SIEMENS
NIXDORF

SINIX

INFORMIX-SQL V4.0

Kennenlernen

Benutzerhandbuch

DOPPEL

Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
AP Internationales
Dokumentationszentrum
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (0 89) 6 36-4 04 43

email im EUnet:
man@sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
AP Internationales
Dokumentationszentrum
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (0 89) 6 36-4 04 43

email im EUnet:
man@sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

INFORMIX-SQL (SINIX)

Kennenlernen

Benutzerhandbuch

Ausgabe Dezember 1990 (INFORMIX-SQL V4.0)

Anhang

. . . und Schulung?

Zu dem nachstehend beschriebenen Produkt, wie zu fast allen DV-Themen, bieten unsere regionalen Training Center in Berlin, Essen, Frankfurt/Main, Hannover, Hamburg, München, Mainz, Stuttgart, Wien und Zürich Kurse an:

Auskunft und Info-Material:

Systemfamilien 7·500 und 8890	**Telefon (0 89) 6 36-4 89 87**
Ein- und Mehrplatzsysteme	**Telefon (0 89) 6 36-4 24 80**

Siemens Nixdorf Training Center
Postfach 83 09 51, W-8000 München 83

Inhalt

1 Überblick über INFORMIX

Dieses Kapitel vermittelt Ihnen das notwendige Grundlagenwissen über INFORMIX. Sie erfahren,

- was eine Datenbank ist,
- was Tabellen, Spalten und Sätze sind,
- wofür Sie INFORMIX verwenden können,
- welche Werkzeuge Ihnen bei INFORMIX zur Verfügung stehen.

1.1 Die Datenbank

1.1.1 Alltägliche Datenbanken

Nicht nur Versicherungen, Banken und Supermärkte verwenden Datenbanken, wir alle benutzen tagtäglich und ganz selbstverständlich "Datenbanken".
Nämlich z.B.solche wie Lexika, Nachschlagewerke, Notizbücher.
All das kann man als gewöhnliche Datenbanken verstehen, als Datenbanken, die Informationen, Zahlen, Termine, Adressen und andere Daten mehr enthalten.

Lassen Sie sich am Beispiel der "gewöhnlichen Datenbanken" 'Notizbuch' und 'Lexikon' das Prinzip von INFORMIX-Datenbanken erläutern.

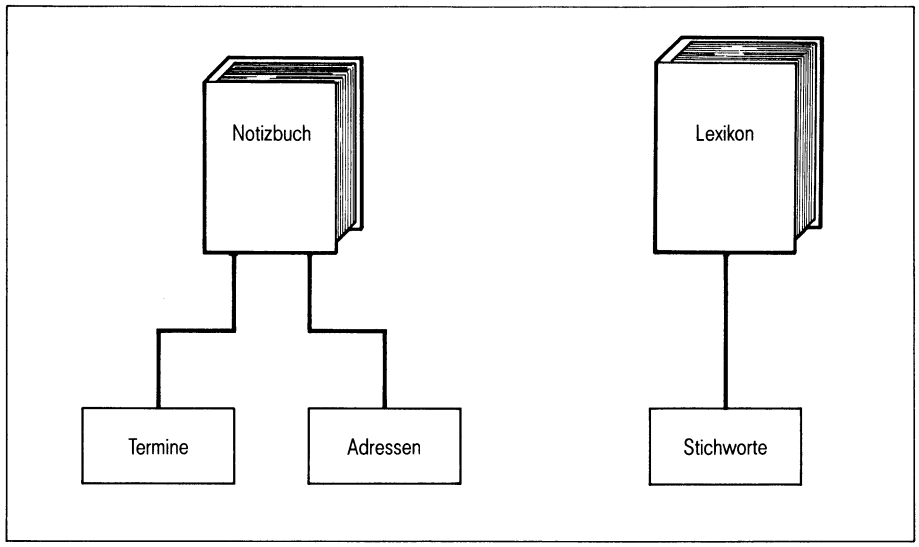


Bild 1-1 Gewöhnliche Datenbanken

Würden Sie versuchen, in Ihrem Notizbuch eine Information über Van Gogh zu finden? Oder umgekehrt, einen Termineintrag im Lexikon zu machen? Natürlich nicht, denn jede Datenbank dient nur einem fest umrissenen Sachgebiet:
Das Lexikon enthält ausschließlich Informationen über Maler, Schriftsteller u.a. und das Notizbuch enthält ausschließlich Termine und Adressen.

So verhält es sich auch bei INFORMIX:

Genauso wie Sie mehrere Lexika, Nachschlagewerke u.a. in Ihrem Bücherschrank führen, so können Sie nach Bedarf auch mehrere INFORMIX-Datenbanken führen.

Jede INFORMIX-Datenbank sollte nur Daten und Informationen aus einem bestimmten, abgegrenzten Sachgebiet enthalten.

Hinweis

Dieses Kapitel demonstriert Ihnen, wie entsprechende INFORMIX-Datenbanken 'notizbuch' und 'lexikon' aussehen könnten; anhand dieser Datenbanken werden einige Grundlagen erläutert. Das Kapitel 3 wird Sie dazu anleiten, den Adressen-Teil der hier vorgestellten Datenbank 'notizbuch' nachzubauen.

Um im weiteren Text zwischen den "gewöhnlichen Datenbanken" 'Notizbuch' und 'Lexikon' und den als Beispiel dienenden INFORMIX-Datenbanken 'notizbuch' und 'lexikon' unterscheiden zu können, wird bei INFORMIX-Datenbanknamen (sowie auch anderen Namen in diesem Zusammenhang) die Kleinschreibweise verwendet; in den Grafiken sind die INFORMIX-Datenbanken durch "Tonnen" symbolisiert.

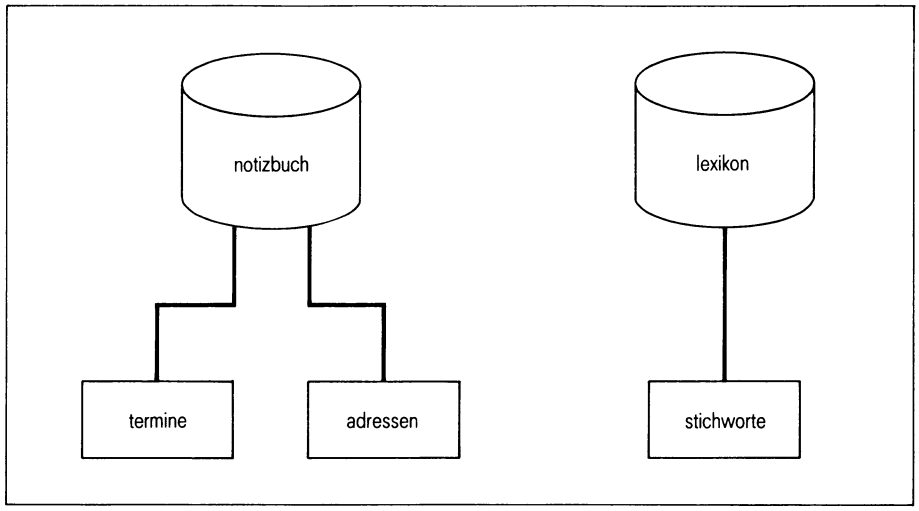


Bild 1-2 INFORMIX-Datenbanken 'notizbuch' und 'lexikon'

Was ist das Besondere an einer INFORMIX-Datenbank?

Eine INFORMIX-Datenbank kann Ihnen Informationen verschaffen, die Ihnen eine "gewöhnliche Datenbank" nicht verschaffen kann.
Wenn Sie beispielsweise in einem normalen Lexikon unter dem Stichwort 'Van Gogh' nachschlagen, dann werden Sie dabei ermitteln können, daß Van Gogh Maler war. Wenn Sie aber umgekehrt unter 'Maler' nachschlagen, wird Ihnen Ihr Lexikon kaum Auskunft über den Maler Van Gogh erteilen.

Wenn Ihr Lexikon aber eine INFORMIX-Datenbank ist, dann können Sie sich z.B. alle Künstlernamen mit dem Beruf Maler und dem Anfangsbuchstaben 'V' ausgeben lassen und würden dabei auch auf den Maler Van Gogh stoßen.

Diesen Vorgang, sich Informationen aus einer Datenbank ausgeben lassen, nennt man 'die Datenbank abfragen'.

1.1.2 Wie hält eine INFORMIX-Datenbank den Datenbestand?

Sie tut das in Form von Tabellen. Was bei einer gewöhnlichen Datenbank auf viele Papierseiten verteilt ist, das hält eine INFORMIX-Datenbank in Tabellen; Sie können sich eine INFORMIX-Tabelle ähnlich vorstellen wie jede bekannte Tabelle aus Büchern und Zeitschriften.

Wieviel Tabellen enthält eine INFORMIX-Datenbank?

Das hängt davon ab, wieviel Informationseinheiten Sie benötigen. Hierzu ein Vergleich mit 'Notizbuch' und 'Lexikon':
In einem gewöhnlichen Notizbuch gibt es wenigstens zwei Informationseinheiten, einen Informationseinheit 'Termine' und eine Informationseinheit 'Adressen'. Und in einem Lexikon gibt es wenigstens eine Informationseinheit 'Stichworte'.

So könnte dann eine INFORMIX-Datenbank 'notizbuch' entsprechend auch zwei Tabellen enthalten: Eine Tabelle 'termine' und eine Tabelle 'adressen'. Und eine INFORMIX-Datenbank 'lexikon' könnte wenigstens eine Tabelle 'stichworte' enthalten.

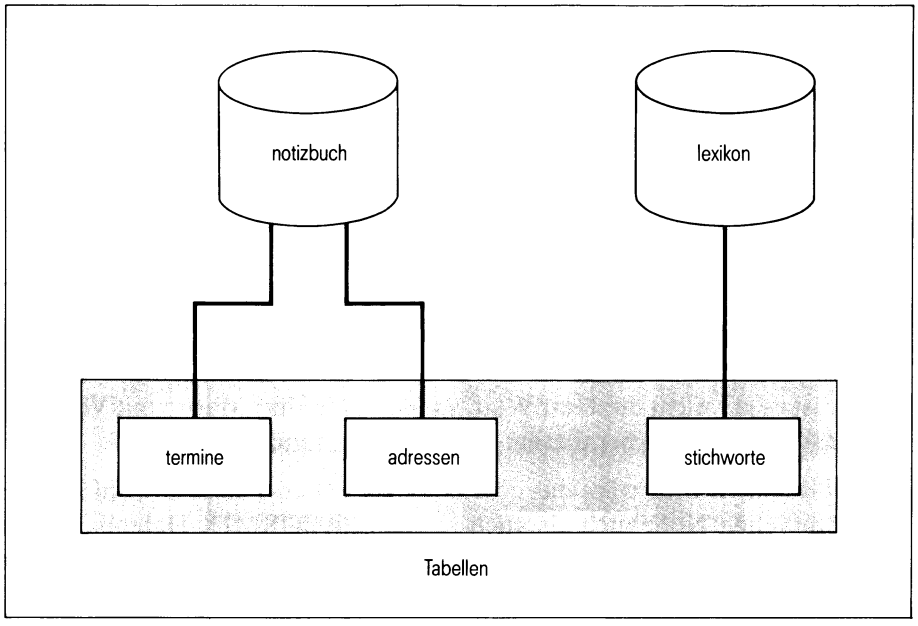


Bild 1-3 Tabellen der Datenbanken

Jede Datenbank enthält mindestens eine Tabelle; sie kann aber ebensoviel Tabellen enthalten, wie Sie für das entsprechende Sachgebiet an Informationseinheiten benötigen. Je eine Tabelle enthält Daten je einer Informationseinheit.

1.1.3 Struktur der Tabelle

Betrachten Sie bitte das folgende Daten-Puzzle:

```
Wolf   Ulm
Waldweg 12
0731/23456
7900 Martin
```

Das Puzzle ein bißchen hin und her gewürfelt, ergibt die folgende Adresse eines Herrn Wolf aus Ulm:

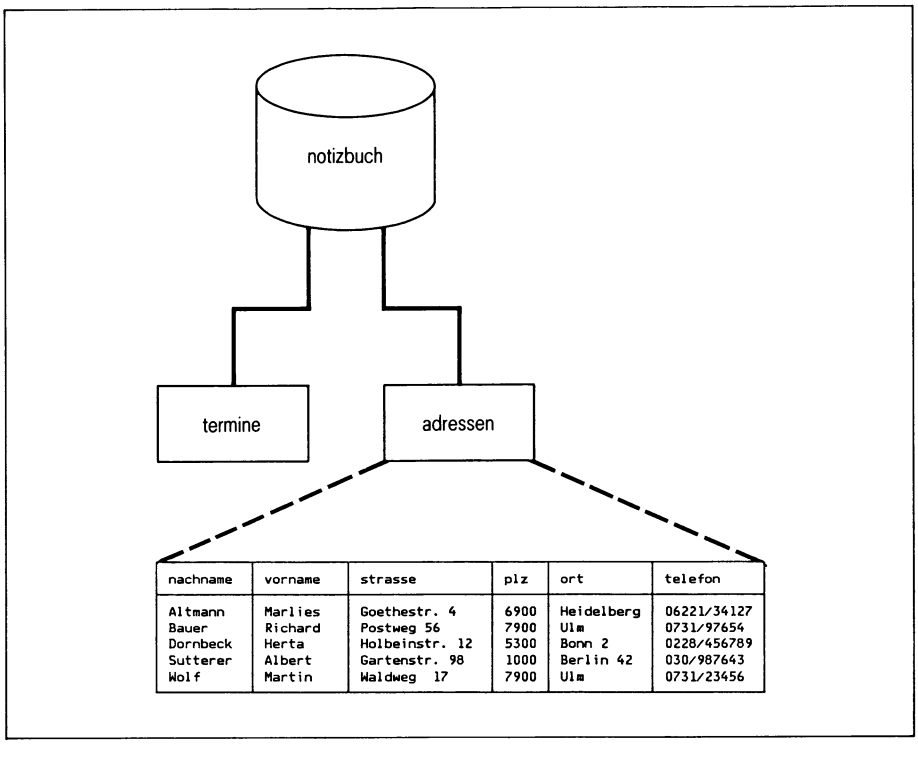
Wolf	Martin	Waldweg 12	7900 Ulm	0731/23456

Oder ist es gar nicht der Herr Wolf, sondern der Herr Martin mit Vornamen Wolf? Das läßt sich bei diesem Informationsstand nicht klären.

Aus diesem Grunde wäre das oben gezeigte Daten-Puzzle sowohl in einem gewöhnlichen Notizbuch als auch in einer INFORMIX-Datenbank wertlos. Offensichtlich müssen die Daten in einer eindeutig zuordnenden Struktur vorliegen. Man benötigt eine Datenstruktur.

Erst die Datenstruktur macht aus einer Anhäufung von Daten verwertbare Daten.

Durch ihren spezifischen Aufbau strukturiert die Tabelle die Daten. Sehen Sie dazu einen Auszug aus der Tabelle 'adressen' der Datenbank 'notizbuch':



nachname	vorname	strasse	plz	ort	telefon
Altmann	Marlies	Goethestr. 4	6900	Heidelberg	06221/34127
Bauer	Richard	Postweg 56	7900	Ulm	0731/97654
Dornbeck	Herta	Holbeinstr. 12	5300	Bonn 2	0228/456789
Sutterer	Albert	Gartenstr. 98	1000	Berlin 42	030/987643
Wolf	Martin	Waldweg 17	7900	Ulm	0731/23456

Bild 1-4 Tabelle 'adressen'

In der Tabelle finden Sie die Daten des oben gezeigten Puzzles wieder. Aber nun ist ganz offensichtlich, daß der Herr aus Ulm mit Nachnamen Wolf und mit Vornamen Martin heißt und nicht umgekehrt.
Daß sich das so eindeutig sagen läßt, liegt an der Struktur der Tabelle: Wie jede bekannte Tabelle aus Büchern und Zeitungen ist auch die Tabelle einer INFORMIX-Datenbank in Spalten und Zeilen unterteilt. Außerdem sind die Spalten mit Überschriften versehen, die die eindeutige Zuordnung der darunterliegenden Daten ermöglichen.

Die einzelnen Teile einer Tabelle einer Datenbank bezeichnet man wie folgt:

Eine Spalte bezeichnet man auch bei einer Tabelle einer Datenbank als 'Spalte'.
Eine Spaltenüberschrift bezeichnet man als 'Spaltennamen'.
Die in einer Spalte enthaltenen Daten bezeichnet man als 'Werte' oder 'Spalteninhalte'.

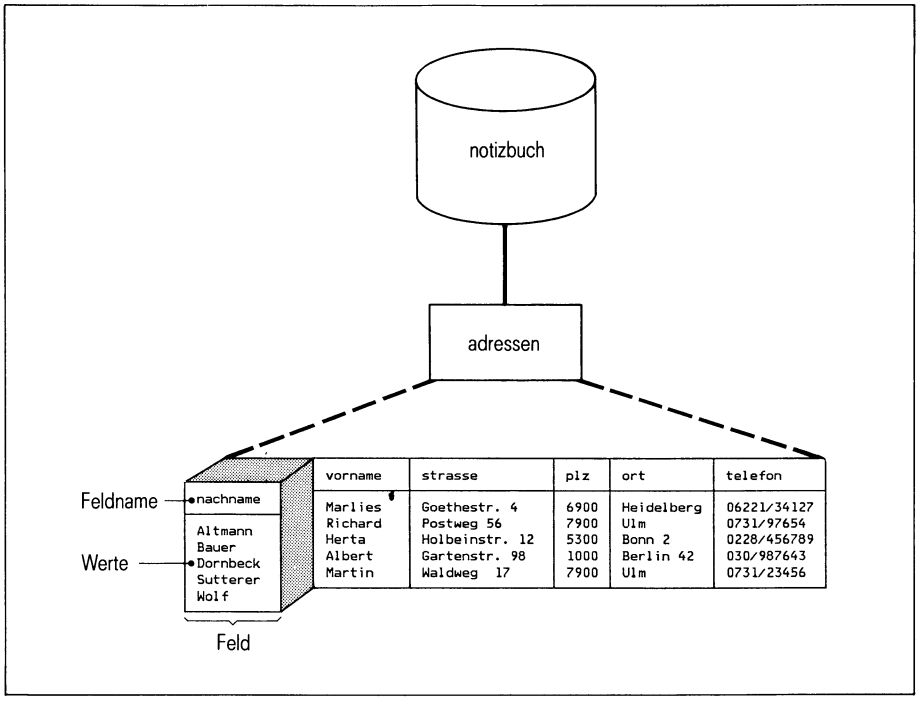


Bild 1-5 Eine Spalte einer Tabelle

Die in Bild 1-5 hervorgehobene Spalte trägt den Spaltennamen 'nachname'. Die Werte dieser Spalte sind:

```
Altmann   Bauer   Dornbeck   Sutterer   Wolf
```

Die Spalte 'nachname' beinhaltet sämtliche Nachnamen von allen Personen, deren Anschriften in der Tabelle 'adressen' enthalten sind.

Eine Zeile einer Tabelle bezeichnet man als 'Satz' oder 'Datensatz'.

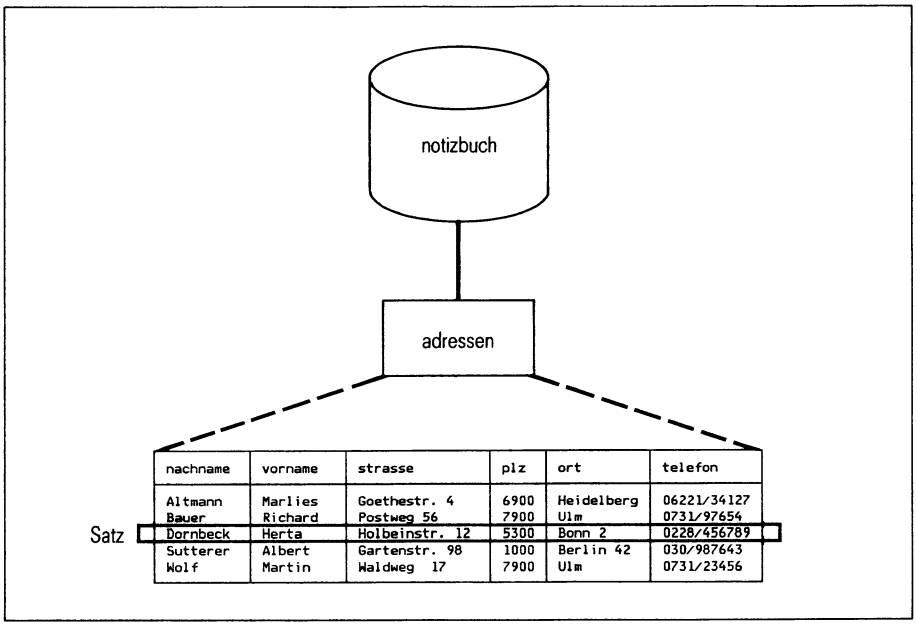


Bild 1-6 Ein Satz einer Tabelle

Der in Bild 1-6 hervorgehobene Satz beinhaltet die Anschrift von Frau Dornbeck.

1.2 Wofür Sie INFORMIX verwenden können

Mit INFORMIX können Sie

- eine Datenbank erzeugen,
- die dafür benötigten Tabellen erzeugen,
- neue Sätze in die Datenbank aufnehmen,
- Sätze löschen,
- Sätze korrigieren,
- die Datenbank abfragen,
- Berechnungen durchführen lassen,
- Listen (z.B. Etiketten) ausdrucken lassen.

Um diese Tätigkeiten ausführen zu können, stellt Ihnen INFORMIX die folgenden Werkzeuge zur Verfügung:

- Werkzeuge zur Erzeugung und Anwendung von Formaten (Bildschirmmasken),
- SQL, eine Datenbanksprache,
- Werkzeuge zur Erzeugung und Anwendung von Listen-Programmen.

Der vorliegende Abschnitt gibt Ihnen einen Einblick in Bedeutung und Gebrauch der Werkzeuge, ausführliche Informationen erhalten Sie in den darauffolgenden Kapiteln.

Bei Ihrer Arbeit unterstützt Sie ein umfangreiches Menüsystem. So können Sie z.B. menügeführt eine Datenbank und die dafür benötigten Tabellen erzeugen.

1.2.1 Formate

Mit INFORMIX können Sie Ihre eigenen Formate (Bildschirmmasken) erzeugen und diese Formate zum Neuaufnehmen, Korrigieren und Löschen von Sätzen, sowie für Datenbank-Abfragen verwenden.

Ein Format entspricht in etwa einem Papier-Formular; allerdings einem Formular, das Sie am Bildschirm angezeigt bekommen und das Sie über die Tastatur ausfüllen und bedienen können.

Zur Erläuterung ein Blick auf ein übliches Papier-Formular:

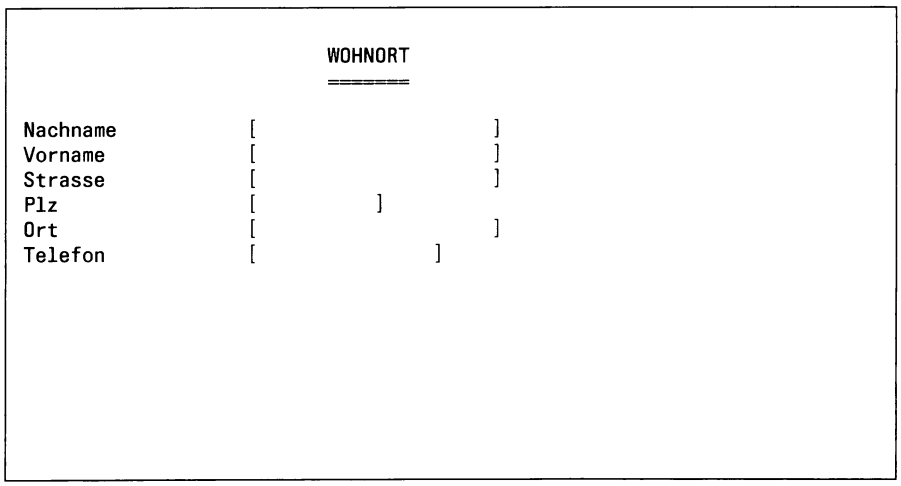


Bild 1-7 Papier-Formular

Das oben gezeigte Papier-Formular enthält einige Schreibfelder [ ]. Schreibfelder, in die Sie alle wesentlichen Daten (einen Satz) für den Wohnort einer Person eintragen können. Sofern Sie die Einträge mit einem Bleistift vornehmen, können Sie den Satz nachträglich korrigieren oder löschen.

Ein Format ist einem Papier-Formular sowohl äußerlich als auch in seiner Bedeutung ähnlich:
Auch ein Format verfügt über derartige Abschnitte [ ]; man nennt sie 'Bildschirmfelder'. Sie dienen den gleichen Zwecken (und weiteren) wie die Schreibfelder in einem Papier-Formular.

So sieht ein INFORMIX-Format aus:

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: adressen Tabelle**

                          WOHNORT
                          =======

Nachname          [                    ]
Vorname           [                    ]
Strasse           [                    ]
Plz               [          ]
Ort               [                    ]
Telefon           [               ]
```

Bild 1-8 Ein Format

Am oberen Bildschirmrand befindet sich ein Menü mit den ausführbaren Funktionen. Sie brauchen sich im Augenblick nicht weiter mit dem Menü zu beschäftigen; die Kapitel 3 und 5 gehen ausführlich auf das Menü ein.

Darunter befinden sich die 'Bildschirmfelder' [ ], die den Schreibfeldern in einem Papier-Formular entsprechen.

Wozu dient ein Format?

Mit einem Format können Sie

- einen Satz in die Datenbank neu aufnehmen,
- einen Satz korrigieren,
- einen Satz löschen.

Darüberhinaus können Sie mit einem Format

- die Datenbank abfragen und sich
- die gefundenen Sätze nacheinander in dem Format anzeigen lassen.

Wenn Sie z.B. eine Datenbank-Abfrage durchführen wollen, weil Sie die Daten von Herrn Wolf wissen möchten, dann können Sie in das Bildschirmfeld 'nachname' den Nachnamen Wolf eintragen:

```
SUCHEN:  START sucht.  F17 loescht alle Felder.  HELP fuer Hilfe.
         DEL bricht ab.                          ** 1: adressen Tabelle**

                          WOHNORT
                          =======

Nachname           [Wolf                ]
Vorname            [                    ]
Strasse            [                    ]
Plz                [          ]
Ort                [                    ]
Telefon            [               ]
```

Bild 1-9 Eintrag in ein Bildschirmfeld

Wenn Sie anschließend die Datenbank-Abfrage ausführen, dann bekommen Sie in den Bildschirmfeldern den kompletten Satz von Herrn Wolf angezeigt:

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: adressen Tabelle**

                          WOHNORT
                          =======

Nachname          [Wolf                ]
Vorname           [Martin              ]
Strasse           [Waldweg 17          ]
Plz               [7900      ]
Ort               [Ulm                 ]
Telefon           [0731/23456     ]
```

Bild 1-10 Anzeige des ermittelten Satzes

Ein Format ist eine Bildschirmmaske. Ein Format enthält Bildschirmfelder, die der Suche, Anzeige, Neuaufnahme, Korrektur und dem Löschen von Daten dienen.

1.2.2 Formate erzeugen

Wenn Sie eine Datenbank neu erzeugt haben, dann steht Ihnen zunächst kein Format zur Verfügung.
Ein Format muß erzeugt werden. Dabei können Sie sich von INFORMIX ein sogenanntes 'Standard-Format' generieren lassen, mit dem Sie die Grundfunktionen - Datenbank abfragen, Sätze neu aufnehmen, korrigieren oder löschen - ausführen können.

1.2.3 SQL, die Datenbanksprache

Ein weiteres Werkzeug von INFORMIX ist SQL, die Datenbanksprache; SQL steht für Structured Query Language.
SQL wurde aus dem weltweiten SQL-Standard weiterentwickelt.

SQL besteht aus einer Reihe von Anweisungen, die an die englische Sprache angelehnt sind.

Mit SQL können Sie unter anderem
- **eine Datenbank abfragen,**
- **eine Datenbank und Tabellen erzeugen,**
- **Sätze neu aufnehmen, ändern oder löschen.**

Auf SQL wird an entsprechender Stelle ausführlich eingegangen. Das folgende Beispiel soll Ihnen einen ersten Eindruck von SQL zu vermitteln:

```
SELECT
   nachname, vorname, strasse, plz, ort, telefon
FROM
   adressen
WHERE
   ort = "Ulm"
```

Sinngemäß bedeutet die Anweisung:

```
ZEIGE DIE WERTE DER SPALTEN
   nachname, vorname, strasse, plz, ort, telefon
AUS DER TABELLE
   adressen
UND DAVON NUR DIEJENIGEN SÄTZE, FÜR DIE GILT
   ort = "Ulm"
```

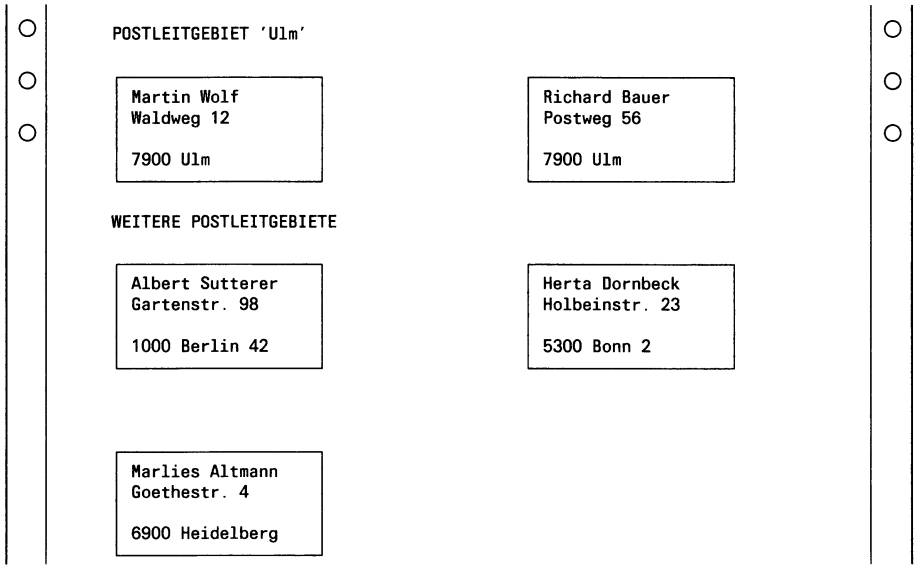
Die Anweisung führt zu einer Ergebnis-Tabelle dieser Art:

nachname	vorname	strasse	plz	ort	telefon
Bauer	Richard	Postweg 56	7900	Ulm	0731/97654
Wolf	Martin	Waldweg 12	7900	Ulm	0731/23456

1.2.4 Listen

Mit Listen-Programmen können Sie Datenbank-Abfragen durchführen und die ermittelten Daten aufbereitet ausdrucken lassen.

Wenn Sie z.B. Klebeetiketten ausdrucken lassen wollen, dann würden Sie dazu ein Listen-Programm verwenden. Ein Listen-Programm, das ein Ergebnis dieser Art liefert:



Um den im Beispiel gezeigten Etikettenausdruck ausführen zu können, muß ein Listen-Programm drei grundlegende Aufgaben erfüllen können:

- Das Listen-Programm muß eine Datenbank-Abfrage durchführen.
  In der Datenbank-Abfrage muß festgelegt sein, welche Informationen der Datenbank (Sätze, Spalten) in der Liste ausgegeben werden sollen.
  Im oberen Beispiel sind das alle Spalten und Sätze der Tabelle 'adressen', sortiert nach Postleitgebieten.
- Das Listen-Programm muß Layoutanweisungen enthalten.
  Das heißt, es muß in ihm festgelegt sein, an welcher Stelle auf dem Papier die Informationen ausgegeben werden sollen.
  Im oberen Beispiel muß festgelegt sein, daß je zwei Datensätze nebeneinander ausgegeben werden sollen, entsprechend zwei Etiketten nebeneinander.
- Das Listen-Programm muß die Ausgabe von zusätzlichen Texten ermöglichen; die Ausgabe von Texten, die ergänzende Informationen liefern.
  Im oberen Beispiel sind es die Texte:
  POSTLEITGEBIET 'Ulm' und WETERE POSTLEITGEBIETE, die die spätere Weiterverarbeitung der Etiketten erleichtern sollen.

Darüber hinaus können Sie von einem Listen-Programm auch Berechnungen durchführen und somit z.B. auch Angebote und Rechnungen erstellen und ausdrucken lassen.

1.2.5 Listen-Programm erzeugen

Wenn Sie eine Datenbank neu erzeugt haben, dann steht Ihnen zunächst kein Listen-Programm zur Verfügung.
Ein Listen-Programm muß erzeugt werden. Dabei können Sie sich von INFORMIX ein sogenanntes 'Standard-Listen-Programm' generieren lassen.
Allerdings bietet ein Standard-Listen-Programm nur sehr eingeschränkte Möglichkeiten; um eine Liste mit einem speziell von Ihnen erwünschten Ausdruck zu erhalten, müssen Sie das Standard-Listen-Programm entsprechend umgestalten.

1.3 Eine Datenbank abfragen

Dieser Abschnitt erläutert Ihnen einige Prinzipien von Datenbank-Abfragen. Sie erfahren, nach welchen Gesichtspunkten Sie eine Datenbank-Abfrage durchführen können und welche Ergebnisse eine Datenbank-Abfrage liefern kann.

Hinweis

Sie erfahren in diesem Abschnitt nicht, welche konkreten Schritte Sie unternehmen müssen, um Datenbank-Abfragen mit SQL oder einem Format durchzuführen. Darauf gehen die Kapitel 3 ff ausführlich ein.

1.3.1 Prinzipien einer Datenbank-Abfrage

Eine Datenbank-Abfrage führt zu einer temporären Ergebnistabelle. Diese enthält die gefundenen Treffersätze.

Welche Treffersätze die Ergebnistabelle enthält, hängt davon ab, wie Sie die Datenbank-Abfrage gestaltet haben.
Sie können eine Datenbank-Abfrage prinzipiell z.B. so gestalten, daß sie alle Sätze einer Tabelle liefert. Oder Sie können eine Datenbank-Abfrage so gestalten, daß sie nur einige ausgewählte Sätze einer Tabelle liefert, z.B. nur diejenigen Sätze mit dem Wohnort 'Ulm'. Oder Sie können eine Datenbank-Abfrage so gestalten, daß Sie nur die Werte einer oder mehrerer Spalten liefert, z.B. nur die Werte der Spalte 'nachname'.

Den Teil einer Datenbank-Abfrage, der Sätze liefert, bezeichnet man als 'Satzauswahl'.
Den Teil einer Datenbank-Abfrage, der Spalten liefert, bezeichnet man als 'Spaltenauswahl'.

1.3.2 Satzauswahl

Die Satzauswahl in einer Datenbank-Abfrage liefert Sätze. Die Satzauswahl kann einen, mehrere oder auch alle Sätze der Tabelle liefern.

Beispielsweise können Sie die Satzauswahl in einer Datenbank-Abfrage so gestalten, daß sie nur die Sätze derjenigen Personen liefert, die in 'Ulm' wohnen:

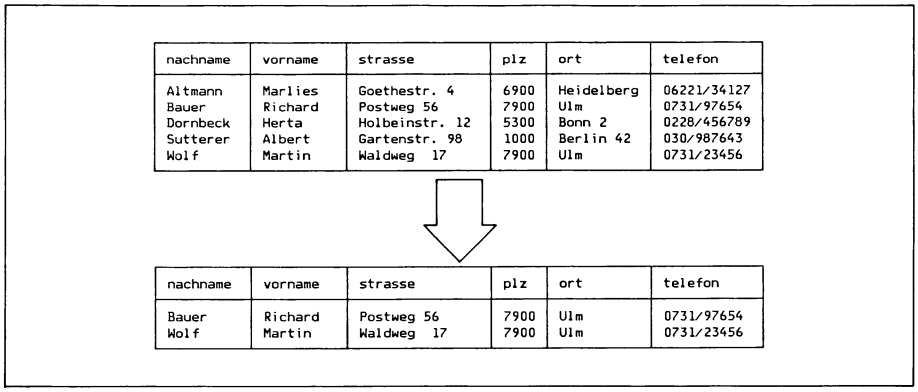


Bild 1-11 Satzauswahl

1.3.3 Spaltenauswahl

Die Spaltenauswahl in einer Datenbank-Abfrage liefert Spalten. Die Spaltenauswahl kann eine Spalte, mehrere Spalten oder auch alle Spalten der Tabelle liefern.

Beispielsweise können Sie die Spaltenauswahl in einer Datenbank-Abfrage so gestalten, daß sie nur Nachnamen liefert:

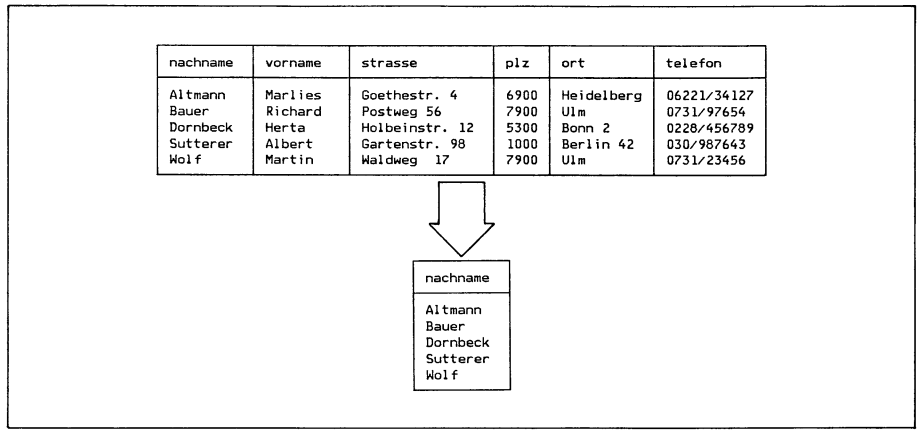


Bild 1-12 Spaltenauswahl

1.3.4 Spaltenauswahl und Satzauswahl

Sie können Spaltenauswahl und Satzauswahl in einer Datenbank-Abfrage auch dergestalt miteinander kombinieren, daß Sie im Ergebnis eine genau bestimmte Datenmenge erhalten.

Beispielsweise können Sie eine Datenbank-Abfrage so gestalten, daß sie nur Nachnamen, Vornamen und den Ort der Personen liefert, die in 'Ulm' wohnen:

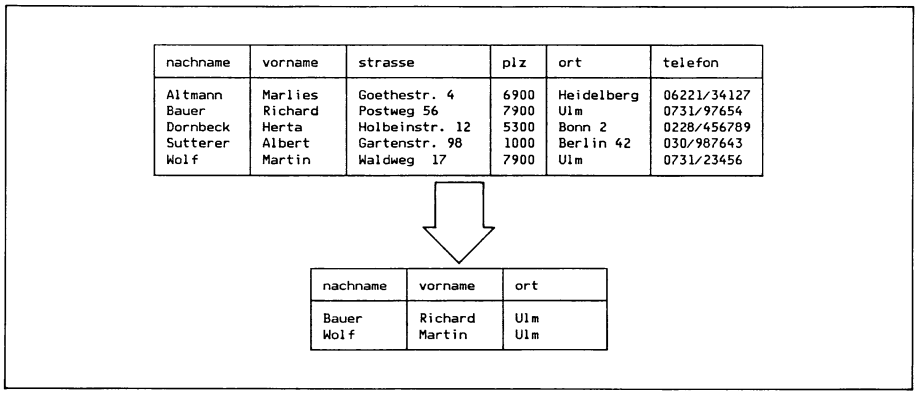


Bild 1-13 Spaltenauswahl und Satzauswahl

1.3.5 Tabellen miteinander verbinden

Dieser Abschnitt erläutert Ihnen das Prinzip einer Technik, mit der Sie Tabellen miteinander verbinden können. Über die dazu erforderlichen praktischen Maßnahmen informieren Sie die Kapitel 5 ff.

Eine Datenbank enthält sehr oft mehr als eine Tabelle. Nämlich ebensoviel Tabellen, wie Sie an Informationseinheiten für die speziellen Zwecke benötigen.
Um eine Datenbank-Abfrage auch über mehrere Tabellen durchführen zu können, gibt es eine Tabellen-Verbindungstechnik, die 'Join' heißt.

Zur Erläuterung ein Blick auf die Tabellen 'adressen' und 'termine' der Datenbank 'notizbuch':

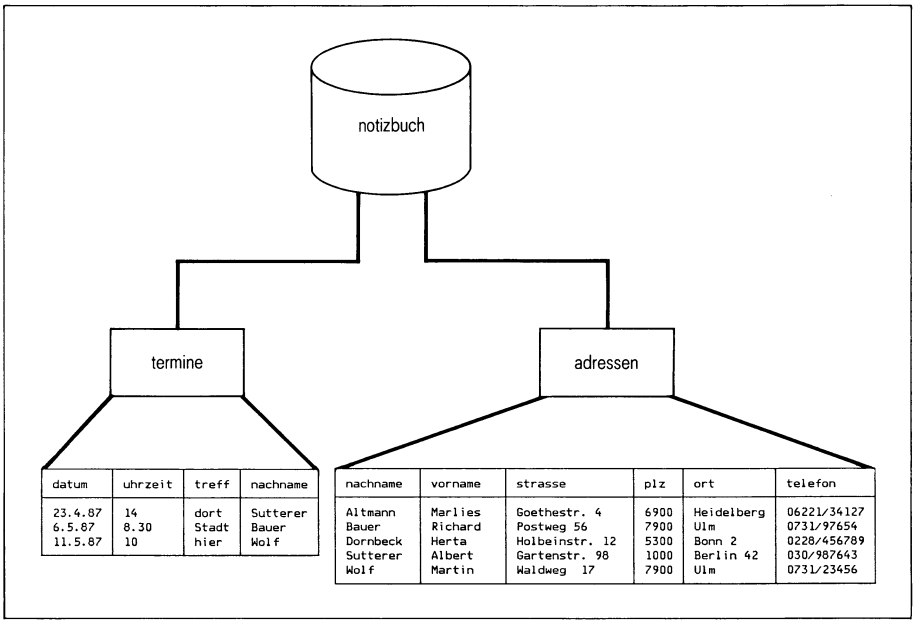


Bild 1-14 Tabellen 'termine' und 'adressen'

Gelegentlich benötigt man Informationen aus mehreren Tabellen zugleich, zum Beispiel im folgenden Fall:
Der Termin am 6.5. muß abgesagt werden; der Gesprächspartner ist davon zu benachrichtigen. Wie lautet seine Telefonnummer?

In einem normalen Notizbuch gehen Sie wie folgt vor:

- Im Terminteil des Notizbuches informieren Sie sich darüber, mit wem Sie am 6.5. einen Termin haben. In diesem Beispiel ist es Herr Bauer.
- Im Adressenteil des Notizbuches informieren sie sich darüber, welche Telefonnummer Herr Bauer hat.

Bei einer INFORMIX-Datenbank 'notizbuch' gehen Sie wie folgt vor: Sie führen eine Datenbank-Abfrage durch, bei der Sie die Tabellen 'termine' und 'adressen' vorübergehend miteinander verbinden und gemeinsam abfragen.

Die Verbindung zwischen den Tabellen stellen Sie über geeignete Tabellenspalten her, die man Join-Spalten nennt.
In der gezeigten Datenbank 'notizbuch' gibt es in der Tabelle 'termine' eine Spalte 'nachname' und in der Tabelle 'adressen' gibt es ebenfalls eine Spalte 'nachname'. Diese Spalten 'nachname' sind als Join-Spalten geeignet; die Spalten 'nachname' der Tabelle 'termine' und 'nachname' der Tabelle 'adressen' enthalten übereinstimmende Werte: In der Tabelle 'termine' gibt es den Herrn Bauer und in der Tabelle 'adressen' gibt es den Herrn Bauer ebenfalls.

Ähnlich wie Sie in einem gewöhnlichen Notizbuch mit dem Nachnamen ermitteln können, ob und wann Sie mit dem entsprechenden Partner einen Termin haben, bzw. wo der Gesprächspartner wohnt, können Sie in diesem Beispiel auch über die jeweiligen Spalten 'nachname' der Tabellen 'termine' und 'adressen' alle verbundenen Sätze ermitteln.

Eine Datenbank-Abfrage, in der die beiden Tabellen 'termine' und 'adressen' über die Join-Spalten 'nachname' miteinander verbunden sind, führt zu diesem Ergebnis:

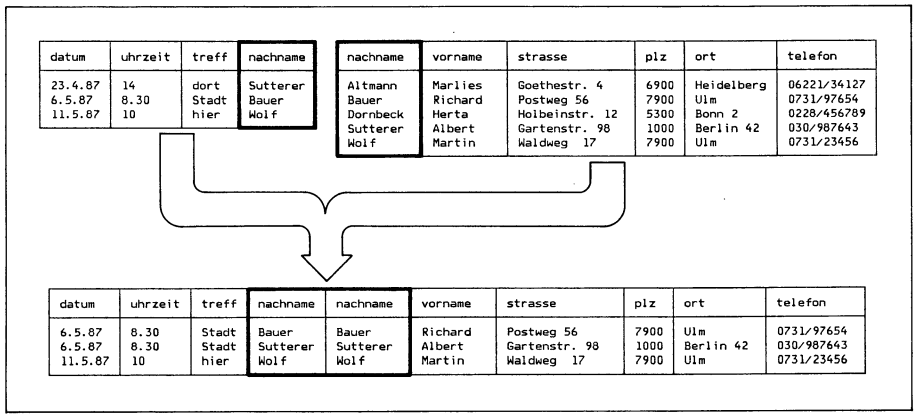


Bild 1-15 Tabellen über Join-Spalten verbinden

Bei der Verbindung der beiden Tabellen ist eine neue, temporäre Ergebnis-Tabelle entstanden. In dieser Tabelle ist nun die erwünschte Information, nämlich die Telefonnummer von Herrn Bauer, abzulesen.

Allerdings zeigt die Ergebnis-Tabelle mehr als für diesen speziellen Fall erwünscht. Nämlich alle Sätze, die sich über die Join-Spalten 'nachname' verbinden lassen.
Um das zu verhindern, können Sie die Satzauswahl und die Spaltenauswahl in der Datenbank-Abfrage so gestalten, daß die Ergebnistabelle ausschließlich die benötigte Information anzeigt:

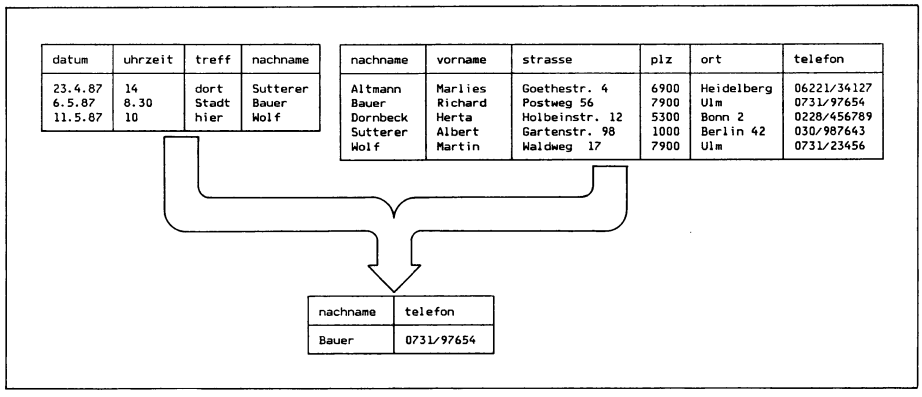


Bild 1-16 Join, Satzauswahl und Spaltenauswahl

Tabellen lassen sich über geeignete Spalten miteinander verbinden. Man nennt diese Spalten 'Join-Spalten'. Beim Aufbau einer Datenbank sind die Join-Spalten von vorneherein zu berücksichtigen.
Eine Datenbank-Abfrage über mehrere Tabellen mittels geeigneter Join-Spalten liefert ebensoviele Sätze, wie sich über die Join-Spalten verbinden lassen. Bei entsprechender Gestaltung von Spaltenauswahl und Satzauswahl liefert eine Datenbank-Abfrage über mehrere Tabellen genau die erwünschte Datenmenge.

1.3.6 Spalten indizieren

Jedes Lexikon ist alphabetisch sortiert. Das alleine ermöglicht, daß Sie eine gesuchte Information schnell und sicher auffinden können.
Aber die Sätze einer Tabelle sind keineswegs alphabetisch sortiert, sie stehen schlichtweg genau in der Reihenfolge in der Tabelle, wie sie nacheinander eingegeben wurden. Die Folge davon ist, daß Datenbank-Abfragen bei umfangreichen Datenbanken u.U. viel Zeit benötigen.
Bedenken Sie bitte: Wie lange würden Sie in Ihrem Notizbuch nach einer bestimmten Adresse suchen, wenn der Adressenteil nicht alphabetisch sortiert wäre?

Um den Zeitaufwand bei einer Datenbank-Abfrage zu reduzieren, können Sie ein oder auch mehrere Spalten 'indizieren'. Das Indizieren bewirkt einige Maßnahmen, die den Datenzugriff beschleunigen. Sie können sich das Ergebnis dieser Maßnahmen ähnlich vorstellen wie die alphabetische Sortierung in einem Lexikon.

Für Join-Spalten gilt: Sie sind grundsätzlich zu indizieren. Denn beim Verbinden von Tabellen sucht INFORMIX über die Join-Spalten die zu verbindenden Sätze.

Beim Einrichten eines Index muß man festlegen, ob der Index 'unique' oder 'duplikate' sein soll. Duplikate bedeutet, daß in der entsprechenden Spalte ein Wert mehrfach vorkommen darf. Unique bedeutet, daß ein Wert nur einmal aufgenommen werden darf.
Beispielsweise sollte ein Index auf die Spalte 'nachname' 'duplikate' sein, sonst wäre es nicht mehr möglich, einen zweiten Satz mit dem Namen 'Bauer' aufzunehmen. INFORMIX würde die Neuaufnahme dieses Satzes abweisen.

Mit dem Unique-Index hingegen kann man erreichen, daß ein Satz nicht versehentlich mit genau den gleichen Werten ein zweites mal aufgenommen wird. Der Unique-Index sorgt damit für die Eindeutigkeit der Sätze. Und das entspricht einem wichtigen Grundsatz bei einer INFORMIX-Datenbank: Jeder Satz einer Tabelle muß sich eindeutig von jedem weiteren Satz der gleichen Tabelle unterscheiden.

Das Indizieren einer Spalte beschleunigt immer dann den Datenzugriff, wenn die Sätze einer Tabelle über die entsprechende Spalte gesucht werden (z.B. ort = "Ulm).
Join-Spalten sind grundsätzlich zu indizieren.
Ein Unique-Index sorgt für die Eindeutigkeit der Sätze in einer Tabelle.

Zusammenfassung

- Mit INFORMIX können Sie mehrere Datenbanken führen.
- Eine Datenbank enthält mindestens eine Tabelle. Sie kann aber ebensoviel Tabellen enthalten, wie Sie an Informationseinheiten benötigen.
- Eine Tabelle besteht aus Spalten und Sätzen.
- Spalten tragen Namen, die Spaltennamen. Die in einer Spalte enthaltenen Daten heißen Werte oder Spalteninhalte.
- Formate sind Bildschirmmasken, mit denen Sie die Datenbank abfragen, Sätze eingeben, korrigieren und löschen können.
- SQL ist eine Datenbanksprache, mit der Sie u.a. die Datenbank abfragen können.
- Listen-Programme führen Datenbank-Abfragen durch und bereiten das Abfrageergebnis für den Ausdruck auf.
- Eine neue Datenbank enthält noch keine Formate und Listen-Programme. Sie können sich von INFORMIX Standard-Formate und Standard-Listen-Programme erzeugen lassen.
- Den Teil einer Datenbank-Abfrage, der die gewünschten Sätze liefert, nennt man Satzauswahl.
- Den Teil einer Datenbank-Abfrage, der die gewünschten Spalten liefert, nennt man Spaltenauswahl.
- Um Tabellen miteinander verbinden zu können, benötigen Sie geeignete Spalten, Join-Spalten.
- Um Datenbank-Abfragen zu beschleunigen und um die Eindeutigkeit von Sätzen zu garantieren, können Sie Spalten indizieren.

2 INFORMIX aufrufen und bedienen

Dieses Kapitel informiert Sie darüber,

- wie Sie INFORMIX aufrufen und
- wie Sie das INFORMIX-Menüsystem bedienen.

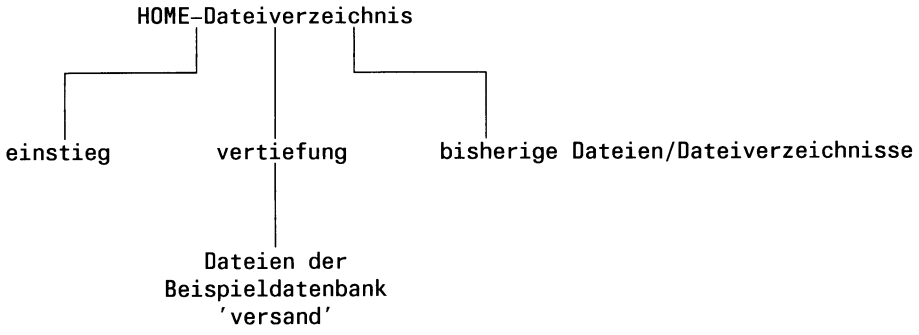
2.1 INFORMIX aufrufen

2.1.1 Die Diskette INFDEX

Im Lieferumfang von INFORMIX ist eine Diskette mit dem Namen INFDEX enthalten. Damit Sie die Erläuterungen in diesem und den folgenden Kapiteln erfolgreich nachvollziehen können, muß der Systemverwalter die Diskette INFDEX über die Funktion 'Installation von Softwareprodukten' einlesen.

Im Rahmen der Installationsprozedur muß der Systemverwalter angeben, welche Benutzerkennung als INFORMIX-Übungskennung zur Verfügung stehen soll. Bei der angegebenen Benutzerkennung werden automatisch im HOME-Dateiverzeichnis (das Dateiverzeichnis, in dem Sie sich nach dem Anmelden befinden) zwei neue untergeordnete Dateiverzeichnisse eingerichtet: 'einstieg' und 'vertiefung'
Das Dateiverzeichnis 'einstieg' dient Ihren ersten Gehversuchen mit INFORMIX, wie sie in Kapitel 3 'Eine Datenbank aufbauen' beschrieben sind. Das Dateiverzeichnis 'vertiefung' enthält eine fertige Beispieldatenbank mit dem Namen 'versand'. Diese Beispieldatenbank benötigen Sie, um die Erläuterungen ab Kapitel 4 nachvollziehen zu können.

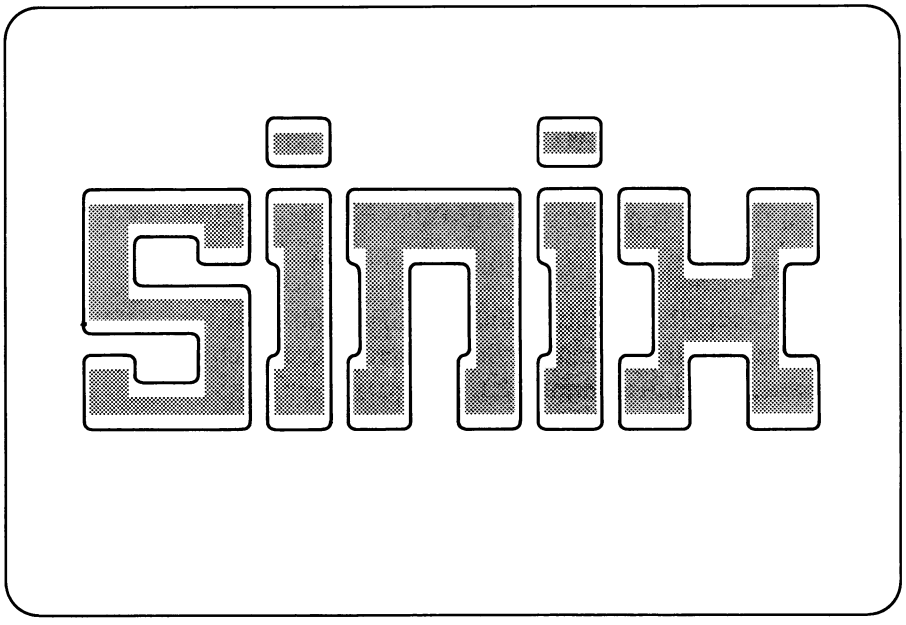
Wenn Ihre Benutzerkennung im Rahmen der Installationsprozedur als INFORMIX-Übungskennung bekanntgegeben wurde, dann ergibt sich für Ihr HOME-Dateiverzeichnis folgende Struktur:



2.1.2 Zugangsweg zu INFORMIX

Anmelden

Wenn Sie den SINIX-Begrüßungsbildschirm vor sich haben:



▷ Geben Sie Ihre Benutzerkennung ein.

▷ Drücken Sie ↵

▷ Geben Sie Ihr Kennwort ein.

▷ Drücken Sie ↵

Sie haben jetzt Ihr Eingabe-Aufforderungszeichen, standardmäßig den '$', erhalten und befinden sich in Ihrem HOME-Dateiverzeichnis.

INFORMIX aufrufen

Für Ihre ersten Gehversuche mit INFORMIX wurde bei der Installation der Diskette INFDEX automatisch das untergeordnete Dateiverzeichnis 'einstieg' erzeugt. Bevor Sie INFORMIX aufrufen, wechseln Sie zunächst mit dem 'cd' Kommando in das Dateiverzeichnis 'einstieg':

▷ Geben Sie ein: cd einstieg

▷ Drücken Sie ↵

Jetzt können Sie INFORMIX aufrufen. Der Programmaufruf für INFORMIX lautet 'isql'.

▷ Geben Sie ein: isql

▷ Drücken Sie ↵

Jetzt meldet sich INFORMIX zunächst mit einem Informationsbildschirm. Nach kurzer Wartezeit erhalten Sie danach automatisch das INFORMIX-Hauptmenü (siehe Abschnitt 2.2.).

2.2 Das INFORMIX-Menüsystem

2.2.1 Das Hauptmenü

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Ablauf, Modifizieren, Neuerstellen oder Loeschen eines Formates.
```

Jedesmal nachdem Sie INFORMIX aufgerufen haben, erhalten Sie zunächst einmal diesen Bildschirm mit dem INFORMIX-Hauptmenü.
Dieser Abschnitt stellt Ihnen zunächst einmal die einzelnen Funktionen des Hauptmenüs vor. Führen Sie an dieser Stelle bitte noch keine Bedienungsvorgänge durch, da Sie sich sonst im INFORMIX-Menüsystem eventuell nicht mehr zurechtfinden.
Weiter unten im Abschnitt 2.2.3 gibt es eine Übung, in der Sie sich mit der Bedienung des INFORMIX-Menüsystems vertraut machen können. Im Anschluß an die Übung finden Sie "allgemeine Bedienungshinweise", denen Sie grundlegende Informationen über die Bedienung des INFORMIX-Menüsystems entnehmen können.

Hier zunächst die Beschreibung des INFORMIX-Hauptmenüs. Das Hauptmenü bietet Ihnen verschiedene ausführbare Funktionen an; jede Funktion führt zu einem Folgemenü mit weiteren Auswahlmöglichkeiten. Das INFORMIX-Hauptmenü bietet mehr Funktionen an, als eine Zeile anzeigen kann. Erkennbar ist dies an der Pseudofunktion '...'. Wenn Sie zur nächsten Zeile blättern (wie das geht, erfahren Sie weiter unten), dann erhalten Sie die Fortsetzung des Menüs:

```
INFORMIX-SQL:  ...  Tabelle  END
Neuerstellen, Modifizieren, oder Loeschen einer Datenbanktabelle.
```

Die Funktionen des INFORMIX-Hauptmenüs:

Format

diese Funktion führt zu weiteren Menüs, in denen Sie folgende Tätigkeiten durchführen können:

- ein bestehendes Format aufrufen und mit ihm arbeiten,
- ein bestehendes Format (Format-Programm) modifizieren,
- ein neues Standard-Format generieren,
- ein neues Format (Format-Programm) erzeugen,
- ein Format-Programm compilieren (übersetzen),
- ein bestehendes Format löschen.

Liste

diese Funktion führt zu weiteren Menüs, in denen Sie folgende Tätigkeiten durchführen können:

- eine bestehende Liste ablaufen lassen,
- ein bestehendes Listen-Programm modifizieren,
- ein neues Standard-Listen-Programm generieren,
- ein neues Listen-Programm erzeugen,
- ein Listen-Programm compilieren (übersetzen),
- ein bestehendes Listen-Programm löschen.

SQL-Dialog

Diese Funktion führt zu SQL, der Datenbanksprache.

Benutzer-Menue

Diese Funktion führt zu weiteren Menüs, die es erlauben, ein benutzerspezifisches Menüsystem zu erstellen bzw. mit einem erstellten benutzerspezifischen Menüsystem zu arbeiten. Das Erstellen eines Benutzermenüs setzt gute Kenntnisse von Formaten voraus. Diese Kenntnisse erlangen Sie erst in Kapitel 5.

Auf die Funktion 'Benutzermenü' geht das vorliegende Handbuch nicht weiter ein. Siehe hierzu [2] INFORMIX-SQL Nachschlagen.

Datenbank

diese Funktion führt zu weiteren Menüs, in denen Sie folgende Tätigkeiten durchführen können:

- eine Datenbank auswählen,
- eine neue Datenbank erzeugen,
- eine Datenbank löschen.

Tabelle

diese Funktion führt zu weiteren Menüs, in denen Sie folgende Tätigkeiten durchführen können:

- eine Tabelle erzeugen,
- eine bestehende Tabelle modifizieren,
- Informationen über die Tabellenstruktur erhalten,
- eine Tabelle löschen.

END

Diese Funktion beendet INFORMIX und führt zurück in die Shell.

2.2.2 Aufbau der INFORMIX-Bildschirme

Anhand des Hauptmenüs erfahren Sie, wie ein INFORMIX-Bildschirm aufgebaut ist:

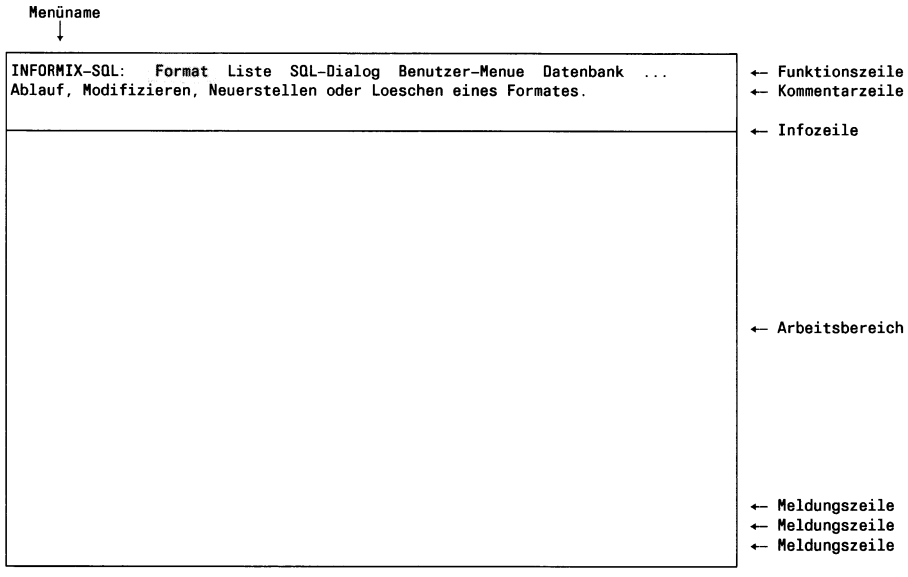


Bild 2-2 Aufbau eines INFORMIX-Bildschirmes

Ein INFORMIX-Bildschirm setzt sich aus den folgenden Teilen zusammen:

Funktionszeile

Die Funktionszeile zeigt den Menünamen (siehe unten) und die ausführbaren Funktionen an. Die jeweils ausgewählte Funktion ist markiert (invertiert) Bei jedem neuen Aufruf von INFORMIX ist automatisch die Funktion 'Format' markiert. In manchen INFORMIX-Bildschirmen erhalten Sie in der Funktionszeile ein Eingabefeld dieser Art:
AUSWAHL DATENBANK >>
Das Eingabefeld dient der Namensgebung bzw. Auswahl eines Objektes (Datenbank, Tabelle, Spalte u.a.).

Menüname

Der Menüname dient Ihrer Orientierung innerhalb des Menüsystems; er informiert Sie über Ihren jeweiligen Standort. Wenn Sie eine Funktion ausgeführt haben und damit in ein Folgemenü gelangt sind, dann beinhaltet der neue Menüname den Namen der zuvor ausgeführten Funktion. Wenn Sie z.B. die Funktion 'Format' ausgeführt haben, dann erhalten Sie ein Folgemenü mit dem Menünamen FORMAT.
Bei Bildschirmen, bei denen es um eine Auswahl geht, zum Beispiel `AUSWAHL DATENBANK` >> gibt es keinen Menünamen.
Im Hauptmenü lautet der Menüname 'INFORMIX-SQL'.

Komentarzeile

In der Kommentarzeile erhalten Sie erläuternde Hinweise zur jeweils markierten Funktion.

Infozeile

Die Infozeile trennt Funktions- und Kommentarzeile vom darunterliegenden Arbeitsbereich. In ihr gibt INFORMIX die folgenden Informationen aus:
Wenn eine Datenbank ausgewählt wurde und damit die aktuelle Datenbank ist, dann bekommen Sie den Namen der ausgewählten Datenbank auf der Infozeile angezeigt.
In bestimmten Menüs bekommen Sie links auf der Infozeile eine Ziffer angezeigt. Die Ziffer zeigt in diesen Menüs die aktuelle Bildschirmseite bzw. die aktuelle Zeile an.

Arbeitsbereich

Im Hauptmenü hat der Arbeitsbereich keine Bedeutung. In anderen Menüs dient der Arbeitsbereich u.a.:

- der Anzeige und Auswahl von Objekten (Tabelle, Datenbank),
- der Eingabe von Anweisungen (SQL-Dialog),
- der Anzeige und Benennung von Spaltennamen (Tabelle),
- der Anzeige des ausgewählten Formates (bzw. der Liste).

Auf die jeweils im Arbeitsbereich durchzuführenden Aktionen wird an entsprechender Stelle eingegangen.

Meldungszeilen

In der 23. und 24. Bildschirmzeile gibt INFORMIX Meldungen aus. Meldungen des Betriebssytems erfolgen auf der darunterliegenden, letzten Bildschirmzeile.

2.2.3 Bedienung der INFORMIX-Menüs

Anschließend erhalten Sie erstmalig eine "Übung", in der Sie sich mit der Bedienung des INFORMIX-Menüsystems vertraut machen können; Sie können die Übung unmittelbar am Bildschirm nachvollziehen.
Im Anschluß an die Übung erhalten Sie "allgemeine Bedienunghinweise", die Sie zusammenfassend über die Menübedienung informieren.

Hinweis

Die Übung will Sie lediglich in die Bedienung der INFORMIX-Menüs einführen. Es geht hier noch nicht um die Erzeugung einer Datenbank oder ähnliches.

Übung 2-1

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Ablauf, Modifizieren, Neuerstellen oder Loeschen eines Formates.
```

Die Eingangssituation ist wie folgt: Sie haben das INFORMIX-Hauptmenü vor sich und die Funktion 'Format' ist markiert.
Der Kommentar in der Kommentarzeile bezieht sich auf die Funktion 'Format'.

Zur Fortsetzung des Menüs blättern

▷ Drücken Sie [↑].

Jetzt sieht Ihr Bildschirm wie folgt aus:

```
INFORMIX-SQL:   ...  Tabelle  END
Neuerstellen, Modifizieren, oder Loeschen einer Datenbanktabelle.

_____________________________________________________________________
```

Sie haben jetzt die Fortsetzung des Hauptmenüs erhalten.

▷ Drücken Sie wieder [↑].

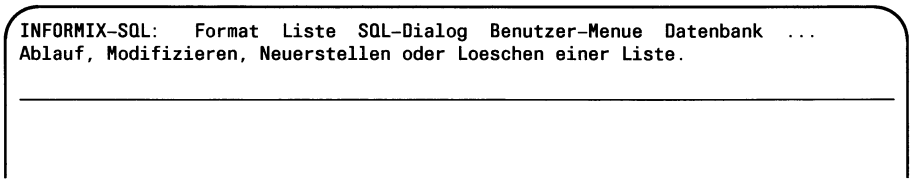
Sie sind damit wieder zum vorherigen Bildschirm zurückgekehrt.

Mit [↑] (oder [↓]) können Sie zwischen den Funktionszeilen von Menüs hin und her blättern, die die Pseudofunktion '...' enthalten.
Das gleiche Ergebnis erzielen Sie, indem Sie die Pseudofunktion '...' markieren (siehe unten).

Funktion markieren

▷ Drücken Sie die Leertaste (Leerzeichentaste).

Jetzt sieht Ihr Bildschirm wie folgt aus:



Nun ist die Funktion 'Liste' markiert und auch die Kommentarzeile hat sich verändert.

▷ Drücken Sie nun ⌫ (Korrekturtaste).

Jetzt ist die vorherige Funktion markiert.

Die Leertaste markiert die jeweils folgende und die Korrekturtaste ⌫ markiert die jeweils vorherige Funktion. Beim Wechsel ändert sich auch die Kommentarzeile.

Hinweis

Die Pfeiltasten → und ← haben die gleiche Wirkung wie Leertaste und ⌫.

Markierte Funktion ausführen

▷ Markieren Sie die Funktion 'Datenbank'.

▷ Drücken Sie ↵

Sie erhalten diesen Bildschirm:

```
DATENBANK:   Auswahl  Neu  Loeschen  END
Eine Datenbank auswaehlen.
```

An dem Menünamen 'DATENBANK' erkennen Sie, daß es sich bei diesem Menü um das Folgemenü der Funktion 'Datenbank' handelt.
Das Menü dient dazu, eine bestehende Datenbank auszuwählen, eine Datenbank neu zu erstellen oder eine Datenbank zu löschen.
In der Übung geht es jedoch zunächst einmal darum, weiter mit dem Menüsystem vertraut zu werden.

↵ führt eine markierte Funktion aus und Sie erhalten ein Folgemenü.

Zweite Möglichkeit, eine Funktion auszuführen

Eine Funktion können Sie auch ausführen, indem Sie den Anfangsbuchstaben der gewünschten Funktion eingeben:

▷ Geben Sie ein: n
Das ist der Anfangsbuchstabe der Funktion 'Neu' - Datenbank neu erstellen.

Sie erhalten diesen Bildschirm:

```
NEUE DATENBANK >>
Namen der neuen Datenbank eingeben. Weiter mit RETURN

_____________________________________________________________________
```

Wenn Sie den Anfangsbuchstaben einer Funktion eingeben, dann wird die entsprechende Funktion nicht markiert, sondern unmittelbar ausgeführt. Der erläuternde Kommentar in der Kommentarzeile entgeht Ihnen dabei.

Hinweis

Wenn Sie diese zweite Möglichkeit zur Ausführung einer Funktion nutzen, dann erhalten Sie unmittelbar einen neuen Bildschirm; das kann dazu führen, daß Sie die Orientierung im Menüsystem verlieren. Nutzen Sie diese zweite, schnelle Möglichkeit also vielleicht erst dann, wenn Sie einigermaßen mit dem INFORMIX-Menüsystem vertraut sind.

In dem gezeigten Bildschirm geht es darum, den Namen einer neu zu erzeugenden Datenbank einzugeben. Zu diesem Zweck haben Sie in der Funktionszeile nun anstatt eines Menüs ein Eingabefeld erhalten.
Bildschirme mit Eingabefeldern erhalten Sie mehrfach innerhalb des INFORMIX-Menüsystems. Und zwar jedesmal dann, wenn es darum geht, den Namen einer Datenbank (oder Tabelle oder Spalte u.a.m.) neu einzugeben oder eine bestehende Datenbank (Tabelle, Spalte u.a.m.) auszuwählen.
Wenn Sie einen Bildschirm mit einem Eingabefeld erhalten haben, dann bedeutet das, daß Sie eine Aktion, wie z.B. in diesem Fall 'Erzeugen einer neuen Datenbank', eingeleitet aber noch nicht ausgeführt haben.

Da in dieser Übung noch keine Datenbank erzeugt werden soll, die Aktion also nicht ausgeführt werden soll, geht es nun wie folgt weiter:

Eine eingeleitete Aktion abbrechen

▷ Drücken Sie [DEL]

Zwei Dinge sind geschehen:

- INFORMIX hat das Drücken von [DEL] mit einem Signalton quittiert.
- Sie sind in das vorherige Menü zurückgekehrt.

[DEL] bricht eine eingeleitete Aktion ab, ein Signalton erfolgt und Sie kehren in das vorherige Menü zurück.

Menü beenden und zum vorherigen Menü zurückkehren

Im augenblicklichen Menü wird Ihnen u.a. die Funktion END angeboten. Um dieses Menü zu beenden, bieten sich Ihnen drei Möglichkeiten an:

- entweder Sie drücken [END]
- oder Sie führen die Funktion END wie folgt aus: Sie markieren die Funktion END mit der Leertaste oder mit [⌫] und drücken [↵]
- oder Sie führen die Funktion END wie folgt aus: Sie geben 'e' als den Anfangsbuchstaben der Funktion END ein.

▷ Beenden Sie das Menü mit einer der genannten Möglichkeiten.

Sie befinden sich nun wieder im Hauptmenü. Sie erkennen das an dem Menünamen links außen in der Funktionszeile; er lautet INFORMIX-SQL

Menüs, die die Funktion END anbieten, können Sie verlassen, indem Sie [END] drücken oder indem Sie die Funktion END markieren und mit [↵] ausführen oder indem Sie 'e' eingeben.
[DEL] wirkt in diesen Menüs nicht. Wenn Sie [DEL] drücken, dann erfolgt lediglich ein Signalton.

