrief speichern> ... Abspeichern eines Serienbriefs
<Serienbrief laden> ... Öffnen eines vorhandenen Serienbriefs

Mit dem Menü **<Serienbrief-Vorschau>** werden die Feldvariablen durch die entsprechenden Feldinhalte aus den Datensätzen der Tabelle schrittweise ersetzt. Dann kann dieser Serienbrief mit **<Serienbrief drucken>** gedruckt werden. Neuerliches Betätigen von **<Serienbrief-Vorschau>** liefert wieder die originale Serienbriefvorlage. Alle mit (E/A) gekennzeichneten Menü-Einträge arbeiten als einfache Wechselschalter.

Wenn statt der Feldvariablen <Fxx> die Namensvariablen <Nxx> in den Text geschrieben werden, dann werden bei der Vorschau dafür die Namen der Datenfelder eingesetzt.

In den Text kann auch EIN virtuelles Rechenfeld eingetragen werden. Dieses wird von zwei geschwungenen Klammern begrenzt, zwischen denen ein Rechenausdruck ohne Leerzeichen steht. Der Rechenausdruck kann Zahlen, Feldvariable <Fxx> von numerischen Datenfeldern und Rechenoperatoren enthalten. So berechnet z.B. {<F2>*<F1>/100} einen Prozentanteil, wobei <F1> der Grundwert und <F2> der Prozentsatz sind.

Spitze und geschwungene Klammern sind reservierte Zeichen zur Kennzeichnung von Feldvariablen. Sie dürfen in dem Text nicht anders verwendet werden.

Der Text kann mit Menü <Datei-Speichern> als Text (TXT,CSV) ohne Textauszeichnungen oder im Rich-Text-Format (RTF) abgespeichert werden. Ein Zeilenvorschub erfolgt mit <Enter>. Ein Seitenvorschub erfolgt mit <Strg><Enter>. Dabei kennzeichnet das reservierte Zeichen "PI" einen Seitenvorschub.

Die Operation **<Suchen>** im Menü **<Bearbeiten>** erfolgt immer ab der aktuellen Cursorposition bis zum Textende. Mit der Taste <F3> kann dann weitergesucht werden.

<Bearbeiten-Sortieren> ermöglicht die Sortierung einer Tabelle entsprechend den Werten in einer bestimmten Spalte.

<Compress> verdichtet Tabellen durch Entfernung aller Blanks. **<Expand>** erweitert alle Felder auf passende Minimallängen. Wenn bei einer Operation die Grenzen der Felder ungleichmäßig verschoben werden, sind **<Compress>** und **<Expand>** auszuführen!

<Bearbeiten-Selektion> führt eine Datenselektion durch, d.h. es werden nur jene Datensätze aus der Tabelle herausgefiltert, welche ein entsprechendes Selektionskriterium erfüllen. **<Bearbeiten-Restore>** macht die letzte Selektion rückgängig.

Mit **<Schrift-Insert ">** werden alle Felder der Tabelle von Hochkommas eingeschlossen. Mit **<Schrift-Remove ">** können alle Hochkommas um die Felder entfernt werden. Mittels **<Schrift-AddLastSep>** wird an jedes Satzende ein Separator platziert. Mittels **<Schrift-DelLastSep>** wird an jedem Satzende der Separator gelöscht.

HINWEIS 1: Wenn eine Tabelle aus Noten besteht, so bedeutet das Zeichen "a" nur "abwesend" bzw. "keine Note erhalten". Als Schulnoten (Eins bis Fünf) werden solche Datenfelder erkannt, deren Typ ganzzahlig numerisch ist und deren Feldlänge 1 ist. Mittels **<Mathematik-Schulnoten>** wird die Interpretation der Daten als Schulnoten ein- und ausgeschaltet, d.h. alle "0" werden durch "a" ersetzt oder umgekehrt.

HINWEIS 2: Wenn eine CSV-Liste in den Editor geladen wird, dann ist der Editor automatisch gesperrt, sodass keine Daten der Liste geändert werden können. Dann sind nur Serienbriefe und statistische Auswertungen möglich. Bei Menü **<Datei-Neu>** wird der Editor automatisch entsperrt. Die Sperre erfolgt auch nicht bei der Verarbeitung von reinen Texten (RTF,TXT).

HINWEIS 3: Steht in einem Datenfeld der Name einer Multimediadatei (Text, Grafik, Sound, Video), dann kann diese einfach aufgerufen werden. Dazu muss zuerst der Cursor auf das Feld platziert und dann die rechte Maustaste gedrückt werden.

---

(2) Die Auswertung von Tabellen

(2a) STATISTIK

Wenn eine Datentabelle mittels Menü **<Datenbank anzeigen>** im Editor erscheint, dann können numerische Felder statistisch ausgewertet werden. Dazu klickt man einfach mit der Maus in die gewünschte Datenspalte und dann auf **<Mathematik-Statistik>**.

Mit Menü **<Mathematik-Statistik-1>** werden eindimensionale Daten (Spalte X1) statistisch ausgewertet. Dabei werden Mittelwert, Streuung und Häufigkeitsverteilung ermittelt.