INFORMIX beenden

Um INFORMIX beenden zu können, müssen Sie sich im Hauptmenü, dem Menü mit dem Menünamen INFORMIX-SQL, befinden.
Wenn Sie sich in einem Folgemenü befinden, dann müssen Sie zunächst in das Hauptmenü zurückkehren.

Da auch das Hauptmenü die Funktion END anbietet, haben Sie wiederum die drei oben genannten Möglichkeiten um das Menü zu beenden: Entweder Sie drücken [END] oder Sie führen die Funktion END aus, indem Sie END markieren und [↵] drücken, oder Sie führen die Funktion END aus, indem Sie 'e' eingeben.

▷ Beenden Sie INFORMIX mit einer der genannten Möglichkeiten.

Jetzt haben Sie INFORMIX verlassen und befinden sich wieder in der Shell.

Wenn Sie INFORMIX wieder aufrufen wollen, dann führen Sie folgende Aktionen durch:

▷ Geben Sie ein: isql

▷ Drücken Sie [↵]

Allgemeine Bedienungshinweise

Das INFORMIX-Menüsystem bedienen Sie wie folgt:

↑	blättert zwischen den Zeilen von Menüs, die die Pseudofunktion '...' enthalten. ↓, ←, →, ⇧, ⇩ haben die gleiche Wirkung.
Leertaste	markiert die jeweils rechts folgende Funktion. Dabei verändert sich auch der erläuternde Hinweis in der Kommentarzeile.
⌫	markiert die jeweils links folgende Funktion. Von links außen springt die Markierung nach rechts außen.
Pfeiltasten	→ und ← wirken wie Leertaste → und ← wirken wie ⌫ **Vorsicht** Andere Wirkung im Menü TABELLE ERSTELLEN! Nähere Hinweise siehe dort. Die Pfeiltasten sollte deshalb nur der geübte Anwender verwenden.
↵	führt die markierte Funktion aus.
Buchstabe	durch die Eingabe des Anfangsbuchstabens wird die gewünschte Funktion unmittelbar ausgeführt. Der erläuternde Kommentar in der Kommentarzeile wird nicht ausgegeben. Um Desorientierung zu vermeiden, sollte nur der geübte Anwender diese Möglichkeit nutzen.
DEL	bricht eine eingeleitete Aktion ab und führt zurück zum vorherigen Menü.
END	beendet Menüs, in denen die Funktion END angeboten wird; Sie kehren zurück zum übergeordneten Menü. END im Hauptmenü beendet INFORMIX.

2.2.4 Hilfe-Texte

An jeder Stelle des Menüsystems stehen Ihnen erläuternde Hilfe-Texte zur Verfügung. Sie können die Texte z.B. dann abrufen, wenn Ihnen der erläuternde Text in der Kommentarzeile nicht ausreicht oder wenn Sie Unterstützung benötigen. Sie gehen in diesem Fall wie folgt vor:

▷ Drücken Sie [HELP]

Im Anschluß daran erhalten Sie auf dem Bildschirm einen Hilfetext mit weitergehenden Erläuterungen. Die Hilfe-Texte beziehen sich auf die jeweils im Menü markierte Funktion.

Das Hilfe-Menü enthält die Funktionen 'Weiter und END'. Wenn sich ein Hilfe-Text über mehrere Bildschirmseiten erstreckt, dann ist die Funktion 'Weiter' markiert, mit der Sie zur nächsten Bildschirmseite blättern können; bei Hilfe-Texten mit nur einer Bildschirmseite ist 'END' markiert. Durch Ausführen der Funktion END kehren Sie zurück zur Ausgangssituation.

Hinweis

Im SQL-Dialog-Menü sind die Hilfe-Funktionen erweitert. Hier erhalten Sie ein Menü, das Ihnen das gezielte Anwählen von Syntaxinformationen ermöglicht.

Zusammenfassung

- Mit dem Programmnamen 'isql' rufen Sie INFORMIX auf.
- Ein INFORMIX-Bildschirm setzt sich aus den folgenden Teilen zusammen: Funktionszeile mit Menünamen, Kommentarzeile, Infozeile, Arbeitsbereich und Meldungszeilen.
- [↑] und [↓] (und anderen Pfeiltasten) blättern zwischen den Zeilen von Menüs, die die Pseudofunktion '...' enthalten.
- Eine Funktion führen Sie wie folgt aus: Markieren und [↵] oder den gewünschten Anfangsbuchstaben eingeben.
- [END] beendet ein INFORMIX-Menü und führt zurück zum vorherigen Menü. [END] im Hauptmenü beendet INFORMIX.
- [DEL] bricht eine eingeleitete Aktion ab und führt zurück zum vorherigen Menü.
- [HELP] führt zu einem Bildschirm mit Hilfe-Texten. Die Hilfe-Texte beziehen sich auf die jeweils markierte Funktion.

3 Eine Datenbank aufbauen

Das vorliegende Kapitel leitet Sie dazu an, die Datenbank 'notizbuch' aufzubauen.
Anhand der Datenbank 'notizbuch' erhalten Sie einen praktischen Überblick über die wichtigsten INFORMIX-Funktionen und -Werkzeuge.

Sie erfahren,

- wie Sie eine Datenbank erzeugen,
- wie Sie eine Tabelle erzeugen,
- wie Sie ein Format erzeugen,
- wie Sie in einem Format neue Sätze eingeben, die Datenbank abfragen, Sätze korrigieren und löschen,
- wie Sie mit SQL die Datenbank abfragen,
- wie Sie eine Liste erzeugen und ablaufen lassen.

Das vorliegende Kapitel setzt voraus, daß Sie in der Lage sind, INFORMIX aufzurufen und das INFORMIX-Menüsystem zu bedienen (siehe Kapitel 2).

3.1 Eine Datenbank erzeugen

Um neue Adressen wie gewohnt mit Papier und Bleistift zu verwalten, benötigt man zunächst ein leeres Notizbuch.

Bei INFORMIX benötigen Sie eine 'leere' Datenbank. In diesem Fall eine Datenbank mit dem Namen 'notizbuch'; diese Datenbank gilt es zu erzeugen.

Eine Datenbank können Sie wie folgt erzeugen:

- menügeführt über die Funktion 'Datenbank' oder
- über eine SQL-Anweisung.

Dieses Kapitel informiert Sie darüber, wie sie eine Datenbank menügeführt erzeugen. Wie Sie eine Datenbank über eine SQL-Anweisung erzeugen, erfahren Sie im Handbuch [2] INFORMIX-SQL Nachschlagen.

3.1.1 Eine Datenbank menügeführt erzeugen

Die folgende Übung leitet Sie dazu an, die Datenbank 'notizbuch' zu erzeugen. Im Anschluß an die Übung finden Sie eine Übersicht mit allgemeinen Bedienungshinweisen zur menügeführten Erzeugung einer Datenbank.

Übung 3-1

```
INFORMIX-SQL:   Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
 Auswaehlen, Neuerstellen oder Loeschen einer Datenbank.

_____________________________________________________________________
```

▷ Im Hauptmenü führen Sie die Funktion 'Datenbank' aus.

Sie haben nun dieses Menü erhalten:

```
DATENBANK:   Auswahl  Neu  Loeschen  END
Eine Datenbank auswaehlen.

_____________________________________________________________________
```

Die Funktionen bedeuten:

Auswahl
: eine Datenbank auswählen und damit zur aktuellen Datenbank machen (siehe unten),

Neu
: eine neue Datenbank erzeugen,

Loeschen
: eine bestehende Datenbank löschen.
Vorsicht
gelöscht wird die Datenbank mit sämtlichen darin enthaltenen Tabellen!

▷ Führen Sie die Funktion 'Neu' aus.

Sie erhalten diesen Bildschirm:

```
NEUE DATENBANK >>
Namen der neuen Datenbank eingeben. Weiter mit RETURN
```

In der Funktionszeile haben Sie ein Eingabefeld erhalten; das Eingabefeld dient der Eingabe des gewünschten Datenbank-Namens.

▷ Tragen Sie hier ein: notizbuch

▷ Drücken Sie [↵]

INFORMIX erzeugt nun die Datenbank 'notizbuch'.

Hinweis

Bei INFORMIX-SE wird dabei im aktuellen Dateiverzeichnis ein neues Dateiverzeichnis mit dem Namen 'notizbuch.dbs' eingerichtet. Dieses Dateiverzeichnis enthält einige Dateien mit Systemtabellen, die INFORMIX zur internen Verwaltung benötigt.
Führen Sie keinesfalls auf Betriebssystemebene Änderungen oder Löschungen an einem 'datenbankname.dbs' Dateiverzeichnis durch, da dabei die entsprechende Datenbank zerstört werden könnte!
Bei INFORMIX-ONLINE ist die Datenbank auf Betriebssystemebene nicht sichtbar.

Nach kurzer Wartezeit geschieht zweierlei:

- INFORMIX führt Sie zurück in das vorherige Menü DATENBANK.
- Auf der Infozeile zwischen Kommentarzeile und Arbeitsbereich bekommen Sie den Namen der neu erzeugten Datenbank angezeigt.

Um in das Hauptmenü zurückzukehren:

▷ Drücken Sie [END]

Die Datenbank ist erzeugt, in einem nächsten Schritt soll es nun darum gehen, eine Tabelle zu erzeugen.

3.1.2 Allgemeine Bedienungshinweise

Die folgenden Bedienungshinweise fassen zusammen, wie Sie eine Datenbank menügeführt erzeugen:

1. Im Hauptmenü führen Sie die Funktion 'Datenbank' aus.
2. Im damit aufgerufenen Menü führen Sie die Funktion 'Neu' aus.
3. Sie erhalten einen Bildschirm mit einem Eingabefeld für den Datenbanknamen.
   Geben Sie den gewünschten Namen ein und drücken Sie ↵
   NAMENSKONVENTIONEN: Der Name darf nicht länger als 10 Zeichen sein; er muß mit einem Buchstaben beginnen und darf aus einer beliebigen Kombination von Buchstaben, Ziffern und Unterstrichen bestehen. Groß- bzw. Kleinschreibweise unterscheidet INFORMIX nicht und gibt den Namen einer Datenbank grundsätzlich in Kleinschreibweise aus. Von INFORMIX reservierte Wörter sollten Sie nicht verwenden (siehe Anhang des Handbuches [1] SQL).
4. Die Datenbank ist damit erzeugt; auf der Infozeile wird der Name der neu erzeugten Datenbank angezeigt. INFORMIX zeigt damit an, daß die neu erzeugte Datenbank nun auch gleichzeitig die aktuelle Datenbank ist.

3.2 Die aktuelle Datenbank

Mit INFORMIX können Sie mehrere Datenbanken im aktuellen Dateiverzeichnis führen. Die Datenbank, die Sie ausgewählt (oder neu erzeugt) haben, nennt man die 'aktuelle Datenbank'.
INFORMIX zeigt den Namen der aktuellen Datenbank rechts außen auf der Infozeile an.

Haben Sie eine Datenbank neu erzeugt, dann macht INFORMIX diese automatisch zur aktuellen Datenbank und zeigt ihren Namen auf der Infozeile an.

Wenn Sie aber INFORMIX neu aufrufen und eine ganz bestimmte Datenbank auswählen wollen, oder wenn Sie während Ihrer INFORMIX-Sitzung eine andere als die zur Zeit aktuelle Datenbank auswählen wollen, dann:

- ▷ Führen Sie im Hauptmenü die Funktion 'Datenbank' aus.
- ▷ Im damit aufgerufenen Menü führen Sie die Funktion 'Auswahl' aus.
- ▷ Sie erhalten nun ein Menü, das Ihnen im Arbeitsbereich die Namen aller verfügbaren Datenbanken an. Der erste Name ist markiert.
- ▷ Die gewünschte Datenbank können Sie wie folgt auswählen:
  entweder Sie markieren den Namen der gewünschten Datenbank mit den Pfeiltasten, oder Sie schreiben den Namen der Datenbank in das Eingabefeld.
- ▷ Danach drücken Sie [↵].

Im Anschluß daran bekommen Sie den Namen der neu ausgewählten Datenbank auf der Infozeile angezeigt.

3.3 Eine Tabelle erstellen

Nachdem die Datenbank erzeugt wurde, können Sie damit beginnen, eine Tabelle zu erstellen.

Eine Tabelle können Sie auf zwei Wegen erstellen:

- menügeführt über die Funktion 'Tabelle' oder
- mit einer SQL-Anweisung.

Dieses Kapitel informiert Sie darüber, wie Sie eine Tabelle menügeführt erstellen. Wie Sie eine Tabelle mit einer SQL-Anweisung erstellen, erfahren Sie im Abschnitt 9.2.1 bzw. im Handbuch [2] INFORMIX-SQL Nachschlagen.

3.3.1 Das 'TABELLE ERSTELLEN'-Menü aufrufen

Die folgende Übung leitet Sie dazu an, das 'TABELLE ERSTELLEN'-Menü aufzurufen.

Übung 3-2

```
INFORMIX-SQL:   ...  Tabelle  END
Neuerstellen, Modifizieren, oder Loeschen einer Datenbanktabelle.

————————— notizbuch ————————————————————————————————————————
```

▷ Führen Sie im Hauptmenü die Funktion 'Tabelle' aus.

Sie erhalten das TABELLE-Menü:

```
TABELLE:  Neu  Modifizieren  Info  Loeschen  END
Erstellen einer neuen Tabelle.

————————— notizbuch ————————————————————————————————————————
```

Das TABELLE-Menü

Das Menü bietet Ihnen die folgenden Funktionen an:

Neu
: erstellen einer neuen Tabelle,

Modifizieren
: die Struktur einer bestehenden Tabelle verändern. Im Anschluß daran müssen Sie noch die Tabelle auswählen, die Sie modifizieren möchten. Danach erhalten Sie das TABELLE-AENDERN-Menü, das bis auf den Menünamen identisch ist mit dem folgend beschriebenen TABELLE-ERSTELLEN-Menü.

Info
: Informationen über Tabellen der aktuellen Datenbank anzeigen. Im Anschluß daran müssen Sie noch die Tabelle auswählen, über deren Struktur Sie sich informieren wollen. Danach erhalten Sie das INFO-Menü. Das INFO-Menü gibt es auch bei SQL; nähere Hinweise siehe dort.

Loeschen
: Tabelle aus der Datenbank löschen. Im Anschluß daran müssen Sie noch die Tabelle auswählen, die gelöscht werden soll. Dann erhalten Sie ein BESTÄTIGEN 'Nein Ja'-Menü. Wenn Sie die Funktion 'Ja' ausführen, dann wird die Tabelle gelöscht.

  Vorsicht

  Gelöscht wird die Tabelle mit sämtlichen darin enthaltenen Sätzen!

END
: in das INFORMIX-Hauptmenü zurückkehren.

Um das TABELLE-ERSTELLEN-Menü aufzurufen, geht es nun wie folgt weiter:

▷ Führen Sie die Funktion 'Neu' aus.

Sie erhalten diesen Bildschirm:

```
NEU TABELLE >>
Namen der neuen Tabelle eingeben. Weiter mit RETURN

————————— notizbuch ————————————————————————————————————————
```

Sie sind nun aufgefordert, der Tabelle einen Namen zu geben.

▷ Geben Sie ein: adressen

▷ Drücken sie ↵

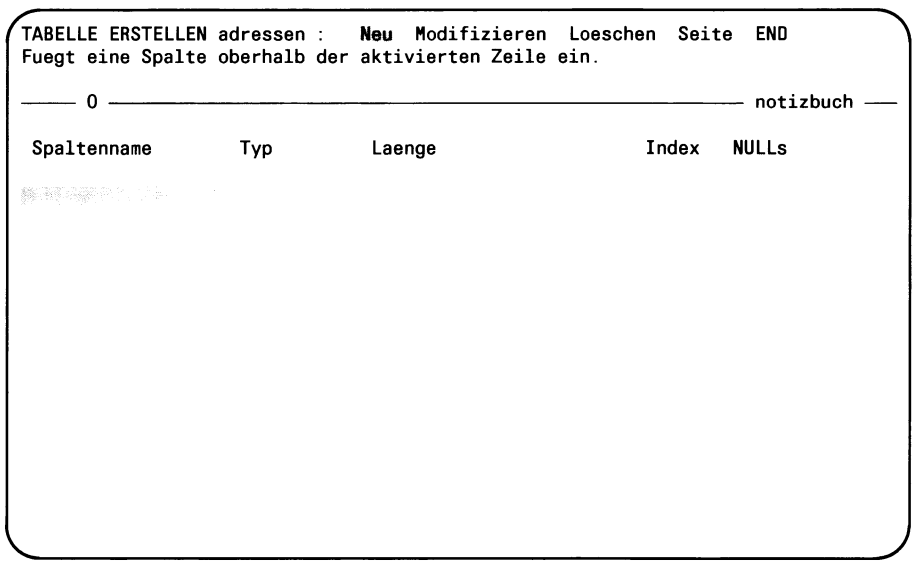
INFORMIX zeigt Ihnen nach kurzer Wartezeit das 'TABELLE ERSTELLEN'-Menü an; hinter dem Menünamen steht der Name der neu zu erstellenden Tabelle, 'adressen'(siehe Abschnitt 3.3.2).

Allgemeine Bedinungshinweise

Um das TABELLE-ERSTELLEN-Menü aufzurufen, führen Sie die folgenden Schritte aus:

1. Im Hauptmenü führen Sie die Funktion 'Tabelle' aus.
2. Wenn noch keine Datenbank ausgewählt wurde, dann erhalten Sie jetzt den AUSWAHL-DATENBANK-Bildschirm. Wählen Sie die gewünschte Datenbank aus.
3. Sie erhalten das TABELLE-Menü. Führen Sie die Funktion 'Neu' aus.
4. Sie erhalten einen Bildschirm mit einem TABELLE-Eingabefeld in der Funktionszeile. Geben Sie den gewünschten Namen der neuen Tabelle ein und drücken Sie ↵
   NAMENSKONVENTIONEN: wie bei Datenbank-Namen; jedoch darf der Name 18 statt 10 Zeichen lang sein. Wenn Sie den Namen einer bereits existierenden Tabelle eingegeben haben, dann wird anschließend nicht das TABELLE-ERSTELLEN-Menü aufgerufen, sondern das TABELLE-AENDERN-Menü.

3.3.2 Ausführbare Funktionen des TABELLE-ERSTELLEN-Menüs



Das TABELLE-ERSTELLEN-Menü bietet in der Funktionszeile die folgenden Funktionen an:

Neu
: eine neue Spalte in die Tabelle einfügen.

Modifizieren
: eine bestehende Spalte verändern,

Loeschen
: eine bestehende Spalte aus der Tabelle löschen.

Seite
: auf die nächste Bildschirmseite blättern (bei Tabellen mit mehr als 16 Spalten).

END
: Nachdem Sie eine Tabelle neu definiert oder modifiziert haben, dann gelangen Sie mit END in das 'END-Menü'. Dieses Menü bietet die Funktionen 'Sichern-und-END' und 'Verwerfen-und-END' an; hier ist darüber zu entscheiden, ob die Tabellen-Definition gesichert oder aber wieder verworfen werden soll.
Danach kehren Sie automatisch wieder in das TABELLE-Menü zurück. In diesem Menü führt Sie END in das Hauptmenü zurück.

Im Arbeitsbereich gibt es fünf Bereiche mit den Spalten-Definitionsmerkmalen.

Wenn Sie im Verlauf der folgenden Übungen schließlich die Tabelle 'adressen' vollständig definiert haben, dann wird Ihr Bildschirm so aussehen:

```
TABELLE ERSTELLEN adressen :   Neu  Modifizieren  Loeschen  Seite  END
Fuegt eine Spalte oberhalb der aktivierten Zeile ein.

——— 1 ——————————————————————————————————————————————— notizbuch ——

 Spaltenname        Typ         Laenge                  Index   NULLs

 nachname           Char        20                              Ja
 vorname            Char        20                              Ja
 strasse            Char        20                              Ja
 plz                Char        10                              Ja
 ort                Char        20                              Ja
 telefon            Char        15                              Ja
```

Jede Zeile im Arbeitsbereich zeigt die Spalten-Definitionsmerkmale je einer Tabellenspalte an. Die folgenden Abschnitte gehen auf die Bedeutung der einzelnen Spalten-Definitionsmerkmale ein. Hier zunächst eine kurze Übersicht:

Spaltenname
: der gewählte Name der Spalte,

Typ
: der zugeordnete Datentyp,

Laenge
: Spaltenlänge,

Index
zeigt mit dem spezifischen Namen (Unique - Dups) an, ob der Spalte ein Index zugeordnet ist; der Index dient u.a dem beschleunigten Datenbank-Zugriff,

NULLs
zeigt an, ob sogenannte NULL-Werte (leere Werte, keine Eingabe) für die Spalte zugelassen sind (ja) oder nicht zugelassen sind (nein).

Der folgende Abschnitt geht ausführlicher auf Datentypen und Spalten-Definitionsmerkmale ein.
Der Abschnitt unterbricht den Fluß Ihrer Aktion am Bildschirm; Sie sollten ihn dann vollständig lesen, wenn Sie sich ausführlich über Datentypen und Spalten-Definitionsmerkmale informieren wollen.
Wenn Sie aber umgehend mit der Tabellen-Definition der Tabelle 'adressen' fortfahren wollen, dann genügt es, wenn Sie weiter unten die Abschnitte '3.3.4 ZEICHEN Datentyp', '3.3.5 Index' und '3.3.6 NULL-Werte' lesen. Danach können Sie in Abschnitt 3.3.7 mit der Erzeugung der Tabelle 'adressen' fortfahren.

3.3.3 Spaltenname

Ein Spaltenname darf maximal 18 Zeichen lang sein und er muß mit einem Buchstaben beginnen. Die weiteren Zeichen dürfen aus Buchstaben, Zahlen und Unterstrichen. Groß- bzw. Kleinschreibung unterscheidet INFORMIX nicht und gibt die Spaltennamen immer in Kleinschreibweise aus. Von INFORMIX reservierte Wörter sollten Sie nicht verwenden. Eine Liste mit den reservierten Wörtern finden Sie im Anhang des Handbuches [1] SQL.

3.3.4 Datentyp

Eine Datenbank kann Daten unterschiedlichster Art aufnehmen, so z.B. Namen, Postleitzahlen, Gehälter, Produktbeschreibungen und vieles andere mehr. Um die Daten optimal speichern und auswerten zu können, sieht INFORMIX unterschiedliche Datentypen vor, z.B. Datentypen, die dazu geeignet sind, Namen, Produktbeschreibungen und beliebige Texte aufzunehmen und andere Datentypen, die dazu geeignet sind, Zahlen aufzunehmen und wiederum andere Datentypen zur Speicherung von Kalenderdaten und Geldbeträgen. Hier eine Übersicht. Weitergehende Erläuterungen siehe unten. Es gibt die folgenden Datentypen:

ZEICHEN
: Kombination aus beliebigen Zeichen; z.B. Buchstaben, Zahlen und Satzzeichen. Es gibt die folgenden ZEICHEN-Datentypen:

  CHAR mit fester Spaltenlänge.

  VARCHAR mit variabler Spaltenlänge (nur INFORMIX-ONLINE). Wird im Menüsystem unter 'Variable-length' angeboten.

NUMERISCH
: es gibt sechs unterschiedliche numerische Datentypen (einschließlich SERIAL, s.u.). Sie alle nehmen Zahlen auf; Festkommazahlen, ganze Zahlen oder Gleitkommazahlen:

  DECIMAL Festkommazahlen:
  : gewünschter maximaler Spaltenumfang und Nachkommastellen lassen sich optional festlegen.

SMALLINT positive und negative ganze Zahlen:
ca. +/-dreißigtausend.

INTEGER positive und negative ganze Zahlen:
ca. +/-zwei Milliarden.

SMALLFLOAT positive und negative Gleitkommazahlen.

FLOAT positive und negative Gleitkommazahlen:
doppelte Genauigkeit gegenüber SMALLFLOAT

ZEIT
es gibt drei unterschiedliche ZEIT-Datentypen:

DATE für ein Datum.

DATETIME für einen Zeitunkt: Datum und/oder Uhrzeit.

INTERVAL für eine Zeitspanne.

SERIAL
spezieller numerischer Datentyp zur automatischen eindeutigen Durchnumerierung.

MONEY
Geldbeträge.

BLOB
Binary Large Objekts; nur INFORMIX-ONLINE. Im Menüsystem werden die BLOB-Datentypen unter 'Variable-length' angeboten. Es gibt folgende BLOB-Datentypen:

TEXT nimmt beliebig lange ASCII-Texte auf.

BYTE nimmt Daten mit beliebigem Inhalt auf, so z.B. HIT-Dokumente, SIPLAN-Tabellen, Grafiken u.a.m.

ZEICHEN Datentypen

Die ZEICHEN-Datentypen CHAR und VARCHAR nehmen außer Steuerzeichen jedes beliebige Zeichen auf. So z.B. Buchstaben, Zahlen und Satzzeichen. Die Datentypen eignen sich beispielsweise zur Aufnahme von Namen, Anschriften, Berufs-, Waren-, Projektbezeichnungen oder beliebigen anderen Texten von beschränktem Umfang.

Beispiel

```
Augsburg
Postweg 65
Systembeauftragter
Projekt zur Sicherung erhaltenswerter Bausubstanz
```

Beim Datentyp CHAR ist die Spaltenlänge anzugeben: Mit der Spaltenlänge legen Sie fest, wieviel Zeichen die Spalte maximal aufnehmen kann. Wenn ein Wert eingegeben wird, der größer ist als bei der Spaltendefinition festgelegt wurde, dann werden die überstehenden Zeichen abgeschnitten und nicht in die Datenbank aufgenommen.
So kann z.B. eine CHAR-Spalte mit der Spaltenlänge 10 den Ortsnamen 'Allmannshausen' nur verstümmelt aufnehmen, nämlich in dieser Form: 'Allmannsha', die überstehenden Zeichen werden abgeschnitten.

Bei der Spaltendefinition ist abzuwägen, daß einerseits die Spaltenlänge groß genug gewählt wird, um auch den umfangreichsten Wert eingeben zu können und daß andererseits die Spaltenlänge nicht überdimensional gewählt wird, um nicht unnötig Speicherplatz zu verschwenden.

Beim Datentyp VARCHAR kann man optional angeben, wieviele Zeichen mindestens aufgenommen werden sollen. Außerdem ist anzugeben, wieviel Zeichen maximal aufgenommen werden sollen. Der Datentyp VARCHAR zeichnet sich dadurch aus, daß er nur den eingebenen Wert (bzw. die Mindestlänge) speichert. Er ist dadurch sehr speicherplatzsparend.

NUMERISCHE Datentypen

Dieser Abschnitt führt Sie in die Handhabung numerischer Datentypen ein; ausführliche Informationen wie z.B. Rundungsverhalten und Speicherbedarf erhalten Sie im Handbuch [1] SQL.

Die numerischen Datentypen lassen sich nach drei Gruppen unterscheiden:

Festkommazahlen
Postive und negative Festkommazahlen
Datentyp DECIMAL

Beim Datentyp DECIMAL können Sie den gewünschten maximalen Spaltenumfang und die Nachkommastellen optional festlegen.
Beispiel: DECIMAL (6.2) legt fest, daß die Spalte maximal 4 Stellen vor dem Komma und 2 Stellen nach dem Komma erlaubt:

```
Maximaler Spaltenumfang - Nachkommastellen = Stellen vor dem Komma
          6             -         2        =           4
```

Vorsicht

Unter Umständen verfälschende Rundungen bei größeren Zahlen als mit Spaltenumfang und Nachkommastellen festgelegt!

Wenn bei DECIMAL keine oder in den Klammern nur eine Zahl angegeben ist, dann ist damit ein Gleitkomma-Datentyp definiert (siehe Handbuch [1] SQL).

Ganze Zahlen
positive und negative ganze Zahlen
Datentypen SMALLINT und INTEGER

Beide Datentypen nehmen positive und negative ganze Zahlen auf (keine Kommazahlen). Sie unterscheiden sich lediglich in der maximal zulässigen Zahlengröße und damit auch im Speicherbedarf:

```
SMALLINT - maximal ca. +/-dreißigtausend: -32767 bis +32767
INTEGER - maximal ca. +/-zwei Milliarden: -2147483647 bis +2147483647
```

Gleitkommazahlen
Datentypen SMALLFLOAT und FLOAT

SMALLFLOAT- und FLOAT-Spalten nehmen binäre Gleitkommazahlen auf.
SMALLFLOAT-Spalten behandeln Gleitkommazahlen mit einfacher Genauigkeit und etwa sieben genauen Stellen.

FLOAT-Spalten behandeln Gleitkommazahlen mit doppelter Genauigkeit und etwa vierzehn genauen Stellen; gegenüber SMALLFLOAT doppelter Speicherbedarf.

Hinweis

FLOAT-Spalten speichern nicht etwa größere Zahlen sondern Zahlen mit größerer Genauigkeit!

Das TABELLE-ERSTELLEN-Menü zeigt aus Platzgründen als Datentyp für Gleitkommazahlen zunächst nur FLOAT an. Wenn Sie diese Funktion auswählen, dann erhalten Sie ein Folgemenü, das Ihnen schließlich die beiden möglichen Datentypen FLOAT und SMALLFLOAT zur Auswahl anbietet.

ZEIT Datentypen

Es gibt drei ZEIT Datentypen: DATE, DATETIME, INTERVAL

DATE

Der Datentyp DATE nimmt Kalenderdaten auf. INFORMIX überprüft bei der Eingabe neuer Kalenderdaten in eine DATE-Spalte, ob das Datum vollständig eingegeben wurde, d.h. ob sowohl Tag als auch Monat als auch Jahr eingegeben wurden. Wenn eine der Angaben fehlt oder falsch ist, weist INFORMIX die Eingabe zurück.

Bei der Eingabe eines Datums sind standardmäßig mehrere Schreibweisen möglich:
Als Trennzeichen darf Punkt oder Komma verwendet werden, führende Null nach DIN ist erlaubt, die Jahreszahl darf abgekürzt oder ausgeschrieben werden:

Beispiel

```
05.02.90        6.9.90        17.4.1990        13,06.1990
```

DATETIME

Der Datentyp DATETIME nimmt Zeitpunkte auf. Die Genauigkeit des Zeitpunktes ist menügeführt mit YEAR (Jahr), MONTH (Monat), DAY (Tag), MINUTE (Minute), SECOND (Sekunde) und FRACTION (Sekundenbruchteil) festlegbar.
Mit YEAR TO MONTH beispielsweise legt man einen Datentyp fest, der DATE entspricht; bei dem Jahr, Monat und Tag gespeichert werden können. Mit YEAR TO YEAR beispielsweise legt man fest, daß nur Jahre gespeichert werden können. Mit MINUTE TO SECOND kann man eine Uhrzeit speichern.

INTERVAL

Mit INTERVAL kann man Zeitspannen abspeichern. Die Genauigkeit der Zeitspanne ist menügeführt mit YEAR (Jahr), MONTH (Monat), DAY (Tag), MINUTE (Minute), SECOND (Sekunde) und FRACTION (Sekundenbruchteil) festlegbar.
Eine Zeitspanne kann maximal entweder über YEAR TO MONTH (Tage bis zu Monaten) oder über DAY TO FRACTION (Tage bis zu Sekundenbruchteilen) gehen.

Datentyp SERIAL

Zahlreiche Datenbank-Anwendungen erfordern, daß jeder Datensatz eine eindeutige, kein zweites Mal mehr vorkommende Nummer erhält.
Zum Beispiel bei Produktnummern muß gewährleistet sein, daß jedes Produkt nur unter seiner eigenen Nummer geführt wird; eine einmal vergebene Nummer darf keinem zweiten Produkt mehr zugeordnet sein.

Für diese Zwecke bietet INFORMIX den Datentyp SERIAL an. Wenn Sie bei der Tabellendefinition eine SERIAL-Spalte festlegen, dann unternimmt INFORMIX alle weiter erforderlichen Schritte automatisch.

Eine SERIAL-Spalte wirkt sich wie folgt aus:

- Jedesmal, wenn Sie einen Datensatz neu aufnehmen, dann weist INFORMIX der SERIAL-Spalte automatisch einen Wert mit einer Nummer zu.
  Wenn Sie beispielsweise den ersten Datensatz aufnehmen, dann erhält die SERIAL-Spalte automatisch den Wert 1 (die gewünschte Startnummer ist festlegbar).
- Bei der Eingabe jedes neuen Datensatzes erhält die SERIAL-Spalte einen Wert, der sich um je 1 erhöht (bzw. der noch nicht vergeben wurde).
  Der zweite eingegebene Datensatz erhält also die Nummer 2, der dritte Datensatz die Nummer 3 u.s.w

Um die Eindeutigkeit der Nummern zu gewährleisten, ist es Ihnen nicht möglich, eine einmal vergebene Nummer nachträglich zu ändern.

Hinweis

Jede Tabelle darf nur eine SERIAL-Spalte enthalten.

Bei der Festlegung einer SERIAL-Spalte fragt Sie INFORMIX nach der gewünschten Startnummer. Sie können jede Nummer größer als '0' angeben. Die höchste von INFORMIX in einer SERIAL-Spalte vergebbare Nummer entspricht der Obergrenze einer INTEGER-Spalte (ca. zwei Milliarden).

Datentyp MONEY

Der Datentyp MONEY nimmt Geldbeträge auf. Es handelt sich bei diesem Datentyp um eine Variante des numerischen Datentyps DECIMAL. Die Gesamtspaltenlänge ist optional festlegbar, nicht aber die Nachkommastellen (immer 2). Standardmäßig bietet INFORMIX für eine MONEY-Spalte dieses Format an: 16,2 Das bedeutet, daß 14 Stellen vor dem Komma und zwei Stellen nach dem Komma zulässig sind.

BLOB Datentypen

Bei den BLOB-Datentypen TEXT und BYTE ist menügeführt anzugeben, ob sie in der Tabelle oder in einem speziellen BLOBSPACE gespeichert werden sollen. Siehe hierzu Handbuch [4] INFORMIX-ONLINE Administrator-Handbuch.

BLOB-Daten kann der Benutzer nicht unmittelbar bearbeiten sondern nur über ein von INFORMIX aufzurufendes Programm (z.B. CED oder HIT). Siehe hierzu die Kapitel 5 und 6.

Es gibt zwei BLOB Datentypen: TEXT, BYTE

TEXT

TEXT kann ASCII-Texte von beliebiger Länge speichern. Speicherbar sind die gleichen Zeichen wie bei CHAR und VARCHAR und zusätzlich die Steuerzeichen HT (Tabulatorsprung), FF (Seitenvorschub) und CR (Neue Zeile).

BYTE

BYTE kann Daten mit beliebigem Inhalt und von beliebiger Länge speichern. So zum Beispiel HIT-Dokumente, SIPLAN-Tabellen, Grafiken und anderes mehr.

3.3.5 Index

Das Indizieren einer Spalte dient dem beschleunigten Datenzugriff. Sie können sich das ähnlich vorstellen wie die alphabetische Sortierung der Stichwörter in einem Lexikon (siehe Kapitel 1).

Wann indiziert man eine Spalte?

Sie sollten eine Spalte nur dann indizieren, wenn eine oder mehrere der folgenden Bedingungen zutrifft:

- die Tabelle enthält mehr als 200 Sätze,
- die Spalte wird als Join-Spalte für die Verbindung von Tabellen verwendet,
- die Spalte wird regelmäßig für Suchanfragen verwendet, (z.B. Personaldaten über Nachnamen suchen),
- die Spalte wird regelmäßig für Sortierungen verwendet (z.B. alphabetisch sortierte Ausgabe der Personaldaten nach Nachnamen),
- Die Spalte ist geeignet, die Eindeutigkeit der zu speichernden Sätze zu garantieren.

Hinweis

Indizieren können Sie eine einzelne Spalte oder auch mehrere Spalten. Darüber hinaus können Sie auch mehrere Spalten einem gemeinsamen zusammengesetzten Index zuordnen (nur mit SQL). Mehr Informationen zum diesem Thema erhalten Sie im Kapitel 9 bzw. im Handbuch [1] SQL.

Nicht indizieren sollten Sie Spalten, die eine sehr größe Anzahl von mehrfach vorkommenden Werten enthalten.

Unique und Dups

Bei der Erzeugung eines Index können Sie entscheiden, ob die Eingabe von mehrfach vorkommenden Werten erlaubt (Dups) oder nicht erlaubt ist (Unique).
Zum Beispiel kann eine Tabelle mehrere Sätze enthalten, bei denen der Nachname 'Schmidt' lautet. Wenn die entsprechende Spalte einen Index erhalten soll, dann einen Dups-Index, der die Eingabe eines bereits enthaltenen Wertes erlaubt.

Umgekehrt ist es dann richtig, einer Spalte einen Unique-Index zuzuordnen, wenn die Werte dieser Spalte in der Tabelle nur einmal vorkommen dürfen. Mit einem Unique-Index kann man so die geforderte Eindeutigkeit der Sätze in einer Tabelle erzwingen.

Aufsteigende und absteigende Reihe

Standardmäßig erzeugt INFORMIX einen Index mit aufsteigender Reihe (0-n bzw. von A-z, wobei 'n' eine ganze positive Zahl ist). Maßgebend hierfür ist die Reihenfolge im ASCII-Code (siehe Anhang).
Die Wirkung hiervon ist u.a., daß die Ausgabe der Sätze in bestimmten Fällen automatisch sortiert erfolgt (z.B Augsburg vor Berlin u.s.w.).

In bestimmten Fällen kann es aber sinnvoll sein, einen Index mit absteigender Reihe zu erzeugen. Einen Index dieser Art können Sie nur mit SQL definieren.

3.3.6 NULL-Werte

In manchen Fällen ist es nicht möglich oder auch nicht sinnvoll, in eine Spalte einen Wert einzutragen, wenn ein Satz neu aufgenommen wird.
Mit dem Definitionsmerkmal NULL können Sie festlegen, ob in eine Spalte ein Wert eingetragen werden muß ("NULL's nein") oder der Eintrag unterbleiben darf ("NULL's ja").
Ist einer Spalte "NULL's nein" zugeordnet, dann weist INFORMIX die Neuaufnahme eines Satzes immer dann zurück, wenn in die entsprechende Spalte kein Wert eingetragen wurde.
Ist einer Spalte "NULL's ja" zugeordnet, so ist es erlaubt, in die entsprechende Spalte nichts einzutragen oder einen vorhandenen Eintrag zu löschen. INFORMIX nimmt dann einen NULL-Wert auf. Dieser NULL-Wert trägt den Informationsgehalt von "Wert nicht bekannt".

Sie sollten einer Spalte immer dann "NULL's ja" zuordnen, wenn es vorkommen kann, daß der entsprechende Wert zum Zeitpunkt der Neuaufnahme eines Satzes nicht bekannt ist. Beispielsweise bei einer Spalte mit Telefonnummern empfiehlt sich "NULL's ja", da eine Telefonnummer nicht immer bekannt ist oder eventuell auch gar nicht existiert.

Sie sollten einer Spalte immer dann "NULL's nein" zuordnen, wenn die Aufnahme eines Wertes in der entsprechenden Spalte logisch zwingend ist. Beispielsweise bei einer Tabelle zur Teileverwaltung sollte der Spalte mit der Produktnummer "NULL's nein" zugeordnet sein. Denn damit verhindert man, daß versehentlich ein Produkt ohne seine Produktnummer aufgenommen wird. Ein solcher Satz wäre u.U. sinnlos und würde die Datenbank inkosistent machen.

Haben Sie einer Spalte "NULL's nein" zugeordnet, dann müssen Sie einen Wert in die entsprechende Spalte eintragen. Ist der Wert dennoch nicht bekannt, dann müssen Sie einen Platzhalter eintragen; bei numerischen Spalten z.B. numerisch '0' und bei CHAR-Spalten ein beliebiges Zeichen.

Es gibt einen fundamentalen logischen Unterschied zwischen einem Platzhalter (numerisch '0') und keinem Eintrag (NULL-Wert), wie das folgende Beispiel verdeutlichen soll:

Gibt es in einer Tabelle zur Lagerverwaltung eine Spalte 'anzahl', so haben die Einträge numerisch '0' bzw. NULL-Wert (kein Eintrag) dort folgende Bedeutungen:

- Numerisch '0': Kein Teil auf Lager.
- NULL-Wert: Lagerstand nicht bekannt.

3.3.7 Bedienung des TABELLE-ERSTELLEN-Menüs

Dieser Abschnitt erläutert Ihnen, wie Sie das TABELLE-ERSTELLEN-Menü bedienen müssen. Zunächst folgen einige vorausgehende Bemerkungen zur grundsätzlichen Handhabung des Menüs.
Anschließend finden Sie eine Übung, in der Sie mit der Erzeugung der Tabelle 'adressen' fortfahren. Im Abschnitt 3.3.8 finden Sie "allgemeine Bedienungshinweise".

Grundsätzliche Handhabung

In TABELLE-ERSTELLEN-Menü sind zwei Bereiche mit Inversmarkierungen zu unterscheiden:

- die Funktionszeile am oberen Bildschirmrand und
- der Arbeitsbereich im mittleren Bildschirmabschnitt.

Der Arbeitsbereich ist in fünf Abschnitte mit den entsprechenden Spalten-Definitionsmerkmalen 'Spaltenname', 'Typ', 'Laenge', 'Index' und 'NULLs' unterteilt. Bis auf diese Anzeigen ist der Arbeitsbereich bei einer neu zu erstellenden Tabelle zunächst einmal vollständig leer.

Sowohl Funktionszeile als auch Arbeitsbereich verfügen über Inversmarkierungen:

- im Funktionsbereich zeigt sie die jeweils ausführbare Funktion an, und im
- Arbeitsbereich zeigt sie das jeweils aktuelle Spalten-Definitionsmerkmal an.

Die Funktionen in der Funktionszeile führen Sie wie in anderen Menüs aus, indem Sie den Anfangsbuchstaben bzw. den jeweils groß geschriebenen Buchstaben der gewünschten Funktion eingeben oder indem Sie mit der Leertaste oder mit [⌫] die gewünschte Funktion markieren und danach [↵] drücken. Die Pfeiltasten können Sie hierzu nicht verwenden. Diese wirken auf die Inversmarkierung im Arbeitsbereich. Bei INFORMIX-ONLINE gibt es im TYP-Menü eine zweite Zeile. um dorthin zu blättern, müssen Sie die Pseudofunktion '...' markieren. [↑] und [↓] können Sie hier nicht verwenden!

Der Arbeitsbereich dient beim Erzeugen einer Tabelle lediglich der Anzeige, hier gibt es für Sie nichts zu tun. Alle erforderlichen Tätigkeiten zur Spaltendefinition führen Sie im Funktionsbereich aus. Die Inversmarkierung im Arbeitsbereich wird von INFORMIX automatisch richtig plaziert!

Lediglich beim nachträglichen Hinzufügen, Ändern oder Löschen ist es u.U. erforderlich, die Inversmarkierung im Arbeitsbereich von Hand zu dirigieren; verwenden Sie in diesem Fall, aber nur in diesem Fall, die Pfeiltasten.

Wenn Sie eine Funktion ausgeführt haben, dann erhalten Sie in der Funktionszeile

- Eingabefelder, z.B. zur Eingabe des Spaltennamens oder zur Eingabe einer gewünschten Spaltenlänge, oder Sie erhalten weitere
- Funktionsmenüs zur Auswahl von Spaltenmerkmalen, z.B. zur Auswahl des gewünschten Datentyps.

Der ganze Ablauf vollzieht sich vollkommen menügeführt. INFORMIX führt Sie automatisch zur jeweils logisch folgenden Tätigkeit hin und macht Ihnen außerdem Vorschläge zu den Spalten-Definitionsmerkmalen. Nachdem Sie beispielsweise einen Spaltennamen eingegeben haben, springt die Inversmarkierung im Arbeitsbereich automatisch auf die folgende Spalte 'Laenge' und INFORMIX macht Ihnen hier bereits einen sinnvollen Vorschlag über die Spaltenlänge.
Wenn Sie einen automatisch erfolgten Vorschlag akzeptieren wollen, brauchen Sie lediglich [↵] zu drücken. Wenn Sie aber eigene Angaben machen wollen, dann führen Sie die entsprechenden Tätigkeiten im Funktionsbereich aus; hier erhalten Sie je nachdem ein Eingabefeld oder ein weiteres Funktionsmenü zur optionalen Auswahl.

Tabelle 'adressen' erstellen

Die folgende Übung unterweist Sie darin, die erste Spalte 'nachname' der Tabelle 'adressen' zu erstellen. Im Anschluß daran werden Sie die weiteren Spalten 'vorname', 'strasse', 'plz', 'ort' und 'telefon' selbständig erstellen. Wie Sie eventuelle Fehleingaben korrgieren, erfahren Sie in den anschließenden allgemeinen Bedienungshinweisen.

Hier eine Übersicht darüber, welche Spalten und Spalten-Definitionsmerkmale die Tabelle 'adressen' erhalten soll:

Spaltenname	nachname	vorname	strasse	plz	ort	telefon
Typ	char	char	char	char	char	char
Laenge	20	20	20	10	20	15
Index	nein	nein	nein	nein	nein	nein
Null	ja	ja	ja	ja	ja	ja

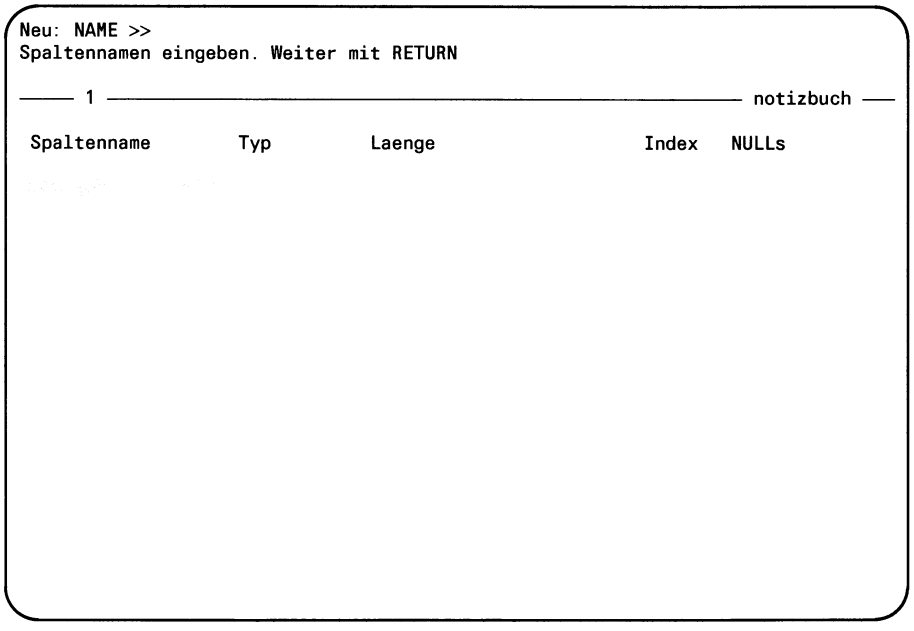
Übung 3-3

Diesen Bildschirm sollten Sie vor sich haben:

```
TABELLE ERSTELLEN adressen :   Neu  Modifizieren  Loeschen  Seite  END
Fuegt eine Spalte oberhalb der aktivierten Zeile ein.

——— 0 ——————————————————————————————————————————— notizbuch ——

 Spaltenname        Typ         Laenge                  Index   NULLs
```

▷ Im TABELLE-ERSTELLEN-Menü führen Sie die Funktion 'Neu' aus.

Sie erhalten diesen Bildschirm:



INFORMIX fordert Sie auf, den gewüschten Spaltennamen einzugeben.

▷ Geben Sie ein: nachname

▷ Drücken Sie ↵

Bei INFORMIX-ONLINE erhalten Sie diesen Bildschirm (Bei INFORMIX-SE wird statt der Pseudofunktion '...' 'Interval' angezeigt; eine zweite Funktionszeile gibt es dort nicht):

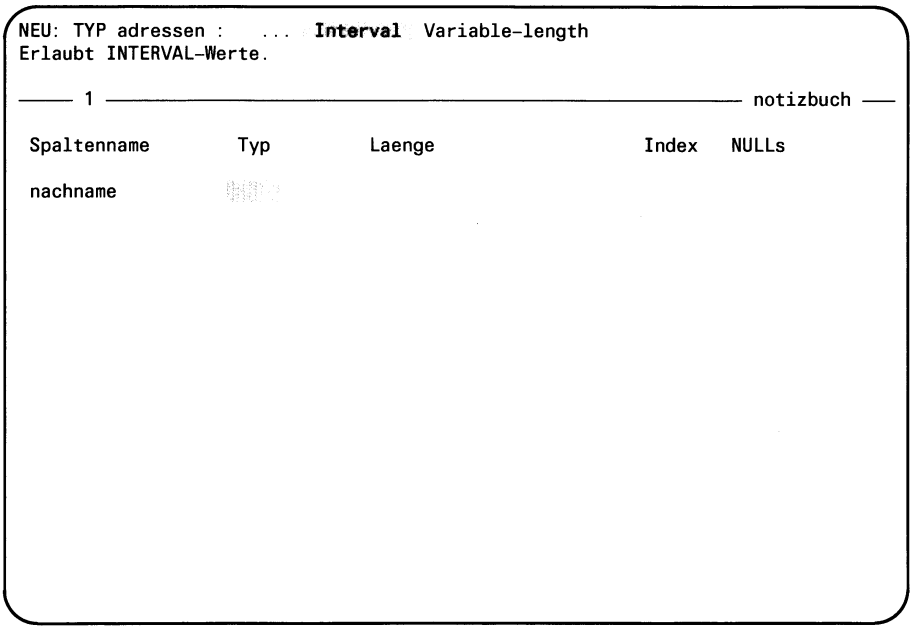
```
NEU: TYP adressen :  Char  Numeric  Serial  Date  Money  date-Time  ...
Erlaubt jede Kombination von Buchstaben, Zahlen und Symbolen.

------ 1 ------------------------------------------------------ notizbuch --

 Spaltenname        Typ          Laenge                    Index   NULLs

 nachname
```

Drei Dinge sind geschehen:

- INFORMIX zeigt den Spaltennamen nun im Arbeitsbereich in der Spalte 'Spaltenname' an.
- Die Inversmarkierung ist auf die nächste Spalte 'Typ' gesprungen.
- In der Funktionszeile haben Sie ein neues Menü erhalten; INFORMIX erwartet von Ihnen nun die Auswahl des gewünschten Datentyps; der Datentyp 'Char' ist markiert.

Um die Fortsetzung des Menüs zu sehen, können sie mit der Leertaste die Pseudofunktion '...' markieren.

Sie erhalten dann den folgenden Bildschirm:

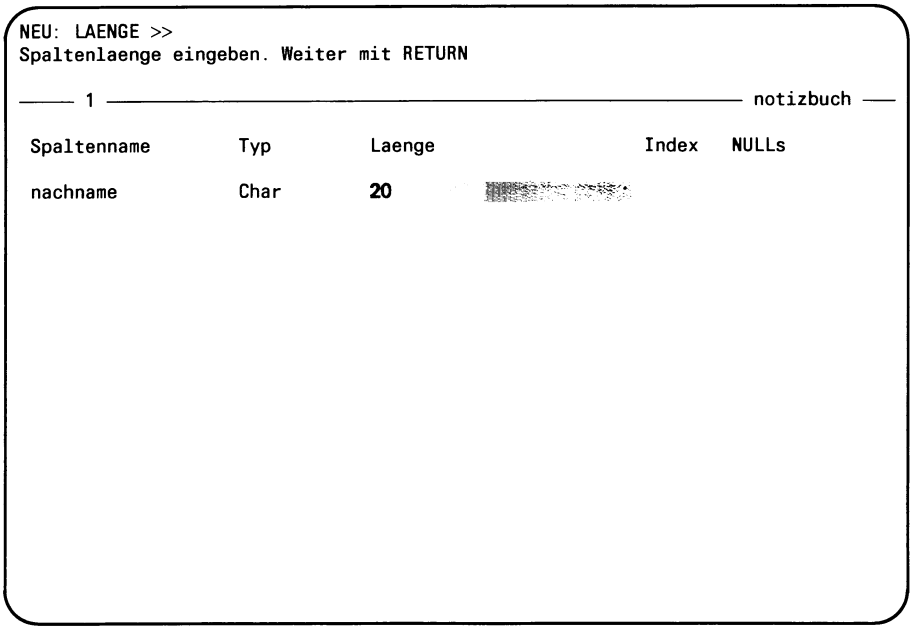


Die Funktion 'Variable-length' führt zu einem Untermenü mit den INFORMIX-ONLINE Datentypen VARCHAR, TEXT und BYTE.

▷ Blättern Sie mit Leertaste oder ⌫ zurück zum vorherigen Bildschirm, um mit dem Aufbau der Tabelle fortfahren zu können.

▷ Wählen Sie den Datentyp 'Char' aus.

Sie erhalten diesen Bildschirm:



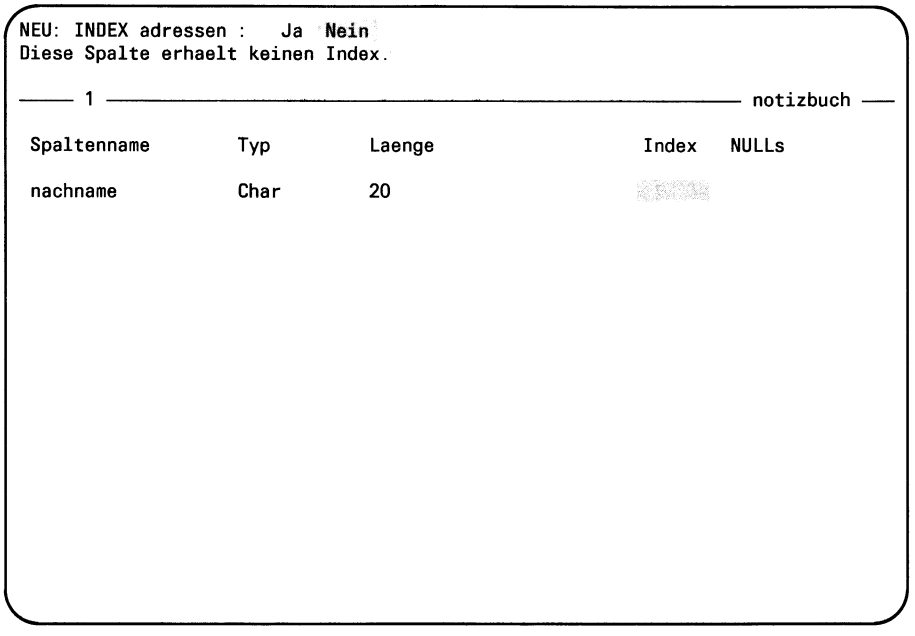
Das ist geschehen:

- INFORMIX zeigt den ausgewählten Datentyp nun im Arbeitsbereich in der Spalte 'Typ' an.
- Die Inversmarkierung ist auf die nächste Spalte 'Laenge' gesprungen. INFORMIX macht Ihnen hier einen Vorschlag zur Spaltenlänge: 20
- In der Funktionszeile haben Sie ein Eingabefeld erhalten, das Ihnen die Eingabe einer individuell erwünschten Spaltenlänge ermöglicht.

Da 20 eine geeignete Spaltenlänge für die Spalte 'nachname' ist, brauchen Sie den Vorschlag lediglich zu bestätigen:

▷ Drücken Sie ↵

Sie erhalten diesen Bildschirm:



Das ist geschehen:

- Die Markierung ist auf die nächste Spalte 'Index' gesprungen. Einen Index-Vorschlag macht INFORMIX nicht, da im allgemeinen nicht jede Spalte einer Tabelle indiziert wird.
- In der Funktionszeile haben Sie ein Menü erhalten, das Ihnen die optionale Einschaltung eines Index ermöglicht.

Einen Index soll die Spalte nicht erhalten:

▷ Bestätigen Sie mit ↵

Sie erhalten diesen Bildschirm:

```
NEU: NULL adressen :   Ja  Nein
Erlaubt 'NULL'-Werte fuer diese Spalte.
------ 1 ------------------------------------------------------ notizbuch --

 Spaltenname        Typ         Laenge                  Index   NULLs

 nachname           Char        20
```

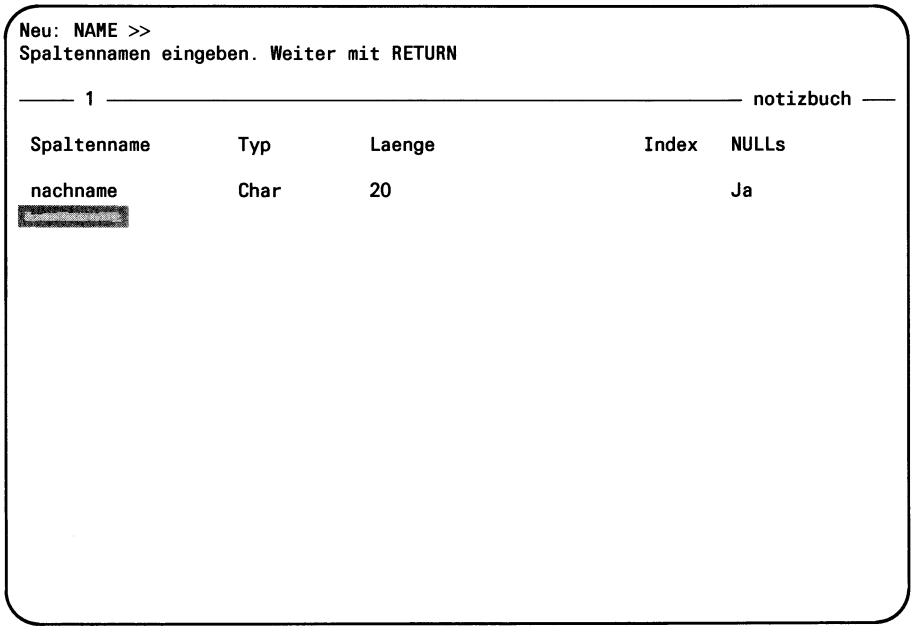
Das ist geschehen:

- Die Markierung ist auf die nächste Spalte NULLs gesprungen.
- In der Funktionszeile haben Sie ein Menü erhalten, das Ihnen die optionale 'Ja - Nein' Auswahl ermöglicht.

Für die Spalte 'nachname' sollen NULL-Werte zugelassen sein:

▷ Bestätigen Sie 'Ja' mit ↵

Sie erhalten diesen Bildschirm:



Die Markierung ist in die nächste Zeile zurück in die Spalte 'Spaltenname' gesprungen. Sie können mit der Eingabe der nächsten Spalte fortfahren:

▷ Geben Sie nach dem gleichen Muster nun auch die Spaltenmerkmale der Spalten 'vorname', 'strasse', 'plz' 'ort' und 'telefon' ein (Übersicht über die anzugebenden Spaltenmerkmale siehe oben).

Nun sollte Ihr Bildschirm so aussehen. Falls eventuelle Korrekturen erforderlich sind, lesen Sie den folgenden Abschnitt 3.3.8 mit den allgemeinen Bedienungshinweisen.

```
Neu: NAME >>
Spaltennamen eingeben. Weiter mit RETURN
------ 1 ------------------------------------------------------ notizbuch --

 Spaltenname        Typ         Laenge                  Index   NULLs

 nachname           Char        20                              Ja
 vorname            Char        20                              Ja
 strasse            Char        20                              Ja
 plz                Char        10                              Ja
 ort                Char        20                              Ja
 telefon            Char        15                              Ja
```

Alle notwendigen Eingaben zur Tabellendefinition sind erfolgt. Um zu beenden:

▷ Drücken Sie ↵

Sie erhalten wieder das TABELLE-ERSTELLEN-Menü:

```
TABELLE ERSTELLEN adressen :   Neu  Modifizieren  Loeschen  Seite  END
Fuegt eine Spalte oberhalb der aktivierten Zeile ein.

——— 1 ——————————————————————————————————————————— notizbuch ——

 Spaltenname        Typ          Laenge                  Index   NULLs

 nachname           Char         20                              Ja
 vorname            Char         20                              Ja
 strasse            Char         20                              Ja
 plz                Char         10                              Ja
 ort                Char         20                              Ja
 telefon            Char         15                              Ja
```

▷ Beenden Sie das Menü mit [END] bzw. durch Ausführen der Funktion END

Sie erhalten das END-Menü:

```
END adressen :   Tabelle-Neuerstellen-und-END  Verwerfen-und-END
Baut eine neue Tabelle auf / Struktur der Tabelle wird veraendert.

——— 1 ——————————————————————————————————————————— notizbuch ——

 Spaltenname        Typ         Laenge                  Index   NULLs

 nachname           Char        20                              Ja
 vorname            Char        20                              Ja
 strasse            Char        20                              Ja
 plz                Char        10                              Ja
 ort                Char        20                              Ja
 telefon            Char        15                              Ja
```

In diesem Menü entscheiden Sie, ob die Tabelle tatsächlich erstellt oder wieder verworfen werden soll.

- ▷ Führen Sie die Funktion 'Tabelle-Neuerstellen-und-END' aus. Sie sind in das TABELLE-Menü zurückgekehrt.
- ▷ Um in das Hauptmenü zurückzukehren, beenden Sie das Menü mit END.

Die Tabelle 'adressen' ist nun erstellt und Sie können mit dem weiteren Aufbau der Datenbank, der Eingabe der Datensätze, fortfahren.

Am einfachsten geht die Eingabe neuer Datensätze mit einem Format. Da für die Datenbank 'notizbuch' noch kein Format existiert, werden Sie im Folgenden zunächst dazu angeleitet, ein Standard-Format zu erzeugen.

3.3.8 Allgemeine Bedienungshinweise

Grundsätzliches zum Erstellen einer Tabelle

Sowohl die Funktionszeile als auch der Arbeitsbereich in Bildschirmmitte verfügen über Inversmarkierungen.

Die Funktionen in der Funktionszeile führen Sie wie folgt aus:
Anfangsbuchstaben (bzw. groß geschriebenen Buchstaben) eingeben oder mit Leertaste bzw. [ESC] (keine Pfeiltasten) markieren und [RETURN]
Beim Hinzufügen einer neuen Spalte positioniert INFORMIX die Inversmarkierung im Arbeitsbereich automatisch auf die logisch folgende Spalte. Beim nachträglichen Hinzufügen, Ändern oder Löschen können Sie die untere Inversmarkierung mit den Pfeiltasten [←] [→] [↑] [↓] [Bild↑] [Bild↓] nach Wunsch positionieren.

Der ganze Ablauf vollzieht sich menügeführt. Wo es angebracht erscheint, macht INFORMIX Vorschläge zu den einzelnen Spaltenmerkmalen. Individuelle Einstellungen sind jedoch jederzeit möglich. Dabei bietet INFORMIX in der Funktionszeile je nachdem entweder 'Eingabefelder' oder weitere 'Menüs' zur optionalen Festlegung der Spaltenmerkmale an.

Die Funktionen

Neu
ermöglicht das Hinzufügen einer neuen Spalte. Der ganze Ablauf erfolgt menügeführt. INFORMIX informiert Sie an jeder Stelle darüber, welche Spaltenmerkmale möglich bzw. sinnvoll sind.
Beim nachträglichen Hinzufügen einer neuen Spalte in eine bestehende Tabellendefinition gehen Sie wie folgt vor: Positionieren Sie zunächst mit den Pfeiltasten die Inversmarkierung auf die Zeile, vor der die neue Spalte hinzugefügt werden soll. Führen Sie danach die Funktion 'Neu' aus. Der weitere Ablauf vollzieht sich wie oben beschrieben.

Modifizieren
ermöglicht die nachträgliche Korrektur eines Spaltenmerkmales. Sie gehen wie folgt vor:
Positionieren Sie mit den Pfeiltasten die untere Markierung auf das Spalten-Definitionsmerkmal, das korrigiert werden soll.

Führen Sie danach die Funktion 'Modifizieren' aus. Der weitere Ablauf vollzieht sich genauso wie bei der Neudefinition einer Spalte.

Vorsicht

Führen Sie keine unbedachten Änderungen durch, die die Werte einer Spalte eventuell zerstören oder verstümmeln könnten. Wenn Sie beispielsweise die Spaltenlänge einer CHAR-Spalte von 20 auf 10 verkürzen, dann werden alle Werte mit mehr als 10 Zeichen um die überstehenden Zeichen abgeschnitten!

Loeschen

löscht eine Spalte. Sie gehen wie folgt vor:
Positionieren Sie die Inversmarkierung in die gewünschte Zeile. Führen Sie danach die Funktion 'Loeschen' aus.

Vorsicht

Mit dem Löschen einer Tabellenspalte werden auch sämtliche in ihr enthaltenen Werte gelöscht!

Seite

ermöglicht das Blättern zur nächsten Bildschirmseite. Die Funktion ist nur sinnvoll bei Tabellen mit mehreren Bildschirmseiten. Bei nur einer Bildschirmseite zeigt sie keine Wirkung.
Links außen auf der Infozeile zeigt INFORMIX die jeweils aktuelle Bildschirmseite an.

END
Beendet das Menü TABELLE ERSTELLEN und führt in das END-Menü (siehe unten 'Das Erstellen einer Tabelle beenden').

Das Erstellen einer Tabelle beenden

Wenn das Erstellen der Tabelle abgeschlossen ist, dann drücken Sie ↵. Danach erhalten Sie das TABELLE-ERSTELLEN-Menü. Dieses Menü beenden Sie mit END. Danach erhalten Sie das END-Menü mit den Funktionen 'Tabelle-Neuerstellen-und-END' und 'Verwerfen-und-END'. Um die Tabelle zu erstellen, führen Sie die Funktion 'Tabelle-Neuerstellen-und-END' aus. Im Anschluß daran erhalten Sie das TABELLE-Menü. Um in das Hauptmenü zurückzukehren, führen Sie die Funktion END aus.
VORSICHT: Die Funktion 'Verwerfen-und-END' in diesem Menü verwirft eine neue Tabellendefinition vollständig, Änderungen sind nicht mehr möglich; eine modifizierte Tabellendefinition verbleibt auf dem ursprünglichen Stand.

3.4 Ein Format erzeugen und verwenden

Der vorliegende und der folgende Abschnitt führen Sie in den Umgang mit Formaten ein. Ausführlichere Informationen zu Formaten erhalten Sie im Handbuch [2] INFORMIX-SQL Nachschlagen.

Prinzipiell erfolgt das Erstellen eines Formats in zwei Schritten:

- Zunächst ist ein Format-Programm zu schreiben, das die erforderlichen Anweisungen enthält.
- Dann ist das Format-Programm zu compilieren, wobei ein ablauffähiges Format entsteht.

Dieser Abschnitt erläutert Ihnen, wie Sie ein Standard-Format erzeugen. Dabei wird das Format-Programm im Rahmen der folgend beschriebenen Vorgehensweise automatisch erzeugt und auch compiliert. Sie müssen das nicht selbst tun.

3.4.1 Ein Standard-Format erzeugen

Die folgende Übung leitet Sie dazu an, ein Standard-Format für die Datenbank 'notizbuch' zu erzeugen. Im Anschluß an die Übung finden Sie eine Übersicht mit "allgemeinen Bedienungshinweisen" zur Erzeugung eines Standard-Formats.

Übung 3-4

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Ablauf, Modifizieren, Neuerstellen oder Loeschen eines Formates.

————————— notizbuch ————————————————————————————————————————
```

▷ Im Hauptmenü führen Sie die Funktion 'Format' aus.

Sie erhalten das FORMAT-Menü:

```
FORMAT:  Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Benutzen eines Formates zur Dateneingabe oder Datenbankabfrage.

————————— notizbuch ————————————————————————————————————————
```

Die Funktionen:

Ablauf
: ein bestehendes Format zur Datenbank-Abfrage und zum Neuaufnehmen, Löschen oder Korrigieren von Sätzen verwenden.

Modifizieren
: ein bestehendes Format-Programm unter Verwendung des Betriebssystem-Editors selbst schreiben.

Generieren
: ein neues Standard-Format automatisch erzeugen.

Neu
: ein neues Format-Pogramm unter Verwendung des Betriebssystem-Editors selbst schreiben.

Compilieren
: ein Format-Programm zu einem ablauffähigen Format compilieren (übersetzen).

Loeschen
: ein Format löschen.

END
: Rückkehr in das INFORMIX-Hauptmenü.

▷ Führen Sie die Funktion 'Generieren' aus.

▷ Drücken Sie ↵

Sie erhalten diesen Bildschirm:

```
GENERIEREN FORMAT >>
Namen des neuen Formates eingeben. Weiter mit RETURN

———————— notizbuch ————————————————————————————————————————
```

Sie sind nun aufgefordert, dem Format einen Namen zu geben:

▷ Geben Sie ein: format1

▷ Drücken Sie ↵

Sie erhalten diesen Bildschirm:

```
AUSWAHL TABELLE >>
Tabelle auswaehlen, die im Standard-Format enthalten sein soll.

———————— notizbuch ————————————————————————————————————————

 adressen
```

Sie sollen nun angeben, für welche Tabelle(n) das Format erstellt werden soll; der Name der Tabelle 'adressen' ist markiert.

▷ Drücken Sie ↵

Sie erhalten dieses Menü:

```
GENERIEREN FORMAT:   Fertig  Weitere  END
Tabellen-Auswahl fuer das Standard-Format fertig.

———————— notizbuch ————————————————————————————————————————
```

INFORMIX fragt Sie nun, ob Sie mit der Auswahl der gewünschten Tabellen 'Fertig' sind oder ob Sie für das Format noch 'Weitere' Tabellen (sofern vorhanden) wünschen. Die Datenbank 'notizbuch' enthält nur eine Tabelle, die Auswahl weiterer Tabellen ist folglich nicht möglich:

▷ Führen Sie die Funktion 'Fertig' aus.

Nun generiert INFORMIX das Standard-Format; nach kurzer Wartezeit werden Sie in das FORMAT-Menü zurückgeführt. Mit einer Meldung zeigt INFORMIX die erfolgreiche Übersetzung des Formates an:

```
FORMAT:  Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Benutzen eines Formates zur Dateneingabe oder Datenbankabfrage.

──────────── notizbuch ──────────────────────────────────────────────────────




 Das Format wurde erfolgreich uebersetzt.
```

Hinweis

Konkret ist dabei folgendes geschehen: Für Sie nicht sichtbar erzeugte INFORMIX ein Format-Programm, das alle erforderlichen Anweisungen enthält.
In einem zweiten Schritt wurde das Format-Programm compiliert (übersetzt) und damit ein ablauffähiges Format erzeugt.

Allgemeine Bedienungshinweise

Um ein Standard-Format zu erzeugen, unternehmen Sie die folgenden Schritte:

1. Im Hauptmenü führen Sie die Funktion 'Format' aus.
2. Im damit aufgerufenen Menü führen Sie die Funktion 'Generieren' aus.
3. Wenn noch keine Datenbank ausgewählt wurde, dann erhalten Sie jetzt einen Bildschirm, der der Auswahl der gewünschten Datenbank dient; ansonsten weiter mit 4.
4. Sie erhalten einen Bildschirm, der der Eingabe des gewünschten Format-Namens dient.
   NAMENSKONVENTIONEN: Der Name darf aus Groß- und Kleinbuchstaben, Zahlen und Unterstrichen bestehen, er darf jedoch nicht länger als 10 Zeichen sein.
5. Sie erhalten ein Menü, in dem Sie die für das Format erwünschte Tabelle auswählen.
6. Um Ihnen auch die optionale Auswahl weiterer Tabellen zu ermöglichen, die für das Format berücksichtigt werden sollen, erhalten Sie nun ein Menü, mit den Funktionen 'Weitere' (Tabellen für das Format) und 'Fertig' (mit der Auswahl).
7. Die Funktion 'Fertig' leitet die automatische Erzeugung des Standard-Formates ein. Nach erfolgreicher Ausführung werden Sie in das FORMAT-Menü zurückgeführt.

3.4.2 Format aufrufen

Dieser Abschnitt erläutert Ihnen anhand einer Übung, gefolgt von allgemeinen Hinweisen, wie man ein gewünschtes Format aufruft.

Übung 3-5

```
INFORMIX-SQL:  Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Ablauf, Modifizieren, Neuerstellen oder Loeschen eines Formates.

————————— notizbuch ————————————————————————————————————————————
```

▷ Im Hauptmenü führen Sie die Funktion 'Format' aus.

Sie erhalten das FORMAT-Menü:

```
FORMAT:  Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Benutzen eines Formates zur Dateneingabe oder Datenbankabfrage.