Mit Menü **<Mathematik-Statistik-2>** werden zweidimensionale Daten (Spalten X1,X2) statistisch ausgewertet. Dabei werden Korrelation, Regression und Streuungsdiagramm ermittelt.

Mit Menü **<Mathematik-Statistik-N>** werden mehrdimensionale Daten (Spalten X1,X2,X3,...) statistisch ausgewertet. Dabei werden auch die Interkorrelationen aller Daten berechnet.

In der letzten Zeile einer auszuwertenden Tabelle muss unbedingt immer das Markierungszeichen # für das Tabellenende stehen. Ansonsten ist keine statistische Auswertung möglich.

(2b) MATHEMATIK

Im Editor können in einer Zeile mathematische Rechenformeln mit beliebigen Zahlen und den konventionellen Operatoren und Funktionen eingegeben werden. Außerdem können genau 26 Variable a,b,c,... y,z zur Speicherung verwendet werden. Eine mathematische Rechnung wird ausgeführt, indem man den Cursor auf die Zeile stellt und <F1> drückt. Folgendes gilt:

```
a = Zahl            <F1>  speichert die Zahl auf a
a = Formel(a,b,...) <F1>  wertet die Formel aus und
                          speichert den Wert auf a
a =                 <F1>  zeigt den Wert von a an
Formel(a,b,...) =   <F1>  wertet die Formel aus und
                          zeigt den Formelwert an
OPERATOREN: (,)+,-,*,/,^
FUNKTIONEN: abs,acos,asin,atan,cos,deg,exp,fak,log,ln,
            rad,round,sin,sqr,sqrt,tan,trunc,pi.
```

Mit **<Mathematik-Funktion-Tabelle>** können beliebige mathematische Funktionen tabelliert werden. Dazu muss eine Textzeile mit der Form F(x),a,b,c genau markiert werden. F(x) ist die Funktionsformel, a der Anfangswert, b der Endwert und c die Schrittweite der Tabellierung. In der Funktionsformel F(x) ist x das Argument und zusätzlich können auch die 23 Variablen d,e,f,... y,z für reelle Zahlen verwendet werden, z.B. sqrt(r^2-x^2),a,b,c. Dabei müssen auf r,a,b,c vorher feste Zahlenwerte abgespeichert worden sein.

Mit **<Mathematik-Funktion-Grafik>** können Funktionen grafisch dargestellt werden. Dazu muss eine Textzeile mit der Form F(x),b genau markiert werden. F(x) ist die Funktionsformel und b ist die Halbbreite des Koordinatensystems.

Das Programm <CSVTEX> kann aus dem Datenbankprogramm <DBSQL> direkt aufgerufen werden.
Die nachfolgenden Anwendungsbeispiele sollen die Arbeitsweise des Programms demonstrieren.

Beispiel 1: Die Datenbank "demodat.csv" als Textliste mit "TAB" als Feldbegrenzer.

NUMMER	ZUNAME	VORNAME	SEX	ORT	TELEFON	GEWICHT	GRÖSSE	WERT
1	Meier	Harald	m	Wien	01/8691405	87	187	1.00
2	Ronka	Heidi	w	Graz	0316/234567	65	167	1.02
3	Dorfer	Maria	w	Graz	0316/406780	83	160	0.77
4	Stanka	Rudolf	m	Linz	0732/123361	61	171	1.10
5	Zenz	Eva	w	Wien	01/2435679	49	167	1.18
6	Wille	Heinz	m	Linz	0732/446570	82	182	1.00
7	Rutger	Herbert	m	Wien	01/6789034	74	178	1.04
8	Haür	Friedl	m	Linz	0732/345210	76	178	1.02
9	Müller	Roland	m	Wien	01/4567891	81	185	1.04
10	Wollner	Christa	w	Linz	0732/543022	67	156	0.89
11	Klaus	Eva	w	Wien	01/2342107	58	160	1.02
12	Holler	Sabine	w	Wien	01/6002342	58	170	1.12
13	Ebner	Harald	m	Graz	0316/432276	78	178	1.00
14	Gral	Walter	m	Graz	0316/503327	72	178	1.06
15	Kurz	Werner	m	Wien	01/4567802	73	185	1.12
16	Stadler	Susi	w	Linz	0732/874533	56	169	1.13
17	Wolf	Maria	w	Graz	0316/705688	58	167	1.09
18	Klad	Eva	w	Linz	0732/106578	52	165	1.13
19	Artner	Heinz	m	Wien	01/7895430	61	173	1.12
20	Baier	Helene	w	Wien	01/5432081	65	171	1.06
21	Wallner	Horst	m	Graz	0316/102340	83	184	1.01
22	Neuner	Eva	w	Graz	0316/453019	56	170	1.14
23	Troll	Rudolf	m	Linz	0732/445687	69	178	1.09
24	Sarg	Willi	m	Wien	01/7864451	93	182	0.89
25	Toth	Maria	w	Wien	01/4453021	61	157	0.96

Beispiel 2: Die Serienbrief-Vorlage "sbrief1.rtf" und eine Serienbrief-Vorschau.