————————— notizbuch ————————————————————————————————————————————
```

▷ Führen Sie die Funktion 'Ablauf' aus.

Sie erhalten diesen Bildschirm, der der Auswahl des gewünschten Formates dient - der Name des Formates 'format1' ist markiert:

```
ABLAUF FORMAT >>
Format auswaehlen oder Namen eingeben. Weiter mit RETURN

————————— notizbuch ————————————————————————————————————————————

format1
```

▷ Um das markierte 'format1' auszuwählen, drücken Sie ↵

Nach kurzer Wartezeit erhalten Sie den PERFORM-Bildschirm mit dem Format 'format1' (siehe den folgenden Abschnitt). PERFORM ist der Name derjenigen INFORMIX-Komponente, die Ihnen das Arbeiten mit einem Format ermöglicht.

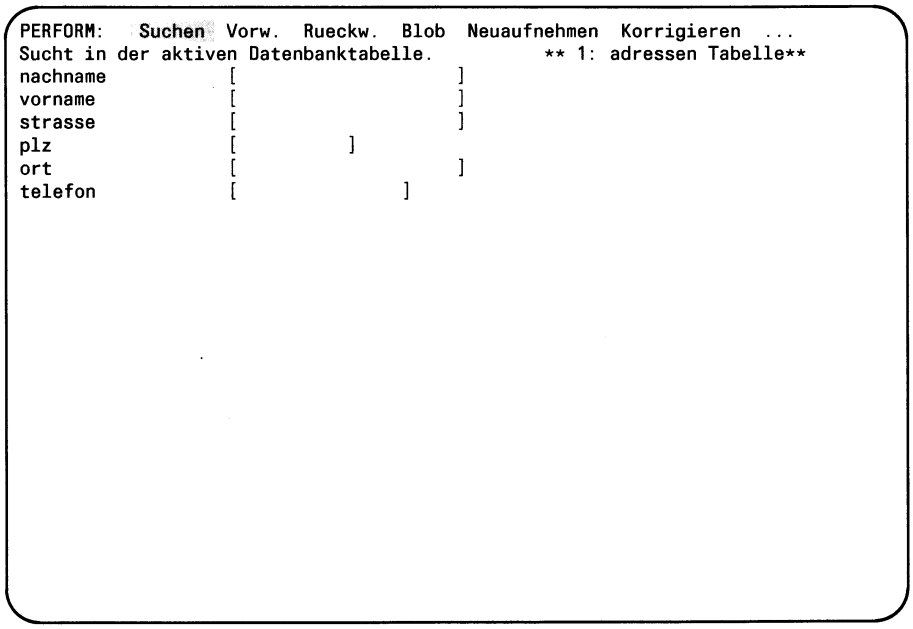
Allgemeine Bedienungshinweise

Um ein gewünschtes Format aufzurufen, unternehmen Sie die folgenden Schritte:

1. Im Hauptmenü führen Sie die Funktion 'Format' aus.
2. Im damit aufgerufenen Menü führen Sie die Funktion 'Ablauf' aus.
3. Sie erhalten einen Bildschirm, der der Auswahl des gewünschten Formates dient.
4. Nach kurzer Wartezeit erhalten Sie den PERFORM-Bildschirm mit dem ausgewählten Format.

3.4.3 Der PERFORM-Bildschirm

Dieser Abschnitt erläutert Ihnen anhand des Formates 'format1' die Besonderheiten des PERFORM-Bildschirmes:



Funktionszeile

Das PERFORM-Menü verfügt über eine zweite, zunächst nicht sichtbare Funktionszeile, da es mehr Funktionen zur Verfügung stellt, als eine einzige Funktionszeile anzeigen kann. Zunächst einmal erhalten Sie die erste Funktionszeile mit den am häufigsten verwendeten Funktionen:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
```

Die meisten dieser Funktionen in dieser Funktionszeile werden Sie in den folgenden Übungen kennenlernen. Sie bedeuten:

Suchen
: ermöglicht die Ausführung einer Datenbank-Abfrage,

Vorw.
: zeigt von den gefundenen Sätzen den jeweils folgenden an,

Rueckw.
: zeigt von den gefundenen Sätzen den jeweils vorherigen an,

Blob
: veranlaßt die Schreibmarke, in das erste BLOB-Bildschirmfeld zu springen. Dies ist dann erforderlich, wenn Sie eine Suche ausgeführt haben und sich die Daten der Blob-Spalte anschauen wollen. Mehr hierzu in Kapitel 5.

Neuaufnehmen
: ermöglicht die Neuaufnahme eines Satzes in die Tabelle,

Korrigieren
: ermöglicht die Korrektur des aktuellen Satzes,

...
: Pseudofunktion; zeigt an, daß noch eine weitere Funktionszeile existiert.

Die nächste Funktionszeile sieht wie folgt aus:

```
PERFORM:   ...  Loeschen  Tabelle  Format  Aktuell  Master  Detail  PRINT  END
```

Von den hier angezeigten Funktionen behandelt dieser Abschnitt nur die Funktion 'Löschen'. Ausführliche Informationen zu den weiteren Funktionen erhalten Sie im Kapitel 5. Hier nur eine kurze Übersicht:

...
: Pseudofunktion zum Blättern zur vorherigen Funktionszeile.

Loeschen
: ermöglicht das Löschen des aktuellen Satzes,

Tabelle
: wechselt die aktuelle Tabelle; zu verwenden bei Formaten mit mehreren Tabellen.

Format
: zeigt die jeweils folgende Bildschirmseite eines Formates; zu verwenden bei Formaten mit mehr als einer Bildschirmseite.

Aktuell
: liest den jeweils aktuellen Datensatz der aktuellen Tabelle erneut ein.

Master
: wechselt zur Master-Tabelle; zu verwenden bei Mehr-Tabellen-Formaten, für die Master-Detail-Beziehungen definiert sind.

Detail
: wechselt zur Detail-Tabelle.

PRINT
: ermöglicht die Ausgabe des gefunden Satzes (der Sätze) in eine Datei.

END
: führt zurück in das Hauptmenü.

Hinweis

Um den PERFORM-Bildschirm zu verlassen, drücken Sie [END] oder Sie geben 'e' als den Anfangsbuchstaben der Funktion END ein.
Die Funktion END wird zwar erst in der zweiten Funktionszeile angeboten, sie ist aber auch von der ersten Funktionszeile aus ausführbar.

Der Arbeitsbereich

Im Arbeitsbereich unterhalb des Menüs zeigt INFORMIX das eigentliche Format an:

```
nachname        [                    ]
vorname         [                    ]
strasse         [                    ]
plz             [          ]
ort             [                    ]
telefon         [               ]
```

Ein Standard-Format besteht aus den beiden folgenden Elementen:

- 'Spaltennamen' und
- 'Bildschirmfeldern', angezeigt durch die [ ] eckigen Klammern.

Die Spaltennamen zeigen an, zu welcher Tabellenspalte ein Bildschirmfeld gehört. Die Spaltennamen dienen lediglich der Information, sie haben keine Bedeutung für die Funktion des Formates.

Die Bildschirmfelder dienen der Eingabe und der Anzeige von Sätzen:

Wenn Sie beispielsweise einen neuen Satz in die Datenbank aufnehmen, dann machen Sie Ihre Eingaben in eben diese Bildschirmfelder. Oder umgekehrt, wenn Sie die Datenbank abfragen, dann zeigt Ihnen INFORMIX in den Bildschirmfeldern den ersten gefundenen Satz an.

Wenn Sie die Funktionen 'Suchen', 'Neuaufnehmen', 'Korrigieren' oder 'Loeschen' ausführen, dann springt die Schreibmarke in den Arbeitsbereich und Sie können die Bildschirmfelder editieren, d.h., Sie können hier Texteingaben machen, die Schreibmarke bewegen und eventuelle Text-Korrekturen vornehmen.

So können Sie beispielsweise mit den Pfeiltasten die Schreibmarke innerhalb eines Bildschirmfeldes bzw. in ein gewünschtes Bildschirmfeld bewegen und mit ⌫ und CHAR DELETE einzelne Zeichen löschen. Ausführliche Informationen zu den Editierfunktionen erhalten Sie in Kapitel 5.

3.4.4 Einen Satz 'Neuaufnehmen'

Die Funktion 'Neuaufnehmen' ermöglicht die Neuaufnahme eines Satzes in die Tabelle. Die folgende Übung erläutert Ihnen anhand der Neuaufnahme einiger Sätze in die Tabelle 'adressen', wie Sie vorgehen müssen. Weiter unten informieren Sie "allgemeine Bedienungshinweise" über die Funktionen 'Neuaufnehmen', 'Suchen', 'Vorw.' 'Rueckw.', 'Korrigieren' und 'Loeschen'.

Die folgenden Sätze können Sie in der anschließenden Übung in die Tabelle 'adressen' aufnehmen. Nehmen Sie bitte keine anderen als die hier gezeigten Sätze auf, da Sie sonst die darauffolgenden Übungen nicht mehr nachvollziehen können:

nachname	vorname	strasse	plz	ort	telefon
Altmann	Marlies	Goethestr. 4	6900	Heidelberg	06221/34127
Bauer	Richard	Postweg 56	7900	Ulm	0731/97654
Dornbeck	Herta	Holbeinstr. 12	5300	Bonn 2	0228/456789
Sutterer	Albert	Gartenstr. 98	1000	Berlin 42	030/987643
Wolf	Martin	Waldweg 17	7900	Ulm	0731/23456

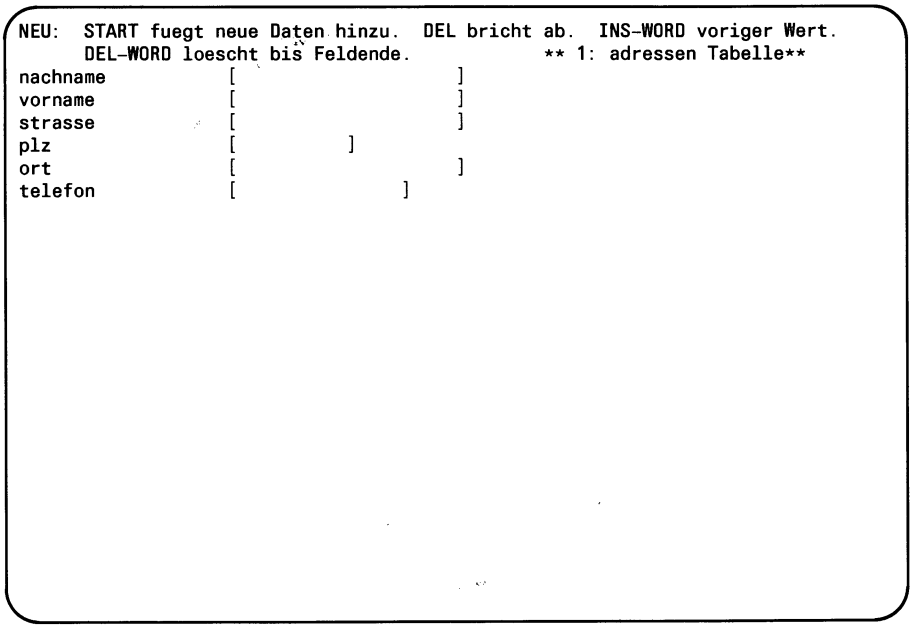
Übung 3-6

Sie sollten diesen Bildschirm vor sich haben:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Nimmt einen neuen Satz in die aktive Tabelle auf** 1: adressen Tabelle**
nachname          [                    ]
vorname           [                    ]
strasse           [                    ]
plz               [          ]
ort               [                    ]
telefon           [               ]
```

▷ Führen Sie die Funktion 'Neuaufnehmen' aus.

Sie erhalten diesen Bildschirm:



Das ist geschehen:

- Sie haben eine neue Funktionszeile erhalten, die Sie auf die Bedeutung der Funktionstasten START, DEL, WORD INSERT, WORD DELETE hinweist.
- Die Schreibmarke befindet sich im Arbeitsbereich im Bildschirmfeld 'nachname'.

Sie können jetzt den Nachnamen des ersten Datensatzes eingeben:

▷ Geben Sie ein: Altmann Eventuelle Schreibfehler können Sie mit ⌫, CHAR DELETE oder WORD DELETE korrigieren.

▷ Um die Schreibmarke in das nächste Bildschirmfeld 'vorname' zu positionieren, drücken Sie ↵

▷ Geben Sie den Vornamen ein: Marlies

▷ Geben Sie nach dem gleichen Muster nun auch die Werte für 'strasse,' 'plz', 'ort' und 'telefon' ein.

Ihr Bildschirm sollte jetzt so aussehen:

```
NEU:  START fuegt neue Daten hinzu.  DEL bricht ab.  INS-WORD voriger Wert.
      DEL-WORD loescht bis Feldende.              ** 1: adressen Tabelle**
nachname          [Altmann            ]
vorname           [Marlies            ]
strasse           [Goethestr. 4       ]
plz               [6900      ]
ort               [Heidelberg         ]
telefon           [06221/34127    ]
```

Um den angezeigten Satz von 'Marlies Altmann' in die Datenbank aufzunehmen, gehen Sie wie folgt vor:

▷ Drücken Sie START

Sie haben wieder den Bildschirm mit dem PERFORM-Menü erhalten:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Nimmt einen neuen Satz in die aktive Tabelle auf** 1: adressen Tabelle**
nachname          [Altmann             ]
vorname           [Marlies             ]
strasse           [Goethestr. 4        ]
plz               [6900      ]
ort               [Heidelberg          ]
telefon           [06221/34127     ]









 Der Satz wurde neu aufgenommen
```

Zwei Dinge sind geschehen:

- Am unteren Bildschirmrand haben Sie eine Meldung darüber erhalten, daß der Satz neu aufgenommen wurde.
- Im Funktionsmenü ist die Funktion 'Neuaufnehmen' markiert.

▷ Geben Sie nun nach dem gleichen Muster auch die vier weiteren Sätze ein.

Die Datenbank enthält nun fünf Sätze; der folgende Abschnitt erläutert Ihnen, wie Sie die Datenbank abfragen und wie Sie sich die gefundenen Sätze am Bildschirm anzeigen lassen können.

3.4.5 'Suchen' - die Datenbank abfragen

Die Funktion 'Suchen' ermöglicht die Durchführung einer Datenbank-Abfrage.

Der reine Bedienungsvorgang gestaltet sich sehr einfach, Sie müssen nur

- die Funktion 'Suchen' ausführen
  (eventuelle Suchbedingungen eingeben - siehe unten) und
- anschließend START drücken.

Danach führt INFORMIX die Datenbank-Abfrage durch und zeigt Ihnen den ersten gefundenen Satz (Treffersatz) in dem Format an.
Mit den Funktionen 'Vorw.' und 'Rueckw.' können Sie sich den jeweils folgenden bzw. den jeweils vorherigen Treffersatz in dem Format anzeigen lassen.

Da Datenbank-Abfragen aber sehr viele Möglichkeiten bieten, soll dieser an der Oberfläche einfache Vorgang hier eingehender beleuchtet werden:

Abläufe bei einer Datenbank-Abfrage

Wenn Sie die Funktion 'Suchen' ausführen und damit eine Datenbank-Abfrage einleiten, dann geschieht zunächst nichts weiter, als daß die Schreibmarke in den Arbeitsbereich (in das erste Bildschirmfeld) springt. INFORMIX eröffnet Ihnen damit die Möglichkeit zur Eingabe sogenannter 'Suchbedingungen'.

Suchbedingung

Mit der Eingabe einer Suchbedingung können Sie festlegen, welche Sätze die Datenbank-Abfrage liefern soll (Satzauswahl).

Beispiel

Eine Datenbank-Abfrage soll lediglich Sätze von Personen liefern, die in Ulm leben. In diesem Fall können Sie in das Bildschirmfeld 'ort' den Wert 'Ulm' eingeben. Nicht sinnvoll ist die Eingabe von 'ulm', denn bei Werten unterscheidet INFORMIX zwischen Groß - und Kleinschreibung.
Der Wert 'Ulm' dient als Suchbedingung - wenn Sie danach START drücken und damit die Datenbank-Abfrage ausführen, dann haben Sie an INFORMIX den Auftrag erteilt, lediglich nach denjenigen Sätzen zu suchen, die in der Spalte 'ort' den Wert 'Ulm' enthalten.
Das gleiche Ergebnis können Sie durch die Eingabe 'U*' erzielen. 'U*' findet alle Sätze, deren Wohnorte mit 'U' beginnen. Der Stern '*' ist ein Suchoperator, der als 'Joker-Zeichen' stellvertretend steht für: ein beliebiges Zeichen, eine Folge von beliebigen Zeichen oder auch kein Zeichen. Auf Suchoperatoren geht das Kapitel 5 ein.

Wenn Sie aber umgekehrt keine Suchbedingungen eingeben, dann ist das für INFORMIX gleichbedeutend damit, daß die Datenbank-Abfrage alle Sätze der Tabelle liefern soll.

Sie haben also zwei Möglichkeiten, eine Datenbank-Abfrage durchzuführen:

- Entweder Sie geben eine (oder mehrere) Suchbedingung(en) ein, in diesem Fall liefert die Datenbank-Abfrage nur diejenigen Sätze, die in der entsprechenden Spalte einen Wert enthalten, der die Suchbedingung erfüllt (Satzauswahl).
- Oder Sie verzichten auf die Eingabe einer Suchbedingung, in diesem Fall liefert die Datenbank-Abfrage alle Sätze der Tabelle.

Die aktuelle Liste

Jede Datenbank-Abfrage, sei es mit einem Format oder mit SQL, führt zu einer Ergebnistabelle mit Treffersätzen. Bei Formaten nennt man diese Ergebnistabelle die 'aktuelle Liste'.

Die aktuelle Liste hat die Bedeutung eines Zwischenspeichers, der sämtliche Treffersätze enthält.
Notwendig wird die aktuelle Liste dadurch, daß ein Format von allen vorhandenen Treffersätzen immer nur einen einzigen Satz anzeigen kann. Während Sie sich einen Satz im Format anzeigen lassen, bleiben die weiteren Treffersätze in der aktuellen Liste auf Abruf verfügbar.

Der aktuelle Satz

Wenn Sie mit START eine Datenbank-Abfrage starten, dann vollzieht sich der folgende Ablauf:

- INFORMIX überprüft, ob und welche Suchbedingungen Sie in die Bildschirmfelder eingegeben haben.
- Abhängig von den Suchbedingungen erstellt INFORMIX die aktuelle Liste mit den Treffersätzen.
- Den ersten Satz der aktuellen Liste zeigt INFORMIX in dem Format an. Diesen Satz bezeichnet man als den 'aktuellen Satz'.

Mit den Funktionen 'Vorw.' und 'Rueckw.' können Sie in der aktuellen Liste blättern, d.h. Sie können sich den jeweils folgenden oder den jeweils vorherigen Satz der aktuellen Liste in dem Format anzeigen lassen.
Der jeweils angezeigte Satz ist der neue aktuelle Satz.

In der folgenden Übung führen Sie zwei Datenbank-Abfragen durch. Eine Datenbank-Abfrage ohne Suchbedingung und eine Datenbank-Abfrage mit Suchbedingung.

Übung 3-7

Um eine Datenbank ohne Suchbedingung auszuführen, unternehmen Sie die folgenden Schritte:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: adressen Tabelle**
nachname          [                    ]
vorname           [                    ]
strasse           [                    ]
plz               [          ]
ort               [                    ]
telefon           [               ]
```

▷ Führen Sie die Funktion 'Suchen' aus.

Sie erhalten diesen Bildschirm:

```
SUCHEN:  START sucht.  F17 loescht alle Felder.  HELP fuer Hilfe.
         DEL bricht ab.                          ** 1: adressen Tabelle**
nachname          [                    ]
vorname           [                    ]
strasse           [                    ]
plz               [          ]
ort               [                    ]
telefon           [               ]
```

Drei Dinge sind geschehen:

- Falls bisher in dem Format ein Satz angezeigt wurde, dann wurde er jetzt vom Bildschirm gelöscht.
- Sie haben eine neue Funktionszeile (und Kommentarzeile) erhalten. Hier wird angezeigt, welche Tasten in diesem Zusammenhang zu verwenden sind:
  START führt die Datenbank-Abfrage aus, DEL bricht ab, HELP führt zu einem Bildschirm mit Hilfe-Texten und F17 löscht alle Bildschirmfelder.
- Die Schreibmarke ist in das erste Bildschirmfeld gesprungen. INFORMIX eröffnet Ihnen damit die Möglichkeit zur Eingabe von Suchbedingungen. Gebrauch von dieser Möglichkeit machen Sie jedoch erst später im zweiten Teil der Übung.

▷ Um die Datenbank-Abfrage auszuführen, drücken Sie START

Sie erhalten diesen Bildschirm:

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**
nachname          [Altmann             ]
vorname           [Marlies             ]
strasse           [Goethestr. 4        ]
plz               [6900      ]
ort               [Heidelberg          ]
telefon           [06221/34127     ]









 Gefundene Saetze: 5
```

Das ist geschehen:

- Sie haben wieder das PERFORM-Menü erhalten; die Funktion 'Suchen' ist markiert.
- Am unteren Bildschirmrand hat INFORMIX diese Meldung ausgegeben:
  Gefundene Saetze: 5
  Das bedeutet, daß die aktuelle Liste 5 Sätze enthält. Und das sind eben diejenigen 5 Sätze, die die Tabelle 'adressen' insgesamt enthält. Die Datenbank-Abfrage ohne Suchbedingungen hat also alle Sätze der Tabelle geliefert.
- In dem Format zeigt INFORMIX den aktuellen Satz, den Satz von 'Marlies Altmann' an. Das ist der erste Satz der aktuellen Liste.

Lassen Sie sich auch die anderen Sätze der aktuellen Liste zeigen:

▷ Führen Sie die Funktion 'Vorw.' aus.

Jetzt ist der Satz von 'Richard Bauer' der aktuelle Satz.

Mit der Funktion 'Vorw.' können Sie also in der aktuellen Liste vorwärts blättern und sich den jeweiligen Satz als den neuen aktuellen Satz am Bildschirm anzeigen lassen.
Wenn Sie am Ende der aktuellen Liste angekommen sind, gibt INFORMIX eine Signalton und eine Meldung aus.
Mit der Funktion 'Rueckw.' können Sie in der umgekehrten Reihenfolge in der aktuellen Liste blättern.

Der folgende Teil der Übung erläutert Ihnen, wie Sie eine Datenbank-Abfrage mit einer Suchbedingung durchführen:

▷ Führen Sie wieder die Funktion 'Suchen' aus.

Der zuletzt angezeigte Satz wurde vom Bildschirm gelöscht und die Schreibmarke ist in den Arbeitsbereich in das erste Bildschirmfeld gesprungen.

Die Übung möchte Sie dazu anleiten, nach den Sätzen derjenigen Personen zu suchen, die in Ulm wohnen, die Such-Bedingung lautet also 'Ulm':

▷ Positionieren Sie mit [↵] die Schreibmarke in das Bildschirmfeld 'ort'

▷ Geben Sie hier ein: Ulm

▷ Drücken Sie [START]

Jetzt erhalten Sie eine Meldung darüber, daß nur zwei Sätze gefunden wurden.
Der Wert 'Ulm' im Bildschirmfeld 'ort' diente als Suchbedingung für die Datenbank-Abfrage - nur zwei Personen wohnen in Ulm und deshalb enthält die aktuelle Liste auch nur zwei Sätze.

Das gleiche Ergebnis können Sie auch durch Eingabe der Postleitzahl '7900' (der Postleitzahl von 'Ulm') als Suchbedingung erzielen:

▷ Führen Sie nochmals die Funktion 'Suchen' aus.

▷ Positionieren Sie die Schreibmarke in das Bildschirmfeld 'plz' und geben Sie hier ein: 7900

▷ Drücken Sie START

Wieder enthält die aktuelle Liste nur zwei Sätze, nämlich diejenigen Sätze, die im Bildschirmfeld 'plz' den Wert '7900' enthalten.

Sie erhalten hier nun erstmals eine Reihe von Aufgaben, die Sie ausführen können, wenn Sie Ihr Wissen überprüfen wollen. Die Lösungen zu den Aufgaben finden Sie im Anhang.

Aufgabe 3-1

Führen Sie nun nacheinander noch mehrere Datenbank-Abfragen mit den folgenden Suchbedingungen durch:

A) vorname 'Herta'

B) strasse 'Waldweg 17'

C) plz '5300'

3.4.6 Einen Satz 'Korrigieren'

Die Funktion 'Korrigieren' ermöglicht die Korrektur eines Satzes. Korrigieren können Sie nur den aktuellen Satz, d.h., Sie müssen vorab eine Datenbank-Abfrage durchführen, die Ihnen genau den gewünschten Satz als den neuen aktuellen Satz am Bildschirm anzeigt.

Die folgende Übung leitet Sie zur Korrektur des Satzes von 'Richard Bauer' an.
Geändert werden soll die Straße von Postweg 56 auf Klenzestr. 29:

Übung 3-8

▷ Führen Sie eine Datenbank-Abfrage durch, die Ihnen als Ergebnis den Satz von 'Richard Bauer' anzeigt (z.B. durch die Suchbedingung 'Richard' im Bildschirmfeld 'vorname').

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: adressen Tabelle**
nachname          [Bauer              ]
vorname           [Richard            ]
strasse           [Postweg 56         ]
plz               [7900      ]
ort               [Ulm                ]
telefon           [0731/97654     ]
```

▷ Führen Sie die Funktion 'Korrigieren' aus.

▷ Positionieren Sie die Schreibmarke in das Bildschirmfeld 'strasse'.

Jetzt können Sie den angezeigten Wert von Postweg 56 in Klenzestr. 29 korrigieren. Sie haben dazu mehrere Möglichkeiten:

- entweder Sie überschreiben den Wert einfach mit dem neuen Wert,
- oder Sie löschen vorab den Wert zeichenweise mit [CHAR DELETE] oder [⌫],
- oder Sie löschen vorab den gesamten Wert mit [WORD DELETE]; die Schreibmarke muß sich am Anfang des Wertes befinden, denn [WORD DELETE] löscht das Bildschirmfeld ab der Schreibmarkenposition.

▷ Verwenden Sie eine der genannten Möglichkeiten, um die Korrektur vorzunehmen.

▷ Drücken Sie danach [START], um den korrigierten Satz in die Tabelle rückschreiben zu lassen.

INFORMIX gibt anschließend eine Meldung über die erfolgte Korrektur des Satzes aus.

3.4.7 Einen Satz 'Löschen'

Mit der Funktion 'Loeschen' können Sie einen Satz aus der Tabelle löschen.
Löschen können Sie nur den aktuellen Satz, d.h., Sie müssen vorab eine Datenbank-Abfrage durchführen, die Ihnen genau den gewünschten Satz am Bildschirm als den aktuellen Satz anzeigt (wie bei der Funktion 'Korrigieren').

Die folgende Übung leitet Sie dazu an, den Satz von 'Albert Sutterer' zu löschen:

Übung 3-9

▷ Führen Sie eine Datenbank-Abfrage durch, die Ihnen als Ergebnis den Satz von 'Albert Sutterer' anzeigt.

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: adressen Tabelle**
nachname          [Sutterer           ]
vorname           [Albert             ]
strasse           [Gartenstr. 98      ]
plz               [1000      ]
ort               [Berlin 42          ]
telefon           [030/987643     ]









 Gefundene Saetze: 1
```

▷ Führen Sie die Funktion 'Loeschen' aus (die Funktion befindet sich in der zweiten Funktionszeile).

Sie erhalten diesen Bildschirm:

```
LOESCHEN:  Ja  Nein
Loescht den gesamten Satz aus der aktiven Tabell** 1: adressen Tabelle**
nachname            [Sutterer            ]
vorname             [Albert              ]
strasse             [Gartenstr. 98       ]
plz                 [1000      ]
ort                 [Berlin 42           ]
telefon             [030/987643     ]
```

INFORMIX will Ihnen hier Gelegenheit geben, eine versehentliche Fehlbedienung zu korrigieren - die Funktion LOESCHEN 'Ja' ist markiert.

▷ Führen Sie die Funktion 'Ja' aus.

Sie haben wieder das PERFORM-Menü erhalten; eine Meldung informiert Sie die erfolgte Löschung des Satzes.

▷ Um in das Hauptmenü zurückzukehren, führen Sie die Funktion END aus. Die Funktion wird zwar erst in der zweiten PERFORM-Menü-Funktionszeile angezeigt, sie ist aber auch von der ersten Funktionszeile aus ausführbar.

Sie sind in das Hauptmenü zurückgekehrt.

Allgemeine Bedienungshinweise

Die folgenden Hinweise erläutern Ihnen anhand der Funktionen des PERFORM-Menüs, welche Schritte Sie unternehmen müssen, um eine Datenbank abzufragen, einen Satz neu aufzunehmen, zu korrigieren oder zu löschen.

Suchen
: ermöglicht die Durchführung einer Datenbank-Abfrage. Sie erhalten eine neue Funktionszeile (und Kommentarzeile), die Sie auf die Bedeutung der Tasten START (führt aus) und DEL (bricht ab) sowie HELP und F17 hinweist.
Die Schreibmarke springt in den Arbeitsbereich, um Ihnen die optionale Eingabe einer (oder mehrerer) Suchbedingungen zu ermöglichen.
START führt die Datenbank-Abfrage aus.
Die Datenbank-Abfrage führt zu einer aktuellen Liste, von der der erste Satz als der aktuelle Satz am Bildschirm angezeigt wird.

Vorw.
: vorwärtsblättern in der aktuellen Liste. Der jeweils nächste Satz wird als der neue aktuelle Satz am Bildschirm angezeigt.

Rueckw.
: rückwärtsblättern in der aktuellen Liste. Der jeweils vorherige Satz wird als der neue aktuelle Satz am Bildschirm angezeigt.

Neuaufnehmen
: ermöglicht die Neuaufnahme eines Satzes in die Tabelle. Sie erhalten eine neue Funktionszeile (und Kommentarzeile), die Sie auf die Bedeutung der Tasten START (führt aus) und DEL (bricht ab) sowie WORD INSERT und WORD DELETE hinweist.
Die Schreibmarke springt in den Arbeitsbereich; Sie können die Werte in die Bildschirmfelder eintragen.
Um die Schreibmarke in das nächste Bildschirmfeld zu positionieren, verwenden Sie ↵
START führt die Neuaufnahme aus.

Korrigieren
ermöglicht die Korrektur des aktuellen Satzes. Der Korrektur muß eine Datenbank-Abfrage vorausgehen, die den zu korrigierenden Satz zum aktuellen Satz macht.
Um die Schreibmarke in eine gewünschtes Bildschirmfeld zu positionieren, verwenden Sie [↵]
Zeichenweise löschen können Sie mit [⌫] oder [CHAR DELETE]; löschen ab Schreibmarkenposition können Sie mit [WORD DELETE]

Loeschen
ermöglicht das Löschen des aktuellen Satzes. Dem Vorgang muß eine Datenbank-Abfrage vorausgehen, die den zu löschenden Satz zum aktuellen Satz macht.
Um eventuelle Fehlbedienung korrigieren zu können, erhalten Sie anschließend ein LOESCHEN 'Ja Nein' Menü. 'Ja' löscht den Satz aus der Tabelle.

END
beendet das PERFORM-Menü und führt zurück zum Hauptmenü. Diese Funktion wird erst auf der zweiten Funktionszeile angezeigt, sie ist aber auch von der ersten Funktionszeile aus ausführbar.

3.5 Arbeiten mit SQL

SQL ist eine Datenbank-Sprache, mit der Sie unter anderem

- eine Datenbank abfragen können,
- eine Datenbank und eine Tabelle erzeugen können,
- Sätze einer Datenbank ändern, löschen oder korrigieren können.

SQL besteht aus einer Reihe von Anweisungen, die der englischen Sprache angelehnt sind. Sie können SQL ähnlich verwenden wie die geschriebene englische Sprache; Sie müssen dabei jedoch die speziellen SQL-Syntaxregeln beachten.

Der vorliegende Abschnitt führt Sie in den Umgang mit SQL ein. Sie erfahren, wie Sie mit SQL eine Datenbank abfragen können. Ausführliche Informationen über Sprachumfang und Syntax erhalten Sie im Kapitel 6 und im Handbuch [2] INFORMIX nachschlagen.

Eine SQL-Anweisung ausführen

Das SQL-Dialog-Menü bietet Ihnen u.a. die Funktionen 'Neu' und 'Korrigieren' an; diese Funktionen führen zu einem Bildschirm, in dem Sie im Arbeitsbereich einen Editor erhalten. Ein Editor dient ganz allgemein zur Eingabe eines Textes; in diesem speziellen Fall dient er zur Eingabe Ihrer SQL-Anweisung.

Nachdem Sie Ihre SQL-Anweisung in den Editor eingegeben haben, können Sie diese mit START ausführen lassen. Dann geschieht folgendes:

- INFORMIX liest die SQL-Anweisung,
- interpretiert ihren Inhalt und
- führt die Anweisung aus.

Wenn Ihre Anweisung eine Anweisung zur Abfrage einer Datenbank ist, dann erhalten Sie am Bildschirm eine Ergebnistabelle mit den Treffersätzen.

3.5.1 Unterschiede zwischen einem Format und einer SQL-Anweisung

Die Datenbank abfragen, Sätze ändern, löschen oder korrigieren können Sie auch mit einem Format, jedoch sind Sie bei einem Format an bestimmte Vorgaben gebunden. So können Sie z.B. bei einem Format zur gleichen Zeit immer nur einen einzigen Satz ändern, löschen oder anzeigen lassen und Sie haben keinen Einfluß auf die Auswahl der Spalten; denn Auswahl und Anzahl der Bildschirmfelder ist fester Bestandteil des Formates.
Den Dialog, den Sie in einem Format mit einer Datenbank führen können, bezeichnet man unter anderem deshalb als 'formatgeführten Dialog'.

In der Unterscheidung dazu bezeichnet man einen Dialog, den Sie mit SQL führen, als 'interaktiven-' oder 'freien Dialog'. Freier Dialog u.a. deshalb, weil Sie bei einer SQL-Anweisung an keine Vorgaben gebunden sind und die erwünschte Spaltenauswahl und Satzauswahl in der Anweisung frei formulieren können.
In den folgenden Abschnitten werden Sie einige Beispiele kennenlernen.

3.5.2 Das SQL-Dialog-Menü aufrufen

Die folgende Übung erläutert Ihnen, wie Sie das SQL-Dialog-Menü aufrufen.

Übung 3-10

```
INFORMIX-SQL:   Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Fuehren eines interaktiven Dialoges mit SQL Anweisungen.

──────────── notizbuch ──────────────────────────────────────────────────
```

▷ Im Hauptmenü führen Sie die Funktion 'SQL-Dialog' aus.

Danach erhalten Sie das SQL-Dialog-Menü.

Allgemeine Bedienungshinweise

Um das SQL-Dialog-Menü aufzurufen, unternehmen Sie die folgenden Schritte:

- Im Hauptmenü führen Sie die Funktion 'SQL-Dialog' aus.
- Wenn bisher noch keine Datenbank ausgewählt wurde, erhalten Sie jetzt den Bildschirm AUSWAHL DATENBANK, der die Auswahl der gewünschten Datenbank ermöglicht (mit den Pfeiltasten markieren und ↵ bzw. den Namen in das Eingabefeld eingeben und ↵)
- Sie erhalten das SQL-Dialog-Menü.

3.5.3 Das SQL-Dialog-Menü

```
SQL-Dialog:  Neu  START  Korrigieren  Ruf-Editor  PRINT  Info  Datei  END
Neue Dialoganweisung eingeben.

————————— notizbuch ————————————————————————————————————————————
```

Das SQL-Dialog-Menü bietet die folgenden Funktionen an:

Neu
: ermöglicht die Eingabe einer neuen SQL-Anweisung und führt zu diesem Zweck in das 'NEU-Menü'.

START
: führt die SQL-Anweisung aus. Wenn die SQL-Anweisung eine Anweisung zu einer Datenbank-Abfrage ist, dann erhalten Sie am Bildschirm eine Ergebnistabelle mit den Treffersätzen.
Sie können die Funktion START ausführen oder auch die Taste START drücken.

Korrigieren
: ermöglicht die Korrektur einer fehlerhaft eingegebenen SQL-Anweisung und führt in das 'KORR.-Menü'; dieses Menü ist bis auf den Menünamen identisch mit dem 'NEU-Menü'.

Ruf-Editor
: ruft vorübergehend den eingestellten Betriebssystem-Editor auf. Der Ruf-Editor bietet mehr Editierfunktionen als der standardmäßige SQL-Editor. Wenn Sie den Ruf-Editor wieder verlassen, kehren Sie automatisch in das SQL-Dialog-Menü zurück.

PRINT
: ermöglicht die Ausführung und Weiterleitung der aktuellen SQL-Anweisung an das Druckprogramm 'lpr', in eine Datei oder über Pipe an ein Shell-Programm.
Sie können die Funktion PRINT ausführen oder auch die Taste PRINT drücken.

Info
: informiert über die Tabellen der aktuellen Datenbank.

Datei
ermöglicht, daß Sie eine SQL-Anweisung auf Abruf in eine Datei sichern bzw. die gesicherte SQL-Anweisung wieder einlesen und ausführen bzw. auch löschen können.

END
führt zurück in das Hauptmenü.

3.5.4 Der SQL-Editor

Bei den Funktionen 'Neu' und 'Korrigieren' erhalten Sie ein neues Menü und die Schreibmarke springt in den Arbeitsbereich, der jetzt als Editor dient; hier können Sie ihre SQL-Anweisung eingeben.

In der folgenden Übersicht sind die grundlegenden Funktionen des Editors aufgelistet. Ausführliche Informationen erhalten Sie im Kapitel 7.

- Die Pfeiltasten positionieren die Schreibmarke. [↖] positioniert die Schreibmarke in die erste Spalte der ersten Zeile.
- Zeichenweise löschen können Sie mit [CHAR DELETE] und [⌫]
- [↵] hat die Bedeutung von 'Neue Zeile' bzw. 'Wagenrücklauf'; die Taste positioniert die Schreibmarke an den Anfang der nächsten Zeile. [↵] bedeutet hier nicht 'führe das Kommando aus'!
- Links oben auf der Infozeile zeigt eine Zahl die aktuelle Zeilenposition der Schreibmarke an. Das sichtbare Eingabefeld besteht aus 18 Zeilen und 79 Zeichen pro Zeile. Bei SQL-Anweisung, die länger als 18 Zeilen sind, "scrollt" der Bildschirm, d.h. der Text wandert um je eine Zeile nach oben.
  Scrollen nach rechts oder links ist nicht möglich.
- Wenn Sie das NEU- oder das KORR.-Menü aufrufen, dann erhalten Sie in der Funktionszeile (und Kommentarzeile) Hinweise auf die Bedeutung dreier Funktionstasten, die Sie zum komfortableren Editieren verwenden können:
  [F15] schaltet um zwischen dem Überschreibmodus und dem Einfügemodus.
  [F16] trennt die Zeile ab der Schreibmarkenposition.
  [F17] löscht ab Position der Schreibmarke bis zum Zeilenende.

3.5.5 Die Datenbank mit SQL abfragen

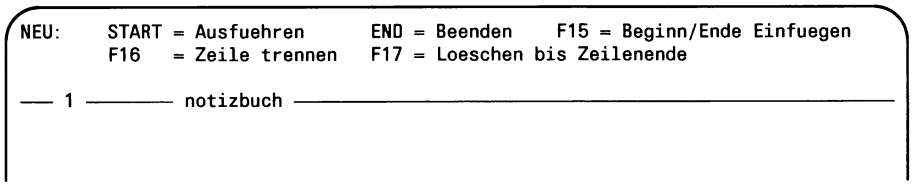
Die folgende Übung und die anschließenden allgemeinen Bedienungshinweise erläutern Ihnen anhand eines Beispiels, welche Schritte Sie unternehmen müssen, um eine SQL-Anweisung auszuführen.
Der darauffolgende Abschnitt führt Sie in die Syntax und den Sprachumfang von SQL ein.

Übung 3-11

Wenn Sie das SQL-Dialog-Menü vor sich haben:

▷ Führen Sie die Funktion 'Neu' aus.

Sie erhalten den NEU-Bildschirm:



Die Funktionen des NEU-Menüs:

START
gemeint ist die Taste START; START läßt sich hier nicht als Funktion ausführen, da die Schreibmarke im Arbeitsbereich steht und die Eingabe 's' als der Anfangsbuchstabe von START als Texteingabe für den Editor interpretiert wird.
START führt die eingegebene SQL-Anweisung aus und führt zurück in das SQL-Dialog-Menü.

END
gemeint ist die Taste END
END beendet das NEU-Menü und führt zurück in das SQL-Menü.

F15
Mit F15 können Sie zwischen dem Überschreibmodus und dem Einfügemodus umschalten.

F16
F16 trennt die aktuelle Zeile ab Position der Schreibmarke.

F17
F17 löscht die aktuelle Zeile ab Position der Schreibmarke.

Die Schreibmarke ist in den Arbeitsbereich gesprungen, der jetzt als Editor dient; hier können Sie Ihre SQL-Anweisung eingeben.

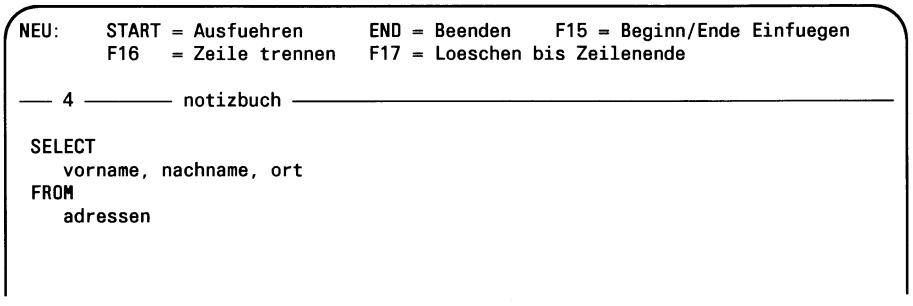
Mit der folgenden SQL-Anweisung führen Sie eine Datenbank-Abfrage durch; Sie fragen nach den Werten der Spalten 'vorname', 'nachname' und 'ort' der Tabelle 'adressen' - grundsätzliche Hinweise zu Aufbau und Syntax einer SQL-Anweisung entnehmen Sie dem darauffolgenden Abschnitt.

▷ Geben Sie die folgende SQL-Anweisung ein:

```
SELECT
   vorname, nachname, ort
FROM
   adressen
```

Die unterschiedlichen Groß- und Kleinschreibweisen sowie die Einrükkungen dienen der besseren Übersicht. Sie sind syntaktisch nicht zwingend und können unterbleiben.

So sollte Ihr Bildschirm aussehen:



▷ Drücken Sie START, um die Datenbank-Abfrage auszuführen.

Wenn Ihnen bei Ihrer Eingabe kein Syntaxfehler unterlaufen ist, dann erhalten Sie jetzt diesen Bildschirm mit dem SQL-Dialog-Menü:

```
SQL-Dialog:  Neu  START  Korrigieren  Ruf-Editor  PRINT  Info  Datei  END
 Dialoganweisung ausfuehren.

------------ notizbuch ------------------------------------------------

 vorname              nachname             ort

 Marlies              Altmann              Heidelberg
 Richard              Bauer                Ulm
 Herta                Dornbeck             Bonn 2
 Martin               Wolf                 Ulm

 Saetze gefunden: 4
```

Der Bildschirm zeigt Ihnen eine Ergebnistabelle mit sämtlichen Werten der Spalten 'vorname', 'nachname' und 'ort' der Tabelle 'adressen'. Am unteren Bildschirmrand erhalten Sie eine Meldung über die Anzahl der gefundenen Treffersätze.

Wenn Ihnen aber ein Syntaxfehler unterlaufen ist, z.B. wenn Sie statt SELECT SELCT geschrieben haben, dann führt Sie INFORMIX zurück in das SQL-Dialog-Menü. Die Funktion 'Korrigieren' ist markiert.

Um den Fehler zu korrigieren:

▷ Führen Sie die Funktion 'Korrigieren' aus.

Sie erhalten dann das KORR.-Menü, das identisch ist mit dem oben gezeigten NEU-Menü.

▷ Führen Sie die notwendigen Korrekturen durch und drücken Sie anschließend wieder START

Allgemeine Bedienungshinweise

Diese Bedienungshinweise erläutern ihnen anhand der Funktionen 'Neu' und 'Korrigieren', welche Schritte Sie unternehmen müssen, um eine SQL-Anweisung einzugeben, auszuführen bzw. um sie zu korrigieren:

Neu

führt in ein Menü, das in der Funktionszeile (und Kommentarzeile) Hinweise auf die Tasten START DEL F15 F16 und F17 enthält.
Die Schreibmarke springt in den Arbeitsbereich, der jetzt als Editor dient; Sie können Ihre SQL-Anweisung eingeben und anschließend mit START ausführen.
Danach werden Sie in das SQL-Dialog-Menü zurückgeführt und erhalten die Ergebnistabelle (bei Datenbank-Abfragen) am Bildschirm. Bei mehrseitigen Ergebnistabellen erhalten Sie ein Menü mit den Funktionen:
'Naechste' - blättern zu der nächsten Bildschirmseite
'Wiederholen' - die erste Bildschirmseite nochmals ausgeben
END - zurück zum SQL-Dialog-Menü. FEHLERBEHANDLUNG: Wenn Ihnen ein Syntaxfehler unterlaufen ist, dann erhalten Sie wieder das SQL-Dialog-Menü. Die Funktion 'Korrigieren' ist markiert.

Korrigieren

führt in das KORR.-Menü, das mit dem NEU-Menü identisch ist. Sie können Ihre Anweisung korrigieren und anschließend wieder mit START ausführen.

3.5.6 Aufbau und Syntax einer SELECT-Anweisung

In der vorausgegangenen Übung haben Sie eine SELECT-Anweisung kennengelernt, mit der Sie eine Datenbank-Abfrage durchgeführt haben:

```
SELECT
   vorname, nachname, ort
FROM
   adressen
```

Eine andere SELECT-Anweisung könnte z.B. so aussehen:

```
SELECT
   nachname, telefon
FROM
   adressen
```

Die beiden gezeigten SELECT-Anweisungen führen zu unterschiedlichen Ergebnissen: Die erste SELECT-Anweisung führt zu einer Ergebnistabelle mit den Werten der Spalten 'vorname', 'nachname' und 'ort' und die zweite SELECT-Anweisung führt zu einer Ergebnistabelle mit den Werten 'nachname' und 'telefon'

Was sich syntaktisch in den beiden Anweisungen unterscheidet, das sind lediglich Anzahl und Auswahl der Spaltennamen. Gleich geblieben dagegen sind die beiden großgeschriebenen Wörter SELECT und FROM - man nennt diese Wörter 'Schlüsselwörter'.
Die Schlüsselwörter sind unverzichtbarer Teil einer jeden SQL-Anweisung; sie bilden das syntaktische Gerüst.

Im Falle einer SELECT-Anweisung sind es eben die Schlüsselwörter SELECT und FROM, die auf jeden Fall in der Anweisung enthalten sein müssen.
Demgegenüber können Sie in die SELECT-Anweisung die Spaltennamen (bzw. auch den Tabellennamen) variabel einsetzen.

Syntaktisch kann man das wie folgt darstellen:

```
SELECT
   spaltenname1, spaltenname2, ...
FROM
   tabellenname
```

Die gezeigte Syntaxdarstellung besagt, daß

- eine SELECT-Anweisung mit dem Schlüsselwort SELECT beginnen muß,
- ein oder beliebig viele Spaltennamen folgen - sie sind durch Kommas voneinander zu trennen,
- darauf das Schlüsselwort FROM folgt und
- darauf der Name derjenigen Tabelle folgt, die die angegebenen Spalten enthält.

Unter Beachtung der hier angeführten Syntaxregeln können Sie also jede beliebige Tabelle der aktuellen Datenbank nach jeder beliebigen Spalte abfragen.

Allerdings können Sie aufgrund der bisherigen Syntaxdarstellung lediglich die von Ihnen gewünschte Spaltenauswahl - noch nicht aber eine Satzauswahl - durchführen. Denn eine Datenbank-Abfrage nach dem bisher gezeigten Syntaxmuster liefert immer alle Sätze der Tabelle.

Um auch eine Satzauswahl durchführen zu können, muß die SELECT-Anweisung um eine optionale WHERE-Klausel ergänzt werden:

Beispiel

```
SELECT
   vorname, nachname, ort
FROM
   adressen
WHERE
   ort = "Heidelberg"
```

Das gezeigte Beispiel enthält eine WHERE-Klausel, bestehend aus dem

- Schlüsselwort WHERE und einer
- Bedingung die lautet: ort = "Heidelberg"

Mit der WHERE-Klausel wird die Datenbank-Abfrage in diesem Beispiel auf diejenigen Sätze eingegrenzt, die in der Spalte 'ort' den Wert 'Heidelberg' enthalten.

Die SQL-Anweisung führt zu dieser Ergebnistabelle:

```
vorname        nachname       ort

Marlies        Altmann        Heidelberg
```

Andere Bedingungen sind zum Beispiel:

```
ort = "Bonn 2"          vorname = "Albert"          telefon = "030/987643"
```

In der erweiterten Syntaxdarstellung, die sowohl Spaltenauswahl als auch Satzauswahl berücksichtigt, lautet die SELECT-Anweisung:

```
SELECT
   spaltenname1, spaltenname2, ...
FROM
   tabellenname
WHERE
   bedingung
```

Wobei sich eine Bedingung u.a. aus den folgenden Elementen zusammensetzen kann:

```
spaltenname = "CHARACTER-Wert"        oder        spaltenname = NUMERISCHER-Wert
```

Die Werte von CHARACTER-Spalten müssen in Anführungszeichen gesetzt sein (DATE-Werte ebenfalls); die Werte von NUMERISCHEN-Spalten dürfen nicht in Anführungszeichen gesetzt sein.

3.5.7 Schreibweise einer SQL-Anweisung

Bei der Schreibweise einer SQL-Anweisung haben Sie sehr viel Freiheiten, beide folgenden SQL-Anweisung bedeuten das gleiche und führen auch zu der gleichen Ergebnistabelle:

```
SELECT
   vorname
FROM
   adressen
```

oder:

```
select vorname from adressen
```

Es gelten die folgenden Regeln:

- Schlüsselwörter, Spaltennamen und Tabellennamen dürfen groß oder klein geschrieben sein.
- Werte müssen genauso geschrieben sein wie sie definiert wurden. Groß- bzw. Kleinschreibung muß hierbei berücksichtigt werden.
- Die Anweisung darf beliebig viele Leerzeichen und Leerzeilen enthalten; mehrzeilige Schreibweise und Einrückungen sind folglich erlaubt.

Um SQL-Anweisungen übersichtlich darstellen zu können, bleibt das Handbuch bei der ursprünglich eingeführten Darstellung mit mehrzeiliger Schreibweise, Einrückungen und Großschreibweise der Schlüsselwörter.

3.5.8 Weitere SELECT-Anweisungen

Um Ihnen Gelegenheit zu geben, das Gelesene zu vertiefen, folgt hier wieder ein Abschnitt mit Aufgaben:

Aufgabe 3-2

Führen Sie mit SQL Datenbank-Abfragen durch, die die folgenden Ergebnisse liefern sollen:

A) Die Spalten 'nachname' und 'ort' aus allen Sätzen

B) Die kompletten Sätze der Personen, die in 'Ulm' wohnen

C) Die Spalten 'vorname' und 'telefon' der Person, die in Heidelberg wohnt

D) Alle Spalten der Person(en), bei denen die Postleitzahl 7900 lautet

3.6 Ein Listenprogramm erzeugen und ablaufen lassen

Dieser Abschnitt informiert Sie darüber, wie Sie ein Standard-Listen-Programm erzeugen und ablaufen lassen.

Ein Standard-Listen-Programm bietet nur eingeschränkte Möglichkeiten, es macht nichts weiter als

- die Datenbank abzufragen und
- die Ergebnistabelle mit einem Standard-Layout aufzubereiten und
- die aufbereitete Ergebnistabelle an das Druckprogramm 'lpr' weiterzuleiten (auszudrucken).

Für spezialisiertere Anwendungen wie z.B. Etikettenausdruck oder Erstellung einer Rechnung müssen Sie sehr weitreichende Änderungen an dem Listen-Programm vornehmen. Das Kapitel 8 informiert Sie ausführlich darüber.
In diesem Abschnitt geht es lediglich darum, die erforderlichen Schritte zur Erzeugung eines Standard-Listen-Programmes kennenzulernen.

3.6.1 Ein Standard-Listen-Programm erzeugen

Ein Standard-Listen-Programm ist ganz ähnlich zu erzeugen wie ein Standard-Format.

Die folgende Übung leitet Sie dazu an, ein Standard-Listen-Programm für die Datenbank 'notizbuch' zu erzeugen. Im Anschluß an die Übung finden Sie eine Übersicht mit "allgemeinen Bedienungshinweisen" zur Erzeugung eines Standard-Listen-Programmes.

Übung 3-12

```
INFORMIX-SQL:   Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Ablauf, Modifizieren, Neuerstellen oder Loeschen einer Liste.

————————— notizbuch ————————————————————————————————————————
```

▷ Im Hauptmenü führen Sie die Funktion 'Liste' aus.

Sie haben das LISTE-Menü erhalten:

```
LISTE:   Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Ablauf eines Listenprogrammes.

————————— notizbuch ————————————————————————————————————————
```

Die Funktionen des LISTE-Menüs:

Ablauf
: ein bestehendes Listen-Programm ablaufen lassen.

Modifizieren
: ein bestehendes Listen-Programm unter Verwendung des Betriebssystem-Editors verändern.

Generieren
: ein neues Standard-Listen-Programm automatisch erzeugen.

Neu
: ein neues Listen-Programm unter Verwendung des Betriebssystem-Editors schreiben.

Compilieren
: ein Listen-Programm zu einer ablauffähigen Liste compilieren (übersetzen).

Loeschen
: ein Listen-Programm löschen.

END
: Rückkehr in das INFORMIX-Hauptmenü.

▷ Führen Sie die Funktion 'Generieren' aus.

Sie erhalten diesen Bildschirm:

```
GENERIEREN LISTE >>
Namen der neuen Liste eingeben. Weiter mit RETURN

————————— notizbuch ————————————————————————————————————————
```

Sie sind nun aufgefordert, dem Listen-Programm einen Namen zu geben:

▷ Geben Sie ein: liste1

▷ Drücken Sie ↵

Sie erhalten den AUSWAHL-TABELLE-Bildschirm:

```
AUSWAHL TABELLE >>
Tabelle auswaehlen, die in der Standard-Liste enthalten sein soll.

————————— notizbuch ————————————————————————————————————————

 adressen
```

Sie sollen nun angeben, für welche Tabelle das Listen-Programm erstellt werden soll; der Name der Tabelle 'adressen' ist markiert.

▷ Drücken Sie ↵

Nun generiert INFORMIX das Standard-Listen-Programm; nach kurzer Wartezeit werden Sie in das LISTE-Menü zurückgeführt - mit einer Meldung zeigt INFORMIX die erfolgreiche Übersetzung des Listen-Programmes an:

```
LISTE:  Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Ablauf eines Listenprogrammes.

————————— notizbuch ————————————————————————————————————————————

 Die Liste wurde erfolgreich uebersetzt.
```

Hinweis

Konkret ist folgendes geschehen: Für Sie nicht sichtbar erzeugte INFORMIX ein Listen-Programm, das alle erforderlichen Anweisungen enthält.
In einem zweiten Schritt wurde das Listen-Programm compiliert und damit eine ablauffähige Liste erzeugt.

Allgemeine Bedienungshinweise

Um ein Standard-Listen-Programm zu erzeugen, unternehmen Sie die folgenden Schritte:

1. Im Hauptmenü führen Sie die Funktion 'Liste' aus.
2. Im damit aufgerufenen Menü führen Sie die Funktion 'Generieren' aus.
3. Wenn noch keine Datenbank ausgewählt wurde, dann erhalten Sie jetzt den AUSWAHL-DATENBANK-Bildschirm, der der Auswahl der gewünschten Datenbank dient - ansonsten gleich weiter mit 4.
4. Sie erhalten den GENERIEREN-LISTE-Bildschirm, der der Eingabe des gewünschten Listen-Namens dient.
   NAMENSKONVENTIONEN: wie bei Format.
5. Sie erhalten den AUSWAHL-TABELLE-Bildschirm, der der Auswahl der erwünschten Tabelle dient.
6. Danach wird automatisch die Standard-Liste erzeugt und Sie werden anschließend in das LISTE-Menü zurückgeführt.

3.6.2 Listen-Programm ablaufen lassen

Dieser Abschnitt erläutert Ihnen anhand einer Übung - gefolgt von "allgemeinen Bedienungshinweisen" - wie Sie ein Listen-Programm ablaufen lassen.

Übung 3-13

```
INFORMIX-SQL:   Format  Liste  SQL-Dialog  Benutzer-Menue  Datenbank  ...
Ablauf, Modifizieren, Neuerstellen oder Loeschen einer Liste.

————————— notizbuch ————————————————————————————————————————————
```

▷ Im Hauptmenü führen Sie die Funktion 'Liste' aus.

Sie erhalten das LISTE-Menü:

```
LISTE:   Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Ablauf eines Listenprogrammes.

————————— notizbuch ————————————————————————————————————————————
```

▷ Führen Sie die Funktion 'Ablauf' aus.

Sie erhalten den ABLAUF-LISTE-Bildschirm, der der Auswahl des gewünschten Listen-Programmes dient - der Name des auszuwählenden Listen-Programmes 'liste1' ist markiert:

```
ABLAUF LISTE >>
Liste auswaehlen oder Namen eingeben. Weiter mit RETURN

————————— notizbuch ————————————————————————————————————————————

 liste1
```

▷ Drücken Sie [↵]

Nach kurzer Wartezeit erhalten Sie im Arbeitsbereich die folgenden Meldungen:

```
ABLAUF LISTE >>
Liste auswaehlen oder Namen eingeben. Weiter mit RETURN

————————— notizbuch ————————————————————————————————————————

 Beginn der Datenbereitstellung.

 Datenbereitstellungs-Anweisung 1 wird verarbeitet.

 Ende der Datenbereitstellung.

 Die Ausgabe wird ueber Pipe an "lpr" geleitet.









 Zur Fortsetzung bitte RETURN
```

Die aufbereitete Liste wird automatisch an das Druckprogramm "lpr" weitergeleitet und über den eingestellten Standard-Drucker ausgedruckt.

Hinweis

Wenn Ihr Drucker nicht mehr als 80 Zeichen ausdrucken kann, bzw. wenn der Druckeraufruf nicht entsprechend gestaltet wurde, dann sind am rechten Rand einige Zeichen abgeschnitten. Um das zu ändern, müßten Sie wesentliche Änderungen an dem automatisch erzeugten Listen-Programm vornehmen. Das Kapitel 8 informiert Sie über die erforderlichen Maßnahmen.

▷ Um in das Hauptmenü zurückzukehren, drücken Sie [↵] und danach [END]

Allgemeine Bedienungshinweise

Um ein gewünschtes Listen-Programm aufzurufen, unternehmen Sie die folgenden Schritte:

1. Im Hauptmenü führen Sie die Funktion 'Liste' aus.
2. Im damit aufgerufenen LISTE-Menü führen Sie die Funktion 'Ablauf' aus.
3. Sie erhalten den ABLAUF-LISTE-Bildschirm, der der Auswahl des gewünschten Listen-Programmes dient. Wählen Sie das gewünschte Listen-Programm aus. Anschließend wird automatisch das ausgewählte Listen-Programm ausgeführt.
4. Um in das Hauptmenü zurückzukehren, drücken Sie ↵ und danach END

Zusammenfassung

- Eine neu erzeugte Datenbank ist zunächst noch leer.
- Den Namen der jeweils aktuellen Datenbank zeigt INFORMIX in jedem Menü auf der Infozeile an.
- Das TABELLE-ERSTELLEN-Menü ermöglicht die menügeführte Erzeugung einer Tabelle.
- An Datentypen stellt INFORMIX zur Verfügung: die ZEICHEN Datentypen CHAR und VARCHAR (nur INFORMIX-ONLINE), die NUMERISCHEN Datentypen DECIMAL, SMALLINT, INTEGER, SMALLFLOAT und FLOAT, die ZEIT Datentypen DATE, DATETIME, INTERVAL, die Datentypen SERIAL und MONEY, die BLOB-Datentypen TEXT und BYTE (nur INFORMIX-ONLINE)
- Sie können sich von INFORMIX Standard-Formate und -Listen automatisch erzeugen lassen.
- Ein Format ist der einfachste Weg, um neue Datensätze eingeben, korrigieren oder löschen zu können. Ein Format dient ebenfalls dazu, die Datenbank abzufragen.
- SQL ist eine dem Englischen angelehnte Datenbanksprache. Mit ihr können Sie die Datenbank in einem freien Dialog abfragen.

4 Die Datenbank 'versand'

Die folgenden Kapitel 5 ff beschäftigen sich eingehender mit Formaten, SQL und Listen.
Dabei beziehen sich die Beispiele und Erläuterungen dieser Kapitel weitgehend auf eine Beispiel-Datenbank mit dem Namen 'versand'; diese Datenbank wurde im Rahmen der Installation der Diskette INFDEX automatisch in Ihrem Dateiverzeichnis/Ordner 'vertiefung' eingerichtet.

Das vorliegende Kapitel informiert Sie über

- Maßnahmen, die Sie ergreifen müssen, um Zugang zu der Beispieldatenbank 'versand' zu erhalten,
- Umfang und Bedeutung der Datenbank 'versand',
- Möglichkeiten, eine veränderte oder zerstörte Datenbank 'versand' wieder zu rekonstruieren.

4.1 Zugang zur Beispieldatenbank 'versand'

Um mit der Beispieldatenbank 'versand' arbeiten können, müssen Sie künftig folgenden, abgeänderten Weg einschlagen:

Nach dem Anmelden wechseln Sie mit dem 'cd' Kommando künftig in das Dateiverzeichnis 'vertiefung'; nicht mehr in das Dateiverzeichnis 'einstieg'. Danach können Sie INFORMIX wie gewohnt aufrufen.

4.2 Umfang und Bedeutung der Datenbank 'versand'

Die Datenbank 'versand' ist die Datenbank eines Sportartikel-Grossisten. Sie dient zwei Zwecken:

- zum einen können Sie an ihr erlernen, wie man Formate sowie Listen-Programme erstellt und bedient und wie man SQL verwendet,
- zum zweiten ermöglicht sie Ihnen einen Einblick in eine praxisnahe, professionelle Datenbank-Anwendung, was Ihnen beim Aufbau Ihrer eignen Datenbank nützlich sein könnte.

Die Datenbank 'versand' besteht aus sechs Tabellen:

'kunde'
Daten über die Kunden,

'staat'
Daten über die Bundesländer,

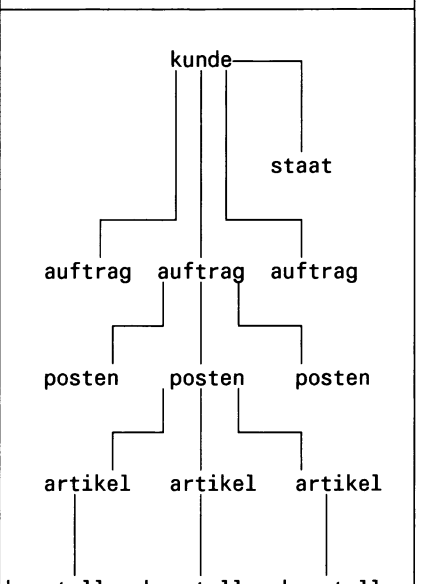
'auftrag'
Daten über die erteilten Kundenaufträge,

'posten'
Daten über die bei einem Auftrag bestellten Posten,

'artikel'
Daten über die geführten Artikel,

'hersteller'
Daten über die Artikelhersteller.

Beziehungen zwischen den Tabellen

TABELLE	BEDEUTUNG	STRUKTUR
kunde	Kundendaten; es gibt 18 Kunden	
staat	Bundesland ausgeschrieben	
auftrag	Auftragsdaten; ein Kunde hat (k)einen oder mehrere Aufträge erteilt	
posten	Postendaten; ein Auftrag besteht aus einem oder mehreren Posten	
artikel	Artikeldaten; ein Posten besteht aus einem oder mehreren Artikeln	
her-steller	Herstellerdaten; jeder Artikel wird von einem Hersteller bezogen	

Es gibt folgende logische Beziehungen zwischen den Tabellen:

- In der Tabelle 'kunde' sind alle Daten über die Kunden gespeichert, die der Sportartikel-Grossist beliefert.
- In der Tabelle 'staat' stehen die Langnamen der Bundesländer; in der Tabelle 'kunde' stehen nur die Abkürzungen. Die Tabelle 'staat' wird in den Erläuterungen der folgenden Kapitel nicht berücksichtigt.
- Ein Kunde erteilt Aufträge; die Daten über die von einem Kunden erteilten Aufträge sind in der Tabelle 'auftrag' gespeichert.
  Es gibt auch Kunden, die bisher noch keinen Auftrag erteilt haben aber dennoch als mögliche künftige Kunden in die Tabelle 'kunde' aufgenommen wurden.
- Ein Auftrag besteht aus einem oder mehreren Posten; die Daten über die Posten sind in der Tabelle 'posten' gespeichert.
- Jeder Posten besteht aus einem oder mehreren Sport-Artikeln; Die Daten über die Sport-Artikel sind in der Tabelle 'artikel' gespeichert.

- In der Tabelle 'hersteller' schließlich sind einige Daten über die Hersteller der einzelnen Sport-Artikel gespeichert.

Die Tabellen lassen sich über Join-Spalten miteinander verbinden. So ist es z.B. möglich, mit Datenbank-Abfragen zu ermitteln,

- welche Aufträge ein Kunde erteilt hat,
- aus welchen Posten ein Auftrag besteht,
- aus welchen Artikeln ein Posten besteht,
- von welchem Hersteller ein Artikel bezogen wird.

Weiter enthält die Datenbank eine ganze Reihe von Formaten und Listen-Programmen sowie eine SQL-Anweisung.
Der Doppelbedeutung der Datenbank entsprechend gibt es Formate und Listen-Programme, die

- dem Erlernen einzelner Funktionsbereiche von Formaten und Listen dienen und die
- beispielhaft professionelle Anwendungen vorführen.

Um die beiden Bereiche deutlich unterscheiden zu können, tragen die Lernformate und -Listen großgeschriebene Namen dieser Art:
FORMAT01
Und die professionellen Formate und Listen tragen kleingeschriebene Namen dieser Art: auftrag

Auf die Lernformate und -Listen (großgeschrieben) nehmen die einzelnen Kapitelabschnitte an entsprechender Stelle Bezug.
Auf die professionellen Anwendungen (kleingeschrieben) geht das Handbuch nicht weiter ein. Es bleibt dem einigermaßen geübten INFORMIX-Anwender überlassen, sich hier gegebenenfalls mit Anregungen und Tips zu versorgen.

Hinweis

Der Anhang informiert ausführlich über die Datenstruktur der Datenbank 'versand' sowie über die Programme der Formate und Listen.

4.3 Möglichkeiten zur Rekonstruktion

Wenn Sie im Verlauf der folgenden Übungen die Datenbank 'versand' verändert haben, dann können Sie die Datenbank wie folgt wieder herstellen:

- Entweder Sie lassen vom Systemverwalter nochmals die Diskette INFDEX installieren. Dabei wird die Datenbank 'versand' automatisch neu erzeugt.
- Oder Sie schlagen folgenden Weg ein (wenn Sie ausreichende Grundkenntnisse über SQL besitzen):
  Zunächst führen Sie die SQL-Anweisung 'CLOSE DATABASE' aus, die die aktuelle Datenbank schließt.
  Danach führen Sie die SQL-Anweisung 'DROP DATABASE versand' aus, die die Datenbank 'versand' löscht.
  Danach lesen Sie über die SQL-Dialog-Menüfunktion 'Datei' (DATEI Laden - siehe Abschnitt 7.2.9) die abgelegte SQL-Anweisung 'c_database' ein und führen die SQL-Anweisung aus. Diese Anweisung (bzw. Anweisungsfolge) erzeugt die Datenbank 'versand' neu.

Hinweis

Die Format- und Listen-Programme der Datenbank 'versand' können Sie nur durch erneute Installation der Diskette INFDEX rekonstrieren.

Zusammenfassung

- Um mit der Datenbank 'versand' arbeiten zu können, müssen Sie künftig in das Dateiverzeichnis 'vertiefung' wechseln (Shell).
- Die Datenbank 'versand' enthält die folgenden sechs Tabellen: 'kunde', 'staat', 'auftrag', 'posten', 'artikel' und 'hersteller'.
  Weiter enthält die Datenbank Lernformate und -Listenprogramme (großgeschrieben) sowie professionelle Formate und Listenprogramme (kleingeschrieben).
- Um den ursprünglichen Stand der Datenbank 'versand' zu erhalten, können Sie die Diskette INFDEX vom Systemverwalter neu installieren lassen; mit SQL-Grundkenntnissen können Sie die Datenbank auch mit SQL-Anweisungen rekonstruieren.

5 Arbeiten mit einem Format

Dieses Kapitel erläutert Ihnen anhand der Formate der Datenbank 'versand' (Dateiverzeichnis 'vertiefung'):

- wie Sie mit einem Format eine Datenbank abfragen,
- wie Sie mit einem Format Sätze neuaufnehmen, korrigieren oder löschen,
- welche Bedeutung die Funktionen im PERFORM-Menü haben,
- was ein 'Mehr-Tabellen-Format' ist,
- wie Sie mit einem 'Mehr-Tabellen-Format' arbeiten.

Dabei gibt es teilweise Wiederholungen aus Kapitel 3; hier in Kapitel 5 können Sie alles in komprimierter Form noch einmal nachlesen.

5.1 Grundlagen

Die folgenden Abschnitte vermitteln Ihnen die Grundlagen der Arbeit mit Formaten.

5.1.1 Format aufrufen

Um ein gewünschtes Format aufzurufen, unternehmen Sie die folgenden Schritte:

- Im Hauptmenü führen Sie die Funktion 'Format' aus.
- Im damit aufgerufenen FORMAT-Menü führen Sie die Funktion 'Ablauf' aus.
- Sie erhalten einen Bildschirm, der alle vorhandenen Formate des aktuellen Dateiverzeichnisses anzeigt. Sie können das gewünschte Format auswählen, indem Sie es mit den Pfeiltasten markieren oder indem Sie seinen Namen in das Eingabefeld in der Funktionszeile eingeben. Danach drücken Sie ↵
- Nach kurzer Wartezeit erhalten Sie den PERFORM-Bildschirm mit dem ausgewählten Format; PERFORM ist diejenige INFORMIX-Komponente, die das Arbeiten mit einem Format ermöglicht.

5.1.2 Der PERFORM-Bildschirm

Im Arbeitsbereich zwischen den beiden Trennlinien zeigt INFORMIX das ausgewählte Format an. Ein Format besteht aus

- Bildschirmfeldern [ ] und
- beliebigen, freien Texten. Bei Standard-Formaten sind es einfach nur die Spaltennamen.

Bildschirmfelder

Die Bildschirmfelder dienen der Anzeige bzw. der Neuaufnahme oder Korrektur von Werten der Tabellenspalten.

Freie Texte

Formate können mit beliebigen, erläuternden freien Texten beschriftet sein. Bei einem Standard-Format besteht die freien Texte lediglich aus den Spaltenamen wie sie in der Tabelle definiert sind. Wenn Sie aber ein eigenes Format erzeugen, dann steht es Ihnen frei, ob und mit welchen freien Texten Sie das Format ausstatten. So kann eine Spalte zum Beispiel 'pers_nr' heißen, in einem Format als freier Text aber ganz anders heißen, z.B. so:

```
Personalnummer:   [          ]
```

Die Formate der Datenbank 'versand' sind ausführlich mit freien Texten versehen.

5.1.3 Editierfunktionen

Beim Suchen, Neuaufnehmen oder Korrigieren eines Satzes stehen Ihnen die folgenden Editierfunktionen zur Verfügung:

[⌫]	löscht das Zeichen links von der Schreibmarke.
[CHAR DELETE]	löscht das Zeichen über der Schreibmarke.
[WORD DELETE]	löscht den Text im Bildschirmfeld ab Position der Schreibmarke.
[CHAR INSERT]	fügt ein Leerzeichen ein.
[WORD INSERT]	zeigt den zuletzt angezeigten Wert wieder an.
[F15]	schaltet um zwischen Überschreib- und Einfügemodus.
[←] [→]	positionieren die Schreibmarke innerhalb eines Bildschirmfeldes.
[↵] [⇥] [⇥]	positionieren die Schreibmarke in das nächste Bildschirmfeld; die Abfolge, in der die Schreibmarke in die Bildschirmfelder springt, ist im Format-Programm festgelegt.
[⇤] [⇤]	positionieren die Schreibmarke in das vorherige Bildschirmfeld.
[!]	steht die Schreibmarke in einem TEXT- oder BYTE-Bildschirmfeld, dann ruft [!] vorübergehend das externe Programm auf, mit dem Sie BLOB-Daten bearbeiten können.

5.1.4 Datentypen

Um unsinnige oder unerwünschte Eingaben zu vermeiden, führt PERFORM einige Prüfungen durch: PERFORM überprüft u.a., ob ein eingegebener Wert auch dem Datentyp der zugehörigen Tabellenspalte enstpricht.

Wenn Sie eine Eingabe machen, die nicht dem definierten Datentyp entspricht, und wenn Sie danach versuchen, das Bildschirmfeld zu verlassen, dann:

- gibt PERFORM einen Signalton und die folgende Meldung aus: Fehler im Feld
- und verhindert, daß Sie das Bildschirmfeld verlassen können.

Sie haben nur die Möglichkeit, entweder einen korrekten Wert einzugeben oder mit [DEL] abzubrechen.

Darüber hinaus können Sie beim Erzeugen eines Formates auch Vorschriften darüber erlassen, wie eine Eingabe zwingend auszusehen hat, damit Sie von PERFORM akzeptiert wird.
So können Sie z.B. dafür sorgen, daß PERFORM in einem numerischen Bildschirmfeld nur Zahlen zwischen 1 und 50 annimmt und sich bei jeder anderen Eingabe so verhält, wie oben beschrieben.

Besonderheiten bei den Datentypen

Ausführliche Informationen über Datentypen entnehmen Sie dem vorangegangenen Abschnitt '3.3.4 Datentyp'.
Der vorliegende Abschnitt geht ausschließlich auf die Besonderheiten bei Datentypen im Zusammenhang mit Formaten ein.

Es gibt die folgenden Datentypen:

- ZEICHEN
  `CHAR, VARCHAR`
- NUMERISCH
  `SMALLINT, INTEGER, SMALLFLOAT, FLOAT, DECIMAL`
- ZEIT
  `DATE, DATETIME, INTERVALL`
- SERIAL
- MONEY
- BLOB
  `TEXT, BYTE`

ZEICHEN-Datentypen

Die Datentypen CHAR und VARCHAR nehmen beliebige Zeichen auf. So z.B. Buchstaben, Nummern, Satztrennzeichen und anderes mehr.

VARCHAR ist nur bei INFORMIX-ONLINE möglich. Der Datentyp VARCHAR speichert Werte speicherplatzsparender als CHAR, ist aber sonst mit CHAR identisch.

Normalerweise sollte es sich so verhalten, daß ein Bildschirmfeld für eine CHAR-Tabellenspalte ebensoviele Zeichen aufnehmen kann, wie bei der Spaltendefinition mit der Länge festgelegt. Bei Standard-Formaten ist das automatisch der Fall: Wenn eine CHAR-Spalte mit der Länge 10 definiert wurde, dann erhalten Sie ein Bildschirmfeld, das 10 Zeichen aufnehmen kann: [0123456789]
Wenn Sie aber ein eigenes Format erzeugen, dann sollten Sie darauf achten, daß die Länge des Bildschirmfeldes mit der Länge der Tabellenspalte übereinstimmt.
So kann es eventuell vorkommen, daß die Länge eines Bildschirmfeldes nicht mit der Länge der Tabellenspalte übereinstimmt.
Ist das Bildschirmfeld zu groß, dann werden die überstehenden Zeichen gegebenenfalls nicht in die Tabelle aufgenommen und einfach abgeschnitten bzw. nicht angezeigt.

NUMERISCHE-Datentypen

An numerischen Datentypen gibt es:

SMALLINT - INTEGER - SMALLFLOAT - FLOAT - DECIMAL

Grundsätzlich gilt: Sie können nur Zahlen eingeben, die dem entsprechenden numerischen Datentyp und der festgelegten Spaltenlänge (bzw. Nachkommastellen) entsprechen:

Bei SMALLINT und INTEGER können Sie nur ganze Zahlen (keine Kommazahlen) eingeben.
Bei SMALLFLOAT und FLOAT können Sie Gleitkommazahlen eingeben (Komma an beliebiger Stelle).
Bei DECIMAL können Sie - je nach Spaltendefinition - ganze Zahlen, Festkommazahlen oder Gleitkommazahlen eingeben.

ZEIT-Datentypen

An ZEIT-Datentypen gibt es:

DATE - DATETIME - INTERVALL

DATE

Bei der Eingabe eines Kalenderdatums können Sie mehrere Schreibweisen wählen, z.B.:

```
6.5.90          6.5.1990          06 05 1990
```

Sie können Punkt, Komma, Schrägstrich und Leerzeichen als Trennzeichen verwenden. Sie können eine führende '0' nach DIN eingeben und das Jahresdatum zwei oder vierstellig eingeben.
Sie müssen sowohl zum Tag, als auch zum Monat, als auch zum Jahr Angaben machen.
Wenn Sie einen Satz neu aufnehmen (bzw. korrigieren), dann wandelt PERFORM das Datum automatisch in ein spezifisch gefordertes Format um, sobald Sie das entsprechende Bildschirmfeld mit der Schreibmarke verlassen. Standardmäßig in dieses Format nach DIN: 06.05.1990
Im Format-Programm ist das gewünschte Ausgabeformat jedoch optional festlegbar.

DATETIME

Bei der Eingabe eines DATETIME-Wertes müssen Sie nach der folgenden Regeln verfahren:

```
jahr-monat-tag stunde:minute:sekunde.bruchteil
```

Bindestrich trennt die Komponenten des Datumanteils, Leerzeichen trennt den Datumanteil vom vom Zeitteil, Doppelpunkt trennt Stunde, Minute und Sekunde, Punkt trennt Sekunde von Bruchteil.
Je nach Definition der DATETIME-Spalte sind u.U. nur Teile hiervon einzugeben.

Beispiel

```
90-12-31 11:20:50
```

INTERVALL

Eine Zeitspanne kann entweder über Jahre bis zu Monaten oder über Tage bis zu Bruchteilen gehen. Für die Eingabe gelten die gleichen Regeln wie bei DATETIME.

Beispiel

```
5-3       Fünf Jahre und drei Monate
12:23     Zwölf Stunden und dreiundzwanzig Minuten
```

SERIAL

Die Werte für SERIAL-Spalten vergibt INFORMIX automatisch; Sie können die Werte von SERIAL-Spalten nicht von Hand eingeben, verändern oder löschen.
Um das zu gewährleisten, unterbindet PERFORM, daß Sie die Schreibmarke beim Korrigieren oder Neuaufnehmen in ein SERIAL-Bildschirmfeld positionieren können.
Beim Suchen erlaubt PERFORM die Positionierung der Schreibmarke in ein SERIAL-Bildschirmfeld, um eine Such-Bedingung eingeben zu können.

MONEY

Bei MONEY-Bildschirmfeldern sind ganzzahlige Eingaben oder aber die Eingabe von Festkommazahlen mit maximal zwei Nachkommastellen möglich.

BLOB

Die BLOB-Datentypen TEXT und BYTE sind nur bei INFORMIX-ONLINE zulässig. In der Beispiel-Datenbank 'versand' gibt es keine BLOB-Datentypen.

BLOB-Daten können Sie nicht unmittelbar in einem Format bearbeiten sondern nur über ein externes Programm (HIT, SIPLAN u.s.w).

Um das externe Programm aufzurufen,

▷ positionieren Sie die Schreibmarke in das BLOB-Bildschirmfeld und

▷ drücken [I]

Danach wird das externe Programm aufgerufen und Sie können die BLOB-Daten bearbeiten. Das externe Programm übernimmt für seine Laufzeit die Kontrolle über den Bildschirm.

Der Name des aufzurufenden Programms ist beim Datentyp BYTE im Format-Programm anzugeben. Beim Datentyp TEXT ist das ebenfalls möglich aber nicht zwingend. Ist bei TEXT kein Programm angegeben, dann wird entweder der Standard-Betriebssystem-Editor (CED) aufgerufen oder der Editor, der mit der Umgebungsvariable DBEDIT vereinbart wurde.

BLOB-Daten werden in einem Format wie folgt angezeigt:

TEXT

in der Länge, die das entsprechende Bildschirmfeld vorgibt.
Beispiel - in einer BLOB-Spalte steht:

```
Dies ist der Text in einer BLOB-Spalte, der mit dem
Betriebssysytem-Editor CED geschrieben wurde.
```

In einem Format könnte das dann wie folgt angezeigt werden:

```
[Dies ist der Text in einer BLOB-Spalte, der mit dem]
```

Um TEXT-Daten in voller Länge sehen zu können, müssen Sie die Schreibmarke in das entsprechende Bildschirmfeld positionieren und mit [I] das vereinbarte Programm aufrufen.

BYTE

BYTE-Daten zeigt ein Format nicht an. Im entsprechenden Bildschirmfeld steht lediglich:
`[<BYTE-Wert> ]`
Um BYTE-Daten sehen zu können, müssen Sie die Schreibmarke in das entsprechende Bildschirmfeld positionieren und mit [I] das vereinbarte Programm aufrufen.

Nachdem Sie eine Suche ausgeführt haben, können Sie mit 'Vorw.' oder 'Rueckw.' in der aktuellen Liste blättern. Dabei sehen Sie die Daten von TEXT-Bildschirmfeldern u.U. nur teilweise und die Daten von BYTE-Bildschirmfeldern überhaupt nicht (nur die Ersatzdarstellung <BYTE-Wert>).

Um auch diese Daten sichtbar machen zu können, gibt es im PERFORM-Menü die Menüfunktion 'BLOB'. Diese Funktion wirkt sich wie folgt aus:

- Die Schreibmarke springt in das erste BLOB-Bildschirmfeld. Sie können die Schreibmarke in das gewünschte BLOB-Bildschirmfeld positionieren.
- Dann drücken Sie [I], um das externe Programm aufzurufen.

Das Programm wird aufgerufen und mit den BLOB-Daten geladen. Wenn Sie das Programm verlassen, kehren Sie wieder in das Format zurück. Um hier wieder mit 'Vorw.' und 'Rueckw.' weiter blättern zu können, müssen Sie zunächst [ESC] drücken.

Sollten Sie Änderungen an den angezeigten BLOB-Daten vorgenommen und diese mit den spezifischen Mitteln des Programms zurückgeschrieben haben, so werden diese Änderungen nicht in die Datenbank mit übernommen.

Veränderungen in der Datenbank erfolgen nur dann, wenn Sie vorausgehend die Funktionen 'Korrigieren' oder 'Neuaufnehmen' (oder 'Loeschen') ausgeführt haben. Bei 'Korrigieren' oder 'Neuaufnehmen' ist es wichtig, daß Sie das externe Programm wie folgt verlassen:

- Sichern Sie die BLOB-Daten mit den spezifischen Mitteln des Programms (beim CED z.B mit [END] [J]).
- Achten Sie dabei darauf, daß die Sicherung auch unter dem gleichen Dateinamen erfolgt, mit dem das Programm bei Aufruf geladen wurde.

Anderenfalls können die BLOB-Daten nicht in die Datenbank übernommen werden.

5.1.5 Format-Programm und PERFORM

In den vorangegangenen Abschnitten war einige Male von Format-Programmen bzw. von selbst erzeugten Formaten die Rede.
Um diese Begriffe richtig einordnen zu können und um einen Eindruck von den Möglichkeiten zu erhalten, die Formate bieten, folgt hier ein Vorgriff auf das folgende Kapitel 6, das sich mit der Erzeugung von Formaten beschäftigt:

Wenn Sie menügeführt ein Standard-Format erzeugen, dann vollzieht sich damit in Wahrheit ein relativ komplexer Vorgang:

Zunächst wird automatisch ein Format-Programm erzeugt.

> Dieses Format-Programm ist einfach ein Text mit einer Reihe von syntaktisch erforderlichen Anweisungen; z.B. enthält er Anweisungen darüber, für welche Datenbank und für welche Tabelle das Format verwendet werden soll. Weiter enthält es Anweisungen darüber, welche Bildschirmfelder das Format erhalten soll und wo sie auf dem Bildschirm angeordnet sein sollen und welchen Tabellenspalten sie zugeordnet sein sollen.
> Dieses Format-Programm können Sie für ein Standard-Format von INFORMIX erstellen lassen. Sie können es aber auch mit Hilfe eines Editors selbst schreiben.

Dann wird das das Format-Programm compiliert (übersetzt).

> Beim Compilieren wird aus den Anweisungen des Format-Programmes ein ablauffähiges Format erzeugt.
> Die Namen dieser Formate bekommen Sie im AUSWAHL-FORMAT-Menü angezeigt.

Wenn Sie nun ein Format ausgewählt haben, um mit ihm zu arbeiten, dann kommt zusätzlich PERFORM in's Spiel.

> Denn nur das Zusammenspiel dieser beiden Elemente - hier das compilierte Format und dort die INFORMIX-Komponente PERFORM - ermöglicht das Arbeiten mit einem Format.

Die Stelle, an der Sie gestalterisch auf das künftige Format einwirken können, ist das Format-Programm.
Dabei spielt es keine Rolle, ob Sie ein Standard-Format-Programm nachträglich modifizieren oder ob Sie das Format-Programm gleich von Anfang an völlig neu schreiben.

Im Format-Programm können Sie unter anderem festlegen:

- freie Texte,
  z.B. Personalnummer statt pers_nr,
- Bildschirm-Layout, d.h., an welcher Stelle auf dem Bildschirm sollen freie Texte und Bildschirmfelder angeordnet sein,
- speziell erwünschte Ausgabeformate, z.B. 6.5.90 statt 06.05.1990,
- zulässige Wertebereiche, z.B. für eine numerische Spalte nur Zahlen zwischen 1 und 50,
- die erwünschte Spaltenauswahl, d.h. Sie können festlegen, welche Tabellenspalten in dem Format berücksichtigt werden sollen, nur diesen Tabellenspalten ordnen Sie in dem Format Bildschirmfelder zu.

In den Formaten der Datenbank 'versand' werden Sie verschiedene Variationsmöglichkeiten der genannten Möglichkeiten kennenlernen.

5.2 'Suchen' - eine Datenbank abfragen

5.2.1 Formaler Ablauf

Wenn Sie mit einem Format eine Datenbank abfragen wollen, dann:

- Rufen Sie das gewünschte Format auf.
- Führen Sie im PERFORM-Menü die Funktion 'Suchen' aus.
- Danach erhalten Sie eine Funktionszeile (und Kommentarzeile) mit Hinweisen auf die Funktionstasten START DEL F17 und HELP
  Die Schreibmarke ist in das erste Bildschirmfeld des Formates gesprungen; das Format ist editierbar. PERFORM eröffnet Ihnen damit die Möglichkeit zur optionalen Eingabe von Such-Bedingungen in den Bildschirmfeldern.
- Um die Datenbank-Abfrage auszuführen, drücken Sie START
- PERFORM führt nun die Datenbank-Abfrage durch, erstellt die aktuelle Liste und zeigt den ersten Satz der aktuellen Liste in dem Format als den aktuellen Satz an.
  Wieviel Sätze die aktuelle Liste enthält, hängt davon ab, ob und welche Such-Bedingungen Sie eingegeben haben. Wenn Sie keine Such-Bedingungen eingegeben haben, dann enthält die aktuelle Liste alle Sätze der aktuellen Tabelle.
- Mit den Funktionen 'Vorw.' und 'Rueckw.' können Sie in der aktuellen Liste blättern und sich den jeweils folgenden bzw. vorherigen Satz als den neuen aktuellen Satz anzeigen lassen.
  Wenn Sie einige Sätze der aktuellen Liste überspringen wollen, dann können Sie vorab auch eine Zahl eingeben und erst danach die Funktion 'Vorw' bzw. 'Rueckw.' ausführen.
  Wenn Sie z.B. 5 Sätze vorwärts springen wollen, dann geben Sie ein: 5 v
- Um nach der Suche auch BLOB-Daten sehen zu können, führen Sie die Menüfunktion 'BLOB' aus und positionieren die Schreibmarke in das gewünschte BLOB-Bildschirmfeld. Drücken Sie I. Damit wird das vereinbarte externe Programm aufgerufen und mit den BLOB-Daten geladen. Nach dem Verlassen des Programmes drücken Sie ESC, um wieder weiter blättern zu können.

5.2.2 Aktuelle Tabelle - aktuelle Liste - aktueller Satz

Die aktuelle Tabelle

Ein Format kann Bildschirmfelder einer einzigen Tabelle oder auch Bildschirmfelder aus mehreren Tabellen enthalten (siehe Abschnitt 5.8). Suchen können Sie aber nur in der jeweils aktuellen Tabelle.
Um Ihnen die Orientierung zu erleichtern, zeigt PERFORM den Namen der aktuellen Tabelle rechts in der zweiten Zeile in dieser Form an:

```
** 1: kunde Tabelle**
```

Die aktuelle Liste

Jede Datenbank-Abfrage führt zu einer Ergebnistabelle mit den gefundenen Treffersätzen. Bei einem Format heißt diese Ergebnistabelle die 'aktuelle Liste'. Ein Format kann zur gleichen Zeit immer nur einen einzigen Satz einer Tabelle anzeigen. Die aktuelle Liste dient als Zwischenspeicher und hält sämtliche Treffersätze auf Abruf bereit. Mit den Funktionen 'Vorw.' und 'Rueckw.' können Sie in der aktuellen Liste blättern und sich damit den jeweils folgenden bzw. vorherigen Satz als den neuen aktuellen Satz am Bildschirm anzeigen lassen.

Der aktuelle Satz

Korrigieren oder löschen können Sie nur den aktuellen Satz; aktueller Satz kann nur ein Satz der aktuellen Liste sein. Bei Ein-Tabellen-Formaten versteht sich das von selbst, denn jeder nach dem Suchen am Bildschirm angezeigte Satz ist automatisch ein Satz der aktuellen Liste und damit auch der aktuelle Satz.
Bei Mehr-Tabellen-Formaten ist das nicht immer der Fall, der Abschnitt 5.8 geht auf die damit verbundenen Umstände ein. Im Zweifelsfall können Sie die Menü-Funktion 'Aktuell' ausführen und sich damit den tatsächlich aktuellen Satz der jeweils aktuellen Liste anzeigen lassen.

5.2.3 Such-Bedingungen

Wenn Sie in einem Format mit der Menüfunktion 'Suchen' eine Datenbank-Abfrage einleiten, dann springt die Schreibmarke in das erste Bildschirmfeld des Formates; das Format ist editierbar, d.h. Sie können Eingaben in die Bildschirmfelder durchführen.
Damit eröffnet Ihnen PERFORM die Möglichkeit zur Eingabe von Such-Bedingungen, mit denen Sie eine erwünschte Satzauswahl durchführen können.
Die aktuelle Liste enthält nur diejenigen Sätze, die die Such-Bedingungen erfüllen.

Die Eingabe von Such-Bedingungen in ein BLOB-Bildschirmfeld ist nicht möglich.

In der folgenden Übung lernen Sie das erste Format, FORMAT01, der Datenbank 'versand' (Dateiverzeichnis/Ordner 'vertiefung') kennen.
Weiter erfahren Sie in der Übung, daß Sie nicht nur eine einzige, sondern auch mehrere Such-Bedingungen eingeben können und dadurch eine Datenbank-Abfrage weiter präzisieren können.

Übung 5-1

▷ Rufen Sie das Format FORMAT01 auf.

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**

                                FORMAT01
                                --------

Kundennummer [                ]


Firma        [                    ]


Vorname      [               ]       Nachname            [               ]
Strasse      [                    ]  evtl. Zweitanschrift [                    ]
Plz          [     ]                 Ort                 [               ]


Telefon      [                  ]    Bundesland          [  ]
```

Hinweis

Von einem Ihnen bisher bekannten Standard-Format unterscheidet sich das FORMAT01 äußerlich in mehreren Merkmalen, die alle die Folge einer entsprechenden Gestaltung des Format-Programmes sind: Das Bildschirmfeld 'Kundennummer' ist invertiert, das Format ist mit seinem Namen überschriftet (freier Text) und die Bildschirmfelder sind teilweise auch nebeinander angeordnet.

Im FORMAT01 gibt es zu jeder Tabellenspalte der Tabelle 'kunde' (Hinweis rechts oben ** 1: kunde Tabelle**) ein Bildschirmfeld. Im FORMAT01 können Sie sich also die vollständigen Kundendaten des Sportartikelversands ermitteln.

▷ Führen Sie mit 'Suchen' und START zunächst eine Datenbank-Abfrage ohne Such-Bedingungen durch.

Sie erhalten diesen Bildschirm:

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**

                                FORMAT01
                                --------

 Kundennummer [101          ]


 Firma        [Pauli Sport          ]


 Vorname      [Ludwig          ]      Nachname            [Pauli          ]
 Strasse      [Forstweg 47         ]  evtl. Zweitanschrift [                ]
 Plz          [8900 ]                 Ort                 [Augsburg       ]


 Telefon      [0821/8075          ]   Bundesland          [BY]




 Gefundene Saetze: 18
```

Am unteren Bildschirmrand erhalten Sie die Meldung:
Gefundene Saetze: 18
Der Sportartikelversand beliefert also 18 Kunden.

▷ Blättern Sie mit 'Vorw.' in der aktuellen Liste. Sie werden dann u.a. feststellen, das es zwei Kunden mit dem Vornamen 'Frank' gibt, von denen der eine seinen Standort in Augsburg und der andere in München hat.

▷ Führen Sie nun erneut eine Datenbank-Abfrage durch, geben Sie aber diesmal in das Bildschirmfeld 'vorname' den Wert 'Frank' als Such-Bedingung ein.

Nun gibt es zwei Treffersätze, denn zwei Kunden tragen den Vornamen Frank.

▷ Führen Sie nun wiederum eine Datenbank-Abfrage durch, geben Sie aber diesmal in das Bildschirmfeld 'vorname' den Wert 'Frank' und zusätzlich in das Bildschirmfeld 'Ort' den Wert 'Augsburg' ein.

Jetzt gibt es nur noch einen Treffersatz, der die beiden gestellten Such-Bedingungen 'Vorname = Frank' und 'Ort = Augsburg' erfüllt.

5.2.4 Such-Bedingungen mit Such-Operatoren

Mittels Such-Operatoren können Sie eine Datenbank-Abfrage weiter präzisieren. So können Sie z.B. mit dem Operator '>' (größer) nach allen Sätzen suchen, bei denen die Postleitzahl größer als '8000' ist.

Diese Such-Operatoren gibt es:

SUCH-OPERATOR	BEDEUTUNG	DATENTYP
*	'Joker' ersetzt bel. Zeichenfolge	(VAR)CHAR
?	'Joker' ersetzt genau ein Zeichen	(VAR)CHAR
=	gleich	alle
>	größer	alle
<	kleiner	alle
>=	größer oder gleich	alle
<=	kleiner oder gleich	alle
<>	ungleich	alle
|	oder	nicht (VAR)CHAR
:	zwischen	nicht DATETIME,INTERVALL
..	zwischen	DATETIME,INTERVALL
>>	höchster Wert	alle
<<	niedrigster Wert	alle

'alle' heißt, alle Datentypen außer den BLOB-Datentypen TEXT und BYTE. In diese Bildschirmfelder kann man grundsätzlich keine Suchbedingungen eintragen.

'nicht' DATENTYP heißt, alle anderen außer den genannten und den BLOB-Datentypen.

Die folgenden Erläuterungen und Beispiele können Sie im FORMAT01 nachvollziehen. Eine Übersicht über die Datensätze der Tabelle 'kunde' finden Sie im Anhang.

Hilfsbildschirmfeld

Wenn ein Bildschirmfeld für die eingegebene Such-Bedingung zu klein ist, dann geschieht folgendes:
Am unteren Bildschirmrand erhalten Sie ein Hilfsbildschirmfeld, das sich über die ganze Bildschirmbreite erstreckt. Die Schreibmarke springt in das Hilfsbildschirmfeld und die bisherigen Eingaben werden mit in das Hilfsbildschirmfeld übernommen. Sie können mit der Eingabe Ihrer Such-Bedingung fortfahren.

Der 'Joker'-Operator '*'

Diesen Operator können Sie nur in (VAR)CHAR-Bildschirmfeldern verwenden.

Den Operator '*' können Sie wie einen Joker verwenden; er ersetzt eine beliebige Zeichenfolge, auch Leerzeichen und keine Zeichen.

Beispiel

```
Firma          [*ort                ]
```

Die Such-Bedingung lautet: Finde alle Sätze, die in der entsprechenden Spalte einen Wert enthalten, der mit einer beliebigen Zeichenkette beginnt und der mit der Zeichenkette 'ort' endet.
PERFORM findet die zwei Sätze mit den Werten 'Pauli Sport' und 'Olympia Sport'.

Der 'Joker'-Operator '?'

Diesen Operator können Sie nur in (VAR)CHAR-Bildschirmfeldern verwenden.

Der Operator '?' ersetzt genau ein Zeichen.

Beispiel

```
Vorname        [?rank               ]
```

Die Such-Bedingung lautet: Finde alle Werte, die mit einem beliebigen einzelnen Zeichen beginnen und die mit der Zeichenkette 'rank' enden.
PERFORM findet zwei Sätze mit den Werten 'Frank'.

Der Operator '='

Den Operator '=' können Sie verwenden, um Sätze zu finden, bei denen in der entsprechenden Spalte ein NULL-Wert bzw.ein Leerzeichen eingetragen ist.

Beispiel

```
evtl. Zweitanschrift  [=                    ]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte einen NULL-Wert enthalten.
PERFORM findet 15 Sätze, die die Such-Bedingung erfüllen.

Der Operator '>'

Den Operator '>' können Sie verwenden, um Sätze zu finden, deren Werte in der entsprechenden Spalte größer sind als in der Suchbedingung vorgegeben.

Beispiel

Wenn Sie die Beispiele nacheinander am Bildschirm nachvollzogen haben, dann erhalten Sie beim folgenden Beispiel erstmals das Hilfsbildschirmfeld am unteren Bildschirmrand.

```
Plz             [>8000]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte einen Wert enthalten, der größer als '8000' ist.
PERFORM findet 11 Sätze, die die Such-Bedingung erfüllen.

Der Operator '<'

Den Operator '<' können Sie verwenden, um Sätze zu finden, deren Werte in der entsprechenden Spalte kleiner sind als in der Suchbedingung vorgegeben.

Beispiel

```
Plz             [<8000]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte einen Wert enthalten, der kleiner als '8000' ist.
PERFORM findet 2 Sätze, die die Such-Bedingung erfüllen.

Der Operator '> ='

Den Operator '> =' können Sie verwenden, um Sätze zu finden, deren Werte in der entsprechenden Spalte größer oder gleich sind als in der Suchbedingung vorgegeben.

Beispiel

```
Plz             [>=8000]
```

0Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte einen Wert enthalten, der größer als '8000' oder gleich '8000' ist.
PERFORM findet 16 Sätze, die die Such-Bedingung erfüllen.

Der Operator '< =>'

Den Operator '< =>' können Sie verwenden, um Sätze zu finden, deren Werte in der entsprechenden Spalte kleiner oder gleich sind als in der Suchbedingung vorgegeben.

Beispiel

```
Plz              [<=8000]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte einen Wert enthalten, der kleiner als '8000' oder gleich '8000' ist.
PERFORM findet 7 Sätze, die die Such-Bedingung erfüllen.

Der Operator '< >'

Den Operator '< >' können Sie verwenden, um Sätze zu finden, deren Werte in der entsprechenden Spalte ungleich dem in der Suchbedingung vorgegeben sind.

Beispiel

```
evtl. Zweitanschrift  [<>                    ]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte keinen NULL-Wert enthalten.
PERFORM findet 3 Sätze, die die Such-Bedingung erfüllen.

Der Operator '|'

Den Operator '|' können Sie verwenden, um Sätze zu finden, die entweder den einen oder den anderen Wert in der entsprechenden Spalte enthalten.

Beispiel

```
Kundennummer [104|108          ]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte entweder den Wert '104' oder den Wert '108' enthalten. PERFORM findet 2 Sätze, die die Such-Bedingung erfüllen.

Der Operator ':'

Den Operator ':' können Sie verwenden, um Sätze zu finden, deren Werte zwischen dem ersten und dem zweiten angegebenen Wert liegen bzw. genau einer der angegebenen Werte sind.

Beispiel

```
Plz             [8200:8900]
```

Die Suchbedingung lautet: finde alle Sätze, die in dieser Spalte einen Wert enthalten, der größer oder gleich '8200' und kleiner oder gleich '8900' ist.
PERFORM findet 8 Sätze, die die Such-Bedingung erfüllen.

Der Operator '..'

Den Operator '..' können Sie nur in DATETIME- und INTERVAL-Bildschirmfeldern verwenden. Er hat bei diesen Bildschirmfeldern logisch die gleiche Bedeutung wie der Operator ':'.

Der Operator '> >'

Den Operator '> >' können Sie verwenden, um den Satz mit dem höchsten Wert in der entsprechenden Spalte zu finden.

Beispiel

```
Kundennummer [>>                ]
```

Die Suchbedingung lautet: finde den Satz mit dem höchsten Wert in dieser Spalte.
PERFORM findet den Satz mit der Kundennummer 118.

Der Operator '< <'

Den Operator '< <' können Sie verwenden, um den Satz mit dem niedrigsten Wert in der entsprechenden Spalte zu finden.

Beispiel

```
Kundennummer [<<                ]
```

Die Suchbedingung lautet: finde den Satz mit dem niedrigsten Wert in dieser Spalte.
PERFORM findet den Satz mit der Kundennummer 101.

Such-Operatoren und der ASCII-Code

Was letztendlich größer oder kleiner bzw. höchster oder niedrigster Wert ist, das hängt standardmäßig (abhängig von der Datensichtstation) von der Reihenfolge der Zeichen im internationalen ASCII-Code ab.
Sie finden eine Tabelle mit den ASCII-Codes im Anhang.

Dieser Tabelle können Sie die festgelegte Reihenfolge entnehmen - es folgen nacheinander:

- Ziffern - 0123456789
- Großbuchstaben - ABCDE...XYZ
- Kleinbuchstaben abcde...xyz

Ab Zeichennummer 48 (dezimal) der Tabelle kommen zunächst die Ziffern 0-9, dann folgen die Großbuchstaben A-Z und zuletzt kommen die Kleinbuchstaben a-z. Die deutschen Umlaute gibt es im ASCII-Code nicht.

Wenn Sie die Operatoren größer bzw. kleiner bei CHAR-Bildschirmfeldern verwenden, dann verhält es sich so, daß der Kleinbuchstabe 'z' gemäß der festgelegten Reihenfolge das letzte und damit 'größte' alphabetische Zeichen ist. Damit gelten die folgenden Regeln:

- 'z' ist größer als 'a'
- 'a' ist größer als 'A'
- 'A' ist größer als die Ziffer '0'

Beispiel

zecke > antilope

ameise > Amsel

zwei > 4

Bei DATE- und DATETIME-Bildschirmfeldern bedeutet größer "später" und kleiner "früher".

Beispiel

```
11.12.90 > 20.9.90
```

Hinweis

In Kapitel 9.5 ist beschrieben, wie Sie vorgehen müssen, um bei 7-Bit-Terminals mit Umlauten arbeiten zu können. Das Arbeiten mit Umlauten hat jedoch den Nachteil, daß die Such-Operatoren u.U. nicht die erwarteten Ergebnisse produzieren, da die den Umlauten 'öÖäÄüÜß' entsprechenden Zeichen '|\{[}]~' im ASCII-Code nicht innerhalb der Buchstaben-Reihenfolge liegen.

Aufgabe 5-1

Führen Sie im FORMAT01 Datenbank-Abfragen mit den folgenden Suchbedingungen durch:

A) Der Nachname soll ein 'g' enthalten.

B) Die Kundennummer soll zwischen/mit 105 und 115 liegen und der Kunde soll in 'Muenchen' wohnen.

C) Die Telefon-Vorwahl soll beliebig sein und die Rufnummer sechsstellig.

5.3 Einen Satz 'Neuaufnehmen'

Wenn Sie einen Satz in die Tabelle neu aufnehmen wollen, dann gehen Sie wie folgt vor:

▷ Führen Sie die Funktion 'Neuaufnehmen' aus.

▷ Geben Sie den neuen Satz ein.

▷ In einem BLOB-Bildschirmfeld drücken Sie [I]. Geben Sie die BLOB-Daten
ein und verlassen Sie das Programm. Sichern Sie dabei die BLOB-Daten unter dem gleichen Dateinamen, mit dem das Programm aufgerufen wurde.

▷ Drücken Sie [START]

Im Anschluß daran erhalten Sie eine Meldung darüber, daß der Satz neu aufgenommen wurde.

Hinweis

Wenn ein eingegebener Wert nicht dem definierten Datentyp entspricht, dann können Sie das entsprechende Bildschirmfeld nicht verlassen und erhalten die Meldung: Fehler im Feld
Sie haben in diesem Fall nur die Möglichkeit, einen korrekten Wert einzugeben oder aber die Eingabe mit [DEL] zu beenden.

5.4 Einen Satz 'Korrigieren'

Korrigieren können Sie nur den aktuellen Satz. Sie müssen also vorab eine Datenbank-Abfrage durchführen und den zu korrigierenden Satz als den aktuellen Satz am Bildschirm anzeigen lassen. Wenn das geschehen ist, dann gehen Sie wie folgt vor:

▷ Führen Sie die Funktion 'Korrigieren' aus.

▷ Führen Sie die erwünschten Korrekturen durch.

▷ In einem BLOB-Bildschirmfeld drücken Sie [I]. Korrigieren Sie die BLOB-Daten und verlassen Sie das Programm. Sichern Sie dabei die BLOB-Daten unter dem gleichen Dateinamen, mit dem das Programm aufgerufen wurde.

▷ Drücken Sie [START]

Im Anschluß daran erhalten Sie eine Meldung darüber, daß der Satz korrigiert wurde.

5.5 Einen Satz 'Löschen'

Löschen können Sie nur den aktuellen Satz. Sie müssen also vorab eine Datenbank-Abfrage durchführen und den zu löschenden Satz als den aktuellen Satz am Bildschirm anzeigen lassen. Wenn das geschehen ist, dann gehen Sie wie folgt vor:

▷ Führen Sie die Funktion 'Löschen' aus.

Um versehentliche Fehl-Bedienungen auszuschließen, erhalten Sie nun eine Funktionszeile mit der Abfrage: LOESCHEN: Ja Nein

▷ Führen Sie die Funktion 'Ja' aus.

Im Anschluß daran erhalten Sie eine Meldung darüber, daß der Satz gelöscht wurde.

5.6 PRINT - Ausgabe in Datei

Die Funktion PRINT ermöglicht Ihnen die Ausgabe des aktuellen Bildschirmes (bzw. der ganzen aktuellen Liste) in eine Datei.

Sie gehen wie folgt vor:

▷ Drücken Sie PRINT
(oder führen Sie die Funktion PRINT aus)

Sie erhalten diesen Bildschirm (hier mit dem Satz Kundennummer '101'):

```
Ausgabeprogramm? (Std: perform.out):

                                FORMAT01
                                --------

Kundennummer [101                ]


Firma        [Pauli Sport         ]


Vorname      [Ludwig         ]      Nachname             [Pauli          ]
Strasse      [Forstweg 47     ]     evtl. Zweitanschrift [                    ]
Plz          [8900 ]                Ort                  [Augsburg       ]


Telefon      [0831/8057       ]     Bundesland           [BY]
```

Sie erhalten hier die Information, daß die Ausgabe standardmäßig in die Datei 'perform.out' geschrieben wird. Sie können jedoch optional einen anderen Dateinamen angeben.

▷ Wenn gewünscht, geben Sie einen anderen Dateinamen an.

▷ Drücken Sie ↵

Sie erhalten diesen Bildschirm:

```
FORMAT AUSGABEPROGRAMM:   Anhaengen  Erstellen
Haengt neue Daten an bestehende Ausgabedatei.   ** 1: kunde Tabelle**

                                FORMAT01
                                --------

Kundennummer [101        ]


Firma        [Pauli Sport         ]


Vorname      [Ludwig         ]      Nachname             [Pauli          ]
Strasse      [Forstweg 47         ] evtl. Zweitanschrift [                    ]
Plz          [8900 ]                Ort                  [Augsburg       ]


Telefon      [0831/8057         ]   Bundesland          [BY]
```

'Anhaengen' schreibt die Daten an das Ende der zuvor bekanntgegebenen (bestehenden) Datei.
'Erstellen' schreibt die Daten an in die zuvor bekanntgegebene Datei.

▷ Führen Sie eine der beiden Funktionen aus.

Sie erhalten diesen Bildschirm:

```
FORMAT-AUSGABE:  aktuelle-Liste  Bildschirm
Die gesamte aktuelle Liste ausgeben.          ** 1: kunde Tabelle**

                                FORMAT01
                                --------

Kundennummer [101          ]


Firma        [Pauli Sport         ]


Vorname      [Ludwig         ]      Nachname             [Pauli          ]
Strasse      [Forstweg 47        ]  evtl. Zweitanschrift [                    ]
Plz          [8900 ]                Ort                  [Augsburg       ]


Telefon      [0831/8057         ]   Bundesland           [BY]
```

'aktuelle-Liste' gibt die gesamte aktuelle Liste aus. 'Bildschirm' gibt nur den aktuelle Bildschirm aus.

▷ Führen Sie eine der beiden Funktionen aus.

Sie erhalten diesen Bildschirm:

```
AUSGABE FORMAT:   Ascii-Format  Bildschirm-Format
Gewaehlte Ausgabe wird in ascii geschrieben.    ** 1: kunde Tabelle**

                                FORMAT01
                                --------

Kundennummer [101          ]


Firma        [Pauli Sport         ]


Vorname      [Ludwig         ]      Nachname             [Pauli          ]
Strasse      [Forstweg 47        ]  evtl. Zweitanschrift [                    ]
Plz          [8900 ]                Ort                  [Augsburg       ]


Telefon      [0831/8057         ]   Bundesland           [BY]
```

'Ascii-Format' gibt die Daten in einem Format aus, wie es von der SQL-Anweisung LOAD erwartet wird:

`wert1|wert2|wert3|...|wertN|`
'Bildschirm-Format' gibt die Daten so aus, wie Sie sie am Bildschirm sehen.

▷ Führen Sie eine der beiden Funktionen aus.

Im Anschluß daran kehren Sie automatisch wieder in das PERFORM-Menü zurück.

5.7 'Format' - ein Format, mehrere Bildschirmseiten

Ebenso wie es mehrseitige Papierformulare geben kann, so kann es auch Formate mit mehreren Bildschirmseiten geben. Das ist z.B. dann sinnvoll, wenn ein Format durch eine sehr große Anzahl von Bildschirmfeldern unübersichtlich würde. In diesem Fall ist es zweckmäßig, die Bildschirmfelder auf mehrere Bildschirmseiten verteilen.

Mit der Funktion 'Format' können Sie zwischen den Bildschirmseiten eines mehrseitigen Formates blättern.

Die folgende Übung erläutert Ihnen die Funktion 'Format' anhand des zweiseitigen Formates FORMAT02. Wie das FORMAT01 ist es ein Format für die Tabelle 'kunde'; zu jeder Tabellenspalte der Tabelle 'kunde' gibt es ein Bildschirmfeld. Jedoch sind die Bildschirmfelder im FORMAT02 auf zwei Bildschirmseiten verteilt.
Auf der zweiten Bildschirmseite gibt es nochmals ein Bildschirmfeld 'Firma', das den Firmennamen anzeigt.

Übung 5-2

▷ Rufen Sie das FORMAT02 auf und führen Sie eine Datenbank-Abfrage ohne Such-Bedingung durch.

Sie erhalten diesen Bildschirm:

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**

                        FORMAT02 - Bildschirmseite 1
                        ----------------------------



     Kundennummer       [101        ]


     Firma              [Pauli Sport          ]

     Vorname            [Ludwig         ]
     Nachname           [Pauli          ]




 Gefundene Saetze: 18
```

▷ Führen Sie die Funktion 'Format' aus. Diese befindet sich in der zweiten Funktionszeile.

Jetzt erhalten Sie diesen Bildschirm:

```
PERFORM:   ...  Loeschen  Tabelle  Format  Aktuell  Master  Detail  PRINT  END
Zeigt die naechste Seite des Formates.          ** 1: kunde Tabelle**

                        FORMAT02 - Bildschirmseite 2
                        ----------------------------


    Firma               [Pauli Sport          ]


    Anschrift           [Forstweg 47          ]
    evt. Zweitanschrift [                     ]

    Plz                 [8900 ]
    Ort                 [Augsburg        ]
    Bundesland          [BY]

    Telefon             [0821/8075         ]
```

▷ Wenn Sie jetzt nochmals die Funktion 'Format' ausführen, dann erhalten Sie wieder den ersten Bildschirm.

Hinweis

Beim Neuaufnehmen oder Korrigieren eines Satzes müssen Sie nicht unbedingt von Hand mit der Funktion 'Format' zur nächsten Bildschirmseite blättern. Sie können einfach die Bildschirmfelder nacheinander ausfüllen. Wenn sich das folgende Bildschirmfeld auf einer anderen Bildschirmseite befindet, dann wechselt PERFORM automatisch zu dieser Bildschirmseite.

Bei Formaten mit mehr als zwei Seiten können Sie auch mit der Eingabe einer Zahl angeben, welche spezielle Bildschirmseite Sie erhalten wollen. Wenn Sie z.B. eingeben: '3 f' dann erhalten Sie unmittelbar die 3. Seite des Formates.

5.8 Mehr-Tabellen-Formate

Ein Format kann Bildschirmfelder aus mehreren Tabellen (der gleichen Datenbank) enthalten. Ein solches Format nennt man ein 'Mehr-Tabellen-Format'.

Komfortabel wird ein Mehr-Tabellen-Format dann, wenn die beteiligten Tabellen über sogenannte "Join-Bildschirmfelder" miteinander verbunden sind. Ein Join-Bildschirmfeld bewirkt, daß bei jeder Datenbank-Abfrage in der einen Tabelle auch ein erster verbundener Satz der anderen Tabelle mit angezeigt wird; und es erleichtert Datenbank-Abfragen in der jeweils verbundenen Tabelle.

Lassen Sie sich das bitte anhand des FORMAT03 erläutern - rufen Sie dazu das FORMAT03 auf:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**
                    FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma     [                    ]
Vorname   [                ]          Nachname             [                ]
Adresse   [                    ]      evtl. Zweitanschrift [                    ]
Plz       [     ]                     Ort                  [                ]

Telefon   [                   ]       Bundesland           [ ]

__auftrag_____________ Kundennummer [           ] ______________________________

Auftragsnummer                       Auftragsdatum
Lieferhinweis
Offen?                               Fremdnummer
Lieferdatum                          Liefergewicht
Zustellgebuehr                       Zahldatum
```

Dieses Format enthält im oberen Teil Bildschirmfelder aus der Tabelle 'kunde' und im unteren Teil Bildschirmfelder aus der Tabelle 'auftrag'. Auf der Trennlinie zwischen den beiden Format-Teilen gibt es ein Bildschirmfeld 'Kundennummer', das invertiert abgebildet ist.

Über dieses Join-Bildschirmfeld sind die beiden Tabellen wie folgt miteinander verbunden:

- in der Tabelle 'kunde' gibt es eine Tabellenspalte 'kunden_nr' und
- in der Tabelle 'auftrag' gibt es ebenfalls eine Tabellenspalte 'kunden_nr'.

Die Verbindung der beiden Tabellen miteinander besteht einfach darin, daß die beiden Tabellenspalten 'kunden_nr' der unterschiedlichen Tabellen dem einen einzigen Bildschirmfeld 'Kundennummer' zugeordnet wurden.

Die folgende Übung erläutert die Auswirkungen eines Join-Bildschirmfeldes anhand des FORMAT03:

Übung 5-3

▷ Führen Sie im FORMAT03 eine Datenbank-Abfrage ohne Such-Bedingungen durch.

Sie erhalten diesen Bildschirm:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**
                    FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma     [Pauli Sport          ]
Vorname   [Ludwig         ]          Nachname             [Pauli          ]
Adresse   [Forstweg 47          ]    evtl. Zweitanschrift [                ]
Plz       [8900 ]                    Ort                  [Augsburg       ]

Telefon   [0821/8075          ]      Bundesland           [BY]

__auftrag___________ Kundennummer [101          ] ____________________________

Auftragsnummer       1002           Auftragsdatum        01.06.1990
Lieferhinweis        Hintertuere klingeln
Offen?               n              Fremdnummer          9270
Lieferdatum          06.06.1990     Liefergewicht        50,60
Zustellgebuehr       15,30          Zahldatum            03.07.1990



 Gefundene Saetze: 18
```

Im oberen Format-Teil zeigt PERFORM den ersten gefundenen Satz der Tabelle 'kunde', den Satz 101 des Kunden 'Pauli Sport' an. Im unteren Teil zeigt PERFORM einen verbundenen Satz der Tabelle 'auftrag' an.
Damit wird in dem Format angezeigt, daß mit dem Kunden Firma 'Pauli Sport' bereits wenigstens ein Auftrag abgewickelt wurde, denn es gibt einen verbundenen Satz der Tabelle 'auftrag'.

▷ Blättern Sie mit 'Vorw.' und 'Rueckw.' in der aktuellen Liste.

Sie stellen fest, daß es nicht zu jedem Satz der Tabelle 'kunde' auch einen verbundenen Satz gibt. Das bedeutet, daß manche Kunden zwar bereits in die Datenbank aufgenommen wurden, bisher aber noch kein Auftrag mit ihnen abgewickelt wurde.

5.8.1 'Tabelle' - die aktuelle Tabelle wechseln

Die aktuelle Tabelle wechseln

Beim bisherigen Kenntnisstand können Sie zwar in der Tabelle 'kunde' eine Datenbank-Abfrage durchführen und sich den jeweils ersten der verbundenen Sätze der Tabelle 'auftrag' anzeigen lassen, nicht aber umgekehrt.

Denn bei Mehr-Tabellen-Formaten gibt es zwar Bildschirmfelder aus mehreren Tabellen; aber die Datenbank abfragen, Sätze löschen oder korrigieren können Sie jeweils nur in einer Tabelle, nämlich in der sogenannten 'aktuellen Tabelle'.

Mit der Funktion 'Tabelle' (in der zweiten Funktionszeile) können Sie die aktuelle Tabelle wechseln. Das hat die folgenden Auswirkungen:

- **Oben rechts oberhalb der Infozeile zeigt PERFORM den Namen der neuen aktuellen Tabelle an - z.B. so: ** 2: auftrag Tabelle****
- **PERFORM wechselt die eckigen Klammern von den Bildschirmfeldern der bisherigen aktuellen Tabelle zu den Bildschirmfeldern der neuen aktuellen Tabelle.**

Hinweis

PERFORM ordnet jeder Tabelle automatisch eine Nummer zu. So ist z.B. im FORMAT03 die Tabelle 'kunde' die Tabelle 1 und die Tabelle 'auftrag' die Tabelle 2.
In welcher Reihenfolge PERFORM diese Nummern vergibt, das hängt von der entsprechenden Reihenfolge im Format-Programm ab.

Wenn Sie ein Format neu aufrufen, dann ist zunächst die Tabelle 1 (entsprechend der Abfolge im Format-Programm) die aktuelle Tabelle. Wenn Sie die Funktion 'Tabelle' ausführen, dann wechselt PERFORM zur Tabelle 2 und von dort zu einer eventuell vorhanden Tabelle 3 u.s.w. und am Schluß wieder zur Tabelle 1.

Wenn sich die Bildschirmfelder der neuen aktuellen Tabelle auf einer anderen Bildschirmseite befinden, dann blättert PERFORM automatisch auf diese Bildschirmseite.

Bei Formaten mit mehr als zwei Tabellen können Sie mit der Eingabe einer Zahl angeben, welche spezielle Tabelle die neue aktuelle Tabelle sein soll. Wenn Sie z.B. eingeben: '3 t' dann machen Sie unmittelbar die 3. Tabelle des Formates (sofern vorhanden) zur neuen aktuellen Tabelle.

Die folgende Übung erläutert Ihnen die Bedeutung der Funktion 'Tabelle' und sie informiert Sie weiter über die Datenstruktur der Datenbank 'versand'.

Übung 5-4

Um die Übung nachvollziehen zu können, benötigen Sie die folgende Ausgangssituation:

- Die Tabelle 'kunde' ist nach wie vor die aktuelle Tabelle.
- Der Satz mit der Kundennummer 101 Firma 'Pauli Sport' ist der aktuelle Satz. Wenn das nicht der Fall ist, dann blättern Sie mit 'Rueckw.' in der aktuellen Liste, bis der Satz mit Kundenummer 101 der aktuelle Satz ist.

Diesen Bildschirm sollten Sie vor sich haben:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**
                  FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma     [Pauli Sport          ]
Vorname   [Ludwig         ]          Nachname              [Pauli          ]
Adresse   [Forstweg 47          ]    evtl. Zweitanschrift  [                    ]
Plz       [8900 ]                    Ort                   [Augsburg       ]

Telefon   [0821/8075         ]       Bundesland            [BY]

__auftrag___________ Kundennummer [101        ] ___________________________

Auftragsnummer      1002             Auftragsdatum        01.06.1990
Lieferhinweis       Hintertuere klingeln
Offen?              n                Fremdnummer          9270
Lieferdatum         06.06.1990       Liefergewicht        50,60
Zustellgebuehr      15,30            Zahldatum            03.07.1990



 Gefundene Saetze: 18
```

▷ Führen Sie die Funktion 'Tabelle' aus.

Sie erhalten diesen Bildschirm:

```
PERFORM:   ... Loeschen Tabelle Format Aktuell Master Detail PRINT END
Waehlt die aktuelle Tabelle.                  ** 2: auftrag Tabelle**
                  FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma      Pauli Sport
Vorname    Ludwig                    Nachname              Pauli
Adresse    Forstweg 47               evtl. Zweitanschrift
Plz        8900                      Ort                   Augsburg

Telefon    0821/8075                 Bundesland            BY

__auftrag___________ Kundennummer [101        ] ____________________________

Auftragsnummer     [1002       ]    Auftragsdatum      [01.06.1990]
Lieferhinweis      [Hintertuere klingeln                      ]
Offen?             [n]              Fremdnummer        [9270       ]
Lieferdatum        [06.06.1990]     Liefergewicht      [50,60      ]
Zustellgebuehr     [15,30    ]      Zahldatum          [03.07.1990]
```

Nun ist die Tabelle 'auftrag' die aktuelle Tabelle. Sie erkennen das an den eckigen Klammern und an dem Hinweis oben rechts: ** 2: auftrag Tabelle**

▷ Führen die Funktion 'Suchen' aus - drücken Sie noch nicht START!

Sie erhalten diesen Bildschirm:

```
SUCHEN:  START sucht.  F17 loescht alle Felder.  HELP fuer Hilfe.
         DEL bricht ab.                          ** 2: auftrag Tabelle**
                  FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma      Pauli Sport
Vorname    Ludwig                    Nachname              Pauli
Adresse    Forstweg 47               evtl. Zweitanschrift
Plz        8900                      Ort                   Augsburg

Telefon    0821/8075                 Bundesland            BY

__auftrag_____________ Kundennummer [101       ] ___________________________

Auftragsnummer     [           ]    Auftragsdatum      [          ]
Lieferhinweis      [                                          ]
Offen?             [ ]              Fremdnummer        [          ]
Lieferdatum        [          ]     Liefergewicht      [          ]
Zustellgebuehr     [        ]       Zahldatum          [          ]
```

Die Bildschirmfelder der Tabelle 'auftrag' wurden gelöscht. Nicht gelöscht wurde allerdings das Bildschirmfeld 'Kundennummer', in dem sich auch die Schreibmarke befindet.
Warum dieses Bildschirmfeld nicht gelöscht wurde, darüber informiert Sie der folgende Abschnitt.

Um in der Tabelle 'auftrag' eine Datenbank-Abfrage durchführen zu können, die alle Sätze der Tabelle liefert:

▷ Löschen Sie zunächst mit WORD DELETE auch das Bildschirmfeld 'Kundennummer'.

▷ Um die Datenbank-Abfrage auszuführen, drücken Sie jetzt START

Sie erhalten diesen Bildschirm:

```
PERFORM:  Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 2: auftrag Tabelle**
                  FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma      Spielball
Vorname    Anton                      Nachname              Hochfeld
Adresse    Goethestr. 34              evtl. Zweitanschrift  Lilienstr.42
Plz        8000                       Ort                   Muenchen

Telefon    089/25430                  Bundesland            BY

__auftrag_____________ Kundennummer [104       ] ____________________________

Auftragsnummer     [1001       ]    Auftragsdatum      [20.01.1990]
Lieferhinweis      [Spedition                                    ]
Offen?             [n]              Fremdnummer        [B77836     ]
Lieferdatum        [01.02.1990]     Liefergewicht      [20,40      ]
Zustellgebuehr     [10,00    ]      Zahldatum          [22.03.1990]



 Gefundene Saetze: 15
```

Der aktuelle Satz ist der Satz mit der Auftragsnummer 1001 und der Kundennummer 104. Zu diesem Satz gibt es in der Tabelle 'kunde' einen verbundenen Satz (Firma 'Spielball').
Die aktuelle Liste enthält insgesamt 15 Sätze. Es existieren also 15 Aufträge.

▷ Blättern Sie in der aktuellen Liste.

Sie stellen dabei fest, daß nun im oberen Format-Teil ein jeweils verbundener Satz der Tabelle 'kunde' angezeigt wird, denn zu jedem Auftrag muß es notwendig auch einen Auftraggeber geben.

Mit der Funktion 'Tabelle' können Sie also die Tabelle wechseln und danach in der neuen aktuellen Tabelle eine Datenbank-Abfrage durchführen.
Wenn Sie mit 'Vorw.' oder 'Rueck.' in der neuen aktuellen Liste blättern, dann bekommen Sie auch je einen ersten verbundenen Satz der anderen Tabelle angezeigt (sofern vorhanden).

5.8.2 Warum Join-Bildschirmfelder nicht gelöscht werden

Sie werden in der vorangegangenen Übung vielleicht festgestellt haben, daß es Kunden gibt, die

- bisher noch gar keinen Auftrag erteilt haben,
- einen Auftrag erteilt haben,
- bereits mehrere Aufträge erteilt haben.

Das zeigt, daß es u.U. nicht nur einen, sondern mehrere verbundene Sätze gibt. Angezeigt wird jedoch nur ein einziger, nämlich der erste gefundene.

Wie können Sie ermitteln, wieviel verbundene Sätze es in der anderen Tabelle gibt (konkret: wieviel Aufträge ein Kunde erteilt hat)?. Durch eine Datenbank-Abfrage in der entsprechenden Tabelle (auftrag). Dabei spielt das Join-Bildschirmfeld eine zentrale Rolle, denn über die Werte des Join-Bildschirmfeldes sind die Sätze miteinander verbunden und über das Join-Bildschirmfeld sind folglich die verbundenen Sätze auch zu ermitteln.

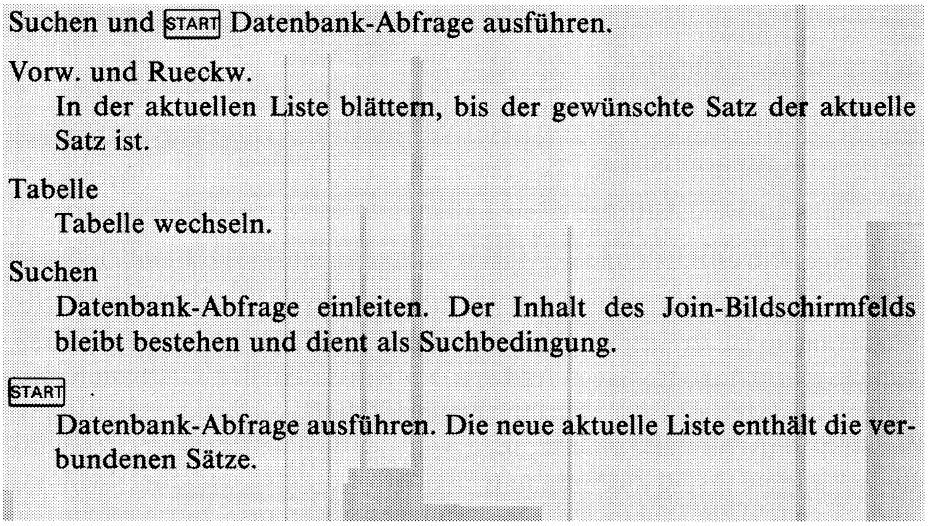
Wenn Sie wissen wollen, wieviel Aufträge ein Kunde erteilt hat, dann können Sie wie folgt vorgehen:

- Sie ermitteln in der Tabelle 'kunde' seine Kundennummer und
- wechseln zur Tabelle 'auftrag' und
- tragen den ermittelten Wert als Such-Bedingung in das Bildschirmfeld Kundenummer ein und
- führen eine Datenbank-Abfrage durch.

Sofern überhaupt ein verbundener Satz existiert, dann erhalten Sie eine aktuelle Liste, die alle Sätze (Aufträge) zu der eingetragenen Kundennummer enthält.

Um Ihnen diesen dritten Arbeitsgang, das Eintragen der Kundennummer von Hand, zu ersparen, wird der Wert in einem Join-Bildschirmfeld eben nicht gelöscht, nachdem Sie eine Datenbank-Abfrage durchgeführt haben und dann die Tabelle gewechselt haben. Denn damit steht der nach wie vor im dem Join-Bildschirmfeld angezeigte Wert gleich als Such-Bedingung für die darauffolgende Datenbank-Abfrage in der verbundenen Tabelle zur Verfügung.

Schematischer Ablauf 'Tabelle' wechseln:



In der folgenden Übung können Sie ermitteln, wieviele Aufträge zum Kunden '104 Firma Spielball' geführt werden.

Übung 5-5

Sie benötigen die folgende Ausgangssituation: Die Tabelle 'kunde' ist die aktuelle Tabelle. Und der Satz Kundennummer 104 Firma 'Spielball' ist der aktuelle Satz.

▷ Wechseln Sie zur Tabelle 'auftrag'.

▷ Leiten Sie mit 'Suchen' eine Datenbank-Abfrage in der neuen aktuellen Tabelle 'auftrag' ein.

Sie stellen fest: Alle Bildschirmfelder der neuen aktuellen Tabelle wurden gelöscht. Nicht aber das Bildschirmfeld 'Kundennummer', hier wird nach wie vor der Wert '104' angezeigt.
Dieser Wert dient als Such-Bedingung für die jetzt folgende Datenbank-Abfrage.

▷ Führen Sie mit START die Datenbank-Abfrage aus.

Sie erhalten die Meldung, daß 4 Sätze gefunden wurden. Mit dem Kunden '104' werden also 4 Aufträge abgewickelt.

5.9 'Master - Detail', hierarchische Beziehung der Tabellen

Bei der Beispiel-Datenbank 'versand' verhält es sich so, daß

- ein Kunde mehrere Aufträge erteilt haben kann und
- ein Auftrag aus mehreren Posten bestehen kann.

Nun sollte man sehr leicht in Erfahrung bringen können, welche Aufträge ein Kunde erteilt hat und welche Posten zu den Aufträgen gehören.
Ein Weg ist Ihnen bereits bekannt; Sie können die Tabelle wechseln und in der neuen aktuellen Tabelle eine erneute Datenbank-Abfrage durchführen.
Mit den Funktionen 'Detail' und 'Master' geht es aber noch einfacher:

Master-Detail-Beziehungen können im Format-Programm für Tabellen definiert sein, die über Join-Bildschirmfelder miteinander verbunden sind. Eine Tabelle wird zur Master-Tabelle erklärt und eine andere zur Detail-Tabelle.
Mit den Funktionen 'Detail' und 'Master' können Datenbank-Abfragen dann komfortabler ausgeführt werden:

Detail
Veranlaßt den Wechsel zur Detail-Tabelle (wie 'Tabelle') und führt dort zusätzlich automatisch eine Datenbank-Abfrage aus (wie START).

Master
Veranlaßt die Rückkehr zur Master-Tabelle. Eine Datenbank-Abfrage wird dabei nicht ausgeführt.

Das FORMAT04 ist ein Mehr-Tabellen-Format für die Tabellen 'kunde', 'auftrag' und 'posten'. In diesem Format sind die folgenden Master-Detail-Beziehungen definiert:

- 'kunde' ist Master-Tabelle von 'auftrag', damit ist 'auftrag' die Detail-Tabelle von 'kunde',
- 'auftrag' ist Master-Tabelle von 'posten', damit ist 'posten' die Detail-Tabelle von 'auftrag'.

Die folgende Übung erläutert die Master-Detail-Beziehung anhand des FORMAT04:

Übung 5-6

▷ Rufen Sie das FORMAT04 auf.

Das ist das FORMAT04:

```
PERFORM:   Suchen  Vorw.  Rueckw.  Blob  Neuaufnehmen  Korrigieren  ...
Sucht in der aktiven Datenbanktabelle.          ** 1: kunde Tabelle**
                  FORMAT04 - Tabellen 'kunde', 'auftrag' und 'posten'
__kunde_____________________________________________________________________

Firma    [                    ]   Adresse          [                    ]
Plz      [     ]                  Ort              [               ]
                                   Bundesland       [  ]
__auftrag__________ Kundennummer [           ] ___________________________

Offen?                             Auftragsdatum
Lieferdatum                        Liefergewicht
Zustellgebuehr                     Zahldatum

__posten__________ Auftragsnummer                ___________________________

Postennummer                       Herstellercode
Artikelnummer                      Artikel
Menge                              Gesamtpreis
```

▷ Führen Sie in der Tabelle 'kunde' eine Datenbank-Abfrage durch und machen Sie den Satz Kundennummer 106 Firma 'Sport Watt' zum aktuellen Satz.

▷ Führen Sie die Funktion 'Detail' aus.

Sie erhalten diesen Bildschirm:

```
PERFORM:   ... Loeschen Tabelle Format Aktuell Master Detail PRINT  END
Waehlt die Detail-Tabelle der aktuellen Tabelle ** 2: auftrag Tabelle**
                FORMAT04 - Tabellen 'kunde', 'auftrag' und 'posten'
__kunde_____________________________________________________________________

Firma     Sport Watt              Adresse           Am Bergsteig 88
Plz       7900                    Ort               Ulm
                                  Bundesland        BY
__auftrag__________ Kundennummer [106        ] _____________________________

Offen?              [j]           Auftragsdatum     [12.04.1990]
Lieferdatum         [30.04.1990]  Liefergewicht     [95,80      ]
Zustellgebuehr      [19,20    ]   Zahldatum         [           ]

__posten__________ Auftragssnummer [1004       ] _____________________________

Postennummer         1            Herstellercode     HR
Artikelnummer        1            Artikel
Menge                     1       Gesamtpreis        960,00




 Gefundene Saetze: 2
```

Die Funktion 'Detail' hat:

- den Wechsel von der Master-Tabelle 'kunde' zur Detail-Tabelle 'auftrag' bewirkt und
- dabei automatisch in der Detail-Tabelle eine Datenbank-Abfrage durchgeführt und eine aktuelle Liste erstellt und
- den ersten Satz der aktuellen Liste als den aktuellen Satz am Bildschirm angezeigt.

Die aktuelle Liste enthält 2 Sätze, es gibt also zwei Aufträge von der Firma 'Sport Watt'. Der Auftrag mit der Auftragsnummer 1004 ist der aktuelle Satz.

▷ Führen Sie nochmals die Funktion 'Detail' aus.

Sie erhalten diesen Bildschirm:

```
PERFORM:   ...  Loeschen  Tabelle  Format  Aktuell  Master  Detail  PRINT  END
Waehlt die Detail-Tabelle der aktuellen Tabelle ** 3: posten Tabelle**
-------------------------------------------------------------------------------
                  FORMAT04 - Tabellen 'kunde', 'auftrag' und 'posten'
__kunde_______________________________________________________________________

Firma     Sport Watt              Adresse            Am Bergsteig 88
Plz       7900                    Ort                Ulm
                                  Bundesland         BY
__auftrag___________ Kundennummer  106          _____________________________

Offen?              j             Auftragsdatum       12.04.1990
Lieferdatum         30.04.1990    Liefergewicht       95,80
Zustellgebuehr      19,20         Zahldatum

__posten__________ Auftragsnummer [1004       ] _____________________________

Postennummer        [1      ]     Herstellercode     [HR]
Artikelnummer       [1      ]     Artikel             Ski-Handschuhe
Menge               [     1]      Gesamtpreis        [960,00    ]



 Gefundene Saetze: 4
```

Nun wurde von der jetzigen Master-Tabelle 'auftrag' zur Detail-Tabelle 'posten' gewechselt und wiederum wurde automatisch in der Detail-Tabelle eine Datenbank-Abfrage durchgeführt.

Die aktuelle Liste enthält 4 Sätze, der Auftrag 1004 besteht also aus 4 Posten.

▷ Führen Sie die Funktion 'Master' aus.

Nun ist nichts weiter geschehen, als daß Sie zur Master-Tabelle 'auftrag' zurückgekehrt sind, die jetzt die neue aktuelle Tabelle ist.

Mit der Funktion 'Master' können Sie also zur Master-Tabelle zurückkehren. Eine Datenbank-Abfrage wird dabei nicht ausgelöst, vielmehr bleibt die vorab erstellte aktuelle Liste der Tabelle 'auftrag' erhalten.

▷ Blättern Sie in der noch vorhandenen aktuellen Liste der Tabelle 'artikel' vorwrrts.

Sie erhalten den zweiten Satz mit der Auftragsnummer 1014

▷ Führen Sie nochmals die Funktion 'Detail' aus.

Damit ist wieder die Tabelle 'posten' die aktuelle Tabelle. Mit der automatischen Datenbank-Abfrage wurde eine neue aktuelle Liste erstellt, die nun 2 Sätze enthält; der Auftrag 1014 besteht also aus 2 Posten.

Mehrere Tabellen bedeutet mehrere aktuelle Listen

PERFORM führt zu jeder Tabelle, in der Sie eine Datenbank-Abfrage durchgeführt haben, eine aktuelle Liste.

Sie können also zum Beispiel

- in einer Tabelle eine Datenbank-Abfrage durchführen und damit eine aktuelle Liste erstellen,
- mit 'Tabelle' (oder 'Detail') die Tabelle wechseln und wiederum eine Datenbank-Abfrage durchführen und damit eine zweite aktuelle Liste erstellen,
- wieder zur ersten Tabelle zurückkehren und in der nach wie vor gespeicherten aktuellen Liste der ersten Tabelle weiterblättern.

Die jeweiligen aktuellen Listen bleiben solange erhalten, bis Sie das Format verlassen oder bis Sie mit einer erneuten Datenbank-Abfrage in der jeweiligen Tabelle die bisherige aktuelle Liste überschreiben.

5.10 'Aktuell' - Rückkehr zum aktuellen Satz

Wenn Sie mehrfach die Tabelle gewechselt haben, dann kann es vorkommen, daß Sie bei einer der aktuellen Listen nicht mehr wissen, welches eigentlich der aktuelle Satz ist.
Wenn das geschehen ist, dann können Sie mit der Funktion 'Aktuell' den aktuellen Satz der aktuellen Liste erneut einlesen.

Die Funktion 'Aktuell' erfüllt auch noch einen weiteren Zweck: Bei Mehrplatz-Systemen können mehrere Anwender zur gleichen Zeit auf die gleichen Datensätze zugreifen. So kann es vorkommen, daß ein Satz, den Sie vermeintlich als den aktuellen Satz auf dem Bildschirm haben, zwischenzeitlich von einem anderen Anwender geändert wurde. Da die Funktion 'Aktuell' den Satz erneut aus der Tabelle einliest, erhalten Sie immer den wirklich aktuellen Stand des Satzes.

Da 'Aktuell' dabei auch den Bildschirm neu aufbaut, können Sie die Funktion ebenfalls dazu verwenden, Meldungen anderer Anwender oder des Betriebssystems vom Bildschirm zu löschen.

'Aktuell' liest den aktuellen Satz der aktuellen Liste erneut aus der Datenbank ein.
Der Bildschirm wird dabei neu aufgebaut.

5.11 Überprüfender Join und LOOKUP-Join

5.11.1 Überprüfender Join

Wenn Sie in einem Mehr-Tabellen-Format einen Satz neu in eine Tabelle aufnehmen, dann sollte gewährleistet sein, daß Sie in ein Join-Bildschirmfeld auch einen richtigen Wert eingeben; nämlich einen Wert, den es in der verbundenen Tabelle bereits gibt.

Wenn Sie zum Beispiel in die Tabelle 'auftrag' einen neuen Satz aufnehmen wollen, dann sollte es nicht passieren, daß Sie in das Bildschirmfeld Kundennummer einen Wert eintragen, den es in der Tabelle 'kunde' nicht gibt. Denn damit gäbe es dann einen Auftrag ohne Auftraggeber, und das wäre unsinnig.

Um zu gewährleisten, daß das nicht passieren kann, kann man im Format-Programm dem Join die Funktion 'prüfen' zuordnen. Wenn das geschehen ist, dann läßt PERFORM nur noch Werte zu, die in der verbundenen Tabelle bereits enthalten sind.

Wenn Sie einen nicht bereits existierenden Wert in ein solches Join-Bildschirmfeld eintragen, dann gibt PERFORM eine Fehlermeldung aus und verhindert die Übernahme des fehlerhaften Wertes in die Tabelle.

Wenn Sie beispielsweise im FORMAT04 einen neuen Satz für die Tabelle auftrag aufnehmen wollen, und wenn Sie dabei einen nicht existierenden Wert in das Bildschirmfeld 'Kundennummer' eintragen, dann erhalten Sie eine Fehlermeldung und PERFORM verhindert die Neuaufnahme des Satzes.

5.11.2 LOOKUP-Join

Neben dem Join mit Bildschirmfeldern, in die Sie Einträge machen können, gibt es noch einen zweiten Join mit Bildschirmfeldern, in die Sie keine Einträge machen können - man nennt diesen Join LOOKUP-Join:

- Die Bildschirmfelder eines LOOKUP-Join zeigen lediglich die Werte einer verbundenen Tabelle an.
- Aber Sie können unter keinen Umständen die Schreibmarke in ein solches Bildschirmfeld positionieren. Und das verhindert, daß Sie Änderungen an dem angezeigten Wert vornehmen (bzw. auch Such-Bedingungen eintragen) können.

Im FORMAT04 gibt es ein Bildschirmfeld mit einem LOOKUP-Join. Es ist das Bildschirmfeld 'Artikel' im dritten Format-Teil (posten).
In diesem Bildschirmfeld zeigt PERFORM an, welche Artikel-Bezeichnung sich hinter einer bestimmten Artikelnummer verbirgt.
Möglich wird das durch eine Anleihe mittels eines LOOKUP-Joins in einer eigentlich gar nicht beteiligten Tabelle; nämlich der Tabelle 'artikel'. Denn nur die Tabelle 'artikel' enthält eine Tabellenspalte 'bezeichnung', dem die hier angezeigten Werte entnommen sind.

Der LOOKUP-Join dient also der ergänzenden Information, er zeigt den Wert einer verbundenen Tabelle an.
Sie können eine solchermaßen verbundene Tabelle aber nicht zur aktuellen Tabelle machen und Sie können deshalb auch keine Änderungen an den angezeigten Werten vornehmen.

Zusammenfassung

- Mittels Such-Bedingungen können Sie eine Satzauswahl durchführen. Sie können mehrere Such-Bedingungen eingeben.
- Such-Bedingungen können auch Such-Operatoren enthalten, mit denen Sie die Such-Bedingung differenzierter gestalten können.
- Eine Datenbank-Abfrage können Sie nur in der aktuellen Tabelle durchführen.
- Wenn Sie eine Datenbank-Abfrage ausgeführt haben, dann erstellt PERFORM eine aktuelle Liste und zeigt den ersten Satz der aktuellen Liste als den aktuellen Satz am Bildschirm an.
  Bei Mehr-Tabellen-Formaten zeigt PERFORM zusätzlich u.U. auch einen ersten Satz einer verbundenen Tabelle an.
- Zu jeder Tabelle, in der Sie eine Datenbank-Abfrage durchgeführt haben, gibt es eine aktuelle Liste. Sie bleibt solange erhalten, bis Sie durch eine erneute Datenbank-Abfrage überschrieben wird oder bis Sie das Format verlassen.
- Mit der Funktion 'Format' können Sie zwischen den Bildschirmseiten eines Formates blättern.
- Mit der Funktion 'Tabelle' können Sie eine gewünschte Tabelle zur neuen aktuellen Tabelle machen.
- Mit den Funktionen 'Master' und 'Detail' können Sie zwischen einer Master-Tabelle und einer Detail-Tabelle wechseln. Die Funktion 'Detail' führt in der Detail-Tabelle automatisch eine Datenbank-Abfrage durch.
- Es gibt den überprüfenden Join und den LOOKUP-Join. Der überprüfende Join verhindert, daß Sie einen Wert in die Tabelle aufnehmen können, den es nicht bereits in der verbundenen Tabelle gibt. Der LOOKUP-Join dient der ergänzenden Information, er zeigt in eigens dafür geschaffenen Bildschirmfeldern Werte einer verbundenen Tabelle an; sie können nicht geändert oder gelöscht werden.

6 Format erzeugen

Dieses Kapitel erläutert Ihnen,

- welche Schritte Sie unternehmen müssen, um ein Format für eine speziell erwünschte Anwendung zu erzeugen,
- aus welchen Teilen ein Format-Programm besteht,
- welche Anweisungen im Format-Programm möglich sind und wie sie sich auswirken.

Das Kapitel erläutert nur einen Teil aller theoretisch verwendbaren Anweisungen; Sie erhalten einen Einblick in die Möglichkeiten und sind danach in der Lage, eigene Formate zu erzeugen.
Vollständige Informationen zu allen verwendbaren Anweisungen erhalten Sie im Handbuch [2] INFORMIX-SQL Nachschlagen.

6.1 Grundsätzliches

Das Erzeugen eines Formates vollzieht sich grundsätzlich in diesen zwei Schritten:

- Zunächst benötigen Sie ein Format-Programm. Das Format-Programm ist einfach ein Text, der alle syntaktisch erforderlichen (und erwünschten) Anweisungen enthält.
- Dann müssen Sie das Format-Programm compilieren (übersetzen). Damit steht Ihnen ein Format zur Verfügung mit dem Sie am Bildschirm arbeiten können und auf das PERFORM zugreifen kann.

Wenn Sie über 'Generieren' ein Standard-Format erzeugen, dann bleibt der oben beschriebene zweiteilige Erzeugungsprozeß für Sie unsichtbar.
Denn was Sie auch von Hand tun könnten, das erledigt FORMBUILD automatisch für Sie (FORMBUILD: INFORMIX-Komponente, die für das Compilieren und das Erzeugen von Standard-Formaten zuständig ist).

Wenn Sie ein Standard-Format erzeugen, dann erzeugt FORMBUILD für Sie

- das Format-Programm und
- führt automatisch das Compilieren (übersetzen) durch.

Um also ein speziell erwünschtes Format zu erzeugen, benötigen Sie zunächst ein Format-Programm.

6.1.1 Zwei Wege, ein Format zu erzeugen

Um ein Format zu erzeugen, gibt es mehrere mögliche Vorgehensweisen. Welche das sind, das erfahren Sie hier anhand der im folgenden beschriebenen Funktionen des FORMAT-Menüs.

Wenn Sie im Hauptmenü die Funktion 'Format' ausführen, dann erhalten Sie dieses Menü:

```
FORMAT:  Ablauf  Modifizieren  Generieren  Neu  Compilieren  Loeschen  END
Benutzen eines Formates zur Dateneingabe oder Datenbankabfrage.
```

Die Funktionen:

Ablauf
: ein bestehendes Format aufrufen.

Modifizieren
: ein bestehendes Format-Programm unter Verwendung des voreingestellten Editors verändern (siehe den folgenden Abschnitt).

Generieren
: ein neues Standard-Format automatisch von FORMBUILD erzeugen lassen.

Neu
: ein neues Format-Programm unter Verwendung des voreingestellten Editors schreiben (siehe den folgenden Abschnitt).

Compilieren
: ein Format-Programm zu einem ablauffähigen Format compilieren (übersetzen).

Loeschen
: ein Format löschen. Gelöscht werden dabei die beiden Dateien 'formatname.per' und 'formatname.frm' (siehe unten).

END
: Rückkehr in das INFORMIX-Hauptmenü.