(a) Serienbrief-Vorlage:

Wien, 1.April.2002
Firma "Werbeflut"

Liebe(r) Frau/Herr,

<F2> <F3>

Irrtümlicher Weise hat uns die Firma "Datenklau GMBH"
Ihre persönlichen Kenndaten übermittelt. So wissen wir,
dass Sie in ***<F5>*** wohnen und die Telefonnummer ***<F6>***
haben. Außerdem sind Sie ***<F8>*** cm groß und ***<F7>*** kg schwer.
Wir bitten Sie um Bestätigung dieser persönlichen Daten,
so dass wir Sie mit unserer Werbung überfluten können.

Hochachtungsvoll
Franz Werbemeister

(b) Erster Serienbrief-Ausdruck:

Wien, 1.April.2002
Firma "Werbeflut"

Liebe(r) Frau/Herr,

Meier Harald

Irrtümlicher Weise hat uns die Firma "Datenklau GMBH"
Ihre persönlichen Kenndaten übermittelt. So wissen wir,
dass Sie in ***Wien*** wohnen und die Telefonnummer ***01/8691405***
haben. Außerdem sind Sie ***187*** cm groß und ***87*** kg schwer.
Wir bitten Sie um Bestätigung dieser persönlichen Daten,
so dass wir Sie mit unserer Werbung überfluten können.

Hochachtungsvoll
Franz Werbemeister

Beispiel 3: Die Serienbrief-Vorlage "sbrief02.rtf" und eine Serienbrief-Vorschau in Tabellenform (ohne Seitenvorschub). Dabei werden nur die drei Datenfelder <F2>, <F3> und <F5> (Zuname, Vorname und Telefon) ausgegeben.

Vorlage:

```
<F2> <F3>: <F5>
```

Vorschau:

```
Meier Harald: 01/8691405
Ronka Heidi: 0316/234567
Dorfer Maria: 0316/406780
Stanka Rudolf: 0732/123361
Zenz Eva: 01/2435679
Wille Heinz: 0732/446570
Rutger Herbert: 01/6789034
Hauer Friedl: 0732/345210
Müller Roland: 01/4567891
Wollner Christa: 0732/543022
Klaus Eva: 01/2342107
Holler Sabine: 01/6002342
Ebner Harald: 0316/432276
Gral Walter: 0316/503327
Kurz Werner: 01/4567802
Stadler Susi: 0732/874533
Wolf Maria: 0316/705688
Klad Eva: 0732/106578
Artner Heinz: 01/7895430
Baier Helene: 01/5432081
Wallner Horst: 0316/102340
Neuner Eva: 0316/453019
Troll Rudolf: 0732/445687
Sarg Willi: 01/7864451
Toth Maria: 01/4453021
Toth Maria: 01/4453021
```