Über die Funktion 'Neu' können Sie ein Format-Programm neu schreiben.

Oft ist es aber einfacher, einen anderen Weg einzuschlagen. Sie können nämlich

- zunächst mit 'Generieren' ein Standard-Format erzeugen und
- dann über 'Modifizieren' das von FORMBUILD erzeugte Format-Programm in den Editor einlesen und dann nach Ihren Wünschen verändern.

Weiter unten können Sie diesen zweiten Weg in einer Übung nachvollziehen.

6.1.2 Einstellen des Betriebssystem-Editors

Sowohl 'Neu' als auch 'Modifizieren' rufen einen Editor des Betriebssystems auf.

Mit der Umgebungsdvariablen DBEDIT ist es möglich, einen gewünschten Editor voreinzustellen. Siehe hierzu das Handbuch [2] INFORMIX-SQL Nachschlagen.

Ist die Umgebungsvariable DBEDIT jedoch nicht gesetzt, dann verhält es sich wie folgt:

- Sobald während der Arbeit mit INFORMIX-SQL erstmalig ein Editor benötigt wird, erhalten Sie einen Bildschirm, der Ihnen den CED als Editor anbietet.
- Wollen Sie mit dem CED arbeiten, dann bestätigen Sie mit ↵.
- Wollen Sie mit einem anderen Editor arbeiten, so tragen Sie dessen Namen ein und drücken ↵.

Der solchermaßen bekanntgegebene Editor ist damit für die aktuelle INFORMIX-Sitzung voreingestellt.

Die Erläuterungen in den folgenden Kapiteln beziehen sich auf den Editor CED. Der folgende Abschnitt führt Sie in den Gebrauch des CED ein.

Der Betriebssystem-Editor CED

Dieser Abschnitt erläutert Ihnen einige Grundfunktionen des CED. Vollständige Informationen zum CED entnehmen Sie bitte dem Handbuch [5] Betriebssystem SINIX Buch 2. Dort finden Sie auch eine umfangreiche Beispielsitzung, die Sie zum Lernen nachvollziehen können.

Wenn INFORMIX den CED aufruft, dann erhalten Sie diesen CED-Bildschirm (bei 'Modifizieren' mit dem Text des entsprechenden Format-Programmes):

```
** CED Texteditor V1.1          Zeile:    1 Spalte:   56  Name: rsq.xxxxx.per
```

```
Sie wollen neuen Text eingeben?
Bitte geben Sie Ihren Text ein!
```

Der Bereich zwischen den beiden Trennlinien ist der Arbeitsbereich; hier können Sie Ihren Text eingeben.
Der Text kann beliebig lang sein. Wenn Sie mit der Schreibmarke am unteren oder oberen Bildschirmrand angekommen sind, dann scrollt der Bildschirm automatisch und Sie können mit dem Editieren fortfahren.

Schreibmarke bewegen

Die folgenden Tasten positionieren die Schreibmarke:

←, →	links und rechts
↑, ↓	oben und unten
⇤, ⇥	in Tabulatorsprüngen nach links und rechts
↖ + ←	linker Bildschirmrand
↖ + →	rechter Bildschirmrand
↖ + ↑	oberer Bildschirmrand
↖ + ↓	unterer Bildschirmrand
↖ + ↖ + ↑	Dateianfang erste Zeile
↖ + ↖ + ↓	Dateiende letzte Zeile

Löschen und Einfügen

CHAR INSERT	fügt Leerzeichen ein
LINE INSERT	fügt Leerzeile ein
CHAR DELETE, ⌫	löscht ein Zeichen
LINE DELETE	löscht eine Zeile

CED verlassen

Wenn Sie mit der Eingabe Ihres Format-Programmes fertig sind und den CED verlassen wollen, dann:

▷ Drücken Sie [END]

Sie erhalten am unteren Bildschirmrand die folgenden Zeilen:

```
Sie wollen den CED verlassen?
Text Sicherung? in Dokument rsq.xxxxxx.per Geben Sie j=Ja oder n=Nein ein!
```

▷ Geben Sie 'j' ein.

Anschließend kehren Sie automatisch in das INFORMIX-Menüsystem zurück und können mit dem nächsten Schritt, dem Compilieren, fortfahren.

Hinweis

Wenn Sie ein Format-Programm neu schreiben oder modifizieren, dann arbeiten Sie zunächst auf einer temporären Datei. FORMBUILD zeigt das mit Dateinamen dieser Art an: rsq.000598.per
Danach wird diese temporäre Datei automatisch gelöscht und Sie erhalten eine Datei dieser Art: 'dateiname.per' Beispiel: FORMAT01.per

6.1.3 Ein Format - zwei Dateien

Zu jedem ablauffähigen Format gehören zwei Dateien:

- Eine Datei mit dem von Ihnen geschriebenen bzw. über die Funktion 'Generieren' automatisch erzeugten Format-Programm.
- Eine Datei, die das compilierte Format enthält.
  Auf diese Datei greift PERFORM zu, wenn Sie mit einem Format arbeiten.

Die Datei mit dem Format-Programm hat den Namen 'formatname.per'; beispielsweise FORMAT01.per
Die Datei mit dem compilierten Format hat den Namen 'formatname.frm'; beispielsweise FORMAT01.frm
Die Dateien befinden sich im aktuellen Dateiverzeichnis (Ordner); dem Dateiverzeichnis, in dem Sie sich bei der Erzeugung eines Formates befinden.
Die Format-Dateien der Datenbank 'versand' befinden sich im Dateiverzeichnis 'vertiefung' Ihres HOME-Dateiverzeichnisses.

6.1.4 Standard-Format erzeugen und modifizieren

Die folgende Übung erläutert Ihnen, wie Sie ein Standard-Format für die Tabelle 'kunde' erzeugen und wie Sie das damit erzeugte Format-Programm anschließend modifizieren können. Darauffolgend finden Sie wieder allgemeine Bedienungshinweise, die die erforderlichen Schritte zusammenfassen.

Standard-Format erzeugen

Übung 6-1

▷ Im INFORMIX-Hauptmenü führen Sie die Funktion 'Format' aus.

Sie erhalten das FORMAT-Menü.

▷ Führen Sie die Funktion 'Generieren' aus.

Sie erhalten jetzt einen Bildschirm, der der Eingabe des gewünschten Formatnamens dient.

▷ Als Formatnamen geben Sie z.B. ein: kunde_std

▷ Drücken Sie ↵

Sie erhalten den Bildschirm AUSWAHL TABELLE

▷ Wählen Sie die Tabelle 'kunde' aus.

Sie erhalten einen Bildschirm, in dem Sie beantworten müssen, ob Sie mit der Auswahl an Tabellen 'Fertig' sind oder ob Sie 'Weitere' Tabellen auswählen wollen.

▷ Führen Sie die Funktion 'Fertig' aus.

Nun erzeugt FORMBUILD ein Standard-Format für die Tabelle 'kunde'. Nach kurzer Wartezeit werden Sie in das FORMAT-Menü zurückgeführt und Sie erhalten eine Meldung darüber, daß das Standard-Format erfolgreich generiert wurde.

Der erste Schritt ist vollzogen, im nächsten Teil der Übung geht es darum, das Format-Programm des Standard-Formates mit 'Modifizieren' aufzurufen und einige Änderungen daran vorzunehmen.

Format-Programm modifizieren

Um das Format-Programm des Standard-Formates 'kunde_std' zu modifizieren, führen Sie weiter die folgenden Schritte aus:

▷ Führen Sie die Funktion 'Modifizieren' aus.

Sie erhalten den MODIFIZIEREN-FORMAT-Bildschirm.

▷ Wählen Sie das Format 'kunde_std' aus.

Sie erhalten einen Bildschirm, der Ihnen den CED als Ruf-Editor anbietet.

▷ Drücken Sie ↵

Die Abfrage nach dem Editor erfolgt während einer INFORMIX-Sitzung nur einmal.

Jetzt ruft FORMBUILD den CED mit dem Format-Programm auf. Nach kurzer Wartezeit erhalten Sie diesen Bildschirm:

```
** CED Texteditor V1.1          Zeile:    1 Spalte:   56  Name: rsqxxxxxx.per
database versand
screen size 24 by 80
{
kunden_nr           [f000       ]
vorname             [f001           ]
nachname            [f002           ]
firma               [f003               ]
adresse1            [f004               ]
adresse2            [f005               ]
ort                 [f006           ]
bundesland          [a0]
plz                 [f007 ]
telefon             [f008               ]
}
end
tables
kunde
attributes
f000 = kunde.kunden_nr, right;

Sie wollen neuen Text eingeben?
Bitte geben Sie Ihren Text ein!
```

Auf dem Bildschirm sehen Sie einen Teil des Format-Programmes. Wenn Sie z.B. mit ⇩ und ⇧ in der Datei blättern, dann können Sie sich auch den Rest des Format-Programmes zeigen lassen. In voller Länge sieht das Format-Programm so aus:

```
database versand
screen size 24 by 80
{
kunden_nr           [f000       ]
vorname             [f001           ]
nachname            [f002           ]
firma               [f003               ]
adresse1            [f004               ]
adresse2            [f005               ]
ort                 [f006           ]
bundesland          [a0]
plz                 [f007 ]
telefon             [f008             ]
}
end
tables
kunde
attributes
f000 = kunde.kunden_nr, right;
f001 = kunde.vorname;
f002 = kunde.nachname;
f003 = kunde.firma;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.ort;
a0 = kunde.bundesland;
f007 = kunde.plz;
f008 = kunde.telefon;
end
```

Die Übung beschäftigt sich noch nicht mit der Syntax eines Format-Programmes; die folgenden Abschnitte gehen ausführlich darauf ein.
Hier geht es zunächst einmal darum, die formalen Schritte kennenzulernen, die Sie unternehmen müssen, um ein Format-Programm zu modifizieren und anschließend zu compilieren.

Der Teil, an dem Sie mit sichtbarem Ergebnis sehr einfach Änderungen vornehmen können, ist der sogenannte SCREEN-Abschnitt ab der zweiten Zeile. In diesem Abschnitt ist u.a. festgelegt, wie das fertige Format am Bildschirm angezeigt wird (Bildschirm-Layout):

```
screen size 24 by 80
{
kunden_nr           [f000       ]
vorname             [f001           ]
nachname            [f002           ]
firma               [f003               ]
adresse1            [f004               ]
adresse2            [f005               ]
ort                 [f006           ]
bundesland          [a0]
plz                 [f007 ]
telefon             [f008               ]
}
end
```

Der SCREEN-Abschnitt dieses Standard-Formates besteht aus folgenden Teilen:

- Links befinden sich die Spaltennamen (freie Texte) und
- in der Mitte befinden sich die Bildschirmfelder, angezeigt durch die eckigen Klammern.

Gegenüber einem fertigen Format wie es am Bildschirm angezeigt wird gibt es nur einen Unterschied: Die Buchstaben/Ziffern-Kombinationen innerhalb der Bildschirmfelder erscheinen im compilierten Format nicht. Man nennt sie 'Feldbezeichner'; ihre Aufgabe ist es, die Zuordnung zwischen einem Bildschirmfeld und einer Tabellenspalte herzustellen.

In diesem SCREEN-Abschnitt können Sie sehr leicht Veränderungen vornehmen. Sie können z.B. die Spaltennamen in eine lesbarere Form bringen und Sie können eine Format-Überschrift hinzufügen.

▷ Verändern Sie den SCREEN-Abschnitt nach dem unten gezeigten Muster, fügen Sie eine Format-Überschrift hinzu und bringen Sie die Spaltennamen in eine lesbarere Form.

```
screen size 24 by 80
{

                  Format 'kunde_std'
                  ------------------

Kundennummer       [f000       ]
Vorname            [f001           ]
Nachname           [f002           ]
Firma              [f003               ]
Adresse            [f004               ]
evtl.Zweitanschrift [f005              ]
Ort                [f006           ]
Bundesland         [a0]
Plz                [f007 ]
Telefon            [f008             ]
}
end
```

Sie haben nun das Format-Programm eines Standard-Formates modifiziert. Jetzt geht es wie folgt weiter:

▷ Um den CED zu verlassen, drücken Sie END

Sie erhalten anschließend am unteren Bildschirmrand eine Abfrage-Zeile dieser Art:

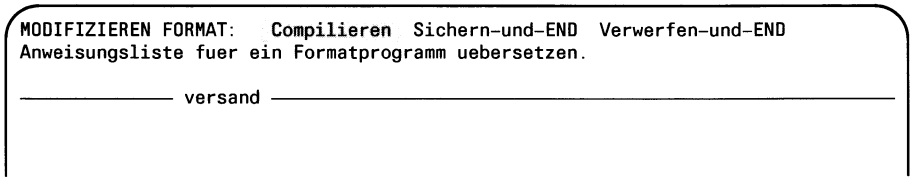
```
Sie wollen den CED verlassen?
Text Sicherung? in Dokument: rsqxxxxxxx.per  Geben Sie j=Ja oder n=Nein ein!
```

▷ Um die Datei zu sichern, geben Sie 'j' ein.

Hinweis

Gesichert wird damit zunächst nur eine temporäre Datei, die später automatisch wieder gelöscht wird. Das endgültige Sichern der Datei mit dem Format-Programm erfolgt in einem späteren Schritt.

Sie sind nun automatisch wieder in das INFORMIX-Menüsystem zurückgekehrt und haben diesen Bildschirm erhalten:



Die Funktionen:

Compilieren
: compiliert das Format-Programm und erzeugt dabei die Datei 'formatname.frm'.
Wenn Syntaxfehler festgestellt wurden, dann kann das Format-Programm nicht erfolgreich compiliert werden; siehe hierzu den folgenden Abschnitt.

Sichern-und-END
: ist nach dem Compilieren auszuführen, wenn die compilierte Datei mit dem ablauffähigen Format und die Datei mit dem Format-Programm gesichert werden sollen. Die Funktion erzeugt aus der bisherigen temporären Datei die Datei 'formatname.per', und sie sichert die Datei 'formatname.per' und die beim Compilieren erzeugte Datei 'formatname.frm'
Anschließend kehren Sie automatisch in das FORMAT-Menü zurück. Sie können die Funktion auch dann ausführen, wenn Sie im Augenblick nicht compilieren wollen, die Datei 'formatname.per' mit Ihrem Format-Programm aber erhalten bleiben soll.

Verwerfen-und-END
: verwirft Änderungen am Format-Programm und führt Sie zurück in das FORMAT-Menü.
Der bisherige Stand des Format-Programmes bleibt erhalten.

Um Ihr Format-Programm zu compilieren:

▷ Führen Sie die Funktion 'Compilieren' aus.

Nach erfolgreicher Compilierung gibt FORMBUILD eine Meldung aus und markiert anschließend automatisch die Funktion 'Sichern-und-END'. Wenn das Compilieren nicht erfolgreich war, weil ein Syntaxfehler festgestellt wurde, dann unternehmen Sie bitte die im folgenden Abschnitt 'Fehlerbehandlung' beschriebenen Schritte.

Um das Format zu sichern:

▷ Führen Sie die Funktion 'Sichern-und-END' aus.

Danach werden Sie automatisch in das FORMAT-Menü zurückgeführt. Jetzt können Sie über die Funktion 'Ablauf' das Format aufrufen und mit ihm arbeiten.

Allgemeine Bedienungshinweise

Um ein Standard-Format zu erzeugen, unternehmen Sie die folgenden Schritte:

1. Im Hauptmenü führen Sie die Funktion 'Format' aus.
2. Im FORMAT-Menü führen Sie die Funktion 'Generieren' aus.
3. Wenn noch keine Datenbank ausgewählt wurde und damit die aktuelle Datenbank ist, dann erhalten Sie jetzt einen Bildschirm, in dem Sie die gewünschte Datenbank auswählen - ansonsten gleich weiter mit 4.
4. Geben Sie dem Format einen Namen.
   NAMENSKONVENTIONEN: Der Name darf aus Groß- und Kleinbuchstaben, Zahlen und Unterstrichen bestehen, er darf jedoch nicht länger als 10 Zeichen sein.
5. Wählen Sie die erste gewünschte Tabelle aus.
6. Wenn Sie noch weitere Tabellen auswählen wollen, dann führen Sie die Funktion 'Weitere' aus und wählen die nächste erwünschte Tabelle aus. Wenn Sie fertig mit der Tabellen-Auswahl sind, dann führen Sie die Funktion 'Fertig' aus.
7. Die Funktion 'Fertig' leitet die automatische Generierung des Standard-Formates ein. Nach erfolgreicher Ausführung werden Sie in das FORMAT-Menü zurückgeführt.

Um ein Format-Programm zu modifizieren, führen Sie die folgenden Schritte aus:

1. Im Hauptmenü führen Sie die Funktion 'Format' aus.
2. Im FORMAT-Menü führen Sie die Funktion 'Modifizieren' aus.
3. Wählen Sie das Format aus, das modifiziert werden soll.
4. Nun wird der voreingestellte Editor aufgerufen und mit der Datei 'formatname.per', die das Format-Programm enthält, geladen. Führen Sie die erwünschten Änderungen durch.
5. Verlassen Sie den Editor. Sie kehren automatisch in das INFORMIX-Menüsystem zurück.
6. Führen Sie die Funktion 'Compilieren' aus. Sollten beim Compilieren Syntaxfehler festgestellt worden sein, dann unternehmen Sie bitte die im folgenden Abschnitt 'Fehlerbehandlung' beschriebenen Schritte.
7. Führen Sie anschließend die Funktion 'Sichern-und-END' aus; Sie kehren automatisch in das FORMAT-Menü zurück.

6.1.5 Fehlerbehandlung

Wenn Ihnen ein Syntaxfehler unterlaufen ist, dann erhalten Sie nach dem Compilieren automatisch diesen Bildschirm mit einer Meldung:

```
COMPILIEREN FORMAT:   Korrigieren  END
Anweisungsliste fuer ein Formatprogramm korrigieren.

————————— versand ————————————————————————————————————————————————————

 nnn: Fehler in den Formatprogramm-Anweisungen.
```

Um den Fehler zu korrigieren, gehen Sie wie folgt vor:

▷ Führen Sie die Funktion 'Korrigieren' aus.

Danach wird automatisch wieder der Editor aufgerufen und mit dem Format-Programm geladen. Das Format-Programm enthält einen erläuternden Kommentar dieser Art:

```
database versand
screen size 24 by 80

                    Format 'kunde_std'
                    ------------------

#
# Die Format-Definition muss mit dem Zeichen "{" beginnen.
# Fehler-Nr.: -2890

Kundennummer        [f000       ]
Vorname             [f001           ]
Nachname            [f002           ]
Firma               [f003               ]
Adresse             [f004               ]
evtl.Zweitanschrift[f005                ]
Ort                 [f006           ]
Bundesland          [a0]
Plz                 [f007 ]
Telefon             [f008               ]
}
end
...
...
...
```

Im gezeigten Beispiel wurde die einleitende geschweifte Klammer des SCREEN-Abschnittes vergessen.
FORMBUILD hat einen Kommentar in das Format-Programm eingefügt, der Sie über die vermutete Fehlerquelle informiert. Die Kommentarzeilen sind mit einleitenden '#' kenntlich gemacht.
Sie brauchen die Kommentarzeilen nicht zu löschen, denn sie werden beim erneuten Compilieren von FORMBUILD nicht als Text interpretiert und ignoriert.

▷ Führen Sie die erforderlichen Korrekturen durch.

▷ Verlassen Sie anschließend wieder den Editor.

▷ Jetzt erhalten Sie wieder den Bildschirm mit den Funktionen 'Compilieren', 'Sichern-und-End', 'Verwerfen-und-END'.

▷ Führen Sie die Funktion 'Compilieren' aus.

Wenn noch ein zweiter Syntaxfehler festgestellt wurde, dann führen Sie wieder die Funktion 'Korrigieren' aus und nehmen die erforderlichen Korrekturen vor. Ansonsten führen Sie jetzt die Funktion 'Sichern-und-END' aus.

6.2 Die Syntax eines Format-Programmes

Ein Format-Programm besteht aus den folgenden Abschnitten:

DATABASE-Abschnitt (Pflicht)
SCREEN-Abschnitt (Pflicht)
TABLES-Abschnitt (Pflicht)
ATTRIBUTES-Abschnitt (Pflicht)
INSTRUCTIONS-Abschnitt (optional)

Im folgenden sind die Abschnitte eines Format-Programmes ausführlich beschrieben.

Anweisungen anordnen

Sie können die Schlüsselwörter groß oder klein schreiben. Sie können beliebig viele Leerzeilen und Leerzeichen einfügen, so daß Sie das Format-Programm übersichtlich anordnen können.
Das gilt nicht für den SCREEN-Abschnitt, denn hier ist jedes Leerzeichen und jede Leerzeile von Bedeutung für das Bildschirmlayout!

Um die Schlüsselwörter deutlich hervorzuheben, verwendet das Handbuch die Großschreibweise der Schlüsselwörter.

Kommentar

Sie können ein Format-Programm mit Kommentaren versehen. Ein Kommentar ist ein Text, den Sie programmerläuternd hinzufügen können und der beim Compilieren ignoriert wird.

Einen Kommentar definieren Sie wie folgt:

```
{Kommentartext}
```

Der Kommentartext wird in geschweifte Klammern eingeschlossen.

Beispiel

```
{ Das ist der DATABASE-Abschnitt des FORMAT01: }

database versand
end
...
```

Hinweis

Der SCREEN-Abschnitt darf wegen der Sonderbedeutung der geschweiften Klammern keinen Kommentar enthalten.

Das Schlüsselwort END

Sie können die einzelnen Abschnitte eines Format-Programmes optional mit dem Schlüsselwort END abschließen. Bei einem Standard-Format schließt FORMBUILD automatisch die Abschnitte SCREEN und ATTRIBUTES mit dem Schlüsselwort END ab.

Format-Programm bei einem Standard-Format

Bei einem Standard-Format für die Tabelle 'kunde' sind das die einzelnen Abschnitte des Format-Programmes:

```
                DATABASE-Abschnitt:
database versand

                SCREEN-Abschnitt:
screen size 24 by 80
{
kunden_nr          [f000       ]
vorname            [f001           ]
nachname           [f002           ]
firma              [f003               ]
adresse1           [f004               ]
adresse2           [f005               ]
ort                [f006           ]
bundesland         [a0]
plz                [f007 ]
telefon            [f008             ]
}
end

                TABLES-Abschnitt:
tables
kunde

                ATTRIBUTES-Abschnitt:
attributes
f000 = kunde.kunden_nr, right;
f001 = kunde.vorname;
f002 = kunde.nachname;
f003 = kunde.firma;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.ort;
a0 = kunde.bundesland;
f007 = kunde.plz;
f008 = kunde.telefon;
end

                INSTRUCTIONS-Abschnitt:
          - standardmäßig nicht vorhanden -
```

Das gezeigte Format-Programm enthält alle Anweisungen, die für ein Standard-Format für die Tabelle 'kunde' benötigt werden. Ein INSTRUCTIONS-Abschnitt wird standardmäßig nicht erzeugt.

6.3 Der DATABASE-Abschnitt

Der DATABASE-Abschnitt besteht aus dem Schlüsselwort DATABASE und dem Namen der Datenbank, für die das Format bestimmt ist.
Bei einem Standard-Format steht hier immer der Name derjenigen Datenbank, die die aktuelle Datenbank war, als Sie das Standard-Format erzeugt haben.

Optional kann der DATABASE-Abschnitt eine 'WITHOUT NULL INPUT' Klausel enthalten. Die Klausel bewirkt Standardvorbelegungen in den Bildschirmfeldern: 'Leerzeichen' in CHAR-Bildschirmfeldern und '0' in numerischen Bildschirmfeldern. Die Klausel kommt jedoch nur dann zur Wirkung, wenn bei der Tabellendefinition NULL's 'Nein' festgelegt wurde (siehe auch Abschnitt 6.6.6).

6.4 Der SCREEN-Abschnitt

Der SCREEN-Abschnitt besteht aus

- dem SchLüsselwort SCREEN,
- Angaben zur Bildschirmgröße (size 24 by 80),
- einer einleitenden geschweiften Klammer '{'
- dem Format-Layout (siehe unten) und
- einer abschließenden geschweiften Klammer '}'
- und u.U. aus dem Schlüsselwort END

```
SCREEN size bildschirmzeilen by bildschirmspalten
{
Format-Layout
}
END
```

Die Angabe:

```
size bildschirmzeilen by bildschirmspalten
```

ist optional und kann bei Bildschirmen mit 25 Zeilen und 80 Zeichen/Zeile entfallen. In diesem Fall sind automatisch die Standardwerte 24 by 80 gültig.
Haben Sie einen Bildschirm mit mehr oder weniger Zeilen und Spalten, so können Sie die Werte hier angeben.

Die geschweiften Klammern müssen jeweils am Anfang einer eigenen Zeile stehen. Optional können Sie den SCREEN-Abschnitt mit dem Schlüsselwort END abschließen.

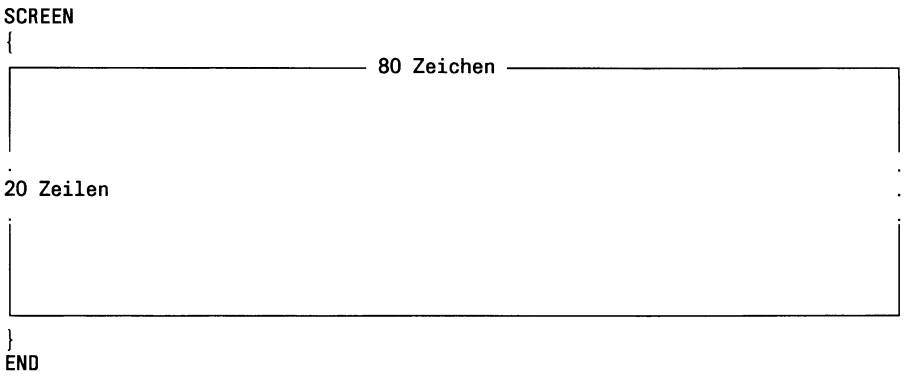
Der SCREEN-Abschnitt erfüllt zwei Aufgaben:

1. Hier gestalten Sie das Aussehen des Formates (Format-Layout).
   So wie Sie hier freie Texte (z.B. Spaltennamen) und Bildschirmfelder anordnen, so werden Sie auch im fertigen Format erscheinen.
2. Im Rahmen des Format-Layoutes treffen Sie mit den Feldbezeichnern (Buchstaben- und Ziffern-Kombinationen innerhalb der Bildschirmfelder) die Zuordnung zwischen Bildschirmfeldern und Tabellenspalten. Die Feldbezeichner müssen gleichlautend auch noch einmal im ATTRIBUTES-Abschnitt vorkommen.

6.4.1 Format-Layout

Im Bereich zwischen den beiden geschweiften Klammern gestalten Sie das Format-Layout.
Dabei geht es um Anordnung und Größe sowohl der Bildschirmfelder als auch der freien Texte. Das sind z.B. Spaltennamen, Trennlinien und weitere Texte, die die Orientierung in einem Format erleichtern sollen.

Für ein Format mit nur einem Format-Bildschirmseite stehen Ihnen Bei einem Bildschirm mit 25 Zeilen und 80 Zeichen/Zeile für das Format-Layout maximal 20 Zeilen und die volle Bildschirmbreite von 80 Zeichen pro Zeile zur Verfügung:



Innerhalb dieses Zeilenraumes können Sie Bildschirmfelder und freie Texte beliebig anordnen.
So können Sie z.B. mehrere Bildschirmfelder nebeneinander anordnen oder Sie können beliebige freie Texte wie z.B. Trennlinien, Format-Überschrift u.a. hinzufügen.

Beispiel

Das hier ist der SCREEN-Abschnitt des Formates FORMAT03 mit einem Format-Layout, das sich weitgehend von dem eines Standard-Formates unterscheidet:

```
screen
{
                    FORMAT03 - Tabellen 'kunde' und 'auftrag'
__kunde_____________________________________________________________________

Firma     [f001                 ]
Vorname   [f002           ]           Nachname              [f003          ]
Adresse   [f004                 ]     evtl. Zweitanschrift [f005              ]
Plz       [f006 ]                     Ort                  [f007          ]

Telefon   [f008              ]        Bundesland           [a0]

__auftrag___________ Kundennummer [f000       ] ____________________________

Auftragsnummer      [f009       ]    Auftragsdatum        [f010      ]
Lieferhinweis       [f011                                       ]
Offen?              [a]              Fremdnummer          [f012      ]
Lieferdatum         [f013       ]    Liefergewicht        [f014      ]
Zustellgebuehr      [f015     ]      Zahldatum            [f016      ]
}
 end
```

Hinweis

Wenn Sie für das Format-Layout mehr als 20 Zeilen verwenden, dann ist das Format bei einem Standardbildschirm nicht mehr auf einem Bildschirm abbildbar und es gibt automatisch eine zweite Bildschirmseite.

Die Format-Programme der Datenbank 'versand' sind in voller Länge im Anhang abgedruckt.
Wenn Sie sich die hier folgend abgedruckten Ausschnitte der Format-Programme am Bildschirm anschauen wollen, dann können Sie wie folgt vorgehen: Im Format-Menü führen Sie die Funktion 'Modifizieren' aus. Dann wählen Sie das entsprechende Format aus. Im Anschluß daran wird automatisch der Editor aufgerufen und mit dem ausgewählten Format geladen.

6.4.2 Bildschirmfelder

Um ein Bildschirmfeld in einem Format von Hand festzulegen, gehen Sie wie folgt vor:

- Mit den eckigen Klammern [ ], den Feldbegrenzern, geben Sie im SCREEEN-Abschnitt den Umfang des Bildschirmfeldes an. Außerdem legen Sie dabei fest, wo ein Bildschirmfeld auf dem Bildschirm angeordnet sein soll.
- Innerhalb des Bildschirmfeldes geben Sie einen Feldbezeichner an. Dieser Feldbezeichner muß gleichlautend noch einmal im ATTRIBUTES-Abschnitt vorkommen.

Beachten Sie dabei bitte, daß das Bildschirmfeld in seinen Abmessungen so gewählt sein sollte, daß es der definierten Spaltenlänge der zugeordneten Tabellenspalte entspricht.
Wenn Sie z.B. ein Bildschirmfeld für eine CHAR-Tabellenspalte mit der Spaltenlänge 20 definieren, dann sollten Sie die Abmessungen des Bildschirmfeldes auch so wählen, daß es genau 20 Zeichen aufnehmen kann. Wurde das Bildschirmfeld zu klein gewählt, dann kann ein Wert u.U. nicht mehr vollständig angezeigt werden.

Hinweis

Um Irrtümer in den Abmessungen der Bildschirmfelder auszuschließen, können Sie den eingangs beschriebenen Weg gehen: Zunächst erzeugen Sie ein Standard-Format; dabei wird automatisch auch ein Format-Programm erzeugt. Wenn Sie anschließend das damit erzeugten Format-Programm modifizieren, dann brauchen Sie sich um die genauen Abmessungen der Bildschirmfelder nicht zu kümmern. Denn bei einem Standard-Format-Programm ist automatisch gewährleistet, daß die Bildschirmfelder genau den Abmessungen entsprechen, wie Sie beim Erzeugen einer Tabelle mit der Spaltenlänge (bzw. Spaltentyp) definiert wurden.
Außerdem ist dann ebenfalls gewährleistet, daß Bildschirmfelder und Tabellenspalten (über die Feldbezeichner) korrekt einander zugeordnet sind.

Standardmäßig sind die eckigen Klammern die Feldbegrenzer. Sie können durch eine entsprechende Anweisung im INSTRUCTIONS-Abschnitt andere Feldbegrenzer definieren.

6.4.3 Bildschirmfelder und Tabellenspalten

Mittels eines Feldbezeichners treffen Sie die Zuordnung zwischen einem Bildschirmfeld und einer Tabellenspalte. Dabei wirken der SCREEN-Abschnitt und der ATTRIBUTES-Abschnitt zusammen, wie das folgende Beispiel zeigen soll:

Beispiel

Im oben gezeigten Standard-Format für die Tabelle 'kunde' gibt es im SCREEN-Abschnitt dieses Bildschirmfeld:

```
vorname              [f001              ]
```

Und im ATTRIBUTES-Abschnitt gibt es diese Anweisung:

```
f001 = kunde.vorname;
```

Die Zuordnung von Bildschirmfeld zu Tabellenspalte erfolgt dadurch, daß

- im SCREEN-Abschnitt im entsprechenden Bildschirmfeld ein Feldbezeichner steht und
- im ATTRIBUTES-Abschnitt eine Anweisung erfolgen muß, die besagt, welche Tabellenspalte dem Feldbezeichner (und damit dem Bildschirmfeld) zugeordnet sein soll.

Im gezeigten Beispiel ist dem Bildschirmfeld 'vorname' über den Feldbezeichener f001 die Tabellenspalte 'vorname' der Tabelle 'kunde' zugeordnet.

Hinweis

Sie können prinzipiell für jede Tabellenspalte der im TABLES-Abschnitt angegebenen Tabelle auch ein Bildschirmfeld definieren. Bei einem Standard-Format ist das automatisch der Fall.
Zwingend erforderlich ist das jedoch nicht. In einem Format können auch weniger Bildschirmfelder definiert sein als es insgesamt Tabellenspalten gibt.

6.4.4 Mehrere Bildschirmseiten eines Formates

Wenn ein Format mehrere Bildschirmseiten erhalten soll, dann können Sie mehrere SCREEN-Abschnitte definieren:

```
SCREEN
{
Format-Layout1
}
SCREEN
{
Format-Layout2
}
END
```

Beispiel

Das ist der SCREEN-Abschnitt des FORMAT02:

```
screen
{
                        FORMAT02 - Bildschirmseite 1
                        ----------------------------


     Kundennummer       [f000       ]


     Firma              [f001                ]

     Vorname            [f002           ]
     Nachname           [f003           ]



}
screen
{
```

```
                        FORMAT02 - Bildschirmseite 2
                        ----------------------------

     Firma              [f001                 ]

     Anschrift          [f004                 ]
     evt. Zweitanschrift [f005                ]

     Plz                [f006 ]
     Ort                [f007          ]
     Bundesland         [a0]

     Telefon            [f008             ]

}
end
```

Der vorangegangene Abschnitt 5.7 'Format - ein Format, mehrere Bildschirmseiten' hat die Auswirkungen beschrieben: Mit der Funktion 'Format' können Sie zwischen den beiden Bildschirmseiten blättern, die hier mit den entsprechenden SCREEN-Abschnitten definiert sind.

Beachten Sie bitte, daß Sie für das Format-Layout je eines SCREEN-Abschnittes standardmäßig nicht mehr als 20 Zeilen verwenden sollten, da Sie anderenfalls zusätzliche, unerwünschte Bildschirmseiten erhalten.

6.5 Der TABLES-Abschnitt

Der TABLES-Abschnitt besteht aus dem Schlüsselwort TABLES; weiter geben Sie hier den Namen derjenigen Tabelle(n) an, für die Bildschirmfelder definiert sind.

Bei einem Standard-Format steht hier immer der Name derjenigen Tabelle(n), die Sie beim Erzeugen des Standard-Formates ausgewählt haben.

Hinweis

Wenn Sie beim Erzeugen eines Standard-Formates mehrere Tabellen ausgewählt haben, dann ist damit noch nicht automatisch eine Join-Verbindung zwischen den Tabellen realisiert. Diese müssen Sie durch entsprechende Modifizierung des Format-Programmes vornehmen (siehe Abschnitt 6.7)

6.6 Der ATTRIBUTES-Abschnitt

Der ATTRIBUTES-Abschnitt besteht aus:

- dem Schlüsselwort ATTRIBUTES,
- einer Liste von Anweisungen, mit denen die Bildschirmfelder den Tabellenspalten zugeordnet werden und
- einem optionalen, abschließenden Schlüsselwort END.

Eine zuordnende Anweisung besteht aus:

- einem Feldbezeichner,
- einem Gleichheitszeichen,
- einem Spaltennamen (und einem optionalen Tabellennamen) und
- optional aus weiteren Anweisungen, mit denen Sie einem Bildschirmfeld bestimmte Eigenschaften zuordnen können.

Beispiel

In dem ATTRIBUTES-Abschnitt des FORMAT01 gibt es diese Anweisung:

```
f000 = kunde.kunden_nr, reverse;
```

Die Anweisung legt fest:

- Daß dem Bildschirmfeld mit dem Feldbezeichner f000 die Tabellenspalte 'kunden_nr' der Tabelle 'kunde' zugewiesen wird.
- Das Schlüsselwort REVERSE bewirkt, daß das Bildschirmfeld invertiert (helle Schrift auf dunklem Grund) abgebildet wird.

Hinweis

Die Reihenfolge, in der die zuordnenden Anweisungen untereinander im ATTRIBUTES-Abschnitt stehen, bestimmt die Reihenfolge, in der die Schreibmarke beim Suchen, Neuaufnehmen oder Korrigieren springt.

6.6.1 Syntax für zuordnende Anweisungen

Eine zuordnende Anweisung kann z.B. wie folgt aufgebaut sein:

```
feldbezeichner = [tabellenname.]spaltenname [,eigenschaft,...];
```

feldbezeichner
: Feldbezeichner muß mit einem Buchstaben beginnen und kann dann weiter aus Buchstaben oder Ziffern bestehen.

=
: Das Gleichheitszeichen stellt die Zuordnung zwischen 'feldbezeichner' und 'spaltenname' her.

tabellenname.
: Der Tabellenname ist optional. Aber wenn Sie einen Tabellennamen angeben, dann müssen Sie ihn mit einem Punkt vom Spaltennamen trennen.
Bei Mehr-Tabellen-Formaten müssen Sie den Tabellennamen dann mit angeben, wenn ein Spaltenname nicht eindeutig ist (Join-Spalten).

spaltenname
: Für 'spaltename' geben Sie den Namen der erwünschten Tabellenspalte an.

,eigenschaft,..
: Für 'eigenschaft' können Sie eine oder mehrere Anweisungen angeben, die die Eigenschaften eines Bildschirmfeldes festlegen.
Sie müssen den Spaltennamen durch ein Komma von der ersten Eigenschafts-Anweisung trennen. Wenn Sie mehrere Anweisungen angeben, dann müssen Sie diese ebenfalls durch Kommas voneinander trennen.

;
: Das Semikolon schließt die Anweisung ab. Wenn Sie keine 'eigenschaft' angegeben haben, dann steht das Semikolon hinter dem Spaltennamen, ansonsten hinter der letzten Anweisung.

6.6.2 Bildschirmfeld-Eigenschaften im FORMAT04

Die folgenden Text-Abschnitte erläutern am Beispiel des FORMAT04-Programmes einige Anweisungen, mit denen Sie die Eigenschaften eines Bildschirmfeldes festlegen können.
Das Handbuch [2] INFORMIX-SQL Nachschlagen informiert Sie ausführlich.

Das Format-Programm des FORMAT04:

```
database versand
screen
{
                    FORMAT04 - Tabellen 'kunde', 'auftrag' und 'posten'
__kunde_____________________________________________________________________________

Firma     [f001                ]   Adresse           [f004                ]
Plz       [f006 ]                  Ort               [f007           ]
                                   Bundesland        [a ]
__auftrag___________ Kundennummer [f000       ] _____________________________________

Offen?              [b]            Auftragsdatum     [f010      ]
Lieferdatum         [f013      ]   Liefergewicht     [f014      ]
Zustellgebuehr      [f015    ]     Zahldatum         [f016      ]

__posten____________ Auftragsnummer [f009       ] _____________________________________

Postennummer        [f017  ]       Herstellercode    [c ]
Artikelnummer       [f018  ]       Artikel           [lo01                ]
Menge               [f019  ]       Gesamtpreis       [f020      ]
}
end
tables
    kunde
    auftrag
    posten
    artikel
attributes
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr, reverse;
f001 = kunde.firma,
       comments = "Bei Neuaufnahme: falls 'GmbH', bitte mit eingeben!";
f004 = kunde.adresse1, required;
f006 = kunde.plz;
f007 = kunde.ort;
a    = kunde.bundesland,
       upshift,
       default="BY";
f009 = * auftrag.auftrags_nr
     =   posten.auftrags_nr, reverse;
b    = auftrag.offen,
       include = (j, n),
       comments = "'j' oder 'n' eingeben";
f010 = auftrag.auftragsdatum,
       default = today;
f013 = auftrag.lieferdatum, verify;
f014 = auftrag.liefergewicht;
f015 = auftrag.zustellgebuehr;
f016 = auftrag.zahldatum;
```

```
f017 = posten.posten_nr;
c    = posten.herstellercode;
f018 = posten.artikel_nr,
       lookup lo01 = artikel.bezeichnung
       joining artikel.artikel_nr;
f019 = posten.menge, right,
       include = (1 to 50),
       comments = "Mindestens 1, maximal 50";
f020 = posten.gesamtpreis;

instructions
kunde master of auftrag;
auftrag master of posten;
end
```

6.6.3 REVERSE - invertierte Abbildung

Das Schlüsselwort REVERSE bewirkt, daß ein Bildschirmfeld invertiert abgebildet wird.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr, reverse;
```

Diese Anweisung bewirkt, daß das Bildschirmfeld 'Kundennummer' invertiert abgebildet wird.

Die im Beispiel gezeigte Anweisung definiert ein Join-Bildschirmfeld. Auf die Besonderheiten eines Join-Bildschirmfeldes geht der Abschnitt 6.7 ein.

6.6.4 COMMENTS - Kommentare am Bildschirm ausgeben

Die COMMENTS-Anweisung bewirkt, daß ein Kommentar am Bildschirm ausgegeben wird, wenn sich die Schreibmarke im entsprechenden Bildschirmfeld befindet.

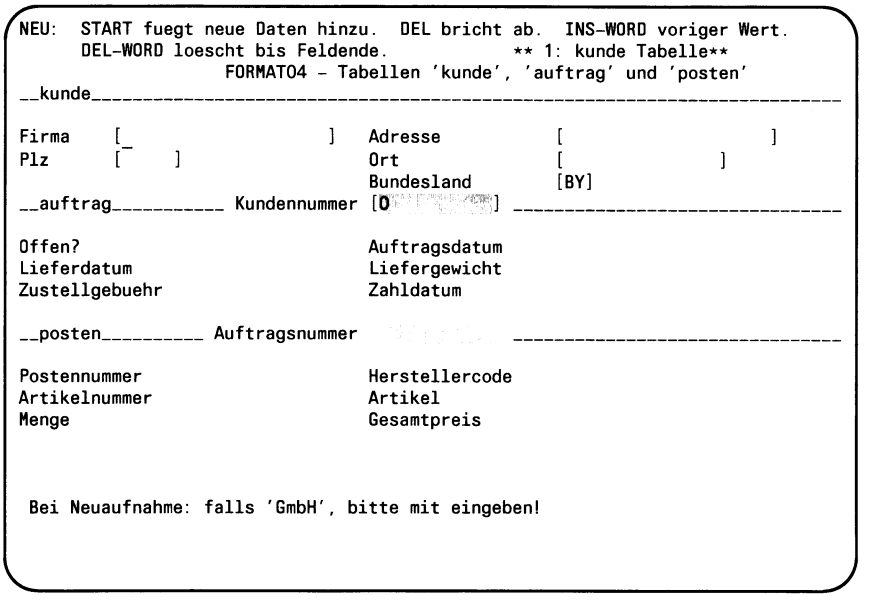
Die COMMENTS-Anweisung verwenden Sie wie folgt: Auf das Schlüsselwort COMMENTS folgt ein Gleichheitszeichen und darauf folgt der erwünschte Kommentartext. Der Kommentartext muß in Anführungszeichen eingeschlossen sein und er muß in eine einzige Zeile passen. Mehrzeilige Kommentartexte sind nicht möglich.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
f001 = kunde.firma,
       comments = "Bei Neuaufnahme: falls 'GmbH', bitte mit eingeben!";
```

Diese Anweisung bewirkt, daß der bei 'comments' definierte Kommentartext:
'Bei Neuaufnahme: falls 'GmbH', bitte mit eingeben!' am unteren Bildschirmrand als Meldung ausgegeben wird, sobald sich die Schreibmarke im Bildschirmfeld 'Firma' befindet:



6.6.5 UPSHIFT und DOWNSHIFT - Groß- und Kleinschreibweise

Das Schlüsselwort UPSHIFT wandelt evtl. kleingeschriebene Eingaben umgehend in Großbuchstaben um; Das Schlüsselwort DOWNSHIFT wandelt evtl. großgeschriebene Eingaben umgehend in Kleinbuchstaben um.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
a      = kunde.bundesland,
         upshift,
         default = "BY";
```

Wenn Sie beim Suchen, Neuaufnehmen oder Korrigieren eine Eingabe in das Bildschirmfeld 'Bundesland' machen, dann wandelt PERFORM Ihre Eingabe umgehend in Großbuchstaben um.
Sie können eingeben: by, By oder bY, in jedem Fall wandelt PERFORM Ihre Eingabe umgehend in BY um.

Das zweite Schlüsselwort in dieser Anweisung, 'default', wird weiter unten behandelt.

Hinweis

Die Verwendung der Schlüsselwörter UPSHIFT und DOWNSHIFT vereinfacht Datenbank-Abfragen und die sortierte Ausgabe von Werten, denn groß- und kleingeschriebene Werte haben unterschiedliche ASCII-Codes.
Ohne das Schlüsselwort UPSHIFT wären beim Neuaufnehmen beispielsweise auch diese Eingaben möglich: by, By und bY. Um alle Sätze mit dem Bundesland 'BY'= Bayern zu finden, müßten Sie dann vier Datenbank-Abfragen durchführen.

6.6.6 DEFAULT - erwünschte Standard-Vorbelegung definieren

Mit einer DEFAULT-Anweisung können Sie vom Standard abweichende Vorbelegungen der Bildschirmfelder festlegen.

Die DEFAULT-Anweisung verwenden Sie wie folgt: Auf das Schlüsselwort DEFAULT folgt ein Gleichheitszeichen und darauf folgt der Wert, der als Standard-Vorbelegung angezeigt werden soll. Den Wert müssen Sie in Anführungszeichen setzen.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
a     = kunde.bundesland,
        upshift,
        default="BY";
```

Wenn Sie die Funktion 'Neuaufnehmen' ausführen, dann bekommen Sie im Bildschirmfeld Bundesland den Wert 'BY' angezeigt.

Vorbelegungen können die Neuaufnahme eines neuen Satzes erleichtern. Ein vorbelegtes Bildschirmfeld braucht dann nicht von Hand ausgefüllt werden, wenn die Vorbelegung dem entspricht, was ohnehin eingetragen werden würde. Wenn beispielsweise ein neuer Kunde aus 'BY = Bayern' aufgenommen wird, dann muß das Bildschirmfeld nicht von Hand ausgefüllt werden, da bereits der richtige Wert angezeigt wird.

TODAY - Standard-Vorbelegung mit dem aktuellen Datum

Die Anweisung DEFAULT = TODAY bewirkt, daß beim Neuaufnehmen das entsprechende DATE-Bildschirmfeld automatisch mit dem aktuellen Tages-Datum vorbelegt wird.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
f010 = auftrag.auftragsdatum,
       default = today;
```

Wenn Sie die Funktion 'Neuaufnehmen' ausführen, dann zeigt PERFORM automatisch im Bildschirmfeld 'Auftragsdatum' das aktuelle Tages-Datum an.

6.6.7 REQUIRED - Pflichteingaben verlangen

Das Schlüsselwort REQUIRED bewirkt, daß bei der Neuaufnahme das entsprechende Bildschirmfeld unbedingt ausgefüllt werden muß.

Wenn Sie einen Satz neu aufnehmen und dabei das entsprechende Bildschirmfeld nicht ausfüllen, dann

- können Sie die Neuaufnahme nicht mit START ausführen,
- positioniert PERFORM die Schreibmarke in das Bildschirmfeld, das unbedingt ausgefüllt werden muß und
- Sie erhalten eine Meldung.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
f004 = kunde.adresse1, required;
```

Wenn Sie einen Satz neu aufnehmen, dann müssen Sie in das Bildschirmfeld 'Adresse' einen Wert eintragen; anderenfalls erfolgen die oben geschilderten drei Maßnahmen.

Hinweis

Das Schlüsselwort REQUIRED wirkt nur bei der Funktion 'Neuaufnehmen'! Es verhindert nicht, daß Sie über die Funktion 'Korrigieren' einen angezeigten Wert löschen können!

Wenn Sie bei der Tabellendefinition NULLs 'Nein' angegeben haben, dann hat das eine ähnliche Wikung: Wenn ein Satz neu aufgenommen wird, dann muß in eine NULLs 'Nein' Spalte eine Eingabe gemacht werden; anderenfalls läßt sich der Satz nicht neu aufnehmen.

6.6.8 INCLUDE - Zulässigen Wertebereich definieren

Mit einer INCLUDE-Anweisung können Sie einen zulässigen Wertebereich definieren.

Sie verwenden die INCLUDE-Anweisung wie folgt: Auf das Schlüsselwort INCLUDE folgt ein Gleichheitszeichen. Darauf folgt der zulässige Wert oder die Liste zulässiger Werte; Sie müssen den Wert oder die Werte-Liste in runde Klammern setzen.
Bei einer Werte-Liste müssen Sie die einzelnen Werte durch Kommas voneinander trennen.
Einzelne CHAR-Werte einer Werte-Liste können in Anführungszeichen gesetzt sein; aber Sie müssen dann in Anführungszeichen gesetzt sein, wenn sie ein Leerzeichen oder ein Komma enthalten.
Mittels des Schlüsselwortes TO können Sie auch eine 'von ... bis ...' Bereichsangabe vornehmen.
Mittels des Schlüsselwortes NULL können Sie NULL-Werte als zulässige Werte erlauben.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
b     = auftrag.offen,
        include = (j, n),
        comments = "'j' oder 'n' eingeben";
```

Bei dem 'Offen?'-Bildschirmfeld geht es darum, ob die Abwicklung eines Auftrages noch offen (j = ja) oder bereits abgeschlossen (n = nein) ist.
Bei der Neuaufnahme oder Korrektur dürfen Sie, als Folge der INCLUDE-Anweisung, keine anderen Werte eingeben als 'j' oder 'n'. Wenn Sie einen anderen Wert eingeben, dann können Sie mit der Schreibmarke das Bildschirmfeld nicht verlassen und Sie erhalten eine Meldung.
Der bei COMMENTS definierte Kommentar weist den Anwender auf die gewollte Einschränkung hin.
Wenn Sie mit INCLUDE einen zulässigen Wertebereich definieren, dann sollten Sie unbedingt mit erläuternden Kommentaren auf die gewollte Einschränkung hinweisen!

Die Liste zulässiger Werte kann auch mehr als zwei Werte umfassen. Im folgenden Beispiel ist eine vorhandene Anweisung des FORMAT04 um eine INCLUDE-Anweisung ergänzt:

Beispiel

```
a     = kunde.bundesland,
        upshift,
        default="BY",
        include = ("BY", "HE", "BW"),
        comments = "Nur 'BY', 'HE' und 'BW' sind zugelassen!";
```

Bei dieser INCLUDE-Anweisung wären nur die Werte BY, HE und BW zulässig.

Wenn zusätzlich eine DEFAULT-Anweisung definiert ist, dann müssen die DEFAULT-Werte in der INCLUDE-Liste enthalten sein.

Hinweis

Wenn, wie im hier folgenden Beispiel, bei CHAR-Werten Leerzeichen oder Kommas in den zulässigen Werten vorkommen, dann müssen die Werte in Anführungszeichen gesetzt sein.
Beispiel:
include = ("Sport Artikel", "Warenhaus, Sportartikel")

TO - Bereichsangabe definieren

Mittels des Schlüsselwortes TO innerhalb einer INCLUDE-Anweisung können Sie auch eine 'von ... bis ...' Bereichsangabe definieren.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
f019 = posten.menge, right,
       include = (1 to 50),
       comments = "Mindestens 1, maximal 50";
```

Diese Anweisung definiert einen zulässigen Wertebereich von 1 - 50. Sie können also den Wert 1 eingeben, den Wert 50 und alle dazwischen liegenden Werte.

Hinweis

Bei einer Bereichsangabe ist standardmäßig die Reihenfolge des ASCII-Codes maßgebend. Wenn Sie z.B. die folgende Bereichsangabe machen: include (A1 to A5), dann sind die Werte A1, A2, A3, A4 und A5 zulässig.
Bei Bereichsangaben muß derjenige Wert, der im ASCII-Code vorangeht, zuerst stehen: (1 to 50) ist syntaktisch richtig, (50 to 1) ist falsch. Oder (A1 to A5) ist richtig und (A5 to A1) ist falsch.

6.6.9 VERIFY - Eingabe wiederholen

Das Schlüsselwort VERIFY dient der Datensicherheit, es vermindert die Gefahr versehentlicher Schreibfehler bei der Eingabe.
Das Schlüsselwort VERIFY bewirkt, daß Sie eine Eingabe in ein Bildschirmfeld noch ein zweitesmal durchführen müssen. Nur dann, wenn die zweite Eingabe mit der ersten identisch ist, kann der Wert in die Tabelle übernommen werden. Anderenfalls erfolgt eine Fehlermeldung.

Beispiel

Das Format-Programm des FORMAT04 enthält diese Anweisung:

```
f013 = auftrag.lieferdatum, verify;
```

Wenn Sie bei der Neuaufnahme eine Eingabe in das Bildschirmfeld 'Lieferdatum' gemacht haben, und wenn Sie versuchen, mit der Schreibmarke das Bildschirmfeld zu verlassen, dann fordert Sie PERFORM mit einer Meldung dazu auf, die Eingabe zu wiederholen.
Nur dann, wenn die zweite Eingabe mit der ersten übereinstimmt, kann der Wert in die Tabelle übernommen werden.

6.6.10 Mehrzeilen-Editor

Bei (VAR)CHAR-Spalten, die länger sind als ein Bildschirmfeld sein kann, gibt es die Möglichkeit, einen PERFORM-eigenen Mehrzeilen-Editor zu aktivieren. Der Mehrzeilen-Editor erleichtert die Eingabe und Korrektur von längeren Texten, er ermöglicht die Eingabe der Texte im Fließtext und erleichtert auch das Löschen.

Standardmäßig wird der Mehrzeilen-Editor nicht aktiviert. Das folgende Beispiel zeigt, wie man vorgehen muß:

```
database notizbuch
screen
{
Bemerkung    [f001                                                    ]
             [f001                                                    ]
             [f001                                                    ]
             [f001                                                    ]
}
tables adressen
attributes
f001 = adressen.bemerkung, wordwrap compress;
end
```

Im SCREEN-Abschnitt fügen Sie weitere Bildschirmfelder mit dem gleichen Feldbezeichner hinzu.
Im ATTRIBUTES-Abschnitt fügen Sie das Schlüsselwort WORDWRAP hinzu. WORDWRAP aktiviert den Mehrzeilen-Editor.
Zusätzlich können Sie auch noch COMPRESS hinzufügen. COMPRESS sorgt dafür, daß weitere Leerzeichen, die durch den automatischen Zeilenumbruch entstehen können, nicht in die Datenbank übernommen werden.

6.6.11 BLOB-Bildschirmfelder

Für BLOB-Bildschirmfelder in einem Format gibt es das Schlüsselwort PROGRAM, mit dem Sie das gewünschte Programm vereinbaren können. Das damit vereinbarte Programm wird jedesmal dann aufgerufen, wenn sich die Schreibmarke in einem BLOB-Bildschirmfeld befindet und Sie [I] drücken. Das folgende Beispiel zeigt, wie Sie vorgehen müssen:

```
database notizbuch
screen
{
Text  [f001          ]
}
tables adressen
attributes
f001 = adressen.langtext, program = "ced";
end
```

Im ATTRIBUTES-Abschnitt fügen Sie hinzu: PROGRAM = "programm"; Für "programm" geben Sie das gewünschte Programm an. In diesem Beispiel ist es "ced".

Bei TEXT-Spalten ist die Vereinbarung eines Programmes nicht zwingend erforderlich. Ist kein Programm vereinbart, so wird entweder der Betriebssystem-Editor oder der mit der Umgebungsvariablen DBEDIT vereinbarte Editor aufgerufen.

6.7 Join-Bildschirmfelder

Bei Mehr-Tabellen-Formaten kann es sinnvoll sein, die beteiligten Tabellen über Join-Bildschirmfelder miteinander zu verbinden.
Denn nur in diesem Fall zeigt PERFORM beim Suchen eventuell vorhandene verbundene Sätze der jeweils nicht aktuellen Tabelle an. Außerdem können Sie in diesem Fall im INSTRUCTIONS-Abschnitt MASTER-DETAIL-Beziehungen definieren, die Datenbank-Abfragen über die Funktionen 'Master' und 'Detail' komfortabler machen.

6.7.1 Definition eines Join-Bildschirmfeldes

Ein Join-Bildschirmfeld definieren Sie dadurch, daß Sie einem einzigen Bildschirmfeld in der zuordnenden Anweisung zwei Tabellenspalten (im ATTRIBUTES-Abschnitt) zuteilen.

Syntax

Die Definition eines Join-Bildschirmfeldes kann wie folgt aussehen:

```
feldbezeichner = [tabellenname1.]spaltenname1
               = [tabellenname2.]spaltenname2;
```

'spaltenname1' und 'spaltenname2' müssen als Join-Spalten geeignete Tabellenspalten sein, über die sich zwei Tabellen sinnvoll miteinander verbinden lassen.
Die Tabellennamen müssen Sie nur dann angeben, wenn 'spaltenname1' und 'spaltenname2' gleichlautend (nicht eindeutig) sind.
Der besseren Übersicht wegen sei Ihnen aber empfohlen, die Tabellennamen grundsätzlich mit anzugeben.

Beispiel

```
f000 = kunde.kunden_nr
     = auftrag.kunden_nr;
```

In diesem Beispiel werden dem einen Bildschirmfeld mit dem Feldbezeichner 'f000' zwei Tabellenspalten zugeordnet; nämlich die Tabellenspalten 'kunden_nr' der Tabelle 'kunde' und 'kunden_nr' der Tabelle 'auftrag'.

6.7.2 Bildschirmfeld-Eigenschaften bei Join-Bildschirmfeldern

Wenn Sie einem Join-Bildschirmfeld Bildschirmfeld-Eigenschaften zuordnen wollen, dann gibt es zwei Möglichkeiten:

- Sie können so verfahren, daß die Anweisung grundsätzlich wirksam ist, unabhängig davon, welche der beiden Tabellen gerade die aktuelle Tabelle ist.
- Oder Sie können so verfahren, daß die Anweisung nur dann wirksam ist, wenn die jeweilige Tabelle die aktuelle Tabelle ist.

Permanente Wirkung einer Bildschirmfeld-Eigenschaft

In diesem Fall ordnen Sie die erwünschte Anweisung (oder mehrere) der zweiten Tabellenspalten-Definition zu.

Beispiel

```
f000 = kunde.kunden_nr
     = auftrag.kunden_nr,
       comments = "Bitte die Kundenummer eingeben";
```

Bei diesem Beispiel gibt PERFORM immer dann die Meldung aus: 'Bitte die Kundenummer eingeben', wenn Sie sich mit der Schreibmarke im Bildschirmfeld f000 befinden. Und das unabhängig davon, ob die Tabelle 'kunde' oder die Tabelle 'auftrag' die jeweils aktuelle Tabelle ist.

Wirkung einer Bildschirmfeld-Eigenschaft nur bei der aktuellen Tabelle

In diesem Fall muß die vorausgehende Tabellenspalten-Definition der zuordnenden Anweisung mit einem Semikolon abgeschlossen sein.

Denn grundsätzlich verhält es sich so, daß eine Eigenschafts-Anweisung für alle vorhergehenden Tabellenspalten-Definitionen wirksam ist, die nicht mit einem Semikolon abgeschlossen wurden.

Wenn Sie folglich erreichen wollen, daß eine Anweisung nur dann wirksam wird, wenn die zweite Tabelle die aktuelle Tabelle ist, dann müssen Sie zusätzlich auch die erste Tabellenfeld-Definition mit einem Semikolon abschließen.

Beispiel

```
f000 = kunde.kunden_nr;
     = auftrag.kunden_nr,
       comments = "Die Kundennummer fuer die Tabelle AUFTRAG eingeben";
```

Bei diesem Beispiel gibt PERFORM den Kommentar nur dann aus, wenn die Tabelle 'auftrag' die aktuelle Tabelle ist.

Wenn Sie erreichen wollen, daß eine Anweisung nur dann wirksam ist, wenn die erste Tabelle die aktuelle Tabelle ist, dann schließen Sie sowohl die erste als auch die zweite Tabellenspalten-Definition mit einem Semikolon ab.
Die zweite Tabellenspalten-Definition müssen Sie deshalb mit einem Semikolon abschließen, weil jede einzelne Anweisung, mit der Sie eine Bildschirmfeld-Definition vornehmen, ein endgültig abschließendes Semikolon benötigt.

Beispiel

```
f000 = kunde.kunden_nr,
       comments = "Die Kundennummer fuer die Tabelle KUNDE eingeben";
     = auftrag.kunden_nr;
```

Bei diesem Beispiel gibt PERFORM den Kommentar nur dann aus, wenn die Tabelle 'kunde' die aktuelle Tabelle ist.

Wenn eine Anweisung jeweils dann wirksam sein soll, wenn die entsprechende Tabelle die aktuelle Tabelle ist: dann ordnen Sie beiden Tabellenspalten-Definitionen die jeweils erwünschten Anweisungen zu und schließen auch beide Tabellenspalten-Definitionen mit Semikolons ab.

Beispiel

```
f000 = kunde.kunden_nr,
       comments = "Die Kundennummer fuer die Tabelle KUNDE eingeben";
     = auftrag.kunden_nr,
       comments = "Die Kundennummer fuer die Tabelle AUFTRAG eingeben";
```

Wenn bei diesem Beispiel die Tabelle 'kunde' die aktuelle Tabelle ist, dann lautet der Kommentar: 'Die Kundennummer fuer die Tabelle KUNDE eingeben' Wenn bei diesem Beispiel die Tabelle 'auftrag' die aktuelle Tabelle ist, dann lautet der Kommentar: 'Die Kundennummer fuer die Tabelle AUFTRAG eingeben'

Hinweis

Das Schlüsselwort REVERSE, das die invertierte Abbildung eines Bildschirmfeldes durchführt, bildet eine Ausnahme: Es wirkt zu jeder Zeit und zwar unabhängig davon, welcher Tabellenspalten-Definition es zugeordnet ist und welche Tabelle die jeweils aktuelle Tabelle ist.

6.7.3 Prüfender Join

Um Unstimmigkeiten zwischen den Tabellen zu verhindern, gibt es die Möglichkeit, einem Join die Funktion 'prüfen' zuzuordnen.
Der 'prüfende Join' wirkt wie folgt: Wenn Sie einen Satz neu aufnehmen, dann überprüft er die Eingabe in das Join-Bildschirmfeld daraufhin, ob der eingegebene Wert in der verbundenen Tabelle bereits existiert. Wenn das nicht der Fall ist, dann wird die Eingabe zurückgewiesen.

Einen prüfenden Join definieren Sie dadurch, daß Sie die Join-Bildschirmfelddefinition um das Zeichen '*' ergänzen. Diejenige Tabelle, der das Zeichen '*' zugeordnet wurde, ist die dominate Tabelle; in ihr wird überprüft, ob ein eingegebener Wert bereits existiert.

Beispiel

```
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr;
```

In diesem Beispiel ist die Tabelle 'kunde' die dominante Tabelle. Wenn Sie in die Tabelle 'auftrag' einen Satz neu aufnehmen wollen, dann überprüft PERFORM Ihre Eingabe in das Bildschirmfeld f000 daraufhin, ob der Wert in der Tabelle 'kunde' bereits existiert. Wenn das nicht der Fall ist, dann weist PERFORM die Eingabe zurück. Damit verhindert der prüfende Join, daß versehentlich ein Satz mit einer falschen Kundennummer aufgenommen wird; nämlich mit einer Kundennummer, die in der Tabelle 'kunde' nicht geführt wird. Denn damit gäbe es dann einen Auftrag ohne Auftraggeber und das wäre unsinnig.

6.7.4 LOOKUP-Join

Der LOOKUP-Join ermöglicht ergänzende Informationen; er zeigt Werte einer weiteren Tabelle an.

Eine Tabelle, die über einen LOOKUP-Join verbunden ist, läßt sich nicht zur aktuellen Tabelle machen; ein angezeigter Wert kann weder geändert noch gelöscht werden.

Um einen LOOKUP-Join zu definieren, müssen Sie die folgenden Maßnahmen ergreifen:

- Im SCREEN-Abschnitt schaffen Sie ein Bildschirmfeld für den LOOKUP-Join.
- Im TABLES-Abschnitt benennen Sie diejenige Tabelle, die verbunden werden soll.
- Im ATTRIBUTES-Abschnitt ergänzen Sie eine geeignete Tabellenspalten-Definition um die unten beschriebenen Anweisungen für den LOOKUP-Join.

Beispiel

Im FORMAT04 gibt es im SCREEN-Abschnitt das LOOKUP-Bildschirmfeld 'Artikel' mit dem Feldbezeichner 'lo01'. In diesem Bildschirmfeld wird ergänzend zur Artikelnummer die Artikelbezeichnung angezeigt. Für diesen LOOKUP-Join ist im TABLES-Abschnitt die Tabelle 'artikel' benannt. Und im ATTRIBUTES-Abschnitt gibt es diese Anweisung:

```
f018 = posten.artikel_nr,
       lookup lo01 = artikel.bezeichnung
       joining artikel.artikel_nr;
```

Die Anweisung wirkt sich wie folgt aus:
Jedesmal dann, wenn im Bildschirmfeld 'f018 Artikelnummer' ein Wert 'artikel_nr' der Tabelle 'posten' angezeigt wird, dann ermittelt PERFORM, ob es in der verbundenen Tabelle 'artikel' in der Tabellenspalte 'artikel_nr' einen identischen Wert gibt.
Wenn das der Fall ist, dann ermittelt PERFORM den Wert aus der Tabellenspalte 'bezeichnung' der Tabelle 'artikel' und zeigt den ermittelten Wert im Bildschirmfeld 'lo01' an.

Die Anweisung besteht aus den folgenden zwei Teilen:
Der erste Teil dieser Anweisung entspricht einer normalen Bildschirmfeld-Definition, mit der dem Bildschirmfeld f018 die Tabellenspalte 'artikel_nr' der Tabelle 'posten' zugeordnet wird:

```
f018 = posten.artikel_nr,
```

Der zweite Anweisungs-Teil definiert den LOOKUP-Join:

```
lookup lo01 = artikel.bezeichnung
joining artikel.artikel_nr;
```

Eine LOOKUP-Join-Definition besteht aus:

- Dem Schlüsselwort LOOKUP
- Einem Feldbezeicher (hier: 'lo01'). Ein entsprechendes Bildschirmfeld muß im SCREEN-Abschnitt existieren, denn darin werden die Werte der verbundenen Tabelle angezeigt.
- Einem zuordnenden Gleichheitszeichen.
- Dem Namen derjenigen Tabellenspalte (plus Tabellennamen), dessen Werte im Bildschirmfeld angezeigt werden sollen (hier: artikel.bezeichnung).
- Dem Schlüsselwort JOINING
- Dem Namen derjenigen Tabellenspalte, über die sich die Tabellen miteinander verbinden lassen (hier: artikel.artikel_nr).

Mit dem letzten Anweisungsteil ist der Kreis zu der an erster Stelle stehende Bildschirmfeld-Definition 'f018 = posten.artikel_nr,' geschlossen: Denn in der Tabelle 'posten' gibt es eine Tabellenspalte 'artikel_nr' und in der verbundenen Tabelle 'artikel' gibt es ebenfalls eine Tabellenspalte 'artikel_nr'. Und über diese beiden Tabellenspalten lassen sich die beiden Tabellen 'posten' und 'artikel' sinnvoll miteinander verbinden, was die Anzeige eines Wertes der verbundenen Tablle überhaupt erst ermöglicht.

LOOKUP-Join als 'prüfender Join'

Den LOOKUP-Join können Sie auch als 'prüfenden Join' einsetzen. Dazu müssen Sie ihn um das Zeichen '*' ergänzen.

Beispiel

```
f018 = posten.artikel_nr,
       lookup lo01 = artikel.bezeichnung
       joining * artikel.artikel_nr;
```

Die Anweisung wirkt sich wie folgt aus:

Wenn Sie bei der Neuaufnahme eines Satzes in das Bildschirmfeld 'f018 Artikelnummer' eine Artikelnummer eingeben, die in der verbundenen Tabelle 'artikel' noch nicht existiert, dann weist PERFORM Ihre Eingabe zurück.

Wenn es Ihnen vor allem auf die Funktion 'Prüfen' ankommt und die Anzeige eines verbundenen Wertes keine Rolle spielt, dann können Sie auch auf die LOOKUP-Bildschirmfeld-Definition verzichten.

Beispiel

```
f018 = posten.artikel_nr,
       joining * artikel.artikel_nr;
```

Die Anweisung wirkt sich wie folgt aus:

Ein Wert der verbundenen Tabelle wird nun nicht mehr angezeigt. Die oben beschriebene Funktion 'Prüfen' bleibt jedoch erhalten.

6.8 Der INSTRUCTIONS-Abschnitt

Im INSTRUCTIONS-Abschnitt eines Format-Programmes können Sie:

- Master-Detail-Beziehungen definieren,
- andere Feldbegrenzer als die standardmäßigen eckigen Klammern definieren,
- Berechnungen durchführen,
- sogenannte COMPOSITE-Joins definieren,
- mit Kontrollblöcken den weiteren Ablauf festlegen, nachdem ein Wert in ein Bildschirmfeld eingegeben wurde.

Der INSTRUCTIONS-Abschnitt bietet damit sehr reichhaltige Möglichkeiten, die ausführlich im Handbuch [2] INFORMIX-SQL Nachschlagen beschrieben sind.
Besondere Beachtung verdient die Möglichkeit, mittels Kontrollblöcken den Ablauf bei der Wert-Eingabe programmierbar zu machen.

Der vorliegende Abschnitt geht nicht weiter auf den INSTRUCTIONS-Abschnitt ein. Wenn Sie sich einen Überblick über die Möglichkeiten im INSTRUCTIONS-Abschnitt verschaffen wollen, dann informieren Sie sich bitte in den Format-Programmen der Formate 'muster' und 'auftrag', die beide über umfangreiche INSTRUCTIONS-Abschnitte verfügen.

Hinweis

Der INSTRUCTIONS-Abschnitt bietet zwar sehr viele Möglichkeiten, es sind aber auch Grenzen gesetzt; nicht jede erwünschte Datenbank-Anwendung läßt sich damit auch realisieren.
Informieren Sie sich deshalb gründlich im entsprechenden Kapitel im Handbuch [2] INFORMIX-SQL Nachschlagen, bevor Sie Ihre spezielle Datenbank-Anwendung konzipieren.

Zusammenfassung

- Um ein Format zu erzeugen, benötigen Sie ein Format-Programm. Das Format-Programm schreiben Sie mit einem Editor.
- Um ein ablauffähiges Format zu erhalten, müssen Sie das Format-Programm compilieren.
- Ein Format-Programm besteht mindestens aus den vier Pflichtabschnitten DATABASE, SCREEN, TABLES und ATTRIBUTES. Es kann zusätzlich auch noch einen INSTRUCTIONS-Abschnitt enthalten.
- Im DATABASE-Abschnitt benennen Sie die Datenbank, für die das Format erstellt werden soll.
- Im SCREEN-Abschnitt gestalten Sie das Bildschirm-Layout.
- Im TABLES-Abschnitt benennen Sie diejenigen Tabellen, für die in dem Format Bildschirmfelder definiert werden.
- Im ATTRIBUTES-Abschnitt treffen Sie die die Zuordnungen zwischen Bildschirmfeldern und Tabellenspalten.
- Die zuordnenden Anweisungen können Eigenschafts-Anweisungen enthalten, mit denen Sie den Bildschirmfeldern bestimmte Eigenschaften zuweisen können.
- Bei Mehr-Tabellen-Formaten können Sie Join-Bildschirmfelder festlegen. Join-Bildschirmfelder bewirken, daß bei Datenbank-Abfragen auch ein erster (vorhandener) Satz der jeweilig verbundenen Tabelle angezeigt wird. Weiter ermöglichen sie, daß Sie 'Master-Detail-Beziehungen' zwischen den Tabellen definieren können.
- Einer Join-Bildschirmfeld-Definition kann die Funktion 'prüfen' zugeordnet werden.
- Ein LOOKUP-Join dient der ergänzenden Information.

7 SQL

Dieses Kapitel erläutert Ihnen,

- wie Sie SQL verwenden,
- welche Menüfunktionen das SQL-Dialog-Menü anbietet,
- wie Sie mit der SQL-SELECT-Anweisung eine Datenbank abfragen,
- wie Sie mit SQL Sätze neu aufnehmen, verändern oder löschen können.

7.1 Was ist SQL?

SQL ist eine Datenbank-Sprache, die der englischen Sprache angelehnt ist. Es gibt SQL-Anweisungen zu den folgenden Bereichen:

- Datendefinition
- Datenmanipulation
- Datenzugriff
- Datenintegrität

Datendefinition

Zur Datendefinition gehören Anweisungen, mit denen Sie die 'Datenstruktur' festlegen können. Dazu gehören u.a. Anweisungen, mit denen Sie eine Datenbank, eine Tabelle oder einen Index erzeugen, löschen oder ändern können.

Datenmanipulation

Zur Datenmanipulation gehören Anweisungen, mit denen Sie eine Datenbank abfragen können (SELECT) und mit denen Sie Sätze in eine Tabelle aufnehmen, löschen oder verändern können.
Die gleichen Tätigkeiten können Sie zwar auch mit einem Format durchführen, jedoch bietet SQL wesentlich umfangreichere Möglichkeiten.