Beispiel 4: Die Datenbank (Tabelle im CSV-Format) "noten5a+.csv" wird in den Editor geladen und automatisch mit der Endmarke # versehen. Mit <Datenbank anzeigen> wird die Tabelle im Editor dargestellt. Dann wird das Menü <Mathematik-Statistik-N> aufgerufen. Damit erfolgt eine N-dimensionale statistische Datenauswertung.

```
NAMEN;RELIGION;DEUTSCH;ENGLISCH;LATEIN;FRANZÖSISCH;GEOGRAFIE;
GESCHICHTE;MATHEMATIK;BIOLOGIE;MUSIK_ERZIEHUNG;BILDN_ERZIEHUNG;
PHYSIK;INFORMATIK;LEIBESÜBUNGEN_K;LEIBESÜBUNGEN_M;ZUSATZ;

ADAM Martina        ;1;2;1;1;1;2;1;1;1;1;1;1;a;1;w,sz      ;
BRINDL Franz        ;1;3;5;2;4;3;2;3;3;1;1;4;2;2;a;m,z,rep  ;
FÜRNDORF Hubert     ;1;2;4;2;4;4;2;5;4;1;1;3;3;2;a;m,sz,rep ;
GANZER Barbara      ;1;2;3;2;4;3;3;3;3;1;1;2;3;a;1;w,sz     ;
GSCHWANDT Erwin     ;1;3;4;3;4;4;3;5;3;1;2;4;4;1;a;m,z,rep  ;
KLEEDORFER Alois    ;1;1;4;3;5;4;2;3;3;1;1;2;4;2;a;m,z,rep  ;
KRISTL Teresa       ;1;2;3;1;2;2;2;2;2;1;1;1;1;a;1;w,sz     ;
MILEK Steffi        ;1;1;2;1;3;2;2;4;2;1;2;2;2;a;1;w,sz     ;
PARKER Heinz        ;a;2;3;1;2;2;2;4;1;1;2;3;1;1;a;m,z      ;
POLGAR Friedrich    ;1;3;3;1;2;3;2;4;2;1;2;1;2;1;a;m,sz     ;
RADINGER Eva        ;1;1;2;2;2;3;2;3;2;1;1;4;3;a;1;w,sz     ;
ROGER Anna          ;a;2;1;1;1;2;1;2;1;1;1;1;1;a;1;w,sz     ;
RUKLINER Sabrina    ;1;3;3;1;2;3;2;3;1;1;1;3;1;a;1;w,sz     ;
SCHRECK Christina   ;1;3;3;1;4;4;1;4;1;1;2;4;1;a;1;w,z      ;
STANKA Roland       ;a;3;4;2;3;4;3;4;2;1;1;2;3;2;a;m,z      ;
STEMMER Michaela    ;1;2;3;2;3;2;2;3;2;1;2;3;1;a;1;w,sz     ;
TASCHNER Christian  ;1;3;4;2;4;3;2;5;3;1;2;4;3;1;a;m,z,rep  ;
WELLIGER Herbert    ;1;2;1;2;2;4;2;4;3;1;2;2;2;1;a;m,z      ;
WENZL Harald        ;1;2;3;2;3;4;3;2;3;1;1;3;1;1;a;m,sz     ;
ZENZ Klara          ;a;2;4;3;4;5;3;4;3;1;1;4;4;a;2;w,z,rep  ;
#
```

```
Statistische Datenauswertung der Zeilen:

20 Zeilen               (Anzahl,Mittel,Streuung)

(  1)  ADAM Martina       ,   14, 1.14, 0.35
(  2)  BRINDL Franz       ,   14, 2.57, 1.18
(  3)  FÜRNDORF Hubert    ,   14, 2.71, 1.28
(  4)  GANZER Barbara     ,   14, 2.29, 0.96
(  5)  GSCHWANDT Erwin    ,   14, 3.00, 1.25
(  6)  KLEEDORFER Alois   ,   14, 2.57, 1.29
(  7)  KRISTL Teresa      ,   14, 1.57, 0.62
(  8)  MILEK Steffi       ,   14, 1.86, 0.83
(  9)  PARKER Heinz       ,   13, 1.92, 0.92
( 10)  POLGAR Friedrich   ,   14, 2.00, 0.93
( 11)  RADINGER Eva       ,   14, 2.00, 0.93
( 12)  ROGER Anna         ,   13, 1.23, 0.42
( 13)  RUKLINER Sabrina   ,   14, 1.86, 0.91
( 14)  SCHRECK Christina  ,   14, 2.21, 1.32
( 15)  STANKA Roland      ,   13, 2.62, 1.00
( 16)  STEMMER Michaela   ,   14, 2.00, 0.76
( 17)  TASCHNER Christian ,   14, 2.71, 1.22
( 18)  WELLIGER Herbert   ,   14, 2.07, 0.96
( 19)  WENZL Harald       ,   14, 2.14, 0.99
( 20)  ZENZ Klara         ,   13, 3.08, 1.21
```

```
Statistische Datenauswertung der Spalten:

15 Spalten               (Anzahl,Mittel,Streuung)

(  2)  RELIGION           ,   16, 1.00, 0.00
(  3)  DEUTSCH            ,   20, 2.20, 0.68
(  4)  ENGLISCH           ,   20, 3.00, 1.10
(  5)  LATEIN             ,   20, 1.75, 0.70
(  6)  FRANZÖSISCH        ,   20, 2.95, 1.12
(  7)  GEOGRAFIE          ,   20, 3.15, 0.91
(  8)  GESCHICHTE         ,   20, 2.10, 0.62
(  9)  MATHEMATIK         ,   20, 3.40, 1.07
( 10)  BIOLOGIE           ,   20, 2.25, 0.89
( 11)  MUSIK_ERZIEHUNG    ,   20, 1.00, 0.00
( 12)  BILDN_ERZIEHUNG    ,   20, 1.40, 0.49
( 13)  PHYSIK             ,   20, 2.65, 1.11
( 14)  INFORMATIK         ,   20, 2.15, 1.11
( 15)  LEIBESÜBUNGEN_K    ,   10, 1.40, 0.49
( 16)  LEIBESÜBUNGEN_M    ,   10, 1.10, 0.30


Interkorrelationen der Spalten:

-------2-----3-----4-----5-----6-----7-----8-----9----10----11----12----13----14----15----16---
--2--- 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 1.00
--3--- 0.00, 1.00, 0.40,-0.11, 0.08, 0.19, 0.07, 0.30,-0.08, 0.00, 0.21, 0.23,-0.11,-0.18,-0.00
--4--- 0.00, 0.40, 1.00, 0.52, 0.78, 0.45, 0.51, 0.47, 0.51, 0.00,-0.09, 0.54, 0.49, 0.60, 0.54
--5--- 0.00,-0.11, 0.52, 1.00, 0.69, 0.69, 0.63, 0.34, 0.75, 0.00,-0.15, 0.47, 0.82, 0.32, 0.75
--6--- 0.00, 0.08, 0.78, 0.69, 1.00, 0.60, 0.44, 0.52, 0.67, 0.00, 0.04, 0.55, 0.65, 0.57, 0.42
--7--- 0.00, 0.19, 0.45, 0.69, 0.60, 1.00, 0.50, 0.45, 0.57, 0.00,-0.13, 0.45, 0.62, 0.30, 0.75
--8--- 0.00, 0.07, 0.51, 0.63, 0.44, 0.50, 1.00, 0.31, 0.59, 0.00,-0.13, 0.27, 0.56,-0.09, 0.52
--9--- 0.00, 0.30, 0.47, 0.34, 0.52, 0.45, 0.31, 1.00, 0.42, 0.00, 0.55, 0.50, 0.54,-0.13, 0.39
-10--- 0.00,-0.08, 0.51, 0.75, 0.67, 0.57, 0.59, 0.42, 1.00, 0.00,-0.12, 0.29, 0.67, 0.31, 0.53
-11--- 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00
-12--- 0.00, 0.21,-0.09,-0.15, 0.04,-0.13,-0.13, 0.55,-0.12, 0.00, 1.00, 0.17,-0.11,-0.82,-0.22
-13--- 0.00, 0.23, 0.54, 0.47, 0.55, 0.45, 0.27, 0.50, 0.29, 0.00, 0.17, 1.00, 0.33,-0.04, 0.42
-14--- 0.00,-0.11, 0.49, 0.82, 0.65, 0.62, 0.56, 0.54, 0.67, 0.00,-0.11, 0.33, 1.00, 0.40, 0.68
-15--- 0.00,-0.18, 0.60, 0.32, 0.57, 0.30,-0.09,-0.13, 0.31, 0.00,-0.82,-0.04, 0.40, 1.00, ***
-16--- 1.00,-0.00, 0.54, 0.75, 0.42, 0.75, 0.52, 0.39, 0.53, 0.00,-0.22, 0.42, 0.68, *** , 1.00

Kleinste Korrelation = -0.82 [LEIBESÜBUNGEN_K (15) / BILDN_ERZIEHUNG (12)]
Größte Korrelation   =  1.00 [LEIBESÜBUNGEN_M (16) / RELIGION (2)]

Gesamter Mittelwert  =  2.18
Mittlere Streuung    =  0.70
```

Das obige Beispiel zeigt die vollständige Statistik einer Schulklasse. Wenn die Notentabelle der Klasse neben dem Notenraster noch eine linke Randspalte (Namen) und eine Kopfzeile (Fächer) aufweist, erkennt das Programm automatisch, dass es sich um Schulnoten (Zahlen von 1 bis 5) handelt und wertet diese entsprechend aus. Zusätzlich kann optional noch eine rechte Randspalte angefügt sein, wo sich weitere Textinformationen befinden. Diese muss unbedingt den Feldnamen ZUSATZ haben. Wenn ein Schüler ein Fach nicht besucht, dann muss in dem entsprechenden Datenfeld entweder ‚0' oder ‚a' stehen. Diese Datenfelder, so wie auch Kopfzeilen und Randspalten werden erkannt und bei der Statistik nicht berücksichtigt.

Handelt es sich bei der Datentabelle um keine Schulnoten, sondern um beliebige Zahlenwerte, dann können diese natürlich ebenfalls automatisch statistisch ausgewertet werden (Menüeintrag <Mathematik-Schulnoten>).

Beispiel 5: Mathematische Berechnungen in einem Dreieck, von dem die drei Seiten a = 8 cm, b = 6 cm und c = 10 cm gegeben sind. Gesucht sind die Winkel x (=α), y (=β), z (=γ), der Umfang u und die Fläche f. (Jede Formel-Zeile muss mit <F1> bestätigt werden !)

```
a =  8.00
b =  6.00
c = 10.00

z = acos((a^2+b^2-c^2)/(2*a*b)) = 90.00
x = asin(a*sin(z)/c) = 53.13
y = 180-(x+z) = 36.87

u = a+b+c = 24.00
s = u/2 = 12.00
f = sqrt(s*(s-a)*(s-b)*(s-c)) = 24.00
```

Beispiel 6: Zeichnen der Kreisfunktion y = sqrt(r^2-x^2) mit Radius r = 5 in einem Koordinatensystem mit der Halbbreite b = 8.

```
b  = 8.00
r  = 5.00
n  = 2.00

sqrt(r^n-x^n), b          ← Diese Zeile markieren und <Mathematik-Funktion-Grafik>
```

Beispiel 7: Tabellieren der mathematischen Funktion $y = \sqrt{(r^2 - x^2)}$ für r = 5 auf dem Intervall [-8,8] mit der Schrittweite 1.

```
a = -8.00
b =  8.00
c =  1.00

r =  5.00
n =  2.00

sqrt(r^n-x^n), a, b, c   ← Diese Zeile markieren und <Mathematik-Funktion-Tabelle>

-8.00  /  *****
-7.00  /  *****
-6.00  /  *****
-5.00  /  0.00
-4.00  /  3.00
-3.00  /  4.00
-2.00  /  4.58
-1.00  /  4.90
0.00   /  5.00
1.00   /  4.90
2.00   /  4.58
3.00   /  4.00
4.00   /  3.00
5.00   /  0.00
6.00   /  *****
7.00   /  *****
8.00   /  *****
```