Datenzugriff

Zum Datenzugriff gehören Anweisungen, mit denen Sie Maßnahmen zum 'Datenschutz' ergreifen können. Sie können Zugriffsrechte erteilen und entziehen, Tabellen für andere Anwender sperren und freigeben, Sichten (Views) auf Ausschnitte der Datenbank definieren und sich Informationen über die Tabellenstruktur beschaffen.

Datenintegrität

Zur Datenintegrität gehören Anweisungen, mit denen Sie 'Datensicherheit' herstellen können. Sie können Transaktionen durchführen, Sie können Protokolle über Änderungen führen und mit Hilfe der Protokolle eine zerstörte Datenbank wieder rekonstruieren.

Dieses Kapitel geht auf den Themenbereich 'Datenmanipulation' ein. Die Kapitel 9 ff beschäftigen sich mit den weiteren Themenbereichen.

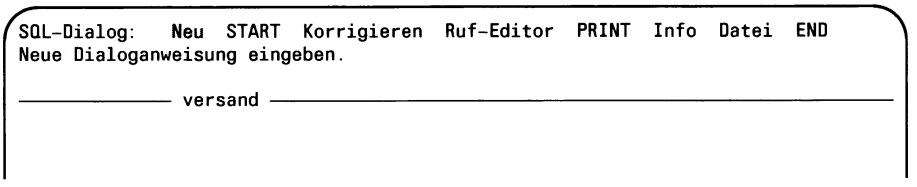
7.2 Das SQL-DIALOG-Menü

7.2.1 Das SQL-Dialog-Menü aufrufen

Um das SQL-Dialog-Menü aufzurufen, müssen Sie die folgenden Schritte unternehmen:

- Im Hauptmenü führen Sie die Funktion 'SQL-Dialog' aus.
- Wenn zuvor noch keine Datenbank ausgewählt wurde, die damit die aktuelle Datenbank ist, dann erhalten Sie jetzt den AUSWAHL-DATENBANK-Bildschirm, in dem Sie die gewünschte Datenbank auswählen.
  Die Auswahl ist nicht zwingend - wenn Sie keine bestimmte Datenbank auswählen wollen, dann drücken Sie DEL
  Anschließend erhalten Sie automatisch das SQL-Dialog-Menü.

7.2.2 Die Funktionen des SQL-DIALOG-Menüs



Im folgenden sind die Funktionen des SQL-DIALOG-Menüs ausführlich beschrieben.

Neu
eine neue SQL-Anweisung eingeben.

START
eine SQL-Anweisung ausführen.

Korrigieren
eine SQL-Anweisung korrigieren.

Ruf-Editor
den Editor des Betriebssystems zur komfortableren Texteingabe aufrufen. Editor - siehe Abschnitt 6.1.2.

PRINT
eine SQL-Anweisung ausführen und die Ergebnistabelle ausdrucken.

Info
Informationen über die Tabellen ausgeben.

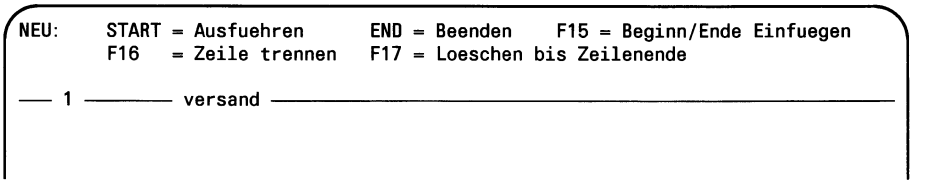
Datei
eine SQL-Anweisung in einer Datei ablegen, laden und löschen.

END
Rückkehr zum Hauptmenü.

7.2.3 'Neu' - eine neue SQL-Anweisung eingeben

Die Funktion 'Neu' ermöglicht die Eingabe einer neuen SQL-Anweisung und bietet dafür im Arbeitsbereich den SQL-Editor an.

Wenn Sie die Funktion 'Neu' ausführen, dann erhalten Sie den NEU-Bildschirm:



In der Funktionszeile erhalten Sie Hinweise auf die folgenden Funktionstasten:

START	führt die SQL-Anweisung aus.
END	beendet und führt zurück in das SQL-Dialog-Menü.
F15	Beginn/Ende Einfügen (siehe unten)
F16	Zeile trennen (siehe unten).
F17	Löschen bis Zeilenende (siehe unten).

Der Arbeitsbereich dient jetzt als Editor, Sie können Ihre gewünschte SQL-Anweisung eingeben.

Der SQL-Editor

Um Ihre SQL-Anweisung eingeben zu können, bieten Ihnen die Funktionen 'Neu' und 'Korrigieren' des SQL-Dialog-Menüs einen Editor an.

Das sichtbare Eingabefeld besteht aus 17 Zeilen und 79 Zeichen pro Zeile. Wenn Sie mit der Schreibmarke am unteren (bzw. oberen) Bildschirmrand angekommen sind, dann 'scrollt' der Bildschirm - der Text wandert um je eine Zeile nach oben (bzw. unten). Scrollen nach rechts oder links ist nicht möglich.
Links oben auf der Infozeile zeigt Ihnen eine Zahl die aktuelle Zeilen-Position der Schreibmarke an.

Die Schreibmarke positionieren

[→]	ein Zeichen nach rechts
[←]	ein Zeichen nach links
[↑]	eine Zeile nach oben
[↓]	eine Zeile nach unten
[⇥]	in Standard-Tabulatorsprüngen 8 Zeichen nach rechts
[⇤]	in Standard-Tabulatorsprüngen 8 Zeichen nach links
[Tab]	in Standard-Tabulatorsprüngen 8 Zeichen nach rechts
[Shift] [Tab]	in Standard-Tabulatorsprüngen 8 Zeichen nach links
[⇧]	um 17 Zeilen nach oben
[⇩]	um 17 Zeilen nach unten
[↵]	in die erste Spalte der nächsten Zeile
[↖]	in die erste Spalte der ersten Zeile

Löschen, einfügen und trennen

CHAR DELETE	löscht das Zeichen über der Schreibmarke
⌫	löscht das Zeichen links von der Schreibmarke
F17	löscht den Rest der Zeile ab der Schreibmarkenposition
CHAR INSERT	fügt ein Leerzeichen ein
LINE INSERT	fügt eine Leerzeile ein
F15	schaltet um zwischen dem Überschreibmodus (bei Aufruf eingeschaltet) und dem Einfügemodus. Wenn Sie sich im Einfügemodus befinden, dann können Sie auf der Schreibmarkenposition neuen Text einfügen; der rechts stehende Text wandert nach rechts. **Vorsicht** Es kann dabei passieren, daß der Text über den Bildschirmrand hinaus wandert und für Sie nicht mehr sichtbar ist; er bleibt aber weiterhin von INFORMIX interpretierbar. Um die Übersicht zu behalten, sollten Sie in diesem Fall die Zeile an geeigneter Stelle mit F16 trennen.
F16	führt einen Zeilenumbruch durch; der Text wird ab Position der Schreibmarke abgetrennt und in der nächsten Zeile ausgegeben.

Hinweis

Wenn Ihre SQL-Anweisung sehr umfangreich ist, oder wenn Sie Wert auf den größeren Komfort des Betriebssystem-Editors legen, dann können Sie auch die Funktion 'Ruf-Editor' ausführen und Ihre SQL-Anweisung mit dem Betriebssystem-Editor schreiben.

7.2.4 START - eine SQL-Anweisung ausführen

Mit START führen Sie eine SQL-Anweisung aus. Im SQL-Dialog-Menü können Sie START mit der Eingabe 's' (bzw. durch Markieren und ↵) oder mit der Funktionstaste START ausführen.
In den NEU- und KORR.-Bildschirmen können Sie nur START drücken. Denn wenn Sie hier 's' als den Anfangsbuchstaben der Funktion START eingeben, dann wird Ihre Eingabe als Texteingabe für den Editor interpretiert; nicht jedoch als eine Eingabe, mit der Sie die Funktion START ausführen wollen.

Wenn Sie Ihre SQL-Anweisung mit START (bzw. START) ausführen, dann wird die Anweisung von INFORMIX gelesen, interpretiert und ausgeführt.

Ist Ihre SQL-Anweisung eine SELECT-Anweisung, mit der Sie eine Datenbank-Abfrage ausführen, dann erhalten Sie den folgenden Bildschirm mit einer Ergebnistabelle (hier die Nachnamen der Tabelle 'kunde'):

```
ABLAUF:   Naechste  Wiederholen  END
Naechste Seite zeigen.

──────────── versand ──────────────────────────────────────────────

nachname

Pauli
Sadler
Korres
Hochfeld
Viktor
Watt
Remark
Korting
Millet
Jaeger
Keyser
Larsinger
Bergen
Albert
```

Die Funktionen:

Naechste
zeigt bei mehrseitigen Ergebnistabellen die folgende Bildschirmseite.

Wiederholen
führt zurück zum ersten Bildschirm.

END
bricht die Bildschirmausgabe der Ergebnistabelle ab und führt zurück in das SQL-Dialog-Menü.

Bei der Ausgabe der Ergebnistabelle gibt es zwei Möglichkeiten:

- Eine einzige Bildschirmseite genügt für die Ausgabe.

  Dann ist automatisch die Funktion END markiert, mit der Sie in das SQL-Dialog-Menü zurückkehren können.

- Die Ergebnistabelle erstreckt sich über mehrere Bildschirmseiten.

  Dann ist automatisch die Funktion 'Naechste' markiert, mit der Sie zu der nächsten Bildschirmseite blättern können.
  Wenn Sie sich die letzte Bildschirmseite ausgeben lassen, dann springt die Leuchtmarkierung automatisch auf die Funktion END.

Auf der jeweils letzten Bildschirmseite erhalten Sie eine Meldung darüber, wieviel Sätze bei der Datenbank-Abfrage gefunden wurden.

Enthält die SQL-Anweisung einen Syntaxfehler, dann gibt INFORMIX eine Meldung aus und führt Sie zurück in das SQL-Dialog-Menü. Die Funktion 'Korrigieren' ist markiert (siehe nächsten Abschnitt).

7.2.5 'Korrigieren' - eine SQL-Anweisung korrigieren

Mit der Funktion 'Korrigieren' erhalten Sie den KORR.-Bildschirm mit Ihrer SQL-Anweisung. Dieser Bildschirm ist bis auf den Menünamen identisch mit dem NEU-Bildschirm.
Sie können Ihre SQL-Anweisung korrigieren oder umgestalten und anschließend mit START ausführen.

INFORMIX unterstützt Sie bei der Fehlersuche; die Schreibmarke steht an der Stelle, an der der erste Fehler vermutet wird. Dabei gibt es zwei Möglichkeiten, wie die Position der Schreibmarke zu interpretieren ist:
Entweder ist der Anweisungsteil, wo die Schreibmarke steht, tatsächlich falsch, z.B. falsch geschrieben. Oder aber der vorangegangene Anweisungsteil ist falsch. Und die Schreibmarke steht deshalb erst unter dem folgenden Anweisungsteil, da dieser dann logisch nicht mehr folgen darf.

Beispiel

```
SELECT
   vorname, nachname, ort,
FROM
   kunde
```

Die gezeigte Anweisung ist falsch und läßt sich nicht ausführen. INFORMIX positioniert die Schreibmarke unter das Schlüsselwort FROM. Aber was ist falsch, das Schlüsselwort FROM oder das davorstehende Komma nach dem Spaltennamen 'ort'?
Denn wenn an dieser Stelle ein Komma steht, dann bedeutet das, daß anschließend noch ein weiterer Spaltename folgen wird. Tatsächlich aber folgt das Schlüsselwort FROM.

7.2.6 'Ruf-Editor' - den Editor des Betriebssystems aufrufen

Wenn Ihre SQL-Anweisung sehr umfangreich ist, oder wenn Sie Wert auf den größeren Komfort des Betriebssystem-Editors legen, dann können Sie die Funktion 'Ruf-Editor' ausführen.

'Ruf-Editor' ruft vorübergehend den Betriebssystem-Editor auf. Über die Möglichkeiten, den gewünschten Editor einzustellen, siehe den Abschnitt 6.1.2.

Wenn Sie den Editor wieder verlassen, dann kehren Sie automatisch wieder in das SQL-Dialog-Menü zurück und Sie können die eingegebene SQL-Anweisung mit START ausführen.

7.2.7 PRINT - Ausgabe an Drucker, Datei oder Programm

Die Funktion PRINT (bzw. PRINT)

- **führt eine Datenbank-Abfrage aus (wie START) und**
- **leitet die erstellte Ergebnistabelle an den Drucker oder in eine Datei oder an ein Shell-Programm weiter.**

Wenn Sie die Funktion PRINT ausführen, dann erhalten Sie das AUSGABE-Menü (hier mit einer Datenbank-Abfrage zur Tabelle 'kunde'):

```
AUSGABE:   Drucker  Neue-Datei  An-Ende-Datei  Pipe  END
Ausgabe zum Drucker schicken.

──────────── versand ──────────────────────────────────────────────────

nachname

Pauli
Sadler
Korres
Hochfeld
Viktor
Watt
Remark
Korting
Millet
Jaeger
Keyser
Larsinger
Bergen
Albert
```

Die Funktionen:

Drucker
: gibt das Ergebnis der Datenbank-Abfrage über den Standard-Drucker aus.

Neue-Datei
: führt zu einem Bildschirm, in dem Sie den Namen der gewünschten Datei eingeben können. Das Ergebnis der Datenbank-Abfrage wird in der angegebenen Datei abgelegt. Wenn die angegebene Datei bereits existiert, dann wird sie überschrieben.

An-Ende-Datei
: führt zu einem Bildschirm, in dem Sie den Namen der gewünschten, bereits existierenden Datei eingeben können. Das Ergebnis der Datenbank-Abfrage wird an das Ende der Datei angehängt.

Pipe
: führt zu einem Bildschirm, in dem Sie den Namen des gewünschten Shell-Programmes angeben können. Das Ergebnis der Datenbank-Abfrage wird an das angegebene Shell-Programm weitergeleitet.

Hinweis

Sie können die Funktion PRINT ausführen, nachdem Sie bereits eine Datenbank-Abfrage mit START ausgeführt haben; Sie können die Funktion aber auch ausführen, ohne die Datenbank-Abfrage zuvor mit START auszuführen, denn PRINT führt ebenfalls eine Datenbank-Abfrage durch. Im zweiten Fall verlassen Sie einfach das NEU- oder KORR.-Menü (bzw. den Ruf-Editor) und führen unmittelbar die Funktion PRINT aus. Eine vorhergehende Bildschirm-Ausgabe erfolgt dann nicht mehr.

7.2.8 'Info' - Informationen über eine Tabelle ausgeben

Mit der Funktion 'Info' können Sie sich Informationen über eine Tabelle ausgeben lassen.

Wenn Sie die Funktion 'Info' ausführen, dann erhalten Sie zunächst den INFO-FUER-TABELLE-Bildschirm, in dem Sie die Tabelle auswählen, über die Sie sich informieren wollen. Anschließend erhalten Sie das INFO-Menü; der Name der ausgewählten Tabelle wird oben links angezeigt (hier die Tabelle 'kunde'):

```
INFO - kunde:   Spalten  Indizes  Rechte  Zustand  Tabelle  END
Spaltennamen und -Typen der Tabelle zeigen.

──────────── versand ──────────────────────────────────────────
```

Die Funktionen:

Spalten
: zeigt an: Spaltennamen und zugeordnete Datentypen.

Indizes
: zeigt an: Index-Namen, Eigentümer eines Index (der den Index erzeugt hat), Indextyp (Eindeutig oder Duplikate), Spalten, die indiziert wurden.

Rechte
: zeigt an: Benutzername und zuerteilte Rechte für die folgenden SQL-Anweisungen: SELECT, UPDATE, INSERT, DELETE, INDEX, ALTER Wenn ein Anwender nicht gesondert aufgeführt ist, dann besitzt er automatisch alle unter 'public' erteilten Rechte.

Zustand
: gibt die folgenden Informationen aus: Name der Tabelle, Eigentümer (der die Tabelle erzeugt hat), Satzlänge, Anzahl der Sätze, Anzahl der Spalten, Erstellungsdatum, Namen des Zugriffsprotokolls (sofern vorhanden).
  Die Anzahl der Sätze wird (nach Neuaufnahmen und Löschungen) nur dann korrekt angezeigt, wenn Sie anschließend die SQL-Anweisung UPDATE STATISTICS ausgeführt haben. Wenn diese SQL-Anweisung noch niemals ausgeführt wurde, dann wird '0' angezeigt.

Tabelle

führt wieder in das INFO-FUER-TABELLE-Menü. Sie können eine weitere Tabelle auswählen, über deren Struktur Sie sich informieren wollen.

END
führt zurück in das SQL-Dialog-Menü.

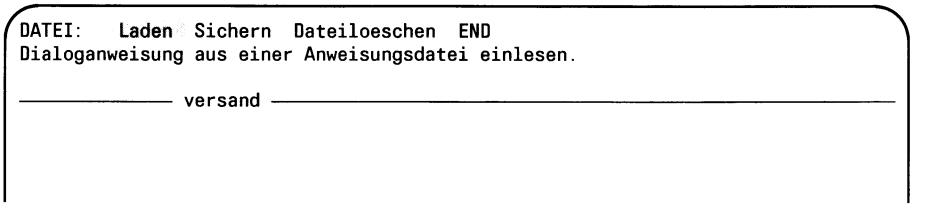
Hinweis

Die hier beschriebenen Informationen können Sie sich auch mit der SQL-Anweisung 'INFO' ausgeben lassen.

7.2.9 'Datei' - SQL-Anweisungen auf Abruf

Über die Funktion 'Datei' können Sie häufig benötigte SQL-Anweisungen in eine Datei ablegen danach beliebig oft wieder laden und ausführen lassen.

Wenn Sie die Funktion 'Datei' ausführen, dann erhalten Sie das DATEI-Menü:



Die Funktionen:

Laden
führt Sie zu einem AUSWAHL-Bildschirm, in dem Sie die Namen der Dateien mit SQL-Anweisungen angeboten bekommen. Die ausgewählte Datei wird in den SQL-Editor geladen.

Sichern
führt Sie zu einem Bildschirm, in dem Sie den gewünschten Namen für die abzulegende SQL-Anweisung angeben. Danach ist die SQL-Anweisung gesichert und Sie können sie über die Funktion 'Laden' immer wieder neu einlesen und ausführen lassen.
NAMENSKONVENTIONEN: Das erste Zeichen muß ein Buchstabe sein, der Rest des Namens kann aus Buchstaben, Ziffern und Unterstrichen bestehen. Der Name darf nicht länger als zehn Zeichen sein.

Dateiloeschen
löscht eine abgelegte SQL-Anweisung.

END
führt zurück in das SQL-Dialog-Menü.

Gesicherte SQL-Anweisungen speichert INFORMIX im aktuellen Dateiverzeichnis in dieser Art: dateiname.sql
Wenn Sie z.B. eine SQL-Anweisung unter dem Namen 'sichern01' abspeichern, dann erhalten Sie in Ihrem aktuellen Dateiverzeichnis die folgende Datei: sichern01.sql
Der AUSWAHL-Bildschirm zeigt den Dateinamen ohne den Zusatz .sql an.

Hinweis

Der AUSWAHL-Bildschirm zeigt eine SQL-Anweisung an, die bereits gesichert ist: c_database
Diese Datei enthält die Anweisungsfolge, mit der die Datenbank 'versand' erzeugt wurde. Das Kapitel 9 geht auf die einzelnen Teile dieser Anweisung ein.

7.3 Grundsätzliches zu SQL-Anweisungen

7.3.1 Schreibweise einer SQL-Anweisung

Die Schlüsselwörter, Datenbank- Tabellen- und Spaltennamen können sie groß oder klein schreiben. Um die Schlüsselwörter deutlich hervorzuheben, verwendet das Handbuch die Großschreibweise.
Werte müssen Sie so schreiben, wie sie in der Tabelle enthalten sind.
Beispiel: "ULM" ist ein anderer Wert als "Ulm"
CHAR-Werte und Kalenderdaten müssen Sie grundsätzlich in Anführungszeichen setzen; numerische Werte können Sie optional in Anführungszeichen setzen.

Sie können in ihre SQL-Anweisung beliebig viele Leerzeichen und Leerzeilen einfügen; das macht es für Sie leichter, Ihre SQL-Anweisung übersichtlich darzustellen.

7.3.2 Kommentare

Sie können eine SQL-Anweisung mit Kommentaren versehen. Ein Kommentar ist ein Text, den Sie einer SQL-Anweisung als Erläuterung hinzufügen können. Der Kommentar wird bei der Ausführung einer SQL-Anweisung überlesen und nicht als Anweisungs-Teil interpretiert.
Sinnvoll sind Kommentare z.B. bei sehr umfangreichen SQL-Anweisungen, die Sie in einer Datei ablegen, um sie bei Bedarf wieder ausführen zu können. Wenn Sie später an der SQL-Anweisung Änderungen vornehmen wollen, dann kann Ihnen ein Kommentar die Orientierung erleichtern.

Einen Kommentar definieren Sie wie folgt:

```
{kommentartext}
```

Sie leiten den Kommentar mit dieser geschweiften Klammer '{' ein, fügen den gewünschten Kommentartext hinzu und schließen den Kommentar mit dieser geschweiften Klammer '}' wieder ab.
Der Kommentar darf mehrzeilig sein; es genügt eine einleitende und eine abschließende Klammer. Weitere Klammern sind nicht erforderlich.

Beispiel

```
{ die folgende SQL-Anweisung gibt die Nachnamen
  der Tabelle 'kunde' aus }

SELECT
   nachname
FROM
   kunde
```

7.3.3 SQL-Prozeduren

Mit Semikolons können Sie mehrere aufeinanderfolgende Anweisungen zu einer Prozedur verbinden. Die Anweisungen werden nacheinander ausgeführt. Die abgelegte Datei 'c_database.sql' enthält eine solche Prozedur.

Beispiel

Diese Anweisung führt zunächst eine Datenbank-Abfrage durch, die nur die Nachnamen liefert und anschließend automatisch eine zweite Datenbank-Abfrage, die nur die Orte liefert.

```
SELECT
   nachname
FROM
   kunde;
SELECT
   ort
FROM
   kunde
```

7.4 Die SELECT-Anweisung - eine Datenbank abfragen

Mit einer SELECT-Anweisung können Sie eine Datenbank abfragen. Die folgenden Abschnitte verschaffen Ihnen einen Eindruck von den Möglichkeiten, die eine SELECT-Anweisung bietet. Das Handbuch [1] SQL informiert Sie umfassend.

7.4.1 Die Klauseln in einer SELECT-Anweisung

Eine SELECT-Anweisung muß mindestens aus den beiden Klauseln SELECT und FROM bestehen. Darüber hinaus sind noch eine ganze Reihe weiterer Klauseln möglich, die im folgenden beschrieben sind. Hier eine Übersicht:

```
SELECT
FROM
WHERE       (optional)
ORDER BY    (optional)
GROUP BY    (optional)
HAVING      (optional)
INTO TEMP   (optional)
```

7.4.2 SELECT und FROM

In der SELECT-Klausel können Sie angeben, welche Spalten einer Tabelle die Datenbank-Abfrage liefern soll.
Diesen Teil einer Datenbank-Abfrage bezeichnet man als 'Spaltenauswahl'.

Die SELECT und FROM Klauseln verwenden Sie z.B. wie folgt:

```
SELECT
   spaltenauswahl
FROM
   tabellenname
```

Für 'spaltenauswahl' können Sie beispielsweise einfach die gewünschten Spaltennamen einsetzen und für 'tabellenname' den Namen der Tabelle. Wenn Sie für 'spaltenauswahl' mehr als einen Spaltennamen angegeben, dann müssen Sie die Spaltennamen durch Kommas voneinander trennen.

Beispiel

```
SELECT
   vorname, nachname, ort
FROM
   kunde
```

Suchen nach allen Spalten

Wenn eine Datenbank-Abfrage alle Spalten einer Tabelle liefern soll, dann können Sie anstelle einzelner Spaltennamen auch dieses Zeichen '*' als Jokerzeichen verwenden:

```
SELECT
   *
FROM
   tabellenname
```

Beispiel

```
SELECT
   *
FROM
   posten
```

Eine Datenbank-Abfrage mit dieser SELECT-Anweisung führt zu folgendem Ergebnis (hier die erste Bildschirmseite):

posten_nr	auftrags_nr	artikel_nr	herstellercode	menge	gesamtpreis
1	1001	1	HRO	1	250,00
1	1002	4	HSK	1	960,00
2	1002	3	HSK	1	240,00
1	1003	9	ANZ	1	20,00
2	1003	8	ANZ	1	840,00
3	1003	5	ANZ	5	99,00
1	1004	1	HRO	1	960,00
2	1004	2	HRO	1	126,00
3	1004	3	HSK	1	240,00
4	1004	1	HSK	1	800,00
1	1005	5	NRG	10	280,00
2	1005	5	ANZ	10	198,00
3	1005	6	SMT	1	36,00
4	1005	6	ANZ	1	48,00

Eindeutigkeit der Spaltennamen

Bei Datenbank-Abfragen über mehrere Tabellen (siehe Abschnitt 7.7) muß die Eindeutigkeit der Spaltennamen gewährleistet sein. Wenn die Spaltennamen unterschiedlicher Tabellen gleiche Namen haben (z.B. bei Join-Spalten), dann ist das nicht mehr gewährleistet. Zum Beispiel gibt es sowohl in der Tabelle 'kunde' als auch in der Tabelle 'auftrag' je eine Spalte mit dem Namen 'kunden_nr'.
In diesen Fällen müssen Sie den jeweiligen Tabellennamen wie folgt mit angeben:

```
tabellenname.spaltenname
```

Vor dem Spaltennamen geben Sie den Namen der Tabelle an. 'tabellenname' und 'spaltenname' müssen Sie durch einen Punkt voneinander trennen. Ein Leerzeichen darf nicht enthalten sein.

Beispiel

```
SELECT
   kunde.vorname, kunde.nachname, kunde.ort
FROM
   kunde
```

Einen Spaltennamen zeitweilig umbenennen

Nachdem Sie eine Datenbank-Abfrage durchgeführt haben, bekommen Sie am Bildschirm eine Ergebnistabelle mit den gefundenen Treffersätzen angezeigt. Die einzelnen Spalten der Tabelle sind standardmäßig mit den Spaltennamen überschrieben.

Es gibt aber auch die Möglichkeit, einen Spaltennamen zeitweilig für die aktuelle Datenbank-Abfrage umzubenennen. Dann bekommen Sie in den Spaltenüberschriften der Ergebnistabelle die neuen und nicht mehr die bei der Tabellendefinition festgelegten Spaltennamen angezeigt.

Das Umbenennen hat noch eine weitere und vielleicht bedeutendere Wirkung: Bei SQL-Prozeduren können Sie in einer folgenden Datenbank-Abfrage anstatt der ursprünglichen Spaltennamen auch die neu definierten Spaltennamen verwenden.
Praktischen Nutzen findet das vor allem dann, wenn Sie Sie mit temporären Tabellen arbeiten (siehe Abschnitt 7.4.6).

Wenn Sie einen Spaltennamen zeitweilig umbenennen wollen, dann gilt folgende Schreibweise:

```
spaltenname spaltennameneu
```

Beispiel

```
SELECT
   nachname name, ort wohnort, kunden_nr kundennummer
FROM
   kunde
```

Wenn Sie dabei auch den Tabellennamen mit angeben, dann sieht das Beispiel so aus:

```
SELECT
   kunde.nachname name, kunde.ort wohnort, kunde.kunden_nr kundennummer
FROM
   kunde
```

Die Anweisung führt zu diesem Ergebnis (hier die erste Bildschirmseite):

```
name            wohnort         kundennummer

Pauli           Augsburg                 101
Sadler          Wasserburg               102
Korres          Rosenheim                103
Hochfeld        Muenchen                 104
Viktor          Ingolstadt               105
Watt            Ulm                      106
Remark          Rosenheim                107
Korting         Muenchen                 108
Millet          Augsburg                 109
Jaeger          Muenchen                 110
Keyser          Augsburg                 111
Larsinger       Ingolstadt               112
Bergen          Landshut                 113
Albert          Muenchen                 114
```

Spaltenausschnitt

Manche Datenbank-Anwendungen erfordern, daß nicht die vollständigen Werte einer Spalte, sondern nur ein Ausschnitt davon geliefert wird. Zu diesem Zweck können Sie bei (VAR)CHAR-Spalten wie folgt einen Spaltenausschnitt festlegen:

```
spaltenname[ganzzahl1,ganzzahl2]
```

'ganzzahl1' und 'ganzzahl2' geben das erste und das letzte Zeichen des Ausschnittes an.
'ganzzahl1' und 'ganzzahl2' müssen Sie durch ein Komma voneinander trennen. Ein Leerzeichen darf die Ausschnitts-Definition nicht enthalten. Die eckigen Klammern müssen Sie angeben.
Wenn Sie nur eine Zahl angeben, dann ist der Ausschnitt genau ein Zeichen lang.

Beispiel

```
SELECT
   nachname[1,5]
FROM
   kunde
```

Diese SELECT-Anweisung liefert das folgende Ergebnis:

```
nachname

Pauli
Sadle
Korre
Hochf
Vikto
Watt
Remar
Korti
Mille
Jaege
Keyse
Larsi
Berge
Alber
```

Aufgabe 7-1

Führen Sie Datenbank-Abfragen durch, die die folgenden Ergebnisse liefern sollen:

A) Alle Spalten der Tabelle 'artikel'.

B) Die Spalten 'artikel_nr', 'bezeichnung', 'preis' und 'stueckliste' der Tabelle 'artikel'.
Die Spalte 'artikel_nr' soll in 'artikelnummer' umbenannt werden, Die Spalte 'bezeichnung' soll in 'artikelbezeichnung' umbenannt werden, Die Spalte 'preis' soll in 'grundpreis' umbenannt werden.

C) Die Spalten 'auftrags_nr', 'kunden_nr' und 'lieferhinweis' der Tabelle 'auftrag'.
Die Spalte 'auftrags_nr' soll in 'auftragsummer' umbenannt werden, Die Spalte 'kunden_nr' soll in 'kundennummer' umbenannt werden. Von der Spalte 'lieferhinweis' sollen nur die ersten 11 Zeichen ausgegeben werden.

7.4.3 WHERE - Bedingung stellen

Mit einer optionalen WHERE Klausel können Sie mit einer Bedingung eine Satzauswahl durchführen. Wenn eine SELECT-Anweisung eine WHERE Klausel enthält, dann enthält die Ergebnistabelle nicht mehr alle Sätze der angegebenen Tabelle, sondern nur noch diejenigen, die die gestellte Bedingung erfüllen.

Die WHERE Klausel verwenden Sie wie folgt:

```
WHERE bedingung
```

Wobei sich eine 'bedingung' u.a. wie folgt zusammensetzen kann:

```
spaltenname vergleichsoperator wert
```

Wenn 'wert' ein CHAR-Wert ist, dann müssen Sie ihn in Anführungszeichen setzen. DATE-Werte müssen Sie ebenfalls in Anführzungszeichen setzen. Numerische Werte können aber müssen Sie nicht in Anführungszeichen setzen.

Beispiel

```
SELECT
   vorname, nachname, ort
FROM
   kunde
WHERE
   ort = "Muenchen"
```

Diese Anweisung führt zu folgendem Ergebnis:

```
vorname         nachname        ort

Anton           Hochfeld        Muenchen
Martin          Korting         Muenchen
Roland          Jaeger          Muenchen
Frank           Albert          Muenchen
Arnold          Sipell          Muenchen
```

Die Ergebnistabelle zeigt nur diejenigen Sätze der Tabelle 'kunde', die in der Spalte 'ort' den Wert 'München' enthalten.

Vergleichs-Operator

Diese Vergleichs-Operatoren können Sie in der WHERE Klausel verwenden:

VERGLEICHS-OPERATOR	BEDEUTUNG
=	gleich
<>	ungleich
!=	ungleich
>	größer
>=	größer gleich
<	kleiner
<=	kleiner gleich

Was jeweils größer oder kleiner ist, das hängt standardmäßig von der Reihenfolge im internationalen ASCII-Code ab. Sehen Sie dazu den vorangegangenen Abschnitt 5.2.4

Erweiterte Bedingung

Mit den Schlüsselworten AND und OR können Sie erweiterte Bedingungen stellen:

- Mit dem Schlüsselwort AND können Sie mehrere Bedingungen aneinanderhängen, die dann alle erfüllt sein müssen.
- Mit dem Schlüsselwort OR können Sie alternative Bedingungen angeben, von denen wenigstens eine erfüllt sein muß.

Beispiel

Die folgende Anweisung ermittelt diejenigen Aufträge, bei denen der Auftrag noch nicht abgeschlossen ist (offen = "j") und bei denen das Lieferdatum vor dem 1.6.90 (lieferdatum < "1.6.90") liegt:

```
SELECT
   kunden_nr, auftrags_nr, auftragsdatum,
   offen, lieferdatum
FROM
   auftrag
WHERE
      offen = "j"
   AND
      lieferdatum < "1.6.90"
```

Die Anweisung führt zu diesem Ergebnis:

```
kunden_nr auftrags_nr auftragsdatum offen lieferdatum

      106        1004 12.04.1990    j     30.04.1990
```

Die folgende Anweisung ermittelt Kundendaten derjenigen Kunden, deren Standort sich entweder in "Muenchen" oder in "Augsburg" befindet:

```
SELECT
   kunden_nr, firma, ort
FROM
   kunde
WHERE
      ort = "Muenchen"
   OR
      ort = "Augsburg"
```

Die Anweisung führt zu diesem Ergebnis:

```
kunden_nr firma                ort

      101 Pauli Sport          Augsburg
      104 Spielball            Muenchen
      108 Martin's Shop        Muenchen
      109 Sportausstattung     Augsburg
      110 Sportwaren           Muenchen
      111 Sport Keyser         Augsburg
      114 Der Sport-Albert     Muenchen
      117 Sportecke            Muenchen
```

Aufgabe 7-2

Führen Sie Datenbank-Abfragen durch, die die folgenden Ergebnisse liefern sollen:

– Die Spalten 'kunden_nr', 'lieferdatum', 'zahldatum', 'offen' der Tabelle 'auftrag'.
  Stellen Sie dabei die folgenden Bedingungen: Die Kundennummer soll '106' sein und der Auftrag soll noch offen ("j") sein.

Das Schlüsselwort MATCHES

Um bei (VAR)CHAR-Werten mit Such-Operatoren arbeiten zu können, müssen Sie anstelle eines Vergleichs-Operators das Schlüsselwort MATCHES verwenden.

Beispiel

```
SELECT
   kunden_nr, firma, nachname
FROM
   kunde
WHERE
  firma MATCHES "*ort"
```

Die Anweisung führt zu dem folgenden Ergebnis:

```
kunden_nr firma                nachname

      101 Pauli Sport          Pauli
      116 Olympia Sport        Partellman
```

Die Ergebnistabelle enthält nur zwei Sätze; nämlich diejenigen Sätze, deren Werte in der Spalte 'firma' mit der Zeichenfolge 'ort' enden.

Diese Such-Operatoren können Sie mit dem Schlüsselwort MATCHES verwenden:

SUCH-OPERATOR	BEDEUTUNG	DATENTYP
*	ersetzt beliebige Zeichenfolge	nur CHAR
?	ersetzt genau ein Zeichen	nur CHAR
[...]	zulässiger Wertebereich	nur CHAR

Der Operator '*'

Der Operator '*' ersetzt eine beliebige Zeichenfolge - auch Leerzeichen und keine Zeichen.

Beispiel

```
MATCHES "A*g"
```

Findet alle Werte, die mit 'A' beginnen und mit 'g' enden.

Der Operator '?'

Der Operator '?' ersetzt genau ein Zeichen.

Beispiel

```
MATCHES "?rank"
```

Findet alle Werte, die mit einem beliebigen Zeichen beginnen und mit der Zeichenfolge 'rank' enden (z.B. bei 'vorname' der Tabelle 'kunde').

Zulässiger Wertebereich [...]

Innerhalb der eckigen Klammern können Sie zwei oder mehrere Zeichen angeben, die enthalten sein müssen.
Auch die Angabe eines von-bis Wertebereiches ist möglich; ebenso auch die Ausgrenzung eines Wertebereiches.

Beispiel

```
MATCHES "[AE]s"
```

Findet alle Werte mit zwei Zeichen, bei denen das erste Zeichen ein 'A' oder ein 'E' ist und das zweite Zeichen ein 's' ist.

```
MATCHES "[A-E]at"
```

Findet alle Werte mit drei Zeichen, bei denen das erste Zeichen ein 'A, B, C, D' oder 'E' ist und die zwei folgenden Zeichen 'at' sind.

```
MATCHES "[^A]s"
```

Findet alle Werte mit zwei Zeichen, bei denen das erste Zeichen nicht 'A' ist und das zweite Zeichen ein 's' ist.

Aufgabe 7-3

Führen Sie eine Datenbank-Abfrage durch, die das folgende Ergebnis liefern soll:

– Alle Spalten der Tabelle 'kunde'. Stellen Sie dabei die folgende Bedingung: Der Ortsname soll mit einem 'A' oder einem 'M' beginnen.

Untergeordnete SELECT-Anweisung

Die WHERE Klausel kann auch eine untergeordnete SELECT-Anweisung enthalten, mit der Sie die Bedingung weiter präzisieren können.
Da die Handhabung von untergeordneten SELECT-Anweisungen nicht ganz einfach ist, kann an dieser Stelle nicht weiter darauf eingegangen werden. Zur Demonstration der Möglichkeiten folgt unten ein Beispiel. Das Handbuch [1] SQL liefert vollständige Informationen.

Um das folgende Beispiel verstehen zu können, müssen Sie wissen, daß die Mengenfunktion MAX den höchsten Wert einer Spalte liefert (siehe Abschnitt 7.5).

Beispiel

Diese SELECT-Anweisung liefert nur eine einzige Auftragsnummer. Der erste Teil der WHERE Klausel grenzt die Treffersätze auf diejenigen Sätze ein, bei denen die Artikelnummer '9' ist.
Der zweite Teil der WHERE Klausel mit der untergeordneten SELECT-Anweisung schließlich ermittelt von diesen Sätzen nur denjenigen Satz, der in der Spalte 'menge' den höchsten Wert enthält.

```
SELECT
   auftrags_nr
FROM
   posten
WHERE
      artikel_nr = 9
   AND
      menge = ( SELECT
                   MAX(menge)
                FROM
                   posten
                WHERE
                   artikel_nr = 9 )
```

Die Anweisung führt zu diesem Ergebnis:

```
auftrags_nr

     1012
```

7.4.4 ORDER BY - sortierte Ausgabe

Mit der ORDER BY Klausel können Sie die Ergebnis-Tabelle nach jeder in der SELECT Klausel genannten Spalte sortieren.

Die ORDER BY Klausel verwenden Sie z.B. wie folgt:

```
ORDER BY
   spaltenname1 [spaltenname2, ...spaltennamex]
```

Die in der ORDER BY genannte Spalte muß auch in der 'spaltenauswahl' der SELECT Klausel genannt sein.
Sie können nach einer Spalte sortieren oder auch nach mehreren. Wenn Sie nach mehreren Spalten sortieren, dann müssen Sie die Spaltennamen durch Kommas voneinander trennen.

Beispiel

Die folgende SELECT-Anweisung liefert eine Ergebnistabelle, die alphabetisch nach den Werten der Spalte 'ort' sortiert ist.

```
SELECT
   kunden_nr, firma, ort
FROM
   kunde
ORDER BY
   ort
```

Die Anweisung führt zu dem folgenden Ergebnis:

```
kunden_nr firma                ort

      101 Pauli Sport          Augsburg
      109 Sportausstattung     Augsburg
      111 Sport Keyser         Augsburg
      105 Der Laden            Ingolstadt
      112 Larsinger & Partner  Ingolstadt
      115 Sportladen           Landshut
      113 Sporthaus            Landshut
      117 Sportecke            Muenchen
      104 Spielball            Muenchen
      114 Der Sport-Albert     Muenchen
      110 Sportwaren           Muenchen
      108 Martin's Shop        Muenchen
      118 Sportausstatter      Passau
      107 - Sportwaren -       Rosenheim
```

Hinweis

Standardmäßig verhält es sich so, daß automatisch immer dann eine sortierte Ausgabe erfolgt, wenn eine in der SELECT Klausel genannte Spalte indiziert ist. Dann wird automatisch nach dieser Spalte sortiert. Mit der ORDER BY Klausel können Sie aber eine hiervon abweichende Sortierung nach jeder beliebigen Spalte durchführen.
Wenn die oben gezeigte SELECT-Anweisung keine ORDER BY Klausel enthält, dann erfolgt automatisch eine Sortierung nach der Spalte 'kunden_nr'.

Sortieren nach mehr als einer Spalte

Wenn Sie in der ORDER BY Klausel mehr als eine Spalte angeben, dann führen Sie damit eine geschachtelte "Sortierung in der Sortierung" durch. Sortiert wird zunächst nach der ersten genannten Spalte und dann innerhalb dieser Sortierung nach der zweiten genannten Spalte u.s.w.

Beispiel

Die folgende SELECT-Anweisung sortiert die Ausgabe zunächst nach der Spalte 'ort' und innerhalb dieser Sortierung erfolgt eine Sortierung nach der Spalte 'firma':

```
SELECT
   kunden_nr, firma, ort
FROM
   kunde
ORDER BY
   ort, firma
```

Die SELECT-Anweisung führt zu dem folgenden Ergebnis:

```
kunden_nr firma                ort

      101 Pauli Sport          Augsburg
      111 Sport Keyser         Augsburg
      109 Sportausstattung     Augsburg
      105 Der Laden            Ingolstadt
      112 Larsinger & Partner  Ingolstadt
      113 Sporthaus            Landshut
      115 Sportladen           Landshut
      114 Der Sport-Albert     Muenchen
      108 Martin's Shop        Muenchen
      104 Spielball            Muenchen
      117 Sportecke            Muenchen
      110 Sportwaren           Muenchen
      118 Sportausstatter      Passau
      107 - Sportwaren -       Rosenheim
```

Aufgabe 7-4

Führen Sie eine Datenbank-Abfrage durch, die das folgende Ergebnis liefern soll:

– Die Spalten 'auftrags_nr', 'auftragsdatum', 'kunden_nr', 'zahldatum' der Tabelle 'auftrag'.
  Die Sortierung soll nach der Spalte 'kunden_nr' erfolgen und innerhalb dieser Sortierung nach der Spalte 'zahldatum'.

7.4.5 GROUP BY und HAVING - Ergebnisse für Gruppen

Die GROUP BY Klausel wird fast ausschließlich im Zusammenhang mit Mengenfunktionen verwendet (siehe Abschnitt 7.5).
Die GROUP BY Klausel liefert Ergebnisse für Gruppen. Eine Gruppe ist eine Anzahl von Sätzen, die in einer Spalte einen identischen Wert (Gruppenkennzeichen) enthalten.
Die GROUP BY Klausel verwenden Sie z.B. wie folgt:

```
GROUP BY
   spaltenname
```

Dabei ist 'spaltenname' diejenige Spalte (Gruppenkennzeichen), über das sich die Gruppenzuordnung treffen läßt.

So gibt es z.B. in der Tabelle 'auftrag' mehrere Sätze, die die gleiche Auftragsnummer enthalten. Und das deshalb, weil ein Kunde ja mehrere Aufträge erteilt haben kann, die dann eben alle unter der gleichen Auftragsnummer geführt werden.

Beispiel

Bei der folgenden SELECT-Anweisung ist das Gruppenkennzeichen die Spalte 'kunden_nr'. Die Anweisung ermittelt, wieviel Sätze (Aufträge) je eine Gruppe enthält. Die Anzahl der Aufträge zeigt die Spalte (count(*)) an.

```
SELECT
   kunden_nr, count(*)
FROM
   auftrag
GROUP BY
   kunden_nr
```

Die Anweisung liefert das folgende Ergebnis:

```
kunden_nr  (count(*))

      101           1
      104           4
      106           2
      110           2
      111           1
      112           1
      115           1
      116           1
      117           2
```

HAVING - Bedingung stellen

Mit einer ergänzenden HAVING Klausel können Sie eine Bedingung stellen. Die HAVING Klausel ist der WHERE Klausel vergleichbar, aber sie ist nur gemeinsam mit der GROUP BY Klausel zu verwenden. Die WHERE Klausel ist hier nicht zulässig.

Die HAVING Klausel verwenden Sie wie folgt:

```
HAVING
   bedingung
```

Beispiel

Die folgende SELECT-Anweisung gibt - gegenüber der vorherigen SELECT-Anweisung - nur noch diejenigen Gruppen aus, die aus mehr als einem Satz bestehen. Ausgegeben werden die Kundennummern und die Anzahl der Aufträge derjenigen Kunden, die mehr als einen Auftrag erteilt haben.

```
SELECT
   kunden_nr, count(*)
FROM
   auftrag
GROUP BY
      kunden_nr
   HAVING
      count(*) > 1
```

Diese SELECT-Anweisung liefert das folgende Ergebnis:

```
kunden_nr  (count(*))

     104           4
     106           2
     110           2
     117           2
```

7.4.6 INTO TEMP - Ausgabe in eine temporäre Tabelle

Mit der INTO TEMP Klausel können Sie das Ergebnis einer Datenbank-Abfrage in eine temporäre Tabelle leiten. Die temporäre Tabelle existiert solange, bis Sie Ihre INFORMIX-Sitzung beendet haben oder bis Sie die temporäre Tabelle mit einer DROP TABLE Anweisung gelöscht haben.

Eine temporäre Tabelle ist einer fest installierten Tabelle gleichwertig; Sie können in ihr ebenso Datenbank-Abfragen durchführen wie in einer fest installierten Tabelle.

Damit finden temporäre Tabellen z.B. bei SQL-Prozeduren Verwendung. Sie können zum Beispiel

- mit einer SELECT-Anweisung eine Datenbank-Abfrage durchführen,
- das Ergebnis in eine temporäre Tabelle leiten und
- mit einer weiteren SELECT-Anweisung eine Datenbank-Abfrage in der temporären Tabelle durchführen.

Auf diese Weise können Sie Ergebnisse erzielen, die sich mit einer einzigen SELECT-Anweisung gar nicht oder nur sehr schwer erzielen lassen.

Hinweis

Siehe hierzu auch den UNION-Operator (Handbuch [1] SQL), mit dem Sie für zwei SELECT-Anweisungen ein gemeinsames Ergebnis erzielen können.

Die INTO TEMP Klausel verwenden Sie wie folgt:

```
INTO TEMP
   tabellenname
```

Beispiel

Die folgende SELECT-Anweisung führt in der Tabelle 'kunde' eine Datenbank-Abfrage durch, benennt dabei die Spaltennamen um (das ist nicht zwingend, dient aber der besseren Übersicht), leitet das Ergebnis in die temporäre Tabelle 'k_tab_tmp' und führt eine zweite Datenbank-Abfrage in der temporären Tabelle 'k_tab_tmp' durch.
Das erzielte Ergebnis ließe sich natürlich auch einfacher erreichen; das Beispiel ist konstruiert und dient lediglich der Demonstration.

```
SELECT
   kunden_nr k_nr_tmp, firma firma_tmp, ort ort_tmp
FROM
   kunde
WHERE
   kunden_nr > 106
INTO TEMP
   k_tab_tmp;
SELECT
   k_nr_tmp, firma_tmp
FROM
   k_tab_tmp
WHERE
   k_nr_tmp < 111
```

Die Anweisung liefert das folgende Ergebnis:

```
k_nr_tmp   firma_tmp

     107   - Sportwaren -
     108   Martin's Shop
     119   Sportausstattung
     110   Sportwaren
```

7.5 Berechnungen in einer SELECT-Anweisung

Mit numerischen Spalten können Sie Berechnungen in den vier Grundrechenarten durchführen.

Diese Rechen-Operatoren können Sie dazu verwenden:

RECHEN-OPERATOR	BEDEUTUNG
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Beispiel

Die folgende SELECT-Anweisung kalkuliert eine fünf-prozentige Erhöhung (* 1.05) der Artikel-Preise und zeigt den bisherigen und den neu errechneten Preis an.

```
SELECT
   artikel_nr, bezeichnung, stueckliste,
   preis, preis * 1.05
FROM
   artikel
```

Die SELECT-Anweisung führt zu diesem Ergebnis:

```
artikel_nr bezeichnung     stueckliste       preis (expression)

         1 Ski-Handschuhe  10/Box          250,00     262,5000
         1 Ski-Handschuhe  10/Box          800,00     840,0000
         1 Ski-Handschuhe  10/Box          450,00     472,5000
         2 Ski-Brille      24/Box          126,00     132,3000
         3 Ski-Stock       12/Colli        240,00     252,0000
         4 Fussball        24/Colli        960,00    1008,0000
         4 Fussball        24/Colli        480,00     504,0000
         5 Tennisschlaeger solo             28,00      29,4000
         5 Tennisschlaeger solo             25,00      26,2500
         5 Tennisschlaeger solo             19,80      20,7900
         6 Tennisball      24 Dosen/Box     36,00      37,8000
         6 Tennisball      24 Dosen/Box     48,00      50,4000
         7 Basketball      24/Box          600,00     630,0000
         8 Volleyball      24/Box          840,00     882,0000
         9 Volleyb. profi  solo             20,00      21,0000
```

Die Ergebnistabelle enthält eine zusätzliche Anzeige-Spalte, die mit dem Namen (expression) überschrieben ist. In dieser Spalte werden die neu errechneten Werte angezeigt.

Die Überschrift 'expression' ist für Spalten vorgesehen, die nicht in der Datenbank existieren. Sie können diese Spalte umbenennen; dazu müssen Sie lediglich einen erwünschten Namen hinzufügen.

Beispiel

In der folgenden SELECT-Anweisung ist die Spalte 'expression' in 'neu_preis' umbenannt worden.

```
SELECT
   artikel_nr, bezeichnung, stueckliste,
   preis, preis * 1.05 neu_preis
FROM
   artikel
```

Die SELECT-Anweisung führt zu diesem Ergebnis:

artikel_nr	bezeichnung	stueckliste	preis	neu_preis
1	Ski-Handschuhe	10/Box	250,00	262,5000
1	Ski-Handschuhe	10/Box	800,00	840,0000
1	Ski-Handschuhe	10/Box	450,00	472,5000
2	Ski-Brille	24/Box	126,00	132,3000
3	Ski-Stock	12/Colli	240,00	252,0000
4	Fussball	24/Colli	960,00	1008,0000
4	Fussball	24/Colli	480,00	504,0000
5	Tennisschlaeger	solo	28,00	29,4000
5	Tennisschlaeger	solo	25,00	26,2500
5	Tennisschlaeger	solo	19,80	20,7900
6	Tennisball	24 Dosen/Box	36,00	37,8000
6	Tennisball	24 Dosen/Box	48,00	50,4000
7	Basketball	24/Box	600,00	630,0000
8	Volleyball	24/Box	840,00	882,0000
9	Volleyb. profi	solo	20,00	21,0000

Hinweis

Es gilt die übliche Rechenregel 'Punkt vor Strich'. Bei gewollten Abweichungen von dieser Regel verwenden Sie runde Klammern.

Aufgabe 7-5

Führen Sie eine Datenbank-Abfrage durch, die das folgende Ergebnis liefern soll:

– Die Spalten 'auftrags_nr', 'gesamtpreis', 'menge' der Tabelle 'posten'. Es sollen nur Sätze geliefert werden, bei denen die Menge größer als '1' ist.
  Die Anweisung soll eine Berechnung durchführen, die aus der Menge und dem Gesamtpreis den jeweiligen Einzelpreis eines Artikels errechnet.
  Angezeigt werden sollen die errechneten Werte unter dem Spaltennamen 'einzelpreis'.

  Hinweis

  Es handelt sich dabei um eine Rückrechnung. Denn tatsächlich wird umgekehrt bei der Neuaufnahme eines Postens in einen Auftrag der Gesamtpreis aus Menge und Einzelpreis errechnet (siehe Format 'auftrag').

Mengenfunktionen

Mit Mengenfunktionen können Sie Sätze zählen, Summen bilden, den Durchschnitts-, höchsten- und niedrigsten Wert ermitteln.
Mengenfunktionen liefern einen einzigen Ergebnissatz für die gesamte Tabelle bzw. Ergebnissätze für Gruppen (GROUP BY).
Wenn Sie Mengenfunktionen verwenden, dann können Sie sich nicht zusätzlich einzelne Sätze einer Tabelle ausgeben lassen.

Die folgenden Mengenfunktionen gibt es:

MENGENFUNKTION	BEDEUTUNG
COUNT(*)	zählt Sätze
SUM	Summe
AVG	Durchschnittswert
MAX	höchster Wert
MIN	niedrigster Wert

Wenn Sie mehrere Mengenfunktionen angeben, dann müssen Sie diese durch Kommas voneinander trennen.

Die Mengenfunktion COUNT(*)

Mit der Mengenfunktion COUNT(*) können Sie die Anzahl der Sätze einer Tabelle ermitteln.
Wenn Sie die Mengenfunktion COUNT(*) angeben, dann dürfen Sie in der Spaltenauswahl keinen Spaltennamen zusätzlich angeben.
Ausnahme: Wenn es zusätzlich eine GROUP BY Klausel gibt, dann dürfen Sie den hier genannten Spaltennamen auch in der Spaltenauswahl angeben (siehe Beispiele im Abschnitt 7.4.5).

Beispiel

Die folgende Anweisung zählt die Anzahl der Sätze der Tabelle 'auftrag'.

```
SELECT
   COUNT(*)
FROM
   auftrag
```

Die Mengenfunktionen SUM, AVG, MAX, MIN

Bei diesen Mengenfunktionen geben Sie zusätzlich in Klammern den Spaltennamen an, dessen Werte ausgewertet werden sollen.

SUM
: ermittelt die Summe der Werte der angegebenen Spalte.

AVG
: ermittelt den Durchschnittswert der Werte der angegebenen Spalte.

MAX
: ermittelt den höchsten Wert der angegebenen Spalte.

MIN
: ermittelt den niedrigsten Wert der angegebenen Spalte.

Beispiel

Die folgende Anweisung ermittelt den Durchschnittswert, den niedrigsten und den höchsten Wert der Spalte 'zustellgebuehr' der Tabelle 'auftrag'.

```
SELECT
   AVG(zustellgebuehr),
   MIN(zustellgebuehr),
   MAX(zustellgebuehr)
FROM
   auftrag
```

Die Anweisung liefert dieses Ergebnis:

```
(avg)    (min)    (max)

13,13     5,00    25,20
```

Die Spaltenüberschriften lassen sich durch Hinzufügen der gewünschten Namen umbenennen (wie bei 'expression').

Beispiel

```
SELECT
   AVG(zustellgebuehr) durchschnitt,
   MIN(zustellgebuehr) minimum,
   MAX(zustellgebuehr) maximum
FROM
   auftrag
```

Nun liefert die Anweisung dieses Ergebnis:

```
durchschnitt  minimum  maximum

       13,13     5,00    25,20
```

Hinweis

Sätze mit NULL-Werten werden wie folgt behandelt: Die COUNT(*) Mengenfunktion zählt Sätze mit NULL-Werten. Die anderen Mengenfunktionen ignorieren NULL-Werte.

Aufgabe 7-6

Führen Sie eine Datenbank-Abfrage durch, die das folgende Ergebnis liefern soll:

- Die Anzahl der Sätze und die Spalte 'auftrags_nr' derjenigen Sätze der Tabelle 'posten', die sich über die Spalte 'auftrags_nr' gruppieren lassen.
  Es sollen nur diejenigen Gruppen ausgegeben werden, die aus mehr als einem Satz bestehen.
  Die Ausgabe der ermittelten Anzahl soll in einer Spalte mit dem Namen 'anzahl' erfolgen.

7.6 Zeitfunktionen

Dieser Abschnitt stellt einige ausgewählte Zeitfunktionen vor, mit denen Sie u.a. auf die Zeitteile einer DATE-Spalte Bezug nehmen können:

DAY
gibt den Tag der angegebenen DATE-Spalte aus.

MONTH
gibt den Monat der angegebenen DATE-Spalte aus.

YEAR
gibt das Jahr der angegebenen DATE-Spalte aus (vierstellig).

WEEKDAY
gibt den Wochentag der angegebenen DATE-Spalte aus.

DATE
gibt ein Datum entsprechend dem angegebenen Ausdruck aus.

MDY
gibt einen DATE-Wert entsprechend den drei angegebenen Ausdrükken aus.

Zusätzlich zur Zeitfunktion geben Sie in Klammern den Spaltennamen (bzw. Ausdruck) an, dessen Zeit-Werte ausgewertet werden sollen.

Beispiel

Die folgende Anweisung gibt nur diejenigen Sätze aus, bei denen der Monat des Lieferdatums '6' ist und bei denen das Jahr des Lieferdatums '1990' ist.

```
SELECT
   lieferdatum, auftragsdatum, kunden_nr, auftrags_nr
FROM
   auftrag
WHERE
      MONTH(lieferdatum) = 6
   AND
      YEAR(lieferdatum) = 1990
```

Die Anweisung liefert das folgende Ergebnis:

```
lieferdatum auftragsdatum   kunden_nr auftrags_nr

06.06.1990  01.06.1990           101        1002
08.06.1990  29.05.1990           115        1010
09.06.1990  05.06.1990           117        1012
```

Aufgabe 7-7

Führen Sie eine Datenbank-Abfrage durch, die das folgende Ergebnis liefern soll:

– Die Spalten 'auftrags_nr', 'kunden_nr', 'lieferdatum' der Tabelle 'auftrag'.
  Es sollen nur diejenigen Sätze ausgegeben werden, bei denen das Jahr 1990 ist und der Monat später als Juli (Spalte 'auftragsdatum').

7.7 Datenbank-Abfragen über mehrere Tabellen

7.7.1 Grundsätzliches

Sie können Datenbank-Abfragen über mehrere Tabellen durchführen, und damit gemeinsame Abfrage-Ergebnisse für die beteiligten Tabellen erzielen.
Dabei bedienen Sie sich einer Tabellen-Verbindungstechnik, die über geeignete Spalten (Join-Spalten) erfolgt; die Join-Spalten müssen beim Aufbau einer Datenbank von vorneherein berücksichtigt worden sein.

Mit dieser Technik ist es zum Beispiel möglich, die Tabellen 'kunde' und 'auftrag' vorübergehend miteinander zu verbinden, gemeinsam abzufragen und dabei zu ermitteln, ob und welche Aufträge ein Kunde erteilt hat.

Bei Datenbank-Abfragen über mehrere Tabellen spielen Join-Spalten eine zentrale Rolle, wie das folgende Beispiel zeigen soll:

Beispiel

In der Tabelle 'kunde' gibt es einen Satz mit der Kundennummer 104, von dem hier nur die Spalten 'firma' und 'kunden_nr' (Join-Spalte) abgebildet sind:

```
firma          kunden_nr

Spielball           104
```

Die SELECT-Anweisung, mit der dieses Ergebnis zu erzielen ist, lautet:

```
SELECT
   kunde.firma, kunde.kunden_nr
FROM
   kunde
WHERE
   kunde.kunden_nr = 104
```

In der Tabelle 'auftrag' gibt es vier Sätze mit der Kundennummer 104, von denen hier nur die Spalten 'kunden_nr' (Join-Spalte), 'auftragsdatum' und 'lieferdatum' abgebildet sind:

```
kunden_nr auftragsdatum lieferdatum

      104 20.01.1990    01.02.1990
      104 12.10.1990    13.10.1990
      104 23.03.1990    13.04.1990
      104 01.09.1990    18.09.1990
```

Die SELECT-Anweisung, mit der dieses Ergebnis zu erzielen ist, lautet:

```
SELECT
   auftrag.kunden_nr, auftrag.auftragsdatum, auftrag.lieferdatum
FROM
   auftrag
WHERE
   auftrag.kunden_nr = 104
```

Mit einer einzigen SELECT-Anweisung können Sie die beiden Tabellen 'kunde' und 'auftrag' vorübergehend miteinander verbinden, gemeinsam abfragen und dabei dieses Abfrage-Ergebnis erzielen:

```
firma          kunden_nr   kunden_nr auftragsdatum lieferdatum

Spielball            104         104 20.01.1990    01.02.1990
Spielball            104         104 12.10.1990    13.10.1990
Spielball            104         104 23.03.1990    13.04.1990
Spielball            104         104 01.09.1990    18.09.1990
```

Diese Ergebnistabelle zeigt, welche Aufträge der Kunde '104 Firma Spielball' erteilt hat.

Zur Erzielung dieses Ergebnisses spielen die Join-Spalten 'kunden_nr' der Tabelle 'kunde' und 'kunden_nr' der Tabelle 'auftrag' eine zentrale Rolle. Über sie lassen sich die beiden Tabellen sinnvoll miteinander verbinden. Die Join-Spalten enthalten übereinstimmende Werte:

```
          kunde.kunden_nr = auftrag.kunden_nr
                  |               |
firma       kunden_nr    kunden_nr    auftragsdatum lieferdatum

Spielball         104 =        104    20.01.1990    01.02.1990
Spielball         104 =        104    12.10.1990    13.10.1990
Spielball         104 =        104    23.03.1990    13.04.1990
Spielball         104 =        104    01.09.1990    18.09.1990
```

Daraus läßt sich ableiten, wie die SELECT-Anweisung mindestens beschaffen sein muß, um dieses Abfrage-Ergebnis erzielen zu können. Die SELECT-Anweisung muß eine WHERE Klausel mit der folgenden Bedingung enthalten:
Gebe nur diejenigen Sätze aus, für die gilt: die Werte 'kunde.kunden_nr' sollen identisch sein mit den Werten 'auftrag.kunden_nr'.
Umgesetzt in die SQL-Syntax lautet die Bedingung:

```
WHERE
  kunde.kunden_nr = auftrag.kunden_nr
```

Die SELECT-Anweisung, mit der das oben gezeigte Ergebnis zu erzielen ist, lautet vollständig:

```
SELECT
   kunde.firma, kunde.kunden_nr,
   auftrag.kunden_nr, auftrag.auftragsdatum, auftrag.lieferdatum
FROM
   kunde, auftrag
WHERE
      kunde.kunden_nr = 104
   AND
      kunde.kunden_nr = auftrag.kunden_nr
```

Die Join-Spalten auch in der Spaltenauswahl anzugeben, ist nicht erforderlich. Die SELECT-Anweisung kann verkürzt wie folgt lauten:

```
SELECT
   kunde.firma,
   auftrag.auftragsdatum, auftrag.lieferdatum
FROM
   kunde, auftrag
WHERE
      kunde.kunden_nr = 104
   AND
      kunde.kunden_nr = auftrag.kunden_nr
```

Die SELECT-Anweisung führt zu diesem Ergebnis:

```
firma          auftragsdatum lieferdatum

Spielball      20.01.1990    01.02.1990
Spielball      12.10.1990    13.10.1990
Spielball      23.03.1990    13.04.1990
Spielball      01.09.1990    18.09.1990
```

7.7.2 Syntax

Die folgende Darstellung zeigt, wie eine SELECT-Anweisung aufgebaut sein kann, mit der Sie zwei Tabellen über Join-Spalten miteinander verbinden können, um ein gemeinsames Abfrageergebnis zu erzielen:

```
SELECT
   [tabelle1.]spalte1 [, [tabelle1.]spalte2, ...spaltex]
   [tabelle2.]spalte1 [, [tabelle2.]spalte2, ...spaltex]
FROM
   tabelle1, tabelle2
WHERE
   [tabelle1.]join-spalte1 = [tabelle2.]join-spalte2
```

In der SELECT Klausel geben Sie diejenigen Spalten der beiden Tabellen an, die ausgegeben werden sollen.
Dabei können Sie zusätzlich die Tabellennamen mit angeben. Aber Sie müssen die Tabellennamen dann mit angeben, wenn die Spaltennamen gleichlautend sind (z.B. 'kunde.kunden_nr' und 'auftrag.kunden_nr').

In der FROM Klausel geben Sie die Namen derjenigen Tabellen an, die miteinander verbunden werden sollen.
Die Tabellennamen müssen durch Kommas voneinander getrennt sein.

In der WHERE-Klausel formulieren Sie mittels der Join-Spalten eine sinnvolle Bedingung. Die Bedingung in der WHERE Klausel kann mittels der Schlüsselwörter AND und OR beliebig erweitert werden.

Hinweis

Bei der hier erläuterten Stand der Tabellen-Verbindungstechnik werden nur diejenigen Sätze ausgegeben, die sich überhaupt über die angegebenen Join-Spalten verbinden lassen.
Mittels des Schlüsselwortes OUTER (in der FROM Klausel) können Sie sich zusätzlich weitere Sätze ausgeben lassen, die sich nicht verbinden lassen. Sehen Sie hierzu das Handbuch [1] SQL.

7.7.3 Mehr als zwei Tabellen verbinden

Eine Abfrage über mehr als zwei Tabellen erfolgt prinzipell analog zu den im vorangegangenen Abschnitt 7.7.2 genannten Regeln:

- In der SELECT Klausel geben Sie diejenigen Spalten der Tabellen an, die ausgegeben werden sollen.
- In der FROM Klausel geben Sie diejenigen Tabellen an, deren Spalten in der SELECT Klausel genannt sind und die Sie miteinander verbinden wollen. Die Tabellennamen sind durch Kommas voneinander zu trennen.
- In der WHERE Klausel formulieren Sie mittels der Join-Spalten sinnvolle Bedingungen.

Beispiel

Die folgende SELECT-Anweisung führt eine Datenbank-Abfrage über die Tabellen 'kunde', 'auftrag' und 'posten' durch. Die Anweisung liefert sämtliche Aufträge, die noch nicht abgeschlossen sind und sämtliche Posten der einzelnen Aufträge.

```
SELECT
  kunde.firma,
  auftrag.auftrags_nr, auftrag.lieferdatum,
  posten.posten_nr, posten.menge, posten.gesamtpreis
FROM
  kunde, auftrag, posten
WHERE
     kunde.kunden_nr = auftrag.kunden_nr
   AND
     auftrag.auftrags_nr = posten.auftrags_nr
   AND
     auftrag.offen = "j"
ORDER BY
   auftrag.auftrags_nr
```

Die Anweisung führt zu folgendem Ergebnis (Bei einigen Aufträgen wurde noch kein Lieferdatum vereinbart):

firma	auftrags_nr	lieferdatum	posten_nr	menge	gesamtpreis
Sport Watt	1004	30.04.1990	1	1	960,00
Sport Watt	1004	30.04.1990	2	1	126,00
Sport Watt	1004	30.04.1990	3	1	240,00
Sport Watt	1004	30.04.1990	4	1	800,00
Larsinger & Partner	1006		1	5	125,00
Larsinger & Partner	1006		2	5	190,00
Larsinger & Partner	1006		3	5	99,00
Larsinger & Partner	1006		4	1	36,00
Larsinger & Partner	1006		5	1	48,00
Sportwaren	1008	06.12.1990	1	1	840,00
Sportwaren	1008	06.12.1990	2	5	100,00

Aufgabe 7-8

Führen Sie eine Datenbank-Abfrage durch, die das folgende Ergebnis liefern soll:

- Die Datenbank-Abfrage soll über die drei Tabellen 'auftrag', 'posten' und 'artikel' erfolgen. Bedienen Sie sich dabei der Join-Spalten 'auftrags_nr' und 'artikel_nr'. Die Tabellen 'auftrag' und 'posten' lassen sich über die Join-Spalten 'auftrags_nr' verbinden, und die Tabellen 'posten' und 'artikel' lassen sich über die Join-Spalten 'artikel_nr' verbinden.

  Die Datenbank-Abfrage soll liefern:
  - die Spalte 'auftrags_nr' der Tabelle 'auftrag',
  - die Spalten 'menge' und 'gesamtpreis' der Tabelle 'posten',
  - die Spalten 'artikel_nr', 'bezeichnung' und 'preis' der Tabelle 'artikel'.

  Ausgegeben werden sollen nur diejenigen Sätze, bei denen die Menge (posten.menge) größer als '1' ist.

7.8 Sätze mit SQL modifizieren

Die folgenden Abschnitte erläutern weitere SQL-Anweisungen zur Datenmanipulation. Es handelt sich um Anweisungen, mit denen Sie

- Sätze neu aufnehmen können (INSERT INTO),
- die Werte von Sätzen verändern können (UPDATE),
- Sätze löschen können (DELETE).

Sätze neu aufnehmen, verändern oder löschen können sie auch mit einem Format, jedoch bietet Ihnen SQL wesentlich mehr Möglichkeiten. die folgenden Abschnitte sollen Ihnen einen Einblick in die Möglichkeiten bieten. Das Handbuch [1] SQL liefert vollständige Informationen.

7.8.1 INSERT INTO - Sätze neuaufnehmen

Mit der INSERT INTO Anweisung können Sie neue Sätze in eine Tabelle aufnehmen.

Die INSERT INTO Anweisung verwenden Sie z.B. wie folgt:

```
INSERT INTO tabelle
   [ (spalte1 [, spalte2, ..., spaltex]) ]
VALUES
     (wert1 [,wert2, ..., wertx] )
```

Auf die Schlüsselwörter INSERT INTO folgt der Name derjenigen Tabelle, in die der neue Satz aufgenommen werden soll.

Die darauf folgende Liste der Spaltennamen ist optional. Wenn Sie einen vollständigen Satz neu aufnehmen wollen, brauchen Sie die Liste der Spaltennamen nicht anzugeben.
Wenn Sie aber keinen vollständigen Satz aufnehmen, dann geben Sie hier an, in welche Spalten die unten angegebenen Werte aufgenommen werden sollen. Spalten, die hier nicht aufgeführt sind, wird automatisch ein NULL-Wert zugewiesen (sofern zugelassen).
Die Liste der Spaltennamen ist in runde Klammern einzuschließen und die einzelnen Spaltennamen sind durch Kommas voneinander zu trennen.

Auf das Schlüsselwort VALUES folgt die Liste der Werte, die neu in die Tabelle aufgenommen werden sollen.
Eine Werte-Liste muß in runde Klammern eingeschlossen sein. CHAR- und DATE-Werte müssen in Anführungszeichen gesetzt sein. Die Anzahl und die Reihenfolge der Werte-Liste muß mit der (evtl.) oben angegebenen Spalten-Liste übereinstimmen; ansonsten ist Anzahl und Reihenfolge der Tabellenspalten maßgebend.
Soll einer Spalte ein NULL-Wert zugewiesen werden, geben Sie an der entsprechenden Stelle das Schlüsselwort NULL an.
Einer Spalte, die keinen NULL-Wert aufnehmen kann, muß ein Wert in der Werte-Liste zugewiesen sein.
Einer SERIAL-Spalte brauchen Sie lediglich den Wert '0' als Platzhalter zuzuweisen. INFORMIX weist dann der Spalte automatisch eine eindeutige, noch nicht vergebene Nummer zu.

Beispiel

Die folgende INSERT INTO Anweisung fügt in die Tabelle 'kunde' einen neuen Satz ein.
Für die SERIAL-Spalte 'kunden_nr' ist der Platzhalter '0' angegeben. Die Spalte 'adresse2' erhält mit dem Schlüsselwort NULL einen NULL-Wert.

```
INSERT INTO kunde
VALUES
   (0, "Richard", "Bornemann",
    "Tennis Welt", "Buchenstrasse 4", NULL,
     "Rosenheim", "BY", "8200", "08031/3215" )
```

Wenn Sie diese SQL-Anweisung mit START ausführen, dann erhalten Sie die folgende Meldung:

```
Saetze eingefuegt: 1
```

Um das Ergebnis zu überprüfen, können sie die Datenbank danach wie folgt abfragen:

```
SELECT
   *
FROM
   kunde
WHERE
   nachame = "Bornemann"
```

Die Anweisung führt zu folgendem Ergebnis:

```
kunden_nr    120
vorname      Richard
nachname     Bornemann
firma        Tennis Welt
adresse1     Buchenstrasse 4
adresse2
ort          Rosenheim
bundesland   BY
plz          8200
telefon      08031/3215
```

Hinweis

Anstelle der VALUES Klausel mit der Werte-Liste können Sie auch eine SELECT Anweisung angeben.
Damit ist es möglich, die eine Tabelle mit Werten zu laden, die die SELECT Anweisung mit einer Abfrage in einer anderen Tabelle geliefert hat (siehe hierzu das Handbuch [1] SQL).

Wenn Sie mehrere neue Sätze in eine Tabelle aufgenommen haben, dann empfiehlt es sich, anschließend mit der UPDATE STATISTICS Aweisung die interne Statistik auf den neuesten Stand zu bringen (siehe hierzu das Handbuch [1] SQLen).

7.8.2 UPDATE - Sätze ändern

Mit der UPDATE-Anweisung können Sie die Werte eines oder mehrerer Sätze ändern.

Die UPDATE-Anweisung verwenden Sie z.B. wie folgt:

```
UPDATE tabelle
SET ( spalte1 [, spalte2, ..., spaltex] ) =
    ( wert1 [, wert2, ..., wertx] )
[ WHERE
     bedingung ]
```

Auf das Schlüsselwort UPDATE folgt der Name derjenigen Tabelle, deren Sätze geändert werden sollen.

Auf das Schlüsselwort SET folgt die Liste der Spalten, deren Werte geändert werden sollen.
Die Spalten-Liste ist in runde Klammern einzuschließen und die einzelnen Spaltennamen sind durch Kommas voneinander zu trennen.
Wenn alle Spalten geändert werden sollen, dann können Sie anstelle einer Spalten-Liste auch das Jokerzeichen '*' angeben.

Darauf folgt ein Gleichheitszeichen und anschließend die Liste der neuen Werte.
Die Werte-Liste ist in runde Klammern einzuschließen. CHAR- und DATE-Werte müssen in Anführungszeichen gesetzt sein. Anzahl und Reihenfolge der Werte-Liste muß mit der Spalten-Liste übereinstimmen. Wenn ein Spalteninhalt in einen NULL-Wert geändert werden soll, dann geben Sie an entsprechender Stelle das Schlüsselwort NULL an.

Die Anweisung kann eine optionale WHERE-Klausel mit einer Bedingung enthalten. Geändert werden dann nur diejenigen Sätze, für die die Bedingung erfüllt ist.