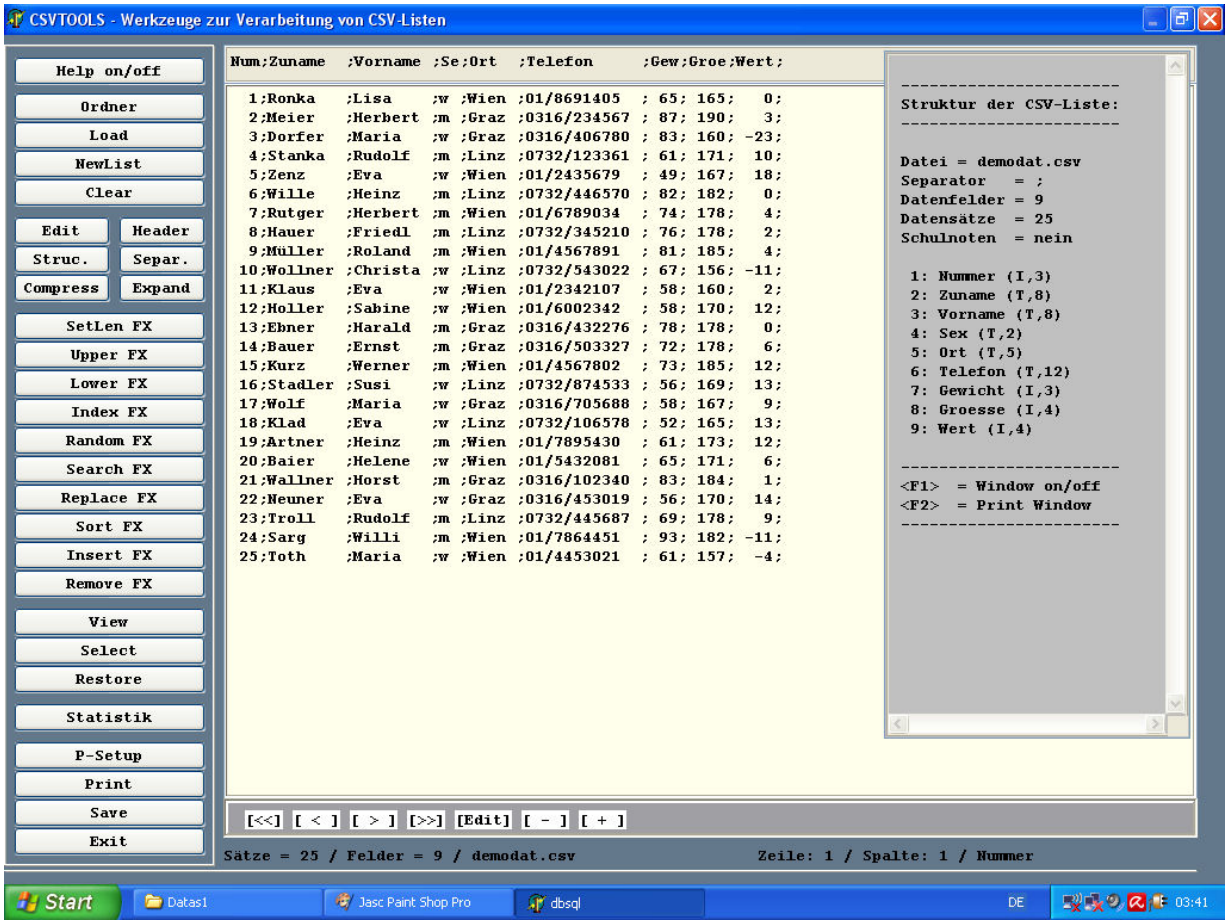
Die Beispiele 5, 6 und 7 finden sich in der Demo-Datei <mathe.rtf>.

Die in <CSVTEX> implementierte Mathematik stellt ein mächtiges Werkzeug dar, mit dem viele Probleme auf einfache Art und Weise gelöst werden können.

==========

<CSVTOOLS> ist eine Sammlung von Werkzeugen zur Verarbeitung von CSV-Listen

==========



Die sog. CSV-Listen (comma separated values) sind extern als reine Textdateien gespeichert und intern als dynamische Stringlisten repräsentiert. Die Zeilen der Liste werden als Datensätze (records), die Spalten werden als Felder (fields) bezeichnet. Die Feldnamen (Spaltennamen) werden in der Kopfzeile (header) angezeigt. Es sind maximal 2000 Datensätze mit maximal 50 Datenfeldern möglich, welche durch Separatoren (Strichpunkte) getrennt sind. Jedes Feld kann maximal 30 Zeichen enthalten (Feldlänge). Grundsätzlich werden drei Feldtypen unterschieden: Textfelder (T), numerische Felder (N) und ganzzahlig numerische Felder (I). Das Programm <CSVTOOLS> kann als ein nützliches und einfach zu bedienendes Datenbankprogramm verwendet werden. Es eignet sich auch gut für eine Einschulung in die Datenbank-Verwaltung. CSV-Listen können in EXCEL direkt importiert oder exportiert werden.

<Help on/off>	Blendet diesen Hilfstext ein/aus. Ausdruck mit Doppelklick.
<Ordner>	Wechselt zwischen aktuellem Ordner und Demo-Ordner.
<Load>	Öffnet eine bestehende CSV-Datei im Editor.
<NewList>	Erzeugt eine neue CSV-Liste mit dem Feldgenerator.
<Clear>	Löscht die Liste im Editor.
<Edit>	Ruft den Maskeneditor zur Datenedition auf.
<Header>	Blendet den Header (Kopfzeile) ein/aus. Dadurch können die Feldnamen beliebig geändert werden.
<Struc.>	Blendet die Listenstruktur ein/aus.
<Separ.>	Ersetzt die Separatoren ";" oder "TAB" oder ",".

<Compress>	Verdichtet die Liste durch Entfernung aller Blanks.
<Expand>	Erweitert alle Felder auf passende Minimallängen. WICHTIG: Wenn bei einer Operation sich die Feldränder ungleich verschieben, dann <Compress> und <Expand> ausführen!
<SetLen FX>	Ändert die Länge von Feld X auf eine gewünschte Zeichenanzahl.
<Upper FX>	Setzt Feld X auf Großschrift.
<Lower FX>	Setzt Feld X auf Kleinschrift.
<Index FX>	Belegt ein bestimmtes Feld X mit fortlaufenden Zeilennummern.
<Random FX>	Belegt ein bestimmtes Feld X mit Zufallswerten. Es wird mit dem Zufallsgenerator ein Zeichen-String erzeugt, wo die ersten Zeichen zufällige Kleinbuchstaben und und die zweiten Zeichen zufällige Ziffern sind. Die Anzahl der zufälligen Buchstaben und Ziffern ist wählbar (1 bis 16).
<Search FX>	Sucht im gewählten Feld X nach einem Suchbegriff. Die Suche ist unabhängig von Groß- oder Kleinschrift.
<Replace FX>	Ersetzt alle Daten eines gewählten Feldes X durch einen Ersatzausdruck. In diesem kann auch auf die Daten anderer Felder mit Hilfe von Feldvariablen <Fxx> Bezug genommen werden. Eine solche Feldvariable besteht aus dem Zeichen F gefolgt von der Feldnummer xx. Beides wird in spitze Klammern gesetzt (field replacement). Alternativ kann auch <Fxx,a,b> eingegeben werden. Dann werden aus den Feldinhalten ab der Position a genau b Zeichen extrahiert. Der verwendete Ersatzausdruck kann in eine Datei abgespeichert oder auch geladen werden. Der Ersatzausdruck kann auch numerisch berechnet werden. Beispielsweise kann das Feld F11 der Demoliste durch den den Rechenausdruck "(2*<F9> - <F10>)/5" ersetzt werden.
<Sort FX>	Sortiert die Liste entsprechend einem gewählten Datenfeld.
<Insert FX>	Fügt an der Spaltenposition X ein neues Datenfeld ein.
<Remove FX>	Löscht das Datenfeld an der Spaltenposition X.
<View>	Projektion auf ausgewählte Felder (sinnvoll für <Print>). Bei Projektionen auf ein Feld kann dieses editiert werden.
<Select>	Selektioniert (filtert) Datensätze mehrstufig.
<Restore>	Stellt nach Selektionen die originale Liste wieder her.
<Statistik>	Wertet numerische Datenfelder statistisch aus.
<P-Setup>	Ermöglicht die Änderung von Drucker-Einstellungen.
<Print>	Druckt auswählbare Datenzeilen der Liste.
<Save>	Speichert die Liste in eine externe CSV-Datei.
<Exit>	Beendet das Programm.

Die besonderen Tastenfunktionen:

Mit der Taste <F1> wird die Listenstruktur ein-/ausgeblendet.
Mit <F2> werden eingeblendete Informationsfenster ausgedruckt.
Mit der Taste <F3> wird bei Suchoperationen immer weitergesucht.
Mit <F4> oder <F5> wird die Schrift verkleinert oder vergrößert.
Mit <F6> oder <F7> werden Hochkommas entfernt oder eingefügt.
Mit <F8> oder <F9> werden Strichpunkte am Zeilenende entweder entfernt oder eingefügt.
Mit <F10> wird Editor und Header gesperrt oder entsperrt.
Mit <F11> wird die Breite von Informationsfenstern verändert.
Mit <F12> wird die Schulnoten-Interpretation ein/-ausgeschaltet.

Die Selektion von Daten:

Selektionieren (Filtern) einer Tabelle nach einer Filterbedingung, d.h. alle Daten eines gewählten Feldes werden mit Hilfe eines Operators mit einem Suchbegriff verglichen. Dann werden nur die Datensätze der Tabelle angezeigt, wo die Filterbedingung erfüllt ist. Die Filterbedingung hat folgende Form: Felddaten (FD) - Operator (OP) - Suchbegriff (SB).

Sechs Filteroperationen (OP) in Textfeldern:

"!" filtert jene FD, welche den SB nicht enthalten.
"=" filtert jene FD, welche gleich SB sind.
"<" filtert jene FD, welche im Alphabet vor SB sind.
">" filtert jene FD, welche im Alphabet hinter SB sind.
"-" filtert jene FD, welche mit SB beginnen.
"+" filtert jene FD, welche den SB enthalten.

Sechs Filteroperationen (OP) in Zahlenfeldern:

"!" filtert jene FD, welche ungleich SB sind.
"=" filtert jene FD, welche gleich SB sind.
"<" filtert jene FD, welche kleiner SB sind.
">" filtert jene FD, welche größer SB sind.
"-" filtert jene FD, welche kleiner/gleich SB sind.
"+" filtert jene FD, welche größer/gleich SB sind.

Die Datensätze der Tabelle, die die Filterbedingung erfüllen, können dann weiter verarbeitet oder neuerlich gefiltert werden. (Mehrstufige Selektion). Die Selektionen werden mit <Restore> wieder aufgehoben.