Vorsicht

Wenn Sie eine UPDATE-Anweisung ohne WHERE-Klausel ablaufen lassen, dann bedeutet daß, daß alle Sätze geändert werden sollen! Deshalb gibt INFORMIX eine entsprechende Meldung aus und Sie erhalten sicherheitshalber in der Funktionszeile eine ABLAUF: 'Ja' 'Nein' Abfrage.
Die Änderungen werden erst dann ausgeführt, wenn Sie die Funktion 'Ja' ausgeführt haben.
Wenn Sie die SQL-Anweisung auf Betriebssystemebene ablaufen lassen, dann unterbleibt die Abfrage.

Beispiel

Die folgende Anweisung ändert einige Werte des Satzes mit der Kundennummer '103'.

```
UPDATE kunde
SET ( vorname, firma, adresse2 ) =
    ( "Marie", "Marie's Sportwaren", "Postfach 4276" )
WHERE
    kunden_nr = 103
```

Nachdem Sie die Anweisung mit START ausgeführt haben, erhalten Sie die folgende Meldung:

```
Satz geaendert: 1
```

Die volle Stärke der UPDATE-Anweisung erweist sich darin, daß Sie gezielt eine ganze Reihe von Sätzen auf einmal ändern können.

Beispiel

Die folgende UPDATE-Anweisung erhöht sämtliche Artikel-Preise um 5 Prozent.

```
UPDATE artikel
SET (preis) = (preis * 1.05)
```

Da bei dieser Anweisung die WHERE-Klausel fehlt, muß die Ausführung dieser Anweisung mit 'Ja' bestätigt werden.

7.8.3 DELETE FROM - Sätze löschen

Mit der DELETE FROM Anweisung können Sie Sätze löschen.

Die DELETE FROM Anweisung verwenden Sie z.B. wie folgt:

```
DELETE FROM tabelle
[ WHERE
    bedingung ]
```

Auf die Schlüsselworte DELETE und FROM folgt der Name derjenigen Tabelle, deren Sätze gelöscht werden sollen.

Darauf folgt eine optionale WHERE Klausel, mit der Sie in einer Bedingung angeben, welche Sätze gelöscht werden sollen.

Vorsicht

Wenn Sie eine DELETE-Anweisung ohne WHERE-Klausel ablaufen lassen, dann bedeutet daß, daß alle Sätze gelöscht werden sollen! Deshalb gibt INFORMIX eine entsprechende Meldung aus und Sie erhalten sicherheitshalber in der Funktionszeile eine ABLAUF: 'Ja' 'Nein' Abfrage.
Die Löschungen werden erst dann ausgeführt, wenn Sie die Funktion 'Ja' ausgeführt haben.

Beispiel

Die folgende Anweisung löscht den Satz, der den Nachnamen 'Bornemann' enthält.

```
DELETE FROM kunde
WHERE
   nachname = "Bornemann"
```

Nachdem Sie diese Anweisung START ausgeführt haben, erhalten Sie die folgende Meldung:

```
Satz geloescht: 1
```

Hinweis

Wenn Sie Sätze gelöscht haben, dann empfiehlt es sich, die interne Statistik mit der UPDATE STATISTICS Anweisung auf den neuesten Stand zu bringen.

Zusammenfassung

- SQL ist eine Datenbank-Sprache, mit der Sie u.a. eine Datenbank abfragen, Sätze neu aufnehmen, verändern oder löschen können.
- Zur Eingabe der gewünschten SQL-Anweisung verwenden Sie entweder den SQL-Editor oder den Ruf-Editor des Betriebssystems.
- Mit der SELECT-Anweisung können Sie eine Datenbank abfragen. Eine SELECT-Anweisung besteht mindestens aus einer SELECT Klausel und einer FROM Klausel.
  Sie kann aber zusätzlich auch noch die folgenden Klauseln enthalten: WHERE Klausel, ORDER BY Klausel, GROUP BY Klausel, INTO TEMP Klausel.
- Sie können Datenbank-Abfragen auch über mehrere Tabellen durchführen. Dabei formulieren Sie in der WHERE Klausel mittels geeigneter Join-Spalten sinnvolle Bedingungen.
- Mit der INSERT-Anweisung können Sie neue Sätze in eine Tabelle aufnehmen.
- Mit der UPDATE-Anweisung können Sie die Sätze einer Tabelle verändern.
- Mit der DELETE-Anweisung können Sie Sätze einer Tabelle löschen.

8 Listen-Programm erzeugen und ausführen

Dieses Kapitel erläutert Ihnen,

- welche Schritte Sie unternehmen müssen, um ein Listen-Programm für eine speziell erwünschte Anwendung zu erzeugen,
- aus welchen Teilen ein Listen-Programm besteht,
- welche Anweisungen im Listen-Programm möglich sind und wie sie sich auswirken.

Das Kapitel erläutert nur einen Teil aller theoretisch verwendbaren Anweisungen; Sie erhalten einen Einblick in die Möglichkeiten und sind danach in der Lage, eigene Listen-Programme zu erzeugen.
Vollständige Informationen zu allen verwendbaren Anweisungen entnehmen Sie bitte dem Handbuch [2] INFORMIX-SQL Nachschlagen.

8.1 Wofür Sie Listen-Programme verwenden können

Mit Listen-Programmen können Sie z.B. die folgenden Ausdrucke erstellen:

- Adressen-Verzeichnisse,
- Telefon-Verzeichnisse,
- Etiketten,
- Formulare,
- Rechnungen,
- Angebote,
- Gehaltsabrechnungen.

Einige Beispiele hierfür finden Sie in den mitgelieferten Listen-Programmen der Datenbank 'versand', aus denen Sie Anregungen für die Gestaltung Ihrer eigenen Listen-Programme entnehmen können.

8.2 Grundsätzliches

Eine ablauffähiges Listen-Programm ist formal ganz ähnlich zu erzeugen wie ein Format.

Wie bei einem Format, so vollzieht sich auch das Erzeugen eines Listen-Programmes grundsätzlich in diesen zwei Schritten:

- Zunächst benötigen Sie ein Listen-Programm mit dem Quelltext.
- Dann müssen Sie das Listen-Quellprogramm compilieren. Damit steht Ihnen ein ablauffähiges Listen-Programm zur Verfügung.

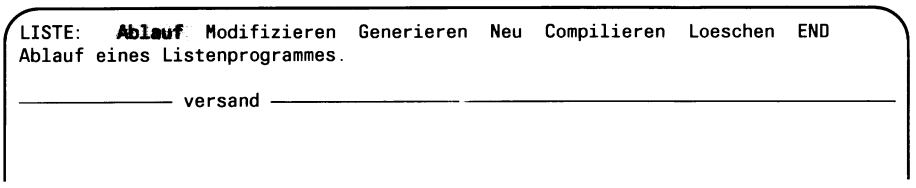
Wenn Sie ein Standard-Listen-Programm erzeugen, dann bleibt der oben beschriebene zweiteilige Erzeugungsprozeß für Sie unsichtbar.
Denn was Sie auch von Hand tun könnten, das erledigt ACE automatisch für Sie (ACE: INFORMIX-Komponente, die für das Compilieren, für den Ablauf von Listen-Programmen und für das Erzeugen von Standard-Listen-Programmen zuständig ist).

Wenn Sie ein Standard-Listen-Programm erzeugen, dann erzeugt ACE für Sie

- das Listen-Programm mit dem Quelltext und
- führt automatisch das Compilieren durch.

8.2.1 Das LISTE-Menü

Wenn Sie im Hauptmenü die Funktion 'Liste' ausführen, dann erhalten Sie das LISTE-Menü:



Die Funktionen:

Ablauf
ein bestehendes Listen-Programm ablaufen lassen.

Modifizieren
ein bestehendes Listen-Programm unter Verwendung des voreingestellten Editors verändern (Editor - siehe den Abschnitt 6.1.2).

Generieren
Ein neues Standard-Listen-Programm automatisch von ACE erzeugen lassen.

Neu
ein neues Listen-Programm unter Verwendung des voreingestellten Editors schreiben (Editor - siehe den Abschnitt 6.1.2).

Compilieren
ein Listen-Quellprogramm zu einem ablauffähigen Listen-Programm compilieren.

Loeschen
ein Listen-Programm löschen. Gelöscht werden die beiden Dateien 'listenname.ace' und 'listenname.arc' (siehe den Abschnitt 8.2.4).

END
Rückkehr in das INFORMIX-Hauptmenü.

Die Funktion 'Neu' ruft den voreingestellten Editor auf (Editor - siehe Abschnitt 6.1.2). Mit Hilfe des Editors können Sie ein Listen-Programm völig neu schreiben.
Oft ist es aber einfacher, wenn Sie zunächst über 'Generieren' ein Standard-Listenprogramm erzeugen und anschließend über 'Modifizieren' das dabei automatisch erzeugte Listen-Programm in den Editor einlesen und dann nach Ihren Wünschen verändern.

8.2.2 Listen-Programm erzeugen, modifizieren und ausführen

Dieser Abschnitt erläutert Ihnen die formalen Schritte, die Sie unternehmen müssen, um ein Listen-Programm (bzw. Standard-Listen-Programm) zu erzeugen, zu modifizieren und auszuführen.

Ein Standard-Listen-Programm erzeugen Sie wie folgt:

- Im Hauptmenü führen Sie die Funktion 'Liste' aus.
- Im LISTE-Menü führen Sie die Funktion 'Generieren' aus.
- Wenn noch keine Datenbank ausgewählt wurde und damit die aktuelle Datenbank ist, dann erhalten Sie jetzt einen Bildschirm, in dem Sie die gewünschte Datenbank auswählen, ansonsten gleich weiter mit dem nächsten Schritt.
- Sie erhalten einen Bildschirm, der der Eingabe des gewünschten Listen-Namens dient.
  NAMENSKONVENTIONEN: Der Name darf aus Groß- und Kleinbuchstaben, Zahlen und Unterstrichen bestehen. Das erste Zeichen muß ein Buchstabe sein. Der Name darf nicht länger als 10 Zeichen sein.
- Wählen Sie die gewünschte Tabelle aus. Sie können nur eine Tabelle auswählen, die Auswahl einer weiteren Tabelle ist nicht möglich.
- Nach erfolgter Auswahl wird automatisch das Standard-Listen-Programm generiert. Danach werden Sie in das LISTE-Menü zurückgeführt.

Ein neues Listen-Programm erzeugen Sie wie folgt:

Wenn Sie ein Listen-Programm von Anfang an völlig neu schreiben wollen, dann unternehmen Sie die hier aufgeführten Schritte.

- Im Hauptmenü führen Sie die Funktion 'Liste' aus.
- Im LISTE-Menü führen Sie die Funktion 'Neu' aus.
- Sie erhalten einen Bildschirm, der der Eingabe des gewünschten Listen-Namens dient.
  NAMENSKONVENTIONEN: Der Name darf aus Groß- und Kleinbuchstaben, Zahlen und Unterstrichen bestehen. Das erste Zeichen muß ein Buchstabe sein. Der Name darf nicht länger als 10 Zeichen sein.
- Danach wird automatisch der eingestellte Editor aufgerufen; Sie können den Text Ihres Listen-Programmes eingeben.
- Verlassen Sie den Editor - Sie kehren automatisch in das INFORMIX-Menüsystem zurück.
- Sie erhalten ein Menü mit den Funktionen 'Compilieren', 'Sichern-und-END', 'Verwerfen-Und-END'.
  Führen Sie die Funktion 'Compilieren' aus.
- Sollten beim Compilieren Syntaxfehler festgestellt worden sein, dann unternehmen Sie bitte die im folgenden Abschnitt "Fehlerbehandlung" beschriebenen Schritte. Im anderen Fall führen Sie abschließend die Funktion 'Sichern-und-END' aus.

Ein Listen-Programm modifizieren Sie wie folgt:

- Im Hauptmenü führen Sie die Funktion 'Liste' aus.
- Im LISTE-Menü führen Sie die Funktion 'Modifizieren' aus.
- Wählen Sie das Listen-Programm aus, das modifiziert werden soll.
- Nun wird automatisch der eingestellte Editor aufgerufen und mit der Datei 'listenname.ace', die das Listen-Programm enthält, geladen. Sie können die erwünschten Änderungen durchführen.
- Verlassen Sie den Editor - Sie kehren automatisch in das INFORMIX-Menüsystem zurück.
- Sie erhalten ein Menü mit den Funktionen 'Compilieren', 'Sichern-und-END', 'Verwerfen-Und-END'.
  Führen Sie die Funktion 'Compilieren' aus. Sollten beim Compilieren Syntaxfehler festgestellt worden sein, dann unternehmen Sie bitte die im folgenden Abschnitt "Fehlerbehandlung" beschriebenen Schritte.
- Sollten keine Syntaxfehler festgestellt worden sein, dann führen Sie anschließend die Funktion 'Sichern-und-END' aus; Sie kehren automatisch in das LISTE-Menü zurück.

Ein Listen-Programm führen Sie wie folgt aus:

- Im Hauptmenü führen Sie die Funktion 'Liste' aus.
- Im LISTE-Menü führen Sie die Funktion 'Ablauf' aus.
- Sie erhalten einen Bildschirm, der der Auswahl des gewünschten Listen-Programmes dient.
- Nach erfolgter Auswahl wird das Listen-Programm ausgeführt. Dabei ist es möglich, daß das Listen-Programm von Ihnen am Bildschirm die Eingabe von Variablen-Werten anfordert, ohne die das Listen-Programm nicht weiter ablaufen kann.
  Wo die Ausgabe der Liste erfolgt (Drucker, Bildschirm, Programm), ist im Quelltext des Listen-Programms festgelegt.

8.2.3 Fehlerbehandlung

Wenn Ihnen ein Syntaxfehler unterlaufen ist, dann erhalten Sie nach dem Compilieren automatisch diesen Bildschirm mit einer Meldung:

```
COMPILIEREN LISTE:   Korrigieren  END
Anweisungsliste fuer ein Listenprogramm korrigieren.

_____________ versand ____________________________________________________________

 nnn: Fehler in den Listenprogramm-Anweisungen.
```

Im Handbuch [3] Fehlermeldungen sind die möglichen Fehlernummern aufgelistet.

Um den Fehler zu korrigieren, gehen Sie wie folgt vor:

1. Führen Sie die Funktion 'Korrigieren' aus.

Danach wird automatisch wieder der Editor aufgerufen und mit dem Listen-Programm geladen. Das Listen-Programm enthält einen erläuternden Kommentar dieser Art:

```
database versand

select
#^
#
#      In Zeile 3, Zeichen 1, wurde ein syntaktischer Fehler gefunden.
# Fehler-Nr.: -8018
#
...
...
...
```

ACE hat einen Kommentar in das Listen-Programm eingefügt, der Sie über die vermutete Fehlerquelle informiert. Die Kommentarzeilen sind mit einleitenden '#' kenntlich gemacht.
Die Fehleranzeige in diesem Beispiel macht darauf aufmerksam, daß an dieser Stelle das Schlüsselwort END erwartet wurde, das den DATABASE-Abschnitt abschließt. Das Schlüsselwort SELECT ist hier falsch.
Sie brauchen die Kommentarzeilen nicht zu löschen, denn sie werden beim erneuten Compilieren von ACE nicht als Text interpretiert und ignoriert.

2. Führen Sie die erforderlichen Korrekturen durch.
3. Verlassen Sie anschließend wieder den Editor.
4. Jetzt erhalten Sie wieder den Bildschirm mit den Funktionen 'Compilieren, Sichern-und-End, Verwerfen-und-END'.
5. Führen Sie die Funktion 'Compilieren' aus.
6. Wenn noch ein zweiter Syntaxfehler festgestellt wurde, dann beginnen Sie wieder mit dem 1. Schritt.
   Ansonsten führen Sie jetzt die Funktion 'Sichern-und-END' aus.

8.2.4 Ein Listen-Programm - zwei Dateien

Zu jedem ablauffähigen Listen-Programm gehören zwei Dateien:

- Eine Datei mit dem von Ihnen geschriebenen (bzw. über die Funktion 'Generieren' automatisch erzeugten) Listen-Programm mit dem Quelltext.
- Eine Datei, die das compilierte Listen-Programm enthält.

Die Datei mit dem Quelltext hat den Namen 'listenname.ace'; beispielsweise LISTE01.ace
Die Datei mit dem compilierten Listen-Programm hat den Namen 'listenname.arc'; beispielsweise LISTE01.arc
Die Dateien befinden sich im aktuellen Dateiverzeichnis; dem Dateiverzeichnis, in dem Sie sich bei der Erzeugung eines Listen-Programmes befinden.
Die Listen-Dateien der Datenbank 'versand' befinden sich im Dateiverzeichnis 'vertiefung' Ihres HOME-Dateiverzeichnisses.

8.3 Die Abschnitte eines Listen-Programmes

Ein Listen-Programm besteht aus den folgenden Abschnitten:

DATABASE-Abschnitt (Pflicht),
DEFINE-Abschnitt (optional),
INPUT-Abschnitt (optional),
OUTPUT-Abschnitt (optional),
SELECT-Abschnitt oder READ-Abschnitt (Pflicht),
FORMAT-Abschnitt (Pflicht).

Jeder Abschnitt muß mit dem Schlüsselwort END abgeschlossenen werden.

Im folgenden sind die Abschnitte eines Listen-Programmes ausführlich beschrieben.

Anweisungen anordnen

Sie können die Schlüsselwörter groß oder klein schreiben; und Sie können beliebig viele Leerzeilen und Leerzeichen einfügen, so daß Sie das Listen-Programm übersichtlich gestalten können.
Um die Schlüsselwörter deutlich hervorzuheben, verwendet das Handbuch die Großschreibweise.

Kommentare

Sie können ein Listen-Programm mit Kommentaren versehen. Ein Kommentar ist ein Text, den Sie programmerläuternd hinzufügen können und der beim Compilieren ignoriert wird.

Einen Kommentar definieren Sie wie folgt:

```
{kommentartext}
```

Der Kommentartext wird in geschweifte Klammern eingeschlossen.

Beispiel

```
{ der DATABASE-Abschnitt der LISTE01: }

database versand
end
...
```

Listen-Programm bei der LISTE01

Dieser Kapitel-Abschnitt gibt Ihnen anhand der LISTE01 einen Überblick über die einzelnen Abschnitte eines Listen-Programmes. Die folgenden Kapitel-Abschnitte erläutern Details.

Bei der LISTE01 sind das die einzelnen Abschnitte:

```
                DATABASE-Abschnitt
database versand
end
```

```
                DEFINE-Abschnitt
define variable firmenname char(20)
end
```

```
                INPUT-Abschnitt
input prompt for firmenname
using "Geben Sie den Namen der Firma ein: "
end
```

```
                OUTPUT-Abschnitt
output report to printer
end
```

```
                SELECT-Abschnitt
select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
where
   firma = $firmenname
end
```

```
                FORMAT-Abschnitt
format
page header
print "K u n d e n d a t e n:"
skip 2 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
end
```

Wenn Sie die LISTE01 ablaufen lassen, dann erhalten Sie am Bildschirm

die Aufforderung, einen Firmennamen einzugeben.
Wenn Sie den Namen 'Sportecke' eingeben, dann erhalten Sie den folgenden Ausdruck:

```
K u n d e n d a t e n:

Kundennummer:      117
Firma:             Sportecke
Nachname:          Sipell
Adresse:           Marktplatz 4
Postleitzahl:      8000
Ort:               Muenchen
```

Das folgende Schaubild zeigt, wie dabei die einzelnen Abschnitte des Listen-Programmes zusammenwirken:

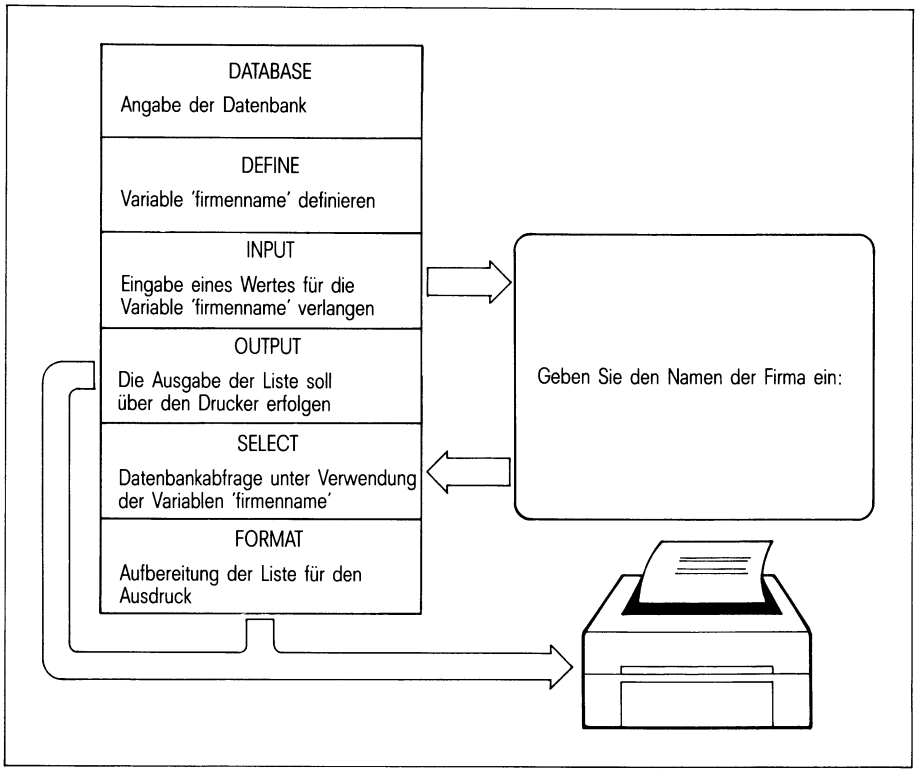


Bild 8-1 Zusammenspiel der einzelnen Abschnitte

8.4 Der DATABASE-Abschnitt

Der DATABASE-Abschnitt besteht aus dem Schlüsselwort DATABASE und dem Namen derjenigen Datenbank, für die das Listen-Programm bestimmt ist.
Bei einem Standard-Listen-Programm steht hier immer der Name derjenigen Datenbank, die die aktuelle Datenbank war, als Sie das Standard-Listen-Programm erzeugt haben.
Der DATABASE-Abschnitt muß, wie jeder andere Abschnitt auch, mit dem Schlüsselwort END abgeschlossen werden.

8.5 Der DEFINE-Abschnitt

Der DEFINE-Abschnitt ist optional. Sie verwenden ihn nur dann, wenn Sie

- Variablen definieren wollen, die im weiteren Ablauf des Listen-Programmes verwandt werden sollen bzw. wenn Sie
- das Listen-Programm von Betriebssystemebene aus aufrufen wollen und dabei Parameter übergeben wollen (Anweisung PARAM) bzw. wenn Sie
- weiter unten statt eines SELECT-Abschnittes einen READ-Abschnitt angegeben haben; in diesem Fall geben Sie hier im DEFINE-Abschnitt an (Anweisung ASCII), welche Datentypen die Daten der mit READ zu lesenden Eingabedatei haben.

Dieses Kapitel behandelt nur die Definition von Variablen. Auf die Anweisungen PARAM und ASCII wird nicht weiter eingegangen. Siehe hierzu das Handbuch [2] INFORMIX-SQL Nachschlagen.

Die Wertzuweisung für die hier definierten Variablen kann auf die folgenden Weisen erfolgen:

- entweder durch eine Eingabe am Bildschirm, wenn eine entsprechende INPUT-Anweisung definiert ist (siehe unten) oder
- durch eine LET-Anweisung im FORMAT-Abschnitt.

Beispiel

Die LISTE01 enthält ein Anwendungsbeispiel für eine Variable. Der Sinn dieser Variablen erschließt sich aus dem Zusammenspiel der drei Abschnitte DEFINE, INPUT und SELECT:
Die LISTE01 gibt einen Satz der Tabelle 'kunde' aus (SELECT). Welchen Satz der Tabelle sie ausgibt, hängt von der Eingabe am Bildschirm ab (INPUT), die vor Ablauf des Listen-Programmes gemacht werden muß.

Dies ist der DEFINE-Abschnitt der LISTE01, in dem die Variable zunächst vereinbart wird:

```
define variable firmenname char(20)
end
```

Diese Anweisung definiert eine Variable mit dem Namen 'firmenname'. Die Variable ist vom Datentyp CHAR und kann maximal 20 Zeichen aufnehmen.

8.6 Der INPUT-Abschnitt

Der INPUT-Abschnitt ist optional. Sie verwenden ihn nur dann, wenn beim Start des Listen-Programmes eine Variable (oder mehrere) mit einem Wert belegt werden soll.
Wenn ein INPUT-Abschnitt definiert wurde, dann wird das Listen-Programm erst dann weiter ausgeführt, nachdem die verlangte Eingabe(n) durchgeführt wurde.

Im INPUT-Abschnitt geben Sie an,

- für welche Variable eine Eingabe gemacht werden soll - die Variable muß im DEFINE-Abschnitt definiert worden sein,
- welcher Text dabei als Eingabeaufforderung am Bildschirm ausgegeben werden soll.

Beispiel

Die LISTE01 enthält den folgenden INPUT-Abschnitt:

```
input prompt for firmenname
using "Geben Sie den Namen der Firma ein: "
end
```

Die Anweisung wirkt sich wie folgt aus: Nachdem das Listen-Programm gestartet wurde, wird zunächst am Bildschirm dieser Text als Eingabeaufforderung ausgegeben: 'Geben Sie den Namen der Firma ein:'
Das Listen-Programm wird erst dann weiter ausgeführt, nachdem eine Eingabe erfolgt ist und die Eingabe mit ↵ abgeschlossen wurde.
Die Eingabe belegt die Variable 'firmenname', deren Wert im weiteren Ablauf des Listen-Programmes im SELECT-Abschnitt verwendet wird (siehe Bild 8-1).

8.7 OUTPUT-Abschnitt

Der OUTPUT-Abschnitt ist optional. Sie verwenden ihn nur dann, wenn Sie zu einem oder beiden der folgenden Bereiche Festlegungen treffen wollen:

- Listenausgabe (Drucker, Datei oder Programm),
- Listenränder

8.7.1 Listenausgabe

Wenn kein OUTPUT-Abschnitt festgelegt ist, dann erfolgt die Ausgabe des Listenprogrammes über den Bildschirm. Mit einer REPORT TO Klausel im OUTPUT-Abschnitt können Sie die Ausgabe über den Drucker oder die Weiterleitung in eine Datei oder an ein Programm veranlassen:

REPORT TO PRINTER
: gibt die Liste über den eingestellten Standard-Drucker aus (lpr).
Ein Standard-Listen-Programm enthält einen OUTPUT-Abschnitt mit einer REPORT TO PRINTER Klausel.

REPORT TO PIPE "programm"
: leitet die Liste an das bei 'programm' benannte Shell-Programm weiter.

REPORT TO "datei"
: leitet die Liste in die bei 'datei' benannte Datei. Eine bereits bestehende Datei des angegebenen Namens wird dabei überschrieben.

Beispiel

```
output
report to printer
end
```

Diese Anweisung veranlaßt den automatischen Ausdruck der Liste über den eingestellten Standard-Drucker (lpr).

8.7.2 Listenränder

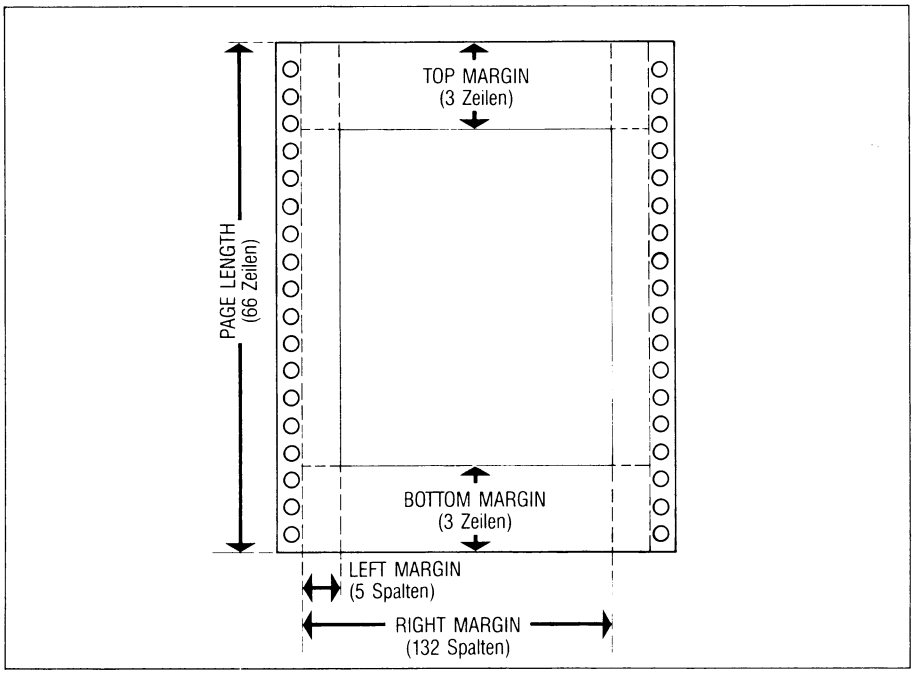


Bild 8-2 Listenränder

Standardmäßig sind die Listenränder so gesetzt, wie in Bild 8-2 gezeigt. Mit den folgenden Anweisungen können Sie vom Standard abweichende Werte für die Listenränder einstellen:

LEFT MARGIN ganzzahl
linker Listenrand (Standard: LEFT MARGIN 5)

RIGHT MARGIN ganzzahl
rechter Listenrand (Standard: RIGHT MARGIN 132)

TOP MARGIN ganzzahl
oberer Listenrand (Standard: TOP MARGIN 3)

BOTTOM MARGIN ganzzahl
unterer Listenrand (Standard: BOTTOM MARGIN 3)

PAGE LENGTH ganzzahl
Seitenlänge (Standard: PAGE LENGTH 66)

Beispiel

```
output
report to printer
right margin 80
page length 70
end
```

Diese Anweisung trifft die folgenden Festlegungen:

Der rechte Listenrand ist auf 80 Zeichen und die Seitenlänge auf 70 Zeilen gesetzt. Alle anderen Listenränder entsprechen den Standard-Werten. Die Ausgabe des Listenprogramms erfolgt über den Drucker.

8.8 SELECT- oder READ-Abschnitt

Die Datenbereitstellung für eine Liste kann auf die folgenden Weisen geschehen:

- durch einen SELECT-Abschnitt, der die benötigten Daten durch eine Datenbankabfrage beschafft, oder alternativ
- durch einen READ-Abschnitt, der die Daten aus einer angegebenen Betriebssystem-Datei einliest. In diesem Fall ist ein DEFINE-Abschnitt mit der ASCII-Anweisung zwingend.

Dieses Kapitel geht auf einen möglichen READ-Abschnitt nicht weiter ein. siehe hierzu das Handbuch [2] INFORMIX-SQL Nachschlagen.

Im SELECT-Abschnitt sorgen Sie für die erwünschte Datenbereitstellung. Dazu definieren Sie nach Muster einer SQL-SELECT-Anweisung eine Datenbank-Abfrage (siehe Kapitel 7). Wenn Sie eine Variable in einer SELECT-Anweisung verwenden, dann müssen Sie ihr einen '$' voranstellen.

Beispiel

Die LISTE01 enthält den folgenden SELECT-Abschnitt:

```
select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
where
   firma = $firmenname
end
```

Die SELECT-Anweisung in diesem Abschnitt stellt einige Daten der Tabelle 'kunde' bereit. Bereitgestellt wird hier entsprechend der WHERE Klausel immer nur ein Satz. Welcher das ist, hängt von der verlangten Eingabe für die Variable 'firmenname' ab.

Hinweis

Der SELECT-Abschnitt kann aus einer einzigen oder aus mehreren aufeinanderfolgenden, durch Semikolons geketteten SELECT-Anweisungen bestehen (siehe den Abschnitt 7.3.3).
Bei geketteten SELECT-Anweisungen gelten folgende Regeln: Jede SELECT-Anweisung (bis auf die letzte) muß eine INTO TEMP Klausel enthalten. Nur die Daten der letzen SELECT-Anweisung werden bereitgestellt.

Nur die letzte SELECT-Anweisung darf eine ORDER BY Klausel enthalten. in Ihr darf kein vorangestellter Tabellenname enthalten sein (nicht 'tabellenname.spaltenname', sondern nur 'spaltenname').

SELECT/READ-Abschnitt und FORMAT-Abschnitt wirken zusammen

Der SELECT- bzw. READ-Abschnitt sorgt lediglich für die Bereitstellung der Daten. Ob und in welchem Umfang die bereitgestellten Daten tatsächlich in der Liste ausgegeben werden, hängt ausschließlich von den entsprechenden Anweisungen im FORMAT-Abschnitt ab.

Beispiel

Die SELECT-Anweisung in der LISTE01 lautet im Original wie folgt:

```
select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
where
   firma = $firmenname
end
```

Die Anweisung könnte bei gleichem Ausgabeergebnis aber auch wie folgt aussehen:

```
select
   *
from
   kunde
where
   firma = $firmenname
end
```

Im ersten Fall werden nur die Spalten 'kunden_nr, firma, nachname, adresse1, plz, ort' der Tabelle 'kunde' bereitgestellt.
Im zweiten Fall werden alle Spalten der Tabelle 'kunde' bereitgestellt. In der Liste werden aber nur diejenigen Spalten ausgegeben, zu denen es im FORMAT-Abschnitt explizite Ausgabe-Anweisungen gibt. Und das sind nur die Spalten 'kunden_nr, firma, nachname, adresse1, plz, ort'.

Umgekehrt allerdings dürfen im FORMAT-Abschnitt keine Ausgabe-Anweisungen zu Spalten vorkommen, die nicht vorab durch den SELECT-Abschnitt bereitgestellt wurden.

8.9 FORMAT-Abschnitt

Im FORMAT-Abschnitt treffen Sie die folgenden Festlegungen:

- was soll in der Liste ausgegeben werden,
- wo soll die Ausgabe erfolgen bzw.
- wann (bei welchem Verarbeitungsstand) soll die Ausgabe erfolgen.

Weiter können Sie im FORMAT-Abschnitt auch Berechnungen durchführen lassen.

8.9.1 Standard-Ausgabe

Alternativ zu genauen 'was-wo-wann' Anweisungen können Sie mit den Schlüsselwörtern EVERY ROW eine Standard-Ausgabe veranlassen. In diesem Fall darf der FORMAT-Abschnitt keine weiteren Anweisungen mehr enthalten.

Beispiel

```
format
   every row
end
```

Bei dieser Anweisung werden alle Sätze und Spalten ausgegeben, die durch die SELECT-Anweisung im SELECT-Abschnitt bereitgestellt wurden. Die Ausgabe erfolgt nach fest eingestellten Standard-Werten.
Der FORMAT-Abschnitt eines automatisch erzeugten Standard-Listen-Programmes enthält eine EVERY ROW Anweisung.

8.9.2 Aufbau des FORMAT-Abschnittes

Soll die Ausgabe nicht mit einer Standard-Ausgabe erfolgen (EVERY ROW), dann muß sich der FORMAT-Abschnitt aus den folgenden zwei Elementen zusammensetzen:
- Kontrollblock-Anweisungen
- und innerhalb einer Kontrollblock-Anweisung Ausgabe-Anweisungen

Beispiel

Das ist der FORMAT-Abschnitt der LISTE01:

```
format
                KONTROLLBLOCK-ANWEISUNG
page header
                AUSGABE-ANWEISUNGEN
print "K u n d e n d a t e n:"
skip 2 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
end
```

Eine Kontrollblock-Anweisung legt fest, 'wo' bzw. 'wann' die darunter definierten Ausgaben erfolgen sollen.
In diesem Beispiel gibt es die Kontrollblock-Anweisung PAGE HEADER. Sie legt fest, wo die folgenden Ausgabe-Anweisungen ausgeführt werden sollen.Hier 'am Anfang jeder neuen Seite' (und in den darauffolgenden Zeilen).
Weitergehende Erläuterungen zu Kontrollblock-Anweisungen erhalten Sie im Abschnitt 8.9.5.

Mit Ausgabe-Anweisungen werden einzelne Zeilen geschrieben bzw. Zeilenvorschübe festgelegt.
In diesem Beispiel gibt es eine ganze Reihe von Ausgabe-Anweisungen, in denen festgelegt ist, welche Texte (z.B. "K u n d e n d a t e n") und welche Spalten (z.B. 'kunden_nr') ausgegeben werden sollen.
Mit den Ausgabe-Anweisungen beschäftigt sich der folgende Abschnitt; hier zunächst noch einige Feststellungen über das Zusammenspiel zwischen Kontrollblock-Anweisungen und Ausgabe-Anweisungen.

Ein FORMAT-Abschnitt besteht aus mindestens einer Kontrollblock-Anweisung und beliebig vielen darauffolgenden Ausgabe-Anweisungen. Er kann aber auch aus mehreren Kontrollblock-Anweisungen mit jeweils darauffolgenden Ausgabe-Anweisungen bestehen, wie die folgende schematische Darstellung zeigt:

```
FORMAT
    kontrollblock-anweisung
       ausgabe-anweisung
       ausgabe-anweisung
        ...
    kontrollblock-anweisung
       ausgabe-anweisung
       ausgabe-anweisung
       ...
    kontrollblock-anweisung
       ausgabe-anweisung
       ausgabe-anweisung
    ....
       ...
       ...
       ...
END
```

8.9.3 Ausgabe-Anweisungen

Die folgenden Kapitel-Abschnitte erläutern einige Ausgabe-Anweisungen. Die anschließenden Kapitel-Abschnitte erläutern die Kontrollblock-Anweisungen.

Eine Zeile schreiben

Die PRINT-Anweisung schreibt eine Zeile einer Liste und bewirkt einen abschließenden Zeilenvorschub zur folgenden Zeile.
Die einzelnen Anweisungsteile in einer PRINT-Anweisung sind durch Kommas voneinander zu trennen. Wenn der abschließende Zeilenvorschub unterdrückt werden soll, dann muß am Ende der PRINT-Anweisung ein Semikolon stehen.

Die PRINT-Anweisung kann lauter einzelne Anweisungsteile enthalten, die genau festlegen

- was ausgegeben werden soll,
- wo in der Zeile die Ausgabe erfolgen soll.

Was kann ausgeben werden?

Ausgeben lassen können Sie u.a.
- beliebige Texte oder
- Spalten, die durch der vorangegangenen Datenbank-Abfrage bereitgestellt wurden (SELECT-Abschnitt).

Auszugebende Texte setzen Sie in Anführungszeichen. Spaltenamen dürfen Sie nicht in Anführungszeichen setzen, da sie sonst als auszugebender Text interpretiert würden.

Beispiel

Das ist der FORMAT-Abschnitt der LISTE01:

```
format

page header
print "K u n d e n d a t e n:"
skip 2 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
end
```

In ihm ist u.a. die folgende Anweisung enthalten:

```
print "K u n d e n d a t e n:"
```

Die Anweisung bewirkt, daß der Text "K u n d e n d a t e n:" ausgegeben wird.

Weiter können Sie im Rahmen einer PRINT-Anweisung Spalten der vorangegangenen Datenbank-Abfrage ausgeben lassen.

Beispiel

In dem oben gezeigten FORMAT-Abschnitt gibt es diese Anweisung:

```
print "Nachname:",     column 20, nachname
```

Diese Anweisung gibt zunächst den Text 'Nachname' und dann auf der Spalte 20 einen Wert der Tabellenspalte 'nachname' aus.

Wo soll die Ausgabe erfolgen?

In der PRINT-Anweisung machen Sie weiter Angaben darüber, wo die Ausgabe in der Zeile erfolgen soll.
Dabei kann die Ausgabe nur innerhalb des Zeilenraumes erfolgen, der durch die Listenränder vorgegebenen ist (siehe OUTPUT-Abschnitt).

Mit den folgenden Anweisungen können Sie Angaben zur erwünschten Ausgabeposition machen:

COLUMN ganzzahl
mit COLUMN können Sie explizite Spaltenpositionierungen vornehmen; gerechnet wird ab dem linken Listenrand.

Beispiel

```
COLUMN 20
```

Diese Anweisung positioniert auf die 20. Spalte nach dem linken Listenrand.

SPACE ganzzahl
mit SPACE können Sie durch das Schreiben von Leerzeichen relative Positionierungen vornehmen; gerechnet wird ab der augenblicklichen Position.

Beispiel

```
15 SPACE
```

Diese Anweisung schreibt 15 Leerzeichen.

Beispiel

```
page header
print "ANFANG", column 20, "MITTE", 15 spaces, "ENDE"
print "123456789 123456789 123456789 123456789 123456789"
end
```

Diese Anweisungen führen, schematisch dargestellt, zu folgendem Ergebnis:

```
                    |<------ Spalte 20
                    |
                    |    <--15 Zeich.-->
ANFANG              MITTE               ENDE
123456789 123456789 123456789 123456789 123456789
```

Zeilenvorschübe

Mit der SKIP-Anweisung können Sie Zeilenvorschübe (Leerzeilen) erzeugen. Mit der Anweisung SKIP TO TOP OF PAGE können Sie einen Vorschub zum Anfang der nächsten Seite erzeugen.

Beispiel

Der FORMAT-Abschnitt der LISTE01 enthält u.a. diese Anweisungen:

```
format

page header
print "K u n d e n d a t e n"
skip 2 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
...
```

Die Anweisung 'skip 2 line' wirkt sich wie folgt aus:
nachdem der Text K u n d e n d a t e n ausgegeben wurde, erfolgen zwei Zeilenvorschübe (zwei Leerzeilen) und erst danach wird die folgende PRINT-Anweisung ausgeführt.

8.9.4 Formatier-Anweisungen

Dieser Abschnitt stellt zwei Formatier-Anweisungen vor:

- CLIPPED, zu verwenden bei CHAR-Spalten und
- USING, zu verwenden bei numerischen Spalten (und DATE-Spalten).

CHAR-Spalten formatieren

Bei Werten von CHAR-Spalten gibt ein Listenprogramm grundsätzlich ebensoviele Zeichen aus, wie die Spalte gemäß Spaltendefinition maximal aufnehmen kann.
Wenn z.B. ein CHAR-Wert aus 10 Zeichen besteht, die Spalte aber 20 Zeichen aufnehmen kann, dann wird der Wert ausgegeben und dann 10 Leerzeichen.
Mit der CLIPPED-Anweisung können Sie die Ausgabe der Leerzeichen unterdrücken.
Der Spaltename und die CLIPPED-Anweisung dürfen nicht durch ein Komma getrennt sein.

Beispiel

Die folgende PRINT-Anweisung schreibt eine Zeile einer Liste. Dabei wird ein Wert 'firma' der Tabelle 'kunde' ausgegeben. Der entsprechende SELECT-Abschnitt ist hier nicht abgebildet.

```
format

page trailer
print "Die Waren werden dem Hause ", firma, " umgehend zugestellt."
end
```

Das Ergebnis:

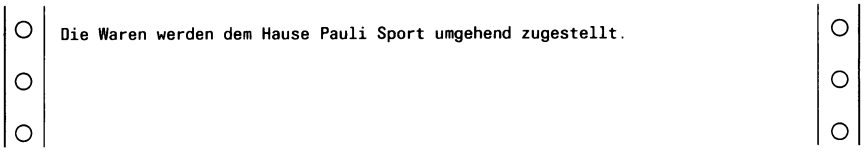
```
Die Waren werden dem Hause Pauli Sport          umgehend zugestellt.
```

Da die Spalte 'firma' maximal 20 Zeichen aufnehmen kann, werden in der Liste an entsprechender Stelle auch 20 Zeichen ausgegeben.

Die CLIPPED-Anweisung unterdrückt die Leerzeichen:

```
format

page trailer
print "Die Waren werden dem Hause ", firma clipped, " umgehend zugestellt."
end
```

Das Ergebnis jetzt:



Automatischer Umbruch

Mit der Anweisung WORDWRAP können Sie längere (VAR)CHAR- oder TEXT-Spalten ausgeben lassen, ohne den Zeilenumbruch programmieren zu müssen.

Beispiel

```
print column 10, beschreibung wordwrap right margin 70
```

Die Folge hiervon ist, daß der Text der Spalte 'beschreibung' mehrzeilig ausgegeben wird. Die Ausgabe pro Zeile beginnt auf Spalte 10 und geht maximal bis Zeile 70. Die Angabe von 'right margin' ist optional. Der hier gesetzte 'right margin' ist nur temporär für die aktuelle Ausgabe gültig. Danach gelten wieder die Standardwerte bzw. der bei OUTPUT gesetzte 'right margin'.

Numerische Spalten formatieren

Bei numerischen Spalten können Sie mittels einer USING-Anweisung ein erwünschtes Ausgabeformat festlegen.
Die Formatieranweisung erfolgt dabei über bestimmte Füllzeichen (bzw. Formatierzeichen), die Sie einzeln oder auch in Kombination miteinander verwenden können.
Die Füllzeichen wirken als Platzhalter; mit ihnen reservieren Sie die erwünschte Anzahl von Zeichen, die für den numerischen Wert zur Verfügung stehen sollen. Wenn ein Wert größer ist, als mit den Füllzeichen reserviert, werden anstelle des Wertes diese Zeichen '*' ausgegeben.
Der Spaltename und die USING-Anweisung dürfen nicht durch ein Komma getrennt sein. Die Füllzeichen sind in Anführungszeichen zu setzen.

Die folgende Übersicht zeigt die möglichen Füllzeichen. Über die Forma-

tierzeichen informiert Sie das Handbuch [2] INFORMIX-SQL Nachschlagen.

- # gibt führende Leerzeichen aus, wenn eine Zahl kleiner ist, als mit den Füllzeichen '#' reserviert.
- & gibt führende '0' aus, wenn eine Zahl kleiner ist, als mit den Füllzeichen '&' reserviert.
- * gibt führende '*' aus, wenn eine Zahl kleiner ist, als mit den Füllzeichen '*' reserviert.
- < gibt einen Wert linksbündig aus.

Beispiel

Die LISTE01 enthält diese USING-Anweisung:

```
print "Kundennummer:", column 20, kundenn_nr using "<<<"
```

Die Wirkung hiervon ist, daß für einen Wert der Spalte 'kunden_nr' drei Zeichen reserviert sind und daß er linksbündig ausgegeben wird.

Weitere Beispiele - der Wert sei '1015':

FÜLLZEICHEN	AUSGABE
USING "######"	1015
USING "&&&&&&"	001015
USING "******"	**1015
USING "<<<<<<"	1015
USING "&&&"	***

8.9.5 Kontrollblock-Anweisungen

Bevor Sie mit PRINT-Anweisungen festlegen können, was ausgegeben werden soll, müssen Sie mit Kontrollblock-Anweisungen festlegen, wo die Ausgaben erfolgen sollen bzw. wann, bei welchem Verarbeitungsstand, die Ausgaben erfolgen sollen.

Die folgenden Kontrollblock-Anweisungen gibt es:

	ANWEISUNG	DIE AUSGABE ERFOLGT:
WO	FIRST PAGE HEADER	am Anfang der ersten Seite.
	PAGE HEADER	bei jedem Seitenanfang.
	PAGE TRAILER	am Ende jeder Seite.
WANN	ON EVERY ROW	bei jedem neuen Satz.
	ON LAST ROW	beim letzten Satz.
	BEFORE GROUP OF	bevor sich das Gruppen-kennzeichen ändert.
	AFTER GROUP OF	nachdem sich das Gruppen-kennzeichen geändert hat.

'Wo'-Kontrollblock-Anweisungen

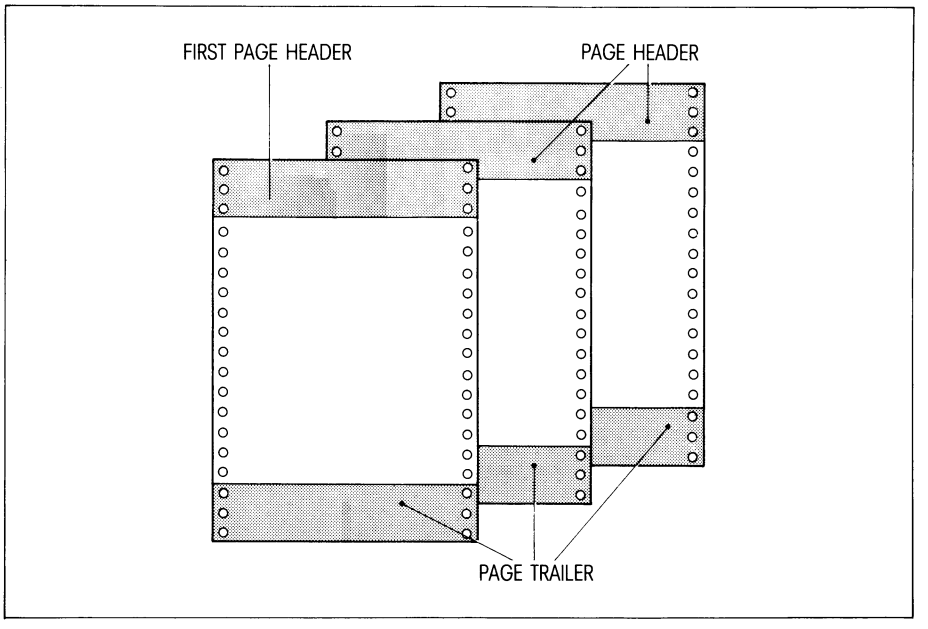


Bild 8-3 Seitenköpfe und Seitenfüße

Mit den Anweisungen FIRST PAGE HEADER und PAGE HEADER können Sie Seitenköpfe festlegen; mit der Anweisung PAGE TRAILER können Sie einen Seitenfuß festlegen.

Dabei gelten die folgenden Regeln:

- In einem FIRST PAGE HEADER Kontrollblock oder einem PAGE HEADER Kontrollblock darf eine beliebige Anzahl von PRINT-Anweisungen enthalten sein, so daß sich ein Seitenkopf maximal auch von der ersten bis zur letzten Zeile erstrecken kann.
- Der insgesamt zur Verfügung stehende Zeilenraum darf nicht überschritten werden.
  Standardmäßig stehen 66 Zeilen zur Verfügung. Im OUTPUT-Abschnitt können Sie vom Standard abweichende Festlegungen treffen.
- Wenn mit einem PAGE TRAILER Kontrollblock ein Seitenfuß festgelegt wurde, dann wird die letzte hier definierte PRINT-Anweisung auf der letzten Zeile ausgegeben (standardmäßig auf Zeile 66).
  Jede darüberliegende PRINT-Anweisung wird auch auf einer darüberliegenden Zeile ausgegeben.
  Wenn es keine (FIRST) PAGE HEADER Anweisung gibt, dann kann sich ein Seitenfuß auch maximal von der letzten bis zur ersten Zeile erstrecken.
- Die PRINT-Anweisungen in einem (FIRST) PAGE HEADER Kontrollblock und in einem PAGE TRAILER Kontrollblock dürfen zusammengenommen den insgesamt zur Verfügung stehenden Zeilenraum nicht überschreiten.

FIRST PAGE HEADER

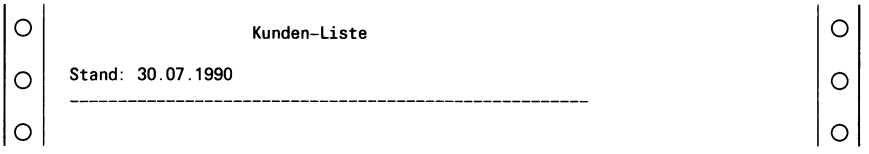
In einem FIRST PAGE HEADER Kontrollblock können Sie Anweisungen festlegen, die nur im Kopf der ersten Seite ausgegeben werden. Die Anweisung ist nur dann sinnvoll, wenn mehrere Seiten ausgegeben werden.

Beispiel

Die Anweisung im folgenden Beispiel legt einen Seitenkopf fest, der nur auf der ersten Seite ausgegeben wird. Die Anweisung 'today' in diesem Beispiel gibt das aktuelle Datum aus.

```
first page header
print column 20, "Kunden-Liste"
skip 2 line
print "Stand: ", today
print "_____________________________________________________"
```

Sie erhalten diesen Ausgabe:



PAGE HEADER

In einem PAGE HEADER Kontrollblock können Sie nach gleichem Muster einen Seitenkopf festlegen, der am Anfang jeder neuen Seite ausgegeben wird.

PAGE TRAILER

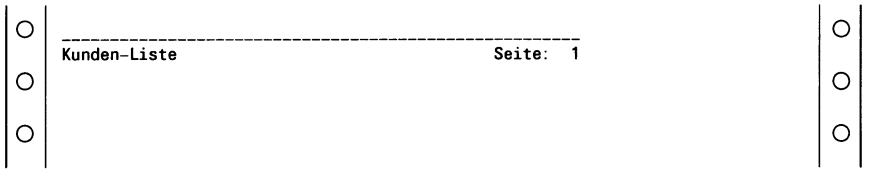
In einem PAGE TRAILER Kontrollblock können Sie einen Seitenfuß festlegen, der am Ende jeder Seite ausgegeben wird.

Beispiel

Das folgende Beispiel legt einen Seitenfuß fest, der an jedem Seitenende ausgegeben wird. Die Anweisung 'pageno' gibt die aktuelle Seitenzahl aus.

```
page trailer
print "_____________________________________________________"
print "Kunden-Liste", column 45, "Seite: ", pageno using "##"
```

Sie erhalten diese Ausgabe:



'Wann'-Kontrollblock-Anweisungen

Die folgend beschriebenen Kontrollblock-Anweisungen ermöglichen Ausgaben, die bei einem bestimmten Verarbeitungsstand erfolgen.

ON EVERY ROW

Die im ON EVERY ROW Kontrollblock festgelegten Ausgaben erfolgen jedesmal dann, wenn ein neuer Satz ausgegeben wird (nicht zu verwechseln mit der EVERY ROW Anweisung, die eine Standard-Ausgabe bewirkt). Dabei ist das Zusammenspiel zwischen dem SELECT-Abschnitt (READ-Abschnitt) und dem FORMAT-Abschnitt zu bedenken. Der SELECT-Abschnitt bewirkt lediglich die Datenbereitstellung auf Abruf. Die solchermaßen bereitgestellten Daten können Sie im Rahmen eines ON EVERY ROW Kontrollblockes nacheinander abrufen und ausgeben lassen.
Umgekehrt können Sie allerdings keine Daten ausgeben lassen, die nicht vorab durch die SELECT-Anweisung bereitgestellt wurden.

Beispiel

Das hier ist das Listen-Programm der LISTE02. Es handelt sich um eine Variante der LISTE01; die LISTE02 gibt alle Sätze der Tabelle 'kunde' aus; erreicht wird das mit einem ON EVERY ROW Kontrollblock. Zusätzlich gibt es in der LISTE02 die Kontrollblöcke PAGE HEADER und PAGE TRAILER, in denen Seitenköpfe und Seitenfüße festgelegt sind.
Die Seitenlänge (im OUTPUT-Abschnitt) ist mit 72 auf die Standard-Zeilenzahl des Druckprogrammes 'lpr' eingestellt. Die 'need 8 line' Anweisung im ON EVERY ROW Kontrollblock verhindert die Trennung der Ausgabe eines Satzes durch einen Seitenwechsel.

```
database versand
end

output report to printer
       page length 72
end

select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
kunde
end

format

page header
print column 30, "Kunden-Liste"
skip 1 line
print "Stand: ", today
print "_____________________________________________________________"
skip 2 line

page trailer
print "_____________________________________________________________"
print "Kunden-Liste", column 54, "Seite: ", pageno using "###"

on every row
need 8 lines
print "K u n d e n d a t e n:"
skip 1 line
print "Kundennummer:", column 20, kunden_nr using "<<<"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
skip 2 line
end
```

Das Listen-Programm liefert das folgende Ergebnis (Auszug der ersten Seite):

```
                                Kunden-Liste

Stand: 19.08.1990
------------------------------------------------------------------


K u n d e n d a t e n:

Kundennummer:       101
Firma:              Pauli Sport
Nachname:           Pauli
Adresse:            Forstweg 47
Postleitzahl:       8900
Ort:                Augsburg


K u n d e n d a t e n:

Kundennummer:       102
Firma:              Sport-Aktiv
Nachname:           Sadler
Adresse:            Blumenstr. 12
Postleitzahl:       8090
Ort:                Wasserburg


...
...
...
------------------------------------------------------------------
Kunden-Liste                                        Seite:  1
```

ON LAST ROW

Im ON LAST ROW Kontrollblock können Sie nach gleichem Muster Ausgaben festlegen, die nur dann erfolgen sollen, nachdem der letzte Satz ausgegeben wurde.
Diese Kontrollblock-Anweisung ermöglicht beispielsweise die abschließende Ausgabe einer Gesamtsumme aller vorausgehend ausgegebenen numerischen Werte einer bestimmten Spalte. Siehe hierzu den Abschnitt 8.10 'Berechnungen in einem Listen-Programm'.

BEFORE GROUP OF und AFTER GROUP OF

Diese beiden Kontrollblock-Anweisungen ermöglichen vorausgehende (BEFORE) oder abschließende (AFTER) Ausgaben für Gruppen.
Gruppen sind Sätze mit einem gemeinsamen Gruppenkennzeichen, einem gemeinsamen Wert in einer bestimmten Spalte.
Zum Beispiel gibt es in der Tabelle 'kunde' mehrere Sätze mit gleichen Werten in der Spalte 'ort'. Diese Sätze lassen sich über das Gruppenkennzeichen 'ort' gruppieren.

In einem BEFORE GROUP OF Kontrollblock können Sie Ausgaben festlegen, die erfolgen sollen, bevor die Sätze einer Gruppe ausgegeben werden.
In einem AFTER GROUP OF Kontrollblock können Sie Ausgaben festlegen, die erfolgen sollen, nachdem die Sätze einer Gruppe ausgegeben wurden.

Achtung

Beide Kontrollblöcke erfordern, daß der SELECT-Abschnitt eine ORDER BY Klausel enthält, in der nach dem angegebenen Gruppenkennzeichen sortiert wird.

Beispiel

Das ist das Listen-Programm der LISTE03; das Listen-Programm gibt die Spalten 'nachname', 'firma' und 'ort' der Tabelle 'kunde' aus. Im BEFORE GROUP OF Kontrollblock sind Ausgaben festgelegt, die erfolgen, bevor eine Gruppe ausgegeben wird. Im AFTER GROUP OF Kontrollblock sind Ausgaben festgelegt, die erfolgen, nachdem eine Gruppe ausgegeben wurde.

```
database versand
end

output report to printer
end

select
   nachname, firma, ort
from
   kunde
order by
   ort
end

format

page header
print column 18, "Kunden-Liste"
print column 18, "============"
skip 2 lines

before group of ort
print "STADT: ", ort
skip 1 line

on every row
print nachname, column 18, firma, column 40, ort
after group of ort
skip 2 lines
end
```

Das Listen-Programm liefert das folgende Ergebnis:

```
                Kunden-Liste
                ============


STADT: Augsburg

Pauli           Pauli Sport          Augsburg
Millet          Sportausstattung     Augsburg
Keyser          Sport Keyser         Augsburg


STADT: Ingolstadt

Viktor          Der Laden            Ingolstadt
Larsinger       Larsinger & Partner  Ingolstadt


STADT: Landshut

Bergen          Sporthaus            Landshut
Grantella       Sportladen           Landshut


STADT: Muenchen

Hochfeld        Spielball            Muenchen
Korting         Martin's Shop        Muenchen
Jaeger          Sportwaren           Muenchen
Albert          Der Sport-Albert     Muenchen
Sipell          Sportecke            Muenchen


STADT: Passau

Bachmann        Sportausstatter      Passau


STADT: Rosenheim

Korres           Philipp's Sportwaren Rosenheim
Remark           - Sportwaren -       Rosenheim


STADT: Ulm

Watt            Sport Watt           Ulm
Partellman      Olympia Sport        Ulm


STADT: Wasserburg

Sadler          Sport-Aktiv          Wasserburg
```

8.10 Berechnungen in einem Listen-Programm

In einem Listen-Programm können Sie folgende Berechnungsarten durchführen lassen:

- Grundrechenarten,
- Berechnungen mit Mengenfunktionen.

8.10.1 Grundrechenarten

Für das Rechnen in den Grundrechenarten stehen Ihnen die folgenden Rechen-Operatoren zur Verfügung:

RECHEN-OPERATOR	BEDEUTUNG
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
**	Exponentialfunktion

Es gilt die übliche Rechenregel 'Punkt vor Strich'. Bei gewollten Abweichungen von dieser Regel setzen Sie runde Klammern.

Beispiel

Die folgende Anweisung multipliziert den Wert von 'preis' mit 1.15 und gibt den um 15% erhöhten Neupreis in der Liste aus.

```
print preis * 1.15
```

Das folgende Beispiel ist das Listen-Programm der LISTE04. Das Listen-Programm kalkuliert, wie sich eine Erhöhung um einen bestimmten Prozentsatz auswirkt.
Dazu wird beim Start des Listen-Programmes zunächst eine Eingabe für den erwünschten Prozentsatz verlangt, die die Variable 'prozent' belegt. Im PAGE HEADER Kontrollblock gibt es eine 'let' Anweisung, die der Variablen aus dem bisherigen Wert einen neu errechneten Wert zuweist, der im weiteren Ablauf als Prozentsatz verwertet werden kann.

```
database versand
end

define variable prozent decimal(3,2)
end

input prompt for prozent
      using "Die Artikel sollen erhoeht werden um Prozent: "
end

output report to printer
end

select
   herstellercode, bezeichnung, preis
from
   artikel
end

format

page header
print "PROZENTSATZ: ", prozent using "##.##"
let prozent = prozent / 100 + 1
skip 1 line
print      "HERSTELLER",
column 12, "BEZEICHNUNG",
column 32, "PREIS",
column 43, "PLUS",
column 49, "NEUPREIS"
skip 1 line

on every row
print herstellercode,
column 12, bezeichnung clipped,
column 30, preis using "####.##",
column 40, preis * prozent - preis using "####.##",
column 50, preis * prozent using "####.##"
end
```

Wenn beim Ablauf der Liste als erwünschter Prozentsatz '6' eingegeben wird, dann ist das das Ergebnis:

```
PROZENTSATZ:  6,00

HERSTELLER BEZEICHNUNG          PREIS      PLUS  NEUPREIS

HRO        Ski-Handschuhe     250,00     15,00   265,00
HSK        Ski-Handschuhe     800,00     48,00   848,00
SMT        Ski-Handschuhe     450,00     27,00   477,00
HRO        Ski-Brille         126,00      7,56   133,56
HSK        Ski-Stock          240,00     14,40   254,40
HSK        Fussball           960,00     57,60  1017,60
HRO        Fussball           480,00     28,80   508,80
NRG        Tennisschlaeger     28,00      1,68    29,68
SMT        Tennisschlaeger     25,00      1,50    26,50
ANZ        Tennisschlaeger     19,80      1,18    20,98
SMT        Tennisball          36,00      2,16    38,16
ANZ        Tennisball          48,00      2,88    50,88
HRO        Basketball         600,00     36,00   636,00
ANZ        Volleyball         840,00     50,40   890,40
ANZ        Volleyb. profi      20,00      1,20    21,20
```

8.10.2 Mengenfunktionen

Mit Mengenfunktionen können Sie Ergebnisse für alle Sätze oder für Gruppen ermitteln.
Sie können z.B. die Werte einer Spalte zu einer Gesamtsumme aufaddieren oder Sie können Zwischenergebnisse ermitteln (mit GROUP-Mengenfunktionen).

Mengenfunktionen, die Ergebnisse für alle Sätze ermitteln:

TOTAL OF
: addiert die Werte einer Spalte.

MIN OF
: ermittelt den kleinsten Wert einer Spalte.

MAX OF
: ermittelt den größten Wert einer Spalte.

AVERAGE OF
: ermittelt den Durchschnittswert einer Spalte.

COUNT
: ermittelt die Anzahl der Sätze.

Mengenfunktionen, die Ergebnisse für Gruppen ermitteln:

GROUP TOTAL OF
addiert die Werte einer Spalte einer Gruppe.

GROUP MIN OF
ermittelt den kleinsten Wert einer Spalte einer Gruppe.

GROUP MAX OF
ermittelt den größten Wert einer Spalte einer Gruppe.

GROUP AVERAGE OF
ermittelt den Durchschnittswert einer Spalte einer Gruppe.

GROUP COUNT
ermittelt die Anzahl der Sätze einer Gruppe.

GROUP PERCENT
ermittelt den prozentualen Anteil der Sätze einer Gruppe gegenüber der Gesamtanzahl der ausgegebenen Sätze.

Mengenfunktionen, die Ergebnisse für Gruppen ermitteln, können Sie nur in einem AFTER GROUP OF Kontrollblock verwenden.

Beispiel

Das ist das Listen-Programm der LISTE05. Im AFTER GROUP Kontrollblock gibt es eine GROUP COUNT Anweisung, die die ausgegebenen Sätze einer Gruppe zählt und den ermittelten Wert ausgibt. Im ON LAST ROW Kontrollblock gibt es eine COUNT Anweisung, die die Gesamtanzahl aller ausgegebenen Sätze zählt.

```
database versand end

output report to printer
        page length 72
end

select
   kunde.kunden_nr knr, firma,
   auftrag.auftrags_nr anr
from
   kunde, auftrag
where
   kunde.kunden_nr = auftrag.kunden_nr
order by
   knr
end

format

first page header
print             "KUNDENNUMMER",
       column 20, "FIRMA",
       column 37, "AUFTRAGSNUMMER"
skip 2 lines

on every row
print  column 10,  knr using "###",
       column 20,  firma clipped,
       column 47,  anr using "###"

after group of knr
print "_____________________________________________________"
print column 30, "ANZAHL AUFTRAEGE ",
      column 49, group count using "##"
print "_____________________________________________________"

on last row
skip 2 lines
print "====================================================="
print column 23, "GESAMT-ANZAHL AUFTRAEGE ",
      column 49, count using "##"
print "====================================================="
end
```

Sie erhalten diese Ausgabe:

KUNDENNUMMER	FIRMA	AUFTRAGSNUMMER
101	Pauli Sport	1002
	ANZAHL AUFTRAEGE	1
104	Spielball	1001
104	Spielball	1003
104	Spielball	1011
104	Spielball	1013
	ANZAHL AUFTRAEGE	4
106	Sport Watt	1004
106	Sport Watt	1014
	ANZAHL AUFTRAEGE	2
110	Sportwaren	1008
110	Sportwaren	1015
	ANZAHL AUFTRAEGE	2
111	Sport Keyser	1009
	ANZAHL AUFTRAEGE	1
112	Larsinger & Partner	1006
	ANZAHL AUFTRAEGE	1
115	Sportladen	1010
	ANZAHL AUFTRAEGE	1
116	Olympia Sport	1005
	ANZAHL AUFTRAEGE	1
117	Sportecke	1007
117	Sportecke	1012
	ANZAHL AUFTRAEGE	2
	GESAMT-ANZAHL AUFTRAEGE	15

Zusammenfassung

- Ein Listen-Programm ermöglicht Ihnen, eine Datenbank abzufragen und die bereitgestellten Daten formatiert auszugeben.
- Um ein Listen-Programm zu erzeugen, benötigen Sie zunächst ein Listen-Quellprogramm. Das Listen-Quellprogramm schreiben (bzw. modifizieren) Sie mit einem Editor.
- Um ein ablauffähiges Listen-Programm zu erhalten, müssen Sie das Listen-Quellprogramm compilieren.
- Ein Listen-Quellprogramm besteht mindestens aus den drei Pflichtabschnitten DATABASE , SELECT und FORMAT. Es kann aber optional zusätzlich diese Abschnitte enthalten: DEFINE, INPUT, OUTPUT
- Im DATABASE-Abschnitt benennen Sie die Datenbank, für die die Liste erstellt werden soll.
- Im DEFINE-Abschnitt können Sie bei Bedarf Variablen definieren bzw. mit PARAM Übergabeparamter festlegen bzw. mit ASCII die Datentypen der mit READ einzulesenden Daten festlegen.
- Im INPUT-Abschnitt können Sie eine Eingabe definieren, die beim Start des Listen-Programmes erfolgen soll. Die Eingabe belegt eine Variable, die im DEFINE-Abschnitt definiert sein muß.
- Im OUTPUT-Abschnitt können Sie festlegen, wo die Ausgabe der fertig erstellten Liste erfolgen soll. Weiter können Sie vom Standard abweichende Listenränder festlegen.
- Im SELECT- bzw. READ-Abschnitt sorgen Sie für die erwünschte Datenbereitstellung.
- Ein FORMAT-Abschnitt besteht aus zwei Elementen: Kontrollblock-Anweisungen und Ausgabe-Anweisungen.
- Es gibt 'wo' Kontrollblock-Anweisungen, mit denen Sie Seitenköpfe und Seitenfüße festlegen können und es gibt 'wann' Kontrollblock-Anweisungen, mit denen Sie festlegen können, nach welchem Verarbeitungsstand bestimmte Ausgaben erfolgen sollen.

- Die PRINT-Anweisung innerhalb eines Kontrollblocks schreibt eine Zeile einer Liste.
- Die SKIP-Anweisung erzeugt Zeilenvorschübe.
- Im FORMAT-Abschnitt können sie Berechnungen durchführen lassen. Zur Verfügung stehen die Grundrechenarten und das Rechnen mit Mengenfunktionen, die Ergebnisse für alle Sätze oder für Gruppen von Sätzen liefern.

9 Datenstruktur

Dieses Kapitel erläutert Ihnen SQL-Anweisungen, mit denen Sie die Datenstruktur (Datendefinition) herstellen und verändern können. Sie erfahren anhand von Beispielen, wie man mit SQL

- eine Datenbank erzeugt und löscht,
- eine Tabelle erzeugt, modifiziert, löscht und umbenennt,
- einen Index erzeugt und löscht.

Verwiesen sei in diesem Zusammenhang auf die SQL-Anweisung 'c_database', mit der die Datenbank 'versand' nebst Tabellen und Indizes erzeugt wurde; Sie können diese SQL-Anweisung im SQL-Dialog-Menü über die Funktion 'Datei' laden.
Einige Beispiele in diesem Kapitel sind dieser SQL-Anweisung entnommen.

Weiter erfahren Sie in diesem Kapitel, welche Maßnahmen Sie ergreifen müssen, wenn Sie mit simulierten Umlauten arbeiten wollen.

9.1 Eine Datenbank erzeugen und löschen

Eine Datenbank erzeugen oder löschen können Sie auf zwei Wegen:

- menügeführt über die Funktion 'Datenbank' im Hauptmenü oder
- mit einer SQL-Anweisung.

Dieser Abschnitt erläutert Ihnen, wie Sie eine Datenbank mit einer SQL-Anweisung erzeugen oder löschen können.

9.1.1 Datenbank erzeugen

Mit der folgenden Anweisung wird eine Datenbank mit dem Namen 'test' erzeugt:

```
CREATE DATABASE test
```

Wenn es eine Datenbank mit dem angegebenen Namen bereits gibt, dann erfolgt eine Fehlermeldung.

Wenn Sie bei INFORMIX-SE eine Datenbank erzeugen, dann wird in Ihrem aktuellen Dateiverzeichnis ein Dateiverzeichnis mit dem Namen 'datenbankname.dbs' eingerichtet. In diesem Beispiel also ein Dateiverzeichnis mit dem Namen 'test.dbs'
In diesem Dateiverzeichnis 'datenbankname.dbs' befinden sich eine ganze Reihe von Dateien, die INFORMIX zur internen Verwaltung benötigt. Außerdem befinden sich hier auch die Dateien mit Ihren Tabellen.

Vorsicht

Versuchen Sie keinesfalls, das Dateiverzeichnis oder eine der darin enthaltenen Dateien zu löschen, zu verändern oder mit sonstigen Shell-Kommandos zu behandeln, da Sie dabei Ihre Datenbank zerstören könnten!

9.1.2 Datenbank löschen

Die folgende Anweisung löscht die Datenbank mit dem Namen 'test':

```
DROP DATABASE test
```

Die aktuelle Datenbank läßt sich nicht löschen. Wenn Sie die aktuelle Datenbank löschen wollen, dann müssen Sie vorab die folgende Anweisung geben:

```
CLOSE DATABASE
```

Diese Anweisung schließt die aktuelle Datenbank.

Vorsicht

Die DROP DATABASE Anweisung löscht die Datenbank mit sämtlichen darin enthaltenen Tabellen, Inidizes und Werten!

Bei INFORMIX-SE wird dabei das Dateiverzeichnis 'datenbankname.dbs' gelöscht.

9.2 Eine Tabelle erzeugen, modifizieren, löschen oder umbenennen

Eine Tabelle erzeugen, modifizieren oder löschen können Sie auf zwei Wegen:

- menügeführt über die Funktion 'Tabelle' im Hauptmenü oder
- mit einer SQL-Anweisung.

Eine Tabelle umbenennen können Sie nur mit SQL. Dieser Abschnitt erläutert Ihnen die entsprechenden SQL-Anweisungen.

9.2.1 Tabelle erzeugen

Wenn Sie die hier geschilderten Beispiele nachvollziehen wollen, dann sollten Sie dazu, wie oben beschrieben, eine neue Datenbank (z.B. 'test') erzeugen.

Die folgende CREATE TABLE Anweisung erzeugt eine Tabelle mit dem Namen 'kunde' (Datenbank 'test') - das ist die gleiche Anweisung, mit der auch die Tabelle 'kunde' der Datenbank 'versand' erzeugt wurde:

```
CREATE TABLE kunde
(
        kunden_nr       SERIAL(101),
        vorname         CHAR(15),
        nachname        CHAR(15),
        firma           CHAR(20),
        adresse1        CHAR(20),
        adresse2        CHAR(20),
        ort             CHAR(15),
        bundesland      CHAR(2),
        plz             CHAR(5),
        telefon         CHAR(18)
)
```

Diese Anweisung erzeugt die folgenden Spalten:

- die Spalte 'kunden_nr' mit dem Datentyp SERIAL. Als Startnummer ist 101 festgelegt; der erste neu aufgenommene Satz erhält in der Spalte 'kunden_nr' den Wert 101,
- die Spalten 'vorname', 'nachname', ..., 'telefon'. Diesen Spalten ist der Datentyp CHAR zugeordnet. Bei jeder Spalte ist in Klammern der

Umfang angegeben. Die Spalte 'vorname' beispielsweise kann einen Wert von maximal 15 Zeichen aufnehmen.