Die Dateneingabe im Maskeneditor:

Hat man eine bereits bestehende Datenliste geladen, dann wird mit Hilfe von <Edit> ein Maskeneditor aufgerufen. Dieser dient zur übersichtlichen Edition der einzelnen Datensätze. Mit dem Maskeneditor kann man in der Liste navigieren, bestehende Datensätze ausdrucken (<Print>) oder ändern (<Write>) und neue Datenzeilen eingeben (<New> und <Write>). Mit <Exit> wird der Maskeneditor verlassen.

Das Anlegen einer neuen CSV-Liste:

Die Erzeugung einer neuen Liste wird mit dem Schalter <NewList> ausgeführt. Mit einem Feldgenerator wird zuerst die Struktur der neuen CSV-Liste eingegeben, d.h. die Anzahl der Datenfelder, der Namen, die Länge und der Datentyp der Datenfelder. Nach diesen korrekten Felddefinitionen werden mit Hilfe des Maskeneditors die neuen Datensätze eingegeben. Nach Beendigung der Eingabe kann die Liste weiter verarbeitet werden.

Das Aufrufen von Multimedia-Dateien:

In Feldern vom Datentyp "T" können die Namen von Multimedia-Dateien (Text-, Sound-, Grafik- und Video-Dateien) eingetragen werden. Nachdem man den Cursor darauf platziert hat, wird mit einem Klick der rechten Maustaste die Datei geöffnet. Voraussetzungen sind, dass diese Dateien im aktuellen Ordner sind und es dazu ausführende Programme gibt.

Die Nummern (Namen) angeklickter Datenspalten werden automatisch erkannt.
Am unteren Fensterrand befindet sich zusätzlich eine Navigationsleiste.
Dort können u.a. Datensätze gelöscht oder neue Datensätze angefügt werden.
Alle Informationsfenster sind mit gehaltener linker Maustaste verschiebbar.

VI. Kurzbeschreibung des Datenbanksystems

Das komplette Datenbanksystem besteht aus genau vier Dateien:

(1) Das Installationsprogramm ***DBSETUP.EXE***.
(2) Die eigentliche Datenbankverwaltung ***DBSQL.EXE***.
(3) Die Einführung in Datenbanken ***DBHELP.PDF***.
(4) Diese Informationsdatei ***README.TXT***.

Das Programm DBSQL.EXE dient zur Erzeugung und Verwaltung von Datenbanken im Paradox-Format. Die Steuerung erfolgt entweder mit Hilfe der Structured Query Language (SQL) in einem eigenen SQL-Editor oder alternativ mit Hilfe von einfachen Schaltern. Das geschieht in zwei getrennten Programmteilen DBSQL und DBRUN, wobei immer vom einen zum anderen Modul gewechselt werden kann.

Eine Umwandlung von CSV-Tabellen aus EXCEL in Paradox-Datenbanken und umgekehrt ist möglich - auch ein Datenaustausch mit ACCESS.

Das System enthält einen multifunktionalen Texteditor CSVTEX zur Verarbeitung von CSV-Listen und zur statistischen Datenauswertung. Mit einem eigenen REPORT-Generator können sehr einfach beliebige Reports, Formulare und Serienbriefe im RTF-Format erzeugt werden. Ein ausführlicher Hilfetext ist im System integriert.

Mit dem Programm können auch Grafik-, Sound-, Video-, PDF-, Text-dateien archiviert und aufgerufen werden.

Zur Demonstration und Übung sind mehrere verschiedene Datenbanken (adressdat, demodat, mediadat, noten5a) und Hilfsdateien vorhanden. Das System enthält auch ein komplettes Firmenprojekt mit genau fünf Datenbanken (artikel, kunden, rechnung, redaten, rechn01).

Beim Programmstart von DBSQL.EXE werden zusätzlich zum aktuellen Ordner zwei neue Ordner angelegt, ein Ordner für die Demo-Projekte und ein Ordner für das Firmen-Projekt. Bei jedem Start des Programms werden die originalen Demonstrations-Datenbanken immer neu angelegt, d.h. alle Änderungen in diesen Datenbanken werden rückgängig gemacht. Für das Anlegen und Verwalten von eigenen Datenbanken sollte immer das aktuelle Verzeichnis gewählt werden.

Damit man mit dem Programm DBSQL.EXE arbeiten kann, muss zuerst einmalig das Installationsprogramm DBSETUP.EXE aufgerufen werden. Dadurch werden die verschiedenen Dateien der Borland Database Engine (BDE) und die Paradox-Treiberdateien auf der Festplatte installiert. Etwaige interne Dateihinweise liegen im Ordner "X:\Programme\dbpau". Die komplette BDE mit allen Treibern und Hilfsdateien wird im Ordner "X:\Programme\Gemeinsame Dateien\Borland Shared\BDE" gespeichert.

DBSQL.EXE speichert automatisch im aktuellen Ordner die Hilfsdatei PDOXUSRS.NET ab. Sie enthält nur Informationen zum Netzwerkbetrieb.

HINWEIS: Schreibt man den Namen einer Datenbank in eine Textdatei "dbname.txt", dann wird diese Datenbank beim Programmstart auto-matisch im aktuellen Ordner geöffnet.

==========