9.2.2 Tabelle modifizieren

Mit der ALTER TABLE Anweisung können Sie bei einer Tabelle

- eine neue Spalte einfügen (ALTER TABLE ADD),
- eine Spalte löschen (ALTER TABLE DROP),
- den Datentyp einer Spalte ändern (ALTER TABLE MODIFY).

Diese Tätigkeiten können Sie auch menügeführt über die Funktion 'Tabelle' im Hauptmenü ausführen.

Mit der RENAME TABLE Anweisung können Sie eine Tabelle umbenennen und mit der RENAME COLUMN Anweisung können Sie einen Spaltennamen umbenennen.

Eine Spalte einfügen

Die folgende Anweisung fügt bei der Tabelle 'kunde' die Spalte 'manager' ein. Die Spalte wird an das Ende der Tabelle angefügt:

```
ALTER TABLE kunde
   ADD (manager CHAR(30))
```

Mit einer BEFORE-Klausel können Sie angeben, vor welcher Spalte die Spalte eingefügt werden soll.

Die folgende Anweisung fügt die Spalte 'ansprechpartner' vor der Spalte 'adresse1' ein:

```
ALTER TABLE kunde
   ADD (ansprechpartner CHAR(20) BEFORE adresse1)
```

Hinweis

Einer neu eingefügte Spalte wird standardmäßig (bei jedem Satz) ein NULL-Wert zugewiesen.

Eine Spalte löschen

Die folgende Anweisung löscht aus der Tabelle 'kunde' die Spalte 'manager':

```
ALTER TABLE kunde
   DROP (manager)
```

Vorsicht

Gelöscht wird dabei die Spalte mit sämtlichen enthaltenen Werten!

Beim Löschen einer Spalte korrigiert INFORMIX automatisch eventuell erteilte Indizes und die Zugriffsrechte, so daß der Rest der Tabelle intakt bleibt.

Den Datentyp einer Spalte ändern

Mit der ALTER TABLE MODIFY Anweisung können Sie den Datentyp einer Spalte ändern. Sie können z.B. bei CHAR-Spalten den Umfang vergrößern oder verkleinern oder Sie können einen numerischen Datentyp in einen anderen numerischen Datentyp umwandeln.

Die folgende Anweisung vergrößert den Umfang der CHAR-Spalte 'nachname' von 15 auf 20:

```
ALTER TABLE kunde
   MODIFY (nachname CHAR(20))
```

Vorsicht

Wenn die Werte einer Spalte größer sind als der neue Spaltentyp zuläßt, dann bricht INFORMIX ab, beläßt die Tabelle auf dem alten Stand und gibt eine Fehlermeldung aus.
Wenn z.B. eine INTEGER-Spalte die folgenden Werte enthält: 100 700 50000
Und wenn die Spalte in eine SMALLINT-Spalte umgewandelt werden soll, dann bricht INFORMIX beim Wert 50000 ab und gibt eine Fehlermeldung aus.

Wenn Sie einen numerischen Datentyp in einen anderen numerischen Datentyp umwandeln, dann kann es zu Rundungsfehlern kommen. Wenn Sie z.B. eine FLOAT-Spalte in eine DECIMAL(4,2) Spalte umwandeln, dann rundet INFORMIX die Werte auf, bevor Sie in den DECIMAL-Datentyp umgewandelt werden.

Einen CHAR-Datentyp können Sie nur dann in einen numerischen Datentyp umwandeln, wenn die Spalte ausschließlich numerische Zeichen (Ziffern, Rechenzeichen) enthält.
Beispielsweise läßt sich der folgende CHAR-Wert in einen numerischen Wert umwandeln: '8900'
Und dieser CHAR-Wert läßt sich nicht umwandeln: '8900 Muenchen'

Eine Tabelle umbenennen, eine Spalte umbenennen

Die folgende Anweisung benennt die Tabelle 'kunde' in 'kundenstamm' um:

```
RENAME TABLE kunde
TO kundenstamm
```

Die folgende Anweisung benennt die Spalte 'kunden_nr' der Tabelle 'kundenstamm' in 'kd_nummer' um:

```
RENAME COLUMN kundenstamm.kunden_nr
TO kd_nummer
```

9.3 Einen Index erzeugen und löschen

Der Abschnitt 3.3.5 hat Sie bereits über das Indizieren von Spalten informiert. Hier eine zusammenfassende Übersicht über die wichtigsten Merkmale:

- Das Indizieren von Spalten beschleunigt den Datenzugriff bei Datenbank-Abfragen.
- Wenn eine Spalte indiziert ist, dann erfolgt bei Datenbank-Abfragen automatisch eine sortierte Ausgabe nach den Werten der indizierten Spalte.
- Eine Spalte sollten Sie nur dann indizieren, wenn eine oder mehrere der folgenden Bedingungen erfüllt sind: Die Tabelle enthält mehr als 200 Sätze; die indizierte Spalte wird regelmäßig als Join-Spalte, für Such-Bedingungen oder für Sortierungen verwendet, oder über die Spalte läßt sich mit einem UNIQUE-Index die geforderte Eindeutigkeit der Sätze erzwingen.
- Nicht indizieren sollten Sie Spalten, die eine sehr große Anzahl von mehrfach vorkommenden Werten enthalten, da in diesem Fall der Datenzugriff u.U. nicht beschleunigt, sondern im Gegenteil verlangsamt wird. Weiter sollten Sie keine einzelne Spalte oder mehrere Spalten gemeinsam indizieren, wenn die Gesamtlänge 120 Bytes überschreitet.
- Sie können eine einzelne Spalte indizieren oder auch mehrere Spalten gemeinsam in einem zusammengesetzten Index.
- Ein Index kann so gestaltet sein, daß nur eindeutige, noch nicht enthaltene Werte aufgenommen werden können (UNIQUE) oder Werte aufgenommen werden können, die bereits enthalten sind (DUPLIKATE).
- Ein Index kann so gestaltet sein, daß die Sortierung in aufsteigender Reihe (ASC) oder in absteigender Reihe (DESC) erfolgt.

Die folgende Anweisung erzeugt einen eindeutigen Index für die Spalte 'kd_nummer'. Der Index heißt 'k_nr_ix':

```
CREATE UNIQUE INDEX k_nr_ix
ON kundenstamm (kd_nummer)
```

Index löschen

Die folgende Anweisung löscht den Index mit dem Namen 'k_nr_ix':

```
DROP INDEX k_nr_ix
```

9.4 Datenbank-Design

Zu den schwierigsten Aufgaben eines Datenbank-Verwalters gehört es, eine sinnvolle Datenstruktur herzustellen; sich zu überlegen, wie die zu speichernden Daten sinnvoll auf Spalten und Tabellen zu verteilen sind. Der vorliegende Abschnitt kann Sie nicht in die Regeln des Datenbank-Designs einführen, das würde den Rahmen dieses Manuals sprengen. Informieren Sie sich zu diesem Thema bitte in der einschlägigen Literatur. Der vorliegende Abschnitt möchte Ihnen lediglich einen möglichen Weg aufzeigen, wie Sie Korrekturen an der Datenstruktur vornehmen können, nachdem die Datenbank bereits aufgebaut wurde. Transaktionskonzepte (siehe Abschnitt 10.2 und Handbuch [1] SQL) berücksichtigt das unten geschilderte Vorgehen nicht.

Sie können z.B. wie folgt vorgehen:

- Um sicherzustellen, daß bei Ihren Aktionen die gespeicherten Daten nicht durch versehentliche Fehlbedienungen verloren gehen, können Sie zunächst die alten Tabellen mit einer UNLOAD-Anweisung (siehe das Handbuch [1] SQL) in ASCII-Dateien entladen. Damit liegen Ihre Daten doppelt vor: einmal in den bisherigen Tabellen und zum zweiten in ASCII-Dateien. Dieses Vorgehen ermöglicht, daß Sie notfalls die alte Datenstruktur wieder vollständig herstellen können, indem Sie die alten Tabellen wieder neu erzeugen und anschließend die Tabellen mit LOAD-Anweisungen mit den Daten laden.
- Dann erzeugen Sie (mit CREATE TABLE oder menügeführt) neue Tabellen und stellen damit die neue Datenstruktur her.
- Dann können Sie die neuen Tabellen mit INSERT INTO Anweisungen mit den bisherigen Daten laden. Dabei können die INSERT INTO Anweisungen SELECT Anweisungen (VALUES SELECT ...) enthalten, mit denen Sie die zu ladende Datenmenge genau auswählen können. Auf diese Weise können Sie z.B. eine Datenmenge, die bisher in einer einzigen Tabelle geführt wurde, auf nunmehr zwei Tabellen verteilen.
- Danach können Sie die alten Tabellen löschen. Mit RENAME-Anweisungen können Sie den neuen Tabellen die Namen der alten Tabellen geben.
- Abschließend können Sie Ihre Formate und Listen-Programme entsprechend anpassen und neu compilieren und die mit UNLOAD erzeugten ASCII-Dateien löschen.

9.5 Umlautbehandlung

Manche Datensichtstationen verarbeiten nur 7-Bit ASCII-Codes. (Tabelle mit den ASCII-Codes siehe Anhang). Das bringt es mit sich, daß die Daten keine Umlaute enthalten dürfen (also z.B. nicht 'München' sondern 'Muenchen').
Es gibt jedoch die Möglichkeit, Umlaute zu simulieren. Dazu sind Maßnahmen auf den folgenden Ebenen erforderlich:

Bildschirm

Wenn Sie über eine deutsche Tastatur bzw. über eine deutsch konfigurierte Tastatur verfügen, dann können Sie mit der Taste [CH CODE] umschalten zwischen der standardmäßigen Bildschirm-Abbildung und einer Abbildung, die anstelle der Klammern die deutschen Umlaute ausgibt. Hier eine Übersicht:

ZEICHEN	WIRD ABGEBILDET ALS
{	ä
[	Ä
|	ö
\	Ö
}	ü
]	Ü
~	ß

Wenn Sie (bei entsprechender Schalterstellung von [CH CODE]) also beispielsweise dieses Zeichen eingeben: '{', dann wird am Bildschirm dieses Zeichen ausgegeben: 'ä'
Es handelt sich dabei aber ausschließlich um einen veränderten Anzeigemodus des Bildschirmes, nicht jedoch um eine veränderte Datenhaltung von INFORMIX!

Um die deutschen Umlaute am Bildschirm sichtbar zu machen, müssen Sie also jedesmal dann, nachdem Sie INFORMIX neu aufgerufen haben, zunächst mit [CH CODE] in den Bildschirm-Abbildungsmodus schalten, in dem anstelle der Klammern die deutschen Umlaute ausgegeben werden.

Drucker

Bei einigen Druckern ist es möglich, den 'lpr'-Druckeraufruf um den Schalter '-dt' ('dt' = deutsch) zu erweitern. Die Wirkung hiervon ist vergleichbar der veränderten Bildschirmdarstellung durch [CH CODE] : die Klammern werden als Umlaute ausgedruckt.

Konkret müssen Sie folgende Maßnahmen treffen:

Bei Listen führt eine OUTPUT REPORT TO PRINTER Anweisung zum falschen Ergebnis, da dabei beim 'lpr'-Aufruf nicht der Schalter '-dt' übergeben wird. Die Anweisung muß lauten:

```
OUTPUT REPORT TO PIPE "lpr -dt"
```

Weiter müssen Sie jedesmal dort, wo Ihnen das Menüsystem bei der Funktion PRINT den Standard 'lpr' anbietet,

```
lpr -dt
```

als Ausgabeprogramm eintragen.

Format

Bei Formaten sind standardmäßig die eckigen Klammern die Feldbegrenzer; z.B. so:

```
Nachname  [              ]
```

Wenn Sie mit [CH CODE] auf die deutsche Bildschirmdarstellung schalten, dann sieht ein Bildschirmfeld so aus:

```
Nachname  Ä              Ü
```

Um das zu verhindern, müssen Sie die Format-Programme um eine Anweisung im INSTRUCTIONS-Abschnitt ergänzen, mit der Sie neue Feldbegrenzer definieren. Sie können z.B. diese Zeichen '<' und '>' als Feldbegrenzer definieren, so daß ein Bildschirmfeld so aussehen kann:

```
Nachname  <              >
```

Kommentare

Bei Kommentaren in SQL-Anweisungen, Format- und Listen-Programmen ist folgendes zu beachten: Das Zeichen '{' eröffnet einen Kommentar und das Zeichen '}' beendet einen Kommentar. Bei veränderter Bildschirmdarstellung werden die Zeichen wie folgt abgebildet: 'ä' und 'ü'
Der Kommentar selbst darf kein 'ü' (bzw. '}') enthalten, da dieses Zeichen das Kommentarende-Zeichen ist. Beispiel:

```
ä Diese Anweisung zeigt alle Saetze mit dem Ort Muenchen ü

SELECT
   *
FROM
   kunde
WHERE
   ort = "München"
```

Die gezeigt Anweisung liefert natürlich nur dann ein Ergebnis, wenn der Wert "München" wie folgt abgespeichert ist: "M}nchen" nicht aber so: "Muenchen"

Werteingabe

Wenn alle oben genannten Voraussetzungen erfüllt sind, dann können Sie Werte mit simulierten Umlauten eingeben. Bei SQL wie im vorherigen Beispiel gezeigt. Bei einem Format z.B. wie folgt:

```
Ort    <München      >
```

Nebenwirkungen

Das ganze oben geschilderte Verfahren wirkt sich allerdings nachteilig auf die sortierte Ausgabe bzw. auf das Suchen mit Vergleichsoperatoren aus, denn die Klammern kommen im ASCII-Code erst nach den alphabetischen Zeichen.
Wenn Sie z.B. diese Werte neu aufnehmen: Alltag, Ära, Zügel
Dann nehmen Sie tatsächlich auf: Alltag, [ra, Z}gel
Eine sortierte Ausgabe würde so aussehen: Alltag, Zügel, Ära

Zusammenfassung

- Mit einer CREATE DATABASE Anweisung können Sie eine Datenbank erzeugen. Mit einer DROP DATABASE Anweisung können Sie eine Datenbank löschen. Mit einer CLOSE DATABASE Anweisung können Sie die aktuelle Datenbank schließen.
- Mit einer CREATE TABLE Anweisung können Sie eine Tabelle erzeugen.
- Mit einer ALTER TABLE Anweisung können Sie in einer Tabelle eine Spalte einfügen, eine Spalte ändern, eine Spalte löschen.
- Mit einer RENAME TABLE Anweisung können Sie eine Tabelle umbenennen.
- Mit einer RENAME COLUMN Anweisung können Sie eine Spalte umbenennen.
- Mit einer CREATE INDEX Anweisung können Sie einen INDEX erzeugen. Mit einer DROP INDEX Anweisung können Sie einen Index löschen.
- Um mit simulierten Umlauten zu arbeiten, müssen Sie Änderungen auf den folgenden Gebieten vornehmen: Bildschirm, Druckeraufruf, Bildschirmfeldern. Bei Kommentaren in Programmen ist die veränderte Situation zu berücksichtigen.

10 Datenschutz und Datensicherheit

Dieses Kapitel erläutert Ihnen anhand von Beispielen einige Maßnahmen

- zum Datenschutz und
- zur Datensicherheit.

Vollständige Informationen erhalten Sie im Handbuch [1] SQL.

10.1 Datenschutz

Dieser Abschnitt informiert Sie über

- Zugriffsrechte, die Sie INFORMIX intern vergeben können,
- VIEWS (Fenster auf die Datenbank).

10.1.1 Zugriffsrechte

Mit Zugriffsrechten legen Sie fest, welche Aktionen ein Anwender (gemäß Benutzerkennung) an einer Tabelle bzw. an der gesamten Datenbank durchführen darf.
Mit einer GRANT-Anweisung können Sie einem Anwender Zugriffsrechte erteilen. Mit einer REVOKE-Anweisung können Sie einem Anwender Zugriffsrechte entziehen.

Die Funktion 'Info' (Rechte) im SQL-Dialog-Menü (oder eine INFO-Anweisung) zeigt, wie die Zugriffsrechte für eine Tabelle standardmäßig gesetzt sind:

```
Benutzer        Select      Update      Insert   Delete   Index   Alter

public          Alle        Alle         Ja       Ja       Ja      Nein
```

Wenn ein Benutzer nicht gesondert aufgeführt ist, dann besitzt er automatisch alle unter 'public' vergebenen Zugriffsrechte.

Ein Anwender darf hier:

Select
alle Spalten der Tabelle in einer SELECT-Anweisung abfragen.

Update
alle Spalten der Tabelle ändern.

Insert
Sätze neu aufnehmen.

Delete
Sätze löschen.

Index
Spalten indizieren.

Alter
die Tabellenstruktur ändern - für 'public' standardmäßig nicht erlaubt.

Mit einer GRANT-Anweisung können Sie einem Anwender gesondert Zugriffsrechte erteilen.

Dabei sind Zugriffsrechte auf zwei Ebenen zu erteilen:

- Datenbank,
- Tabelle.

Die Erteilung der Zugriffsrechte auf Datenbankebene bilden die Voraussetzung für die Zugriffsrechte auf Tabellenebene.

Die folgende GRANT-Anweisung erteilt dem Anwender 'gast' die Zugriffsrechte auf Datenbankebene:

```
GRANT
  connect
TO
  gast
```

Die folgende GRANT-Anweisung erteilt dem Anwender 'gast' die Zugriffsrechte für die SELECT-Anweisung, für das Ändern, Neuaufnehmen oder Löschen von Sätzen der Tabelle kunde:

```
GRANT
  select, update, insert, delete
ON
  kunde
TO
  gast
```

Nach dieser Anweisung zeigt 'Info' an:

```
Benutzer        Select      Update      Insert   Delete   Index   Alter

public          Alle        Alle         Ja       Ja       Ja      Nein
gast            Alle        Alle         Ja       Ja       Nein    Nein
```

Dem Anwender 'gast' sind damit alle Zugriffsrechte außer 'Index' und 'Alter' erteilt.

Mit einer REVOKE-Anweisung können Sie einem Anwender Zugriffsrechte entziehen. Die folgende REVOKE-Anweisung entzieht dem Anwender 'gast' das Zugriffsrecht für 'Delete':

```
REVOKE
  delete
ON
  kunde
FROM
```

```
gast
```

Nach dieser Anweisung zeigt 'Info' an:

```
Benutzer        Select      Update      Insert   Delete   Index   Alter

public          Alle        Alle        Ja       Ja       Ja      Nein
gast            Alle        Alle        Ja       Nein     Nein    Nein
```

Zugriffsrechte auf Betriebbsystemebene

Damit ein Anwender mit Formaten oder Listen eines anderen Eigentümers arbeiten kann, ist es erforderlich, auf Betriebssystemebene auch die SINIX-Zugriffsrechte entsprechend zu ändern.

10.1.2 VIEW

Eine View entspricht einem Fenster auf die Datenbank. Eine View zeigt nur einen ganz bestimmten, festgelegten Ausschnitt der Datenbank und andere Teile nicht.
Eine View ermöglicht, daß Sie z.B. erwünschte Dauerverbindungen zwischen zwei oder mehreren Tabellen herstellen können, oder daß Sie Datenschutzmaßnahmen treffen können, indem sie einem Anwender nur einen ganz bestimmten, einschränkenden Ausschnitt auf die Datenbank erlauben und ihm die Zugriffsrechte zu anderen Tabellen und Views entziehen.

Eine View besitzt ähnliche Eigenschaften wie eine Tabelle, ohne aber tatsächlich eine Tabelle zu sein. Sie können z.B.:

- eine View mit einer SELECT-Anweisung abfragen,
- neue Sätze in eine View aufnehmen,
- Sätze einer View verändern,
- Sätze einer View löschen,
- Zugriffsrechte für eine View erteilen und entziehen.

View als Dauerverbindung von Tabellen

Neben anderen denkbaren Anwendungen können Sie eine View dazu verwenden, eine erwüschte Dauerverbindung zwischen zwei oder mehreren Tabellen herzustellen. Das erleichtert in der Folge Datenmanipulationen wie z.B. Datenbank-Abfragen oder die Neuaufnahme von Sätzen.

Die folgende Anweisung erzeugt eine View mit dem Namen 'kund_auftr'. Die View definiert einen Ausschnitt aus der Datenbank 'versand'. Der Ausschnitt zeigt nur einige Spalten der Tabellen 'kunde' und 'auftrag'.
Die optionale WITH CHECK OPTION Klausel gewährleistet, daß der Anwender nur solche Sätze neuaufnehmen, ändern usw. kann, die die in der SELECT-Klausel gestellten Bedingungen erfüllen.

```
CREATE VIEW
   kund_auftr
AS
   select
       firma, ort,
       lieferdatum, zustellgebuehr, zahldatum
   from
       kunde, auftrag
   where
       kunde.kunden_nr = auftrag.kunden_nr
WITH CHECK OPTION
```

Die mit dieser Anweisung erzeugte View kann nun wie eine Tabelle behandelt werden. Z.B. ist es sehr einfach möglich, wie folgt eine Datenbank-Abfrage durchzuführen:

```
SELECT
   *
FROM
   kund_auftr
```

Die Anweisung liefert das folgende Ergebnis:

firma	ort	lieferdatum	zustellgebuehr	zahldatum
Spielball	Muenchen	01.02.1990	10,00	22.03.1990
Pauli Sport	Augsburg	06.06.1990	15,30	03.07.1990
Spielball	Muenchen	13.10.1990	10,80	04.11.1990
Sport Watt	Ulm	30.04.1990	19,20	
Olympia Sport	Ulm	19.12.1990	16,20	30.12.1990
Larsinger & Partner	Ingolstadt		14,20	
Sportecke	Muenchen	23.04.1990	25,20	
Sportwaren	Muenchen	06.12.1990	13,80	21.12.1990
Sport Keyser	Augsburg	04.03.1990	10,00	21.04.1990
Sportladen	Landshut	08.06.1990	12,30	22.07.1990
Spielball	Muenchen	13.04.1990	5,00	01.06.1990
Sportecke	Muenchen	09.06.1990	14,20	
Spielball	Muenchen	18.09.1990	12,20	10.10.1990
Sport Watt	Ulm	10.05.1990	12,30	18.07.1990
Sportwaren	Muenchen	01.08.1990	6,30	31.08.1990

View als Maßnahme zum Datenschutz

Eine View können Sie auch dazu verwenden, einem Anwender aus Gründen des Datenschutzes nur einen ganz bestimmten, einschränkenden Ausschnitt aus der Datenbank zu erlauben.

Sie gehen in diesem Fall wie folgt vor:

- Mit einer CREATE VIEW Anweisung legen Sie den Ausschnitt fest, den der Anwender verwenden darf. Die Anweisung sollte die WITH CHECK OPTIONS Klausel enthalten, die dem Anwender auschließlich Datenmanipulationen entsprechend der einschränkenden SELECT-Anweisung erlaubt.
- Mit einer GRANT-Anweisung erteilen Sie dem Anwender die erforderlichen Rechte an der View.
- Mit REVOKE-Anweisungen entziehen Sie dem Anwender den Zugriff auf andere Tabellen und Views der Datenbank.

10.2 Datensicherheit

Dieser Abschnitt erläutert Ihnen in Grundzügen, wie Sie

- mit einem sogenannten Transaktionsprotokoll eine zerstörte Datenbank wieder restaurieren können (die Beschreibung gilt für die Vorgehensweise bei INFORMIX-SE, die entsprechende Vorgehensweise bei INFORMIX-ONLINE entnehmen Sie dem [4] INFORMIX-ONLINE Administrator-Handbuch).
- Datenmanipulationen (Sätze hinzufügen, löschen oder modifizieren) im Rahmen einer Transaktion "auf Probe" durchführen können.

10.2.1 Transaktionsprotokoll

Um eine zerstörte Datenbank restaurieren zu können, benötigen Sie je eine Sicherungskopie

- der sogenannten Transaktionsprotokoll-Datei und
- der Datenbank

Welche einzelnen Schritte Sie unternehmen müssen, um eine zerstörte Datenbank wieder restaurieren zu können, ist weiter unten beschrieben. Hier zunächst einige Erläuterungen zum Transaktionsprotokoll.

In einem Transaktionsprotokoll werden sämtliche erfolgreich abgeschlossenen Änderungen an der Datenbank wie z.B. Sätze löschen, modifizieren oder hinzufügen sowie Änderungen an der Datenstruktur mit protokolliert.

Ein Transaktionsprotokoll können Sie auf die folgenden Weisen erzeugen:

- Wenn ein Transaktionsprotokoll von Anfang an geführt werden soll, dann geben Sie bei der Erzeugung der Datenbank (CREATE DATABASE) die optionale WITH LOG Klausel an.
- Wenn ein Transaktionsprotokoll nachträglich erzeugt werden soll, dann führen Sie die START DATABASE Anweisung aus (nur bei INFORMIX-SE möglich).

Beispiel

Die folgende Anweisung erzeugt die Datenbank 'test2' und sie erzeugt dabei eine Transaktionsprotokoll-Datei mit dem Namen 'protokoll1'.

```
CREATE DATABASE test2 WITH LOG IN "/usr/gast/vertiefung/protokoll1"
```

Die folgende Anweisung erzeugt für eine bereits existierende Datenbank 'test' eine Transaktionsprotokoll-Datei mit dem Namen 'protokoll2'.

```
START DATABASE test WITH LOG IN "/usr/gast/vertiefung/protokoll2"
```

Hinweis

Sie müssen den Namen der Transaktionsprotokoll-Datei mit dem absoluten Pfadnamen angeben. Die angegebene Transaktionsprotokoll-Datei darf noch nicht existieren, sonst wird die Anweisung abgewiesen.

Die START DATABASE Anweisung läßt sich nicht für die aktuell eröffnete Datenbank ausführen werden; führen Sie gegebenenfalls vorab die CLOSE DATABASE Anweisung aus.
START DATABASE sperrt die Datenbank exklusiv gegen jeglichen Zugriff anderer Prozesse. CLOSE DATABASE hebt die Sperre wieder auf.

10.2.2 Sicherungsmaßnahmen

Die hier beschriebenen Sicherungsmaßnahmen gelten nur für INFORMIX-SE. Zu den Maßnahmen bei INFORMIX-ONLINE siehe das Handbuch [4] INFORMIX-ONLINE Administrator-Handbuch.

Damit Sie gegebenenfalls eine zerstörte Datenbank wieder restaurieren können, müssen Sie vorab die folgenden Sicherungsmaßnahmen ergreifen:

1. Mit einer CREATE DATABASE Anweisung oder einer START DATABASE Anweisung sorgen Sie dafür, daß eine Transaktionsprotokoll-Datei zur Verfügung steht.

2. Die Transaktionsprotokoll-Datei sollten Sie in kurzen Abständen (z.B. täglich) auf externem Datenträger (Diskette, Magnetband-Kassette) sichern, so daß Sie immer über einen annähernd aktuellen Stand der Transaktionsprotokoll-Datei verfügen.
   In der Shell mit einem far oder tar Kommando, z.B. so:
   sie far cv /usr/gast/vertiefung/protokoll1

3. In regelmäßigen Abständen führen Sie vollständige Sicherungen des Dateiverzeichnisses 'datenbankname.dbs' auf externem Datenträger durch.
   In der Shell mit einem far oder tar Kommando, z.B. so:
   sie far cv test2.dbs
   Diese Anweisung sichert die Datenbank 'test2' auf Diskette(n).

4. Weiter müssen Sie dafür sorgen, daß auch künftig eine brauchbare Transaktionsprotokoll-Datei zur Verfügung steht. Es gibt dazu zwei mögliche Vorgehensweisen:

   Entweder Sie verwenden die alte Transaktionsprotokoll-Datei weiter; in diesem Fall müssen Sie dafür sorgen, daß die Datei vollständig leer ist. Das geht mit dem folgenden Shell-Kommando: cat /dev/null > pfaddatei, z.B. so:
   cat /dev/null > /usr/gast/vertiefung/protokoll1

   Oder Sie erzeugen mit START DATABASE eine neue Transaktionsprotokoll-Datei (siehe oben) und löschen mit einem Shell-Kommando die alte, z.B. so:
   rm /usr/gast/vertiefung/protokoll1
   Wenn Sie diese Vorgehensweise wählen, dann müssen Sie folgenden, veränderten Weg einschlagen: Zunächst erzeugen Sie mit START DATABASE eine neue Transaktionsprotokoll-Datei. Erst dann sichern Sie wie unter 3. beschrieben die Datenbank 'datenbankname.dbs'. Denn nur in diesem Fall ist der Datenbank der Name der Transaktionsprotokoll-Datei bekannt.

Die Transaktionsprotokoll-Datei kann sehr schnell recht umfangreich werden, was sich in entsprechend langen Sicherungszeiten bemerkbar macht. In diesem Fall sollten Sie die oben beschriebenen Punke 3. und 4. erneut durchführen.

10.2.3 Zerstörte Datenbank restaurieren

Wenn Sie die oben beschriebenen Maßnahmen durchgeführt haben, dann können Sie Ihre Datenbank wie folgt auf den annähernd aktuellen Stand bringen:

- Sie lesen die zuletzt gesicherte Transaktionsprotokoll-Datei und die zuletzt gesicherte Datenbank von den Datenträgern ein.
- Dann führen Sie (bei geschlossener Datenbank - CLOSE DATABASE) die ROLLFORWARD DATABASE Anweisung aus.
  Im Anschluß daran wird die Datenbank wieder auf den Stand der letzten, erfolgreich abgeschlossenen Aktion gebracht.

Beispiel

Die folgende Anweisung restauriert die Datenbank 'test2'.

```
ROLLFORWARD DATABASE test2
```

ROLLFORWARD DATABASE sperrt die Datenbank exklusiv gegen andere Prozesse; mit CLOSE DATABASE wird die Sperre wieder aufgehoben.

Hinweis

Wenn Sie über eine zweite, zusätzliche Festplatte verfügen, dann kann sich u.U. der folgende, abgeänderte Weg bei den Sicherungsmaßnahmen empfehlen:
Sowohl Transaktionsprotokoll-Datei (über Angabe des Pfadnamens in der WITH LOG IN Klausel) als auch die Sicherungskopie der Datenbank bringen Sie auf der zweiten Fesplatte unter.
In diesem Fall kann sich die regelmäßige, zeitaufwendige Grundsicherung der Datenbank, wie oben unter 3. beschrieben, unter Umständen erübrigen. Denn dann können Sie die Sicherungskopie der Datenbank mit dem Transaktionsprotokoll und der ROLLFORWARD DATABASE Anweisung mit relativ geringem Zeitaufwand auf den jeweils aktuellen Stand bringen. Im Ergebnis entspricht das einer erneuten, vollständigen Grundsicherung der Datenbank.
Aber auch in diesem Fall müssen Sie, wie oben unter 4. beschrieben, dafür sorgen, daß auch weiterhin ein brauchbares Transaktionsprotokoll zur Verfügung steht.

10.2.4 Transaktion

Wenn Sie für Ihre Datenbank ein Transaktionsprotokoll führen, dann können Sie "Transaktionen" durchführen.
Eine Transaktion ist eine Folge von Anweisungen, die als eine geschlossene, unteilbare Einheit behandelt wird.

In einer Transaktion können Sie eine Reihe von Datenmanipulationen (Sätze neu aufnehmen, löschen oder modifizieren) vornehmen, die Sie

- insgesamt durchführen lassen können oder
- gegebenenfalls auch insgesamt wieder rückgängig machen können.

Eine Transaktion ermöglicht damit Datenmanipulationen "auf Probe", da Sie entscheiden können, ob die in der Transaktion durchgeführten Anweisungen beibehalten oder wieder rückgängig gemacht werden sollen.

Transaktion beginnen

Die Anweisung BEGIN WORK beginnt eine Transaktion.

Hinweis

Sätze, die Sie nun neu aufnehmen, modifizieren oder löschen, sind bis zum Abschluß der Transaktion automatisch gegen den schreibenden Zugriff anderer Prozesse gesperrt und können nur gelesen werden (SELECT).

BEGIN WORK wird ignoriert, wenn bereits eine Transaktion eröffnet ist.

Transaktion beenden

Wenn die Anweisungen im Rahmen der Transaktion das erwünschte Ergebnis geliefert haben, dann beenden Sie die Transaktion mit der Anweisung COMMIT WORK. In diesem Fall sind Ihre Änderungen an der Datenbank wirksam.

Hinweis

Bei COMMIT WORK wird die Transaktion im Transaktionsprotokoll als abgeschlossen gekennzeichnet.

COMMIT WORK hebt Satzsperren auf, die automatisch im Rahmen der Transaktion gesetzt wurden.

COMMIT WORK wird ignoriert, wenn keine Transaktion eröffnet ist.

Wenn Sie INFORMIX verlassen, ohne eine Transaktion mit COMMIT WORK zu beenden, dann wird die Transaktion automatisch zurückgesetzt (wie ROLLBACK WORK - siehe unten).

Transaktion zurücksetzen

Wenn die Anweisungen im Rahmen der Transaktion nicht das erwünschte Ergebnis geliefert haben, dann beenden Sie die Transaktion mit ROLLBACK WORK. In diesem Fall wird die Datenbank wieder in den Stand zurückversetzt, der vor Ausführung der Transaktion bestand.

Hinweis

Die Anweisung setzt alle Satzsperren zurück.

Wenn Sie INFORMIX verlassen, ohne eine Transaktion mit ROLLBACK WORK zu beenden, dann wird die Transaktion automatisch zurückgesetzt.

ROLLBACK WORK wird abgewiesen, wenn keine Transaktion eröffnet ist.

Beispiel

Dieses Beispiel demonstriert, wie im Rahmen einer Transaktion neue Sätze in die Tabelle 'hersteller' aufgenommen werden und wie anschließend wieder der alte Stand der Datenbank hergestellt wird.

Die folgende Anweisung zeigt den ursprünglichen Stand der Tabelle:

```
SELECT
   *
FROM
   hersteller
```

Das Ergebnis:

```
herstellercode herstellername

SMT            Schmitt
ANZ            Anzabel
NRG            Norgesta
HSK            Hasken
HRO            Herold
```

Die folgende Anweisung schaltet zunächst ein Transaktionsprotokoll für die Datenbank 'versand' ein; die vorausgehende Anweisung CLOSE DATABASE ist deshalb erforderlich, da für die aktuell eröffnete Datenbank die START DATABASE Anweisung nicht erlaubt ist.

```
CLOSE DATABASE;
START DATABASE versand WITH LOG IN "/usr/gast/vertiefung/protokoll1"
```

Die folgende Anweisung beginnt die Transaktion.

```
BEGIN WORK
```

Mit den folgenden Anweisungen werden zwei neue Sätze in die Tabelle aufgenommen.

```
INSERT INTO hersteller
VALUES
   ("BRN", "Berninger");
INSERT INTO hersteller
VALUES
   ("FRN", "Freimann")
```

Die SELECT Anweisung zeigt nun:

```
herstellercode herstellername

SMT            Schmitt
ANZ            Anzabel
NRG            Norgesta
HSK            Hasken
HRO            Herold
BRN            Berninger
FRN            Freimann
```

Die folgende Anweisung setzt die Transaktion wieder zurück.

```
ROLLBACK WORK
```

Nun zeigt die SELECT Anweisung wieder den ursprünglichen Stand:

```
herstellercode herstellername

SMT            Schmitt
ANZ            Anzabel
NRG            Norgesta
HSK            Hasken
HRO            Herold
```

Zusammenfassung

- Mit einer GRANT Anweisung können Sie einem Anwender Zugriffsrechte auf die Datenbank erteilen.
- Mit einer REVOKE Anweisung können Sie einem Anwender Zugriffsrechte auf die Datenbank entziehen.
- Eine View entspricht einem Fenster auf die Datenbank. Sie können eine View u.a. dazu verwenden, erwünschte Dauerverbindungen zwischen zwei oder mehreren Tabellen herzustellen. Eine View kann auch als Maßnahme zum Datenschutz dienen.
- START DATABASE oder CREATE DATABASE mit der WITH LOG IN Klausel erzeugen ein Transaktionsprotokoll (INFORMIX-SE).
- Das Transaktionsprotokoll und eine Grundsicherung der Datenbank ermöglichen, daß eine zerstörte Datenbank auf den annähernd aktuellen Stand zurückversetzt werden kann (INFORMIX-SE).
- Wenn ein Transaktionsprotokoll eingeschaltet ist, dann können Sie im Rahmen einer Transaktion Datenmanipulationen "auf Probe" durchführen.

A Anhang

A.1 Lösungen der Aufgaben

Aufgabe 3-1

Aufgabe A)

In das Bildschirmfeld 'vorname' tragen Sie 'Herta' ein:

```
nachname        [                    ]
vorname         [Herta               ]
strasse         [                    ]
plz             [          ]
ort             [                    ]
telefon         [               ]
```

Aufgabe B)

In das Bildschirmfeld 'strasse' tragen Sie 'Waldweg 17' ein:

```
nachname        [                    ]
vorname         [                    ]
strasse         [Waldweg 17          ]
plz             [          ]
ort             [                    ]
telefon         [               ]
```

Aufgabe C)

In das Bildschirmfeld 'plz' tragen Sie '5300' ein:

```
nachname        [                    ]
vorname         [                    ]
strasse         [                    ]
plz             [5300      ]
ort             [                    ]
telefon         [               ]
```

Aufgabe 3-2

Aufgabe A)

Die SELECT-Anweisung lautet:

```
SELECT
   nachname, ort
FROM
   adressen
```

Aufgabe B)

Die SELECT-Anweisung lautet:

```
SELECT
   nachname, vorname, strasse, plz, ort, telefon
FROM
   adressen
WHERE
   ort = "Ulm"
```

Aufgabe C)

Die SELECT-Anweisung lautet:

```
SELECT
   vorname, telefon
FROM
   adressen
WHERE
   ort = "Heidelberg"
```

Aufgabe D)

Die SELECT-Anweisung lautet:

```
SELECT
   nachname, vorname, strasse, plz, ort, telefon
FROM
   adressen
WHERE
   plz = 7900
```

Aufgabe 5-1

Aufgabe A)

In das Bildschirmfeld 'Nachname' tragen Sie die Suchbedingung '*g*' ein:

```
Nachname              [*g*              ]
```

Aufgabe B)

In das Bildschirmfeld 'Kundennummer' tragen Sie die Suchbedingung '105:115' und in das Bildschirmfeld 'Ort' tragen Sie die Suchbedingung 'Muenchen' ein:

```
Kundennummer [105:115     ]

Ort          [Muenchen        ]
```

Aufgabe C)

In das Bildschirmfeld 'Telefon' tragen Sie die Suchbedingung '*/??????' ein:

```
Telefon      [*/??????          ]
```

Aufgabe 7-1

Aufgabe A)

Die SELECT-Anweisung lautet:

```
SELECT
   *
FROM
   artikel
```

Aufgabe B)

Die SELECT-Anweisung lautet:

```
SELECT
   artikel_nr artikelnummer, bezeichnung artikelbezeichnung,
   preis grundpreis, stueckliste
FROM
   artikel
```

Aufgabe C)

Die SELECT-Anweisung lautet:

```
SELECT
   auftrags_nr auftragsnummer, kunden_nr kundennummer,
   lieferhinweis[1,11]
FROM
   auftrag
```

Aufgabe 7-2

Die SELECT-Anweisung lautet:

```
SELECT
   kunden_nr, lieferdatum, zahldatum, offen
FROM
   auftrag
WHERE
     kunden_nr = 106
   AND
     offen = "j"
```

Aufgabe 7-3

Die SELECT-Anweisung lautet:

```
SELECT
   *
FROM
   kunde
WHERE
     ort MATCHES "A*"
   OR
     ort MATCHES "M*"
```

Aufgabe 7-4

Die SELECT-Anweisung lautet:

```
SELECT
   auftrags_nr, auftragsdatum, kunden_nr, zahldatum
FROM
   auftrag
ORDER BY
   kunden_nr, zahldatum
```

Aufgabe 7-5

Die SELECT-Anweisung lautet:

```
SELECT
   auftrags_nr, gesamtpreis, menge,
   gesamtpreis / menge einzelpreis
FROM
   posten
WHERE
   menge > 1
```

Aufgabe 7-6

Die SELECT-Anweisung lautet:

```
SELECT
   count(*) anzahl, auftrags_nr
FROM
   posten
GROUP BY
   auftrags_nr
HAVING
   count(*) > 1
```

Aufgabe 7-7

Die SELECT-Anweisung lautet:

```
SELECT
   auftrags_nr, kunden_nr, lieferdatum
FROM
   auftrag
WHERE
      MONTH(auftragsdatum) > 7
   AND
      YEAR(auftragsdatum) = 1990
```

Aufgabe 7-8

Die SELECT-Anweisung lautet:

```
SELECT
   auftrag.auftrags_nr,
   posten.menge, posten.gesamtpreis,
   artikel.artikel_nr, artikel.bezeichnung, artikel.preis
FROM
   auftrag, posten, artikel
WHERE
     auftrag.auftrags_nr = posten.auftrags_nr
   AND
     posten.artikel_nr = artikel.artikel_nr
   AND
     posten.menge > 1
```

A.2 Die ASCII-Zeichen

dezi-mal	oktal	hexa-dez.		Bedeutung	Control
0	00	00	NUL	Null, keine Operation	@
1	01	01	SOH	Start of Heading Vorspannanfang	A
2	02	02	STX	Start of Text Textanfang	B
3	03	03	ETX	End of Text Textende	C
4	04	04	EOT	End of Transmission Taste END Übertragungsende	D
5	05	05	ENQ	Enquiry Stationsanruf	E
6	06	06	ACK	Acknowledge Bestätigung	F
7	07	07	BEL	Bell Klingel	G
8	10	08	BS	Backspace Korrekturtaste	H
9	11	09	HT	Horizontal Tabulation Tabulatorzeichen	I
10	12	0A	LF	Line Feed Zeilenvorschub, neue Zeile	J
11	13	0B	VT	Vertical Tabulation	K
12	14	0C	FF	Form Feed Formularvorschub	L
13	15	0D	CR	Carriage Return Wagenrücklauf	M
14	16	0E	SO	Shift Out Umschalten Zeichensatz	N
15	17	0F	SI	Shift In Zurückschalten Zeichensatz	O
16	20	10	DLE	Data Link Escape Austritt aus der Datenverbindung	P
17	21	11	DC1	Device Control 1 Gerätesteuerung 1, Ausgabe fortsetzen	Q
18	22	12	DC2	Device Control 2	R
19	23	13	DC3	Device Control 3 Ausgabe anhalten	S
20	24	14	DC4	Device Control 4	T
21	25	15	NAK	Negative Acknowledge Fehlermeldung	U
22	26	16	SYN	Synchronous Idle Synchronisierung	V
23	27	17	ETB	End of Transm. Block Datenblockende	W
24	30	18	CAN	Cancel ungültig, Zeilenlöscher	X
25	31	19	EM	End of Medium Datenträgerende, quit (Signal3)	Y
26	32	1A	SUB	Substitute Character Zeichen ersetzen	Z
27	33	1B	ESC	Escape Rücksprung	

dezi-mal	oktal	hexa-dez.		Bedeutung		Control
28	34	1C	FS	File Separator	Dateitrennung	\
29	35	1D	GS	Group Separator	Gruppentrennung	]
30	36	1E	RS	Record Separator	Satztrennung	^
31	37	1F	US	Unit Separator	Einheitentrennung	_ oder DEL
32	40	20	SP	SPACE	Leerzeichen	
33	41	21	!			
34	42	22	"			
35	43	23	#	Nummernzeichen		
36	44	24	$	oder nationales Währungssymbol		
37	45	25	%			
38	46	26	&			
39	47	27	'			
40	50	28	(			
41	51	29	)			
42	52	2A	*	(Stern gilt oft als Multiplikations-zeichen)		
43	53	2B	+			
44	54	2C	'			
45	55	2D	-			
46	56	2E	.			
47	57	2F	/	(dient als Divisionszeichen)		
48	60	30	0			
49	61	31	1			
50	62	32	2			
51	63	33	3			
52	64	34	4			
53	65	35	5			
54	66	36	6			
55	67	37	7			
56	70	38	8			
57	71	39	9			
58	72	3A	:			
59	73	3B	;			
60	74	3C	<			
61	75	3D	=			
62	76	3E	>			
63	77	3F	?			
64	100	40	@	(kaufmännisches "at" oder §)		
65	101	41	A			
66	102	42	B			
67	103	43	C			
68	104	44	D			
69	105	45	E			
70	106	46	F			
71	107	47	G			
72	110	48	H			
73	111	49	I			
74	112	4A	J			
75	113	4B	K			
76	114	4C	L			
77	115	4D	M			

dezi-mal	oktal	hexa-dez.		Bedeutung	Control
78	116	4E	N		
79	117	4F	O		
80	120	50	P		
81	121	51	Q		
82	122	52	R		
83	123	53	S		
84	124	54	T		
85	125	55	U		
86	126	56	V		
87	127	57	W		
88	130	58	X		
89	131	59	Y		
90	132	5A	Z		
91	133	5B	[	oder Ä	
92	134	5C	\	Gegenschrägstrich oder Ö	
93	135	5D	]	oder Ü	
94	136	5E	^	oder ↑	
95	137	5F	_	Unterstrich oder ←	
96	140	60	'		
97	141	61	a		
98	142	62	b		
99	143	63	c		
100	144	64	d		
101	145	65	e		
102	146	66	f		
103	147	67	g		
104	150	68	h		
105	151	69	i		
106	152	6A	j		
107	153	6B	k		
108	154	6C	l		
109	155	6D	m		
110	156	6E	n		
111	157	6F	o		
112	160	70	p		
113	161	71	q		
114	162	72	r		
115	163	73	s		
116	164	74	t		
117	165	75	u		
118	166	76	v		
119	167	77	w		
120	170	78	x		
121	171	79	y		
122	172	7A	z		
123	173	7B	{	oder ä	
124	174	7C	|	oder ö	
125	175	7D	}	oder ü	
126	176	7E	~	oder ß	
127	177	7F	DEL	Delete Löschzeichen, Interrupt (Signal2)	

A.3 Beispieldatenbank versand

In diesem Abschnitt wird die Beispieldatenbank *versand* beschrieben, die für die Beispiele in diesem Handbuch verwendet wird. Die Struktur der Datenbank-Tabellen werden beschrieben, für welche Spalten ein Index definiert ist und über welche Spalten eine Verbindung der Tabellen (Join) möglich ist.

Für jede Tabelle werden die in ihr enthaltenen Daten aufgelistet.

Struktur der Tabellen

Die Datenbank *versand* enthält Informationen über einen fiktiven Sportartikelgroßhändler, der Läden in Süddeutschland beliefert. Die Datenbank besteht aus fünf Tabellen:

- kunde
- auftrag
- posten
- artikel
- hersteller
- bundesland

Tabelle kunde

Die Tabelle *kunde* enthält Informationen über 18 Läden, die Sportartikel vom Großhändler beziehen. Für jeden Laden wird der Name des Ladeninhabers und der Name und die Adresse des Ladens gespeichert. Die Tabelle *kunde* hat folgende Spalten:

```
Spaltenname          Datentyp

kunden_nr            SERIAL
vorname              CHAR(15)
nachname             CHAR(15)
firma                CHAR(20)
adresse1             CHAR(20)
adresse2             CHAR(20)
ort                  CHAR(15)
bundesland           CHAR(2)
plz                  CHAR(5)
telefon              CHAR(18)
```

Für die Spalte *kunden_nr* ist ein eindeutiger Index definiert. Für die Spalte *plz* ist ebenfalls ein Index definiert; bei diesem sind doppelt vorkommende Werte erlaubt.

Tabelle auftrag

Die Tabelle *auftrag* enthält die Aufträge, die die Kunden dem Großhändler erteilt haben. Für jeden Auftrag ist die Nummer des Auftrags gespeichert; zusätzlich das Auftragsdatum, die Kundennummer, Versandanweisungen, evtl. Rückstände, die Nummer des Kundenbestellscheins, Versandkosten und das Datum, an dem der Kunde für den Auftrag bezahlt hat. Die Spalten in der Tabelle *auftrag* sind:

Spaltenname	Datentyp
auftrags_nr	SERIAL
auftragsdatum	DATE
kunden_nr	INTEGER
lieferhinweis	CHAR(40)
offen	CHAR(1)
fremd_nr	CHAR(10)
lieferdatum	DATE
liefergewicht	DECIMAL(8,2)
zustellgebuehr	MONEY(6,2)
zahldatum	DATE

Für die Spalte *auftrags_nr* ist ein eindeutiger Index definiert; die Spalte *kunden_nr* ist ebenfalls indiziert, läßt aber doppelt vorkommende Werte zu.

Tabelle posten

Die Tabelle *posten* enthält die einzelnen Positionen eines Auftrags. Beispiel: Einige Aufträge enthalten nur einen Posten, während andere Aufträge fünf Posten haben. Die Tabelle *posten* enthält die Postennummer, Auftragsnummer, Artikelnummer, Herstellercode, Menge und den Gesamtpreis für jeden Artikel, der bestellt wurde. Die Spalten in der Tabelle *posten* sind:

```
Spaltenname        Datentyp

posten_nr          SMALLINT
auftrags_nr        INTEGER
artikel_nr         SMALLINT
herstellercode     CHAR(3)
menge              SMALLINT
gesamtpreis        MONEY(8,2)
```

Die Spalte *auftrags_nr* ist indiziert und läßt doppelt vorkommende Werte zu. Die Spalten *artikel_nr* und *herstellercode* bilden einen zusammengesetzen Index, doppelt vorkommende Werte sind erlaubt.

Tabelle artikel

Der Großhändler bietet seinen Kunden fünfzehn verschiedene Arten von Sportartikeln an. Beispiel: Der Großhändler bietet SkiHandschuhe von drei verschiedenen Herstellern und Basketbälle von einem Hersteller an.

Die Tabelle *artikel* ist ein Katalog der Artikel, die vom Großhändler verkauft werden. Für jeden Artikel im Lager hat die Tabelle *artikel* eine Nummer, die den Artikel kennzeichnet, einen Code, der den Hersteller bezeichnet, eine Beschreibung des Artikels, den Preis der Packung, die Mindestmenge, die bestellt werden muß und eine Beschreibung der Pakkung (z.B.: Eine Kiste enthält zehn Fußbälle).

Die Tabelle *artikel* enthält folgende Spalten:

Spaltenname	Datentyp
artikel_nr	SMALLINT
herstellercode	CHAR(3)
bezeichnung	CHAR(15)
preis	MONEY(6,2)
liefereinheit	CHAR(4)
stueckliste	CHAR(15)

Die Spalten *artikel_nr* und *herstellercode* bilden einen zusammengesetzten Index, der eindeutige Werte enthalten muß.

Tabelle hersteller

Der Großhändler hat Sportartikel von fünf Herstellern. Informationen über diese Hersteller sind in der Tabelle *hersteller* gespeichert. Diese Informationen bestehen aus einem Herstellercode und dem Namen des Herstellers. Die Spalten der Tabelle *hersteller* sind:

Spaltenname	Datentyp
herstellercode	CHAR(3)
herstellername	CHAR(15)

Für die Spalte *herstellercode* ist ein eindeutiger Index definiert.

Tabelle bundesland

Die Tabelle *bundesland* enthält die Namen und Abkürzungen der 11 Bundesländer der Bundesrepublik Deutschland. Die Spalten der Tabelle *bundesland* sind:

Spaltenname	Datentyp
bcode	CHAR(2)
laendername	CHAR(20)

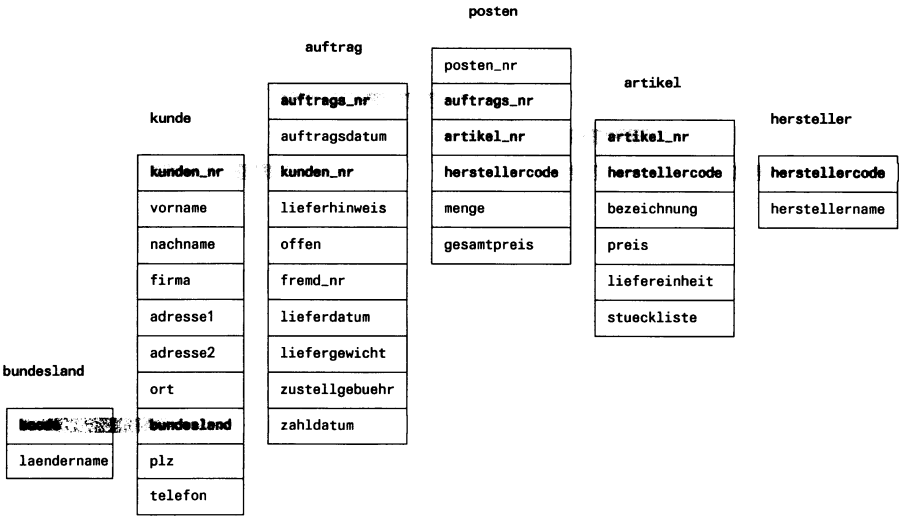
Für die Spalte *bcode* ist ein eindeutiger Index definiert.

Verbindung der Tabellen durch Join-Spalten

Die sechs Tabellen der Datenbank *versand* lassen sich durch Spalten verbinden, die als Join-Spalten geeignet sind. Das ermöglicht eine Datenbankabfrage über mehrere Tabellen gleichzeitig.

Übersicht

Die gerasterten Spalten zeigen die sinnvollen Join-Verbindungen zwischen den Tabellen der Datenbank.



Im folgenden wird jeweils die Verbindung von zwei Tabellen betrachtet.

Join-Spalten in den Tabellen kunde und auftrag

Die Tabellen *kunde* und *auftrag* lassen sich durch die Spalte *kunden_nr* in der Tabelle *kunde* und *kunden_nr* in der Tabelle *auftrag* verbinden.

Ausschnitt aus der Tabelle *kunde*:

kunden_nr	vorname	nachname
101	Ludwig	Pauli
102	Karola	Sadler
103	Philipp	Korres
104	Anton	Hochfeld

Ausschnitt aus der Tabelle *auftrag*:

auftrags_nr	auftragsdatum	kunden_nr
1001	20.01.1990	**104**
1002	01.06.1990	101
1003	12.10.1990	**104**
1004	12.04.1990	106

Die Tabelle *kunde* enthält in der Spalte *kunden_nr* für jeden Kunden eine eindeutige Nummer und zusätzlich Spalten für Namen, Firma, Adresse, Telefonnummer etc. Die Tabelle *auftrag* enthält ebenfalls eine Spalte *kunden_nr*, die die Nummer des Kunden enthält, der den Auftrag gab und weitere Spalten, die den Auftrag betreffen.

Will man die Aufträge von Anton Hochfeld abfragen, so verbindet man die Tabellen *kunde* und *auftrag* über die Spalte *kunden_nr*. Anton Hochfeld hat die Kundennummer 104 (aus Tabelle *kunde* gelesen). In der Tabelle *auftrag* gibt es zwei Aufträge mit der Kundennummer 104, Anton Hochfeld hat also zur Zeit zwei Aufträge gegeben.

Durch die Verbindung der Tabellen *kunde* und *auftrag* über die JoinSpalte *kunden_nr* kann der Name und die Adresse eines Kunden gleichzeitig mit den zugehörigen Aufträgen abgefragt werden.

Join-Spalten in den Tabellen auftrag und posten

Die Tabellen *auftrag* und *posten* lassen sich durch die Spalte *auftrags_nr* in der Tabelle *auftrag* und *auftrags_nr* in der Tabelle *posten* verbinden. In der Tabelle *auftrag* wird jeder Auftrag eines Kunden durch eine eindeutige Auftragsnummer identifiziert. Für jeden Posten dieses Auftrags wird ein Satz mit der zugehörigen Auftragsnummer und Informationen über den bestellten Artikel in die Tabelle *posten* geschrieben.

Durch die Join-Spalte *auftrags_nr* wird die Zuordnung zwischen Auftrag und zugehörigen Posten hergestellt.

Ausschnitt aus der Tabelle *auftrag*:

auftrags_nr	auftragsdatum	kunden_nr
1001	20.01.1990	104
1002	01.06.1990	101
1003	12.10.1990	104

Ausschnitt aus der Tabelle *posten*:

posten_nr	auftrags_nr	artikel_nr	herstellercode
1	1001	1	HRO
1	1002	4	HSK
2	1002	3	HSK
1	1003	9	ANZ
2	1003	8	ANZ
3	1003	5	ANZ

Join-Spalten in den Tabellen posten und artikel

Die Tabellen *posten* und *artikel* lassen sich durch zwei Spalten verbinden: *artikel_nr* und *herstellercode* in der Tabelle *posten* und *artikel_nr* und *herstellercode* in der Tabelle *artikel*.

Beide Spalten werden zur eindeutigen Identifizierung eines Artikels benötigt; z.B. ist der Artikel mit der Artikelnummer 1 und dem Herstellercode HRO ein Herold Ski-Handschuh, während der Artikel mit der Artikelnummer 1 und dem Herstellercode HSK ein Hasken Ski-Handschuh ist.

Die gleichen Artikelnummern und Herstellercodes können in mehreren Sätzen der Tabelle *posten* auftreten, wenn dieser Artikel in verschiedenen Aufträgen bestellt wurde.

Durch diese beiden Spalten *artikel_nr* und *herstellercode* wird der Zusammenhang zwischen Posten eines Auftrags (= bestellter Artikel) in der Tabelle *posten* und dem Artikel in der Tabelle *artikel* hergestellt.

Ausschnitt aus der Tabelle *posten*:

posten_nr	auftrags_nr	artikel_nr	herstellercode
1	1001	**1**	**HRO**
1	1002	4	HSK
2	1002	3	HSK
1	1003	9	ANZ
2	1003	8	ANZ
3	1003	5	ANZ
1	1004	**1**	**HRO**

Ausschnitt aus der Tabelle *artikel*:

artikel_nr	herstellercode	bezeichnung
1	**HRO**	Ski-Handschuhe
1	HSK	Ski-Handschuhe
1	SMT	Ski-Handschuhe

Join-Spalten in den Tabellen artikel und hersteller

Die Tabellen *artikel* und *hersteller* lassen sich durch die Spalte *herstellercode* in der Tabelle *artikel* und *herstellercode* in der Tabelle *hersteller* verbinden. Dieser Herstellercode erscheint in mehreren Sätzen der Tabelle *artikel*, in der Tabelle *hersteller* ist er eindeutig.

Durch die Verbindung der Tabellen *artikel* und *hersteller* über die Spalte *herstellercode* kann zu einem bestimmten Artikel der volle Name des Herstellers gefunden werden.

Ausschnitt aus der Tabelle *artikel*:

artikel_nr	herstellercode	bezeichnung
1	HRO	Ski-Handschuhe
1	HSK	Ski-Handschuhe
1	SMT	Ski-Handschuhe
2	HRO	Ski-Brille

Ausschnitt aus der Tabelle *hersteller*:

herstellercode	herstellername
NRG	Norgesta
HSK	Hasken
HRO	Herold

Join-Spalten in den Tabellen kunde und bundesland

Die Tabellen *kunde* und *bundesland* lassen sich durch die Spalte *bundesland* in der Tabelle *kunde* und *bcode* in der Tabelle *bundesland* verbinden. Wenn mehrere Kunden im selben Bundesland wohnen, tritt die Abkürzung für das Bundesland in mehreren Sätzen der Tabelle *kunde* auf.

Ausschnitt aus der Tabelle *kunde*:

kunden_nr	vorname	nachname	...	bundesland
101	Ludwig	Pauli		**BY**
102	Karola	Sadler		**BY**
103	Philipp	Korres		**BY**

Ausschnitt aus der Tabelle *bundesland*:

bcode	laendername
BW	Baden-Wuerttemberg
BY	Bayern
BE	Berlin (West)
BR	Bremen

Daten in der Datenbank versand

Die Daten, die in der Datenbank *versand* gespeichert sind, finden Sie in den folgenden Tabellen.

Tabelle kunde

kunden_nr	vorname	nachname	firma	adresse1	adresse2	ort	bundesland	plz	telefon
101	Ludwig	Pauli	Pauli Sport	Forstweg 47		Augsburg	BY	8900	0821/8075
102	Karola	Sadler	Sport-Aktiv	Blumenstr. 12		Wasserburg	BY	8090	08071/121289
103	Philipp	Korres	Philipp's Sportwaren	Sandgasse 2	Postfach 3498	Rosenheim	BY	8200	08031/4543
104	Anton	Hochfeld	Spielball	Goethestr. 34	Lilienstr.42	Muenchen	BY	8000	089/25430
105	Raimund	Viktor	Der Laden	Hofstr. 34		Ingolstadt	BY	8070	0841/2321
106	Georg	Watt	Sport Watt	Am Bergsteig 88		Ulm	BY	7900	0731/7334421
107	Karl	Remark	- Sportwaren -	Allee 13		Rosenheim	BY	8200	08031/560324
108	Martin	Korting	Martin's Shop	Buchenstr. 51		Muenchen	BY	8000	089/33730
109	Janette	Millet	Sportausstattung	Munsterstr. 1	Wofgangstr 66d	Augsburg	BY	8900	0821/358789
110	Roland	Jaeger	Sportwaren	Herbststr. 77		Muenchen	BY	8000	089/5032
111	Frank	Keyser	Sport Keyser	Im Tal 2		Augsburg	BY	8900	0821/7845
112	Margarete	Larsinger	Larsinger & Partner	Wiesenweg 11		Ingolstadt	BY	8070	0841/797235
113	Liane	Bergen	Sporthaus	Hochstr. 59		Landshut	BY	8300	0871/8699182
114	Frank	Albert	Der Sport-Albert	Torstr. 33		Muenchen	BY	8000	089/6367
115	Alfred	Grantella	Sportladen	Forchenstr. 99		Landshut	BY	8300	0871/498822
116	Johannes	Partellman	Olympia Sport	Spinosastr. 10		Ulm	BY	7900	0731/601123
117	Arnold	Sipell	Sportecke	Marktplatz 4		Muenchen	BY	8000	089/6321
118	Dieter	Bachmann	Sportausstatter	Lindenweg 4		Passau	BY	8390	0851/560011

Tabelle auftrag

auftrags_nr	auftragsdatum	kunden_nr	lieferhinweis	offen	fremd_nr	lieferdatum	liefergewicht	zustellgebuehr	zahldatum
1001	20.01.1990	104	Spedition	n	B77836	01.02.1990	20,40	10,00	22.03.1990
1002	01.06.1990	101	Hintertuere klingeln	n	9270	06.06.1990	50,60	15,30	03.07.1990
1003	12.10.1990	104	Spedition	n	B77890	13.10.1990	35,60	10,80	04.11.1990
1004	12.04.1990	106	zweimal klingeln	j	8006	30.04.1990	95,80	19,20	
1005	04.12.1990	116	vorher anrufen	n	2865	19.12.1990	80,80	16,20	30.12.1990
1006	19.09.1990	112	nach 10.00 Uhr	j	013557		70,80	14,20	
1007	25.03.1990	117		n	278893	23.04.1990	125,90	25,20	
1008	17.11.1990	110	Samstags geschlossen	j	L2230	06.12.1990	45,60	13,80	21.12.1990
1009	14.02.1990	111	zweite Tuere rechts	n	4745	04.03.1990	20,40	10,00	21.04.1990
1010	29.05.1990	115	wenn geschlossen, nebenan abliefern	n	4290	08.06.1990	40,60	12,30	22.07.1990
1011	23.03.1990	104	Spedition	n	B77897	13.04.1990	10,40	5,00	01.06.1990
1012	05.08.1990	117		n	278701	09.06.1990	70,80	14,20	
1013	01.09.1990	104	Spedition	n	B77930	18.09.1990	60,80	12,20	10.10.1990
1014	01.05.1990	106	mehrfach klingeln	n	8052	10.05.1990	40,60	12,30	18.07.1990
1015	10.07.1990	110	Samstags geschlossen	n	MA003	01.08.1990	20,60	6,30	31.08.1990

Tabelle posten

posten_nr	auftrags_nr	artikel_nr	herstellercode	menge	gesamtpreis
1	1001	1	HRO	1	250,00
1	1002	4	HSK	1	960,00
2	1002	3	HSK	1	240,00
1	1003	9	ANZ	1	20,00
2	1003	8	ANZ	1	840,00
3	1003	5	ANZ	5	99,00
1	1004	1	HRO	1	960,00
2	1004	2	HRO	1	126,00
3	1004	3	HSK	1	240,00
4	1004	1	HSK	1	800,00
1	1005	5	NRG	10	280,00
2	1005	5	ANZ	10	198,00
3	1005	6	SMT	1	36,00
4	1005	6	ANZ	1	48,00
1	1006	5	SMT	5	125,00
2	1006	5	NRG	5	190,00
3	1006	5	ANZ	5	99,00
4	1006	6	SMT	1	36,00
5	1006	6	ANZ	1	48,00
1	1007	1	HRO	1	250,00
2	1007	2	HRO	1	126,00
3	1007	3	HSK	1	240,00
4	1007	4	HRO	1	480,00
5	1007	7	HRO	1	600,00
1	1008	8	ANZ	1	840,00
2	1008	9	ANZ	5	100,00
1	1009	1	SMT	1	450,00
1	1010	6	SMT	1	36,00
2	1010	6	ANZ	1	48,00
1	1011	5	ANZ	5	99,00
1	1012	8	ANZ	1	840,00
2	1012	9	ANZ	10	200,00
1	1013	5	ANZ	1	19,80
2	1013	6	SMT	1	36,00
3	1013	6	ANZ	1	48,00
4	1013	9	ANZ	2	40,00
1	1014	4	HSK	1	960,00
2	1014	4	HRO	1	480,00
1	1015	1	SMT	1	450,00

Tabelle artikel

artikel_nr	herstellercode	bezeichnung	preis	liefereinheit	stueckliste
1	HRO	Ski-Handschuhe	250,00	Box	10/Box
1	HSK	Ski-Handschuhe	800,00	Box	10/Box
1	SMT	Ski-Handschuhe	450,00	Box	10/Box
2	HRO	Ski-Brille	126,00	Box	24/Box
3	HSK	Ski-Stock	240,00	Col.	12/Colli
4	HSK	Fussball	960,00	Col.	24/Colli
4	HRO	Fussball	480,00	Col.	24/Colli
5	NRG	Tennisschlaeger	28,00	solo	solo
5	SMT	Tennisschlaeger	25,00	solo	solo
5	ANZ	Tennisschlaeger	19,80	solo	solo
6	SMT	Tennisball	36,00	Box	24 Dosen/Box
6	ANZ	Tennisball	48,00	Box	24 Dosen/Box
7	HRO	Basketball	600,00	Box	24/Box
8	ANZ	Volleyball	840,00	Box	24/Box
9	ANZ	Volleyb. profi	20,00	solo	solo

Tabelle hersteller

herstellercode	herstellername
SMT	Schmitt
ANZ	Anzabel
NRG	Norgesta
HSK	Hasken
HRO	Herold

Tabelle bundesland

bcode	laendername
BW	Baden-Wuerttemberg
BY	Bayern
BE	Berlin (West)
BR	Bremen
HA	Hamburg
HE	Hessen
NI	Niedersachsen
NW	Nordrhein-Westfalen
RP	Rheinland-Pfalz
SA	Saarland
SH	Schleswig-Holstein

Formatprogramme

auftrag

```
{ Datei: auftrag.per }

database
        versand

screen
{
--------------------------------------------------------------------------------
Kundennr: [k1          ]                    Telefon:  [k10           ]
   Firma: [k4                  ]
 Vorname: [k2              ]               Nachname:  [k3             ]
 Adresse: [k5                  ]                      [k6                  ]
     Plz: [k7    ] Ort: [k8             ] Bundesland: [k9]
--------------------------------------------------------------------------------
Auftragsnummer: [au11         ]        Auftragsdatum: [au12       ]
 Fremde Nummer: [au20       ]
         offen: [a]
   Lieferdatum: [au21       ]             Zahldatum: [au22       ]
--------------------------------------------------------------------------------
       Artikel: [p13]                          Preis: [preis ]
Herstellercode: [p16]                      x  Menge: [p18]
  Beschreibung: [bezeichnung     ]            ----------------------
       Einheit: [einh]                       =  Total: [p19       ]
    Hersteller: [hersteller      ]
   Stueckliste: [stueckliste     ] Auftr.-Gesamtpreis: [dtot        ]
}
end

tables
        kunde
        auftrag
        posten
        artikel
        hersteller

attributes

k1  = * kunde.kunden_nr = auftrag.kunden_nr, queryclear;
k2  = vorname, comments = "Bei Bedarf Vornamen eingeben";
k3  = nachname;
k4  = firma, reverse;
k5  = adresse1;
k6  = adresse2;
k7  = plz;
k8  = ort;
k9  = bundesland, upshift, autonext,
      include = ("BY","HA","BW","BE"), default = "BY",
      comments = "Erlaubte Bundeslaender sind BY, HA, BW, or BE";
k10 = telefon, picture = "######/########";
```

```
au11 = * auftrag.auftrags_nr = posten.auftrags_nr, queryclear;
au12 = auftragsdatum, default = today;
au20 = fremd_nr;
a    = offen, downshift, default = "n", include = ("j","n");
au21 = lieferdatum;
au22 = zahldatum;

p13 = posten.artikel_nr, queryclear, include = (1 to 9),
      comments = "Artikelnummer zwischen 1 und 9";
    = * artikel.artikel_nr, noentry, noupdate, queryclear;
p16 = posten.herstellercode,
         lookup hersteller = hersteller.herstellername,
         joining * hersteller.herstellercode, upshift, queryclear,
         comments = "Codes sind HRO, HSK, SMT, NRG, SMT und ANZ";
    = * artikel.herstellercode, noentry, noupdate, queryclear;
p18 = menge, include = (1 to 100);
p19 = posten.gesamtpreis;

bezeichnung = artikel.bezeichnung, noentry, noupdate;
einh        = artikel.liefereinheit, noentry, noupdate;
stueckliste = artikel.stueckliste,  noentry, noupdate;
preis       = artikel.preis,  noentry, noupdate;

dtot        = displayonly type money;

instructions

kunde master of auftrag;
auftrag master of posten;

composites <posten.artikel_nr, posten.herstellercode>
          * <artikel.artikel_nr, artikel.herstellercode>

before editadd editupdate of posten.menge
        if preis is null
          then begin
            comments bell "Artikel und Hersteller passen nicht zusammen"
            nextfield = p13
        end

after editadd editupdate of posten.menge
        let p19  = preis * p18
        nextfield = p13

after add update query of posten
        let dtot = (total of p19)

end
```

FORMAT01

```
database versand
screen
{

                                FORMAT01
                                --------

Kundennummer [f000       ]


Firma        [f001                ]


Vorname      [f002           ]       Nachname            [f003           ]
Strasse      [f004                ]  evtl. Zweitanschrift [f005               ]
Plz          [f006 ]                 Ort                 [f007           ]


Telefon      [f008              ]    Bundesland          [a0]
}
end
tables
kunde
attributes
f000 = kunde.kunden_nr, reverse;
f001 = kunde.firma;
f002 = kunde.vorname;
f003 = kunde.nachname;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.plz;
f007 = kunde.ort;
f008 = kunde.telefon;
a0 = kunde.bundesland;
end
```

FORMAT02

```
database versand
screen
{

                        FORMAT02 - Bildschirmseite 1
                        ----------------------------


     Kundennummer       [f000       ]


     Firma              [f001                ]

     Vorname            [f002          ]
     Nachname           [f003          ]



}
screen
{
```

```
                          FORMAT02 - Bildschirmseite 2
                          ----------------------------


      Firma               [f001                 ]


      Anschrift           [f004                 ]
      evt. Zweitanschrift [f005                 ]

      Plz                 [f006 ]
      Ort                 [f007           ]
      Bundesland          [a0]

      Telefon             [f008              ]

}
end
tables
kunde
attributes
f000 = kunde.kunden_nr, reverse;
f001 = kunde.firma;
f002 = kunde.vorname;
f003 = kunde.nachname;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.plz;
f007 = kunde.ort;
a0 = kunde.bundesland;
f008 = kunde.telefon;
end
```

FORMAT03

```
database versand
screen
{
                    FORMAT03 - Tabellen 'kunde' und 'auftrag'
--kunde-------------------------------------------------------------------------

Firma     [f001                 ]
Vorname   [f002            ]          Nachname             [f003           ]
Adresse   [f004                 ]     evtl. Zweitanschrift [f005                ]
Plz       [f006 ]                     Ort                  [f007           ]

Telefon   [f008               ]       Bundesland           [a0]

--auftrag------------- Kundennummer [f000       ] ------------------------------

Auftragsnummer      [f009       ]     Auftragsdatum       [f010      ]
Lieferhinweis       [f011                                        ]
Offen?              [a]               Fremdnummer         [f012      ]
Lieferdatum         [f013       ]     Liefergewicht       [f014      ]
Zustellgebuehr      [f015     ]       Zahldatum           [f016      ]
}
end
tables
kunde
auftrag
attributes
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr, reverse;
f001 = kunde.firma;
f002 = kunde.vorname;
f003 = kunde.nachname;
f004 = kunde.adresse1;
f005 = kunde.adresse2;
f006 = kunde.plz;
f007 = kunde.ort;
a0   = kunde.bundesland;
f008 = kunde.telefon;

f009 = auftrag.auftrags_nr;
f010 = auftrag.auftragsdatum;
f011 = auftrag.lieferhinweis;
a    = auftrag.offen;
f012 = auftrag.fremd_nr;
f013 = auftrag.lieferdatum;
f014 = auftrag.liefergewicht;
f015 = auftrag.zustellgebuehr;
f016 = auftrag.zahldatum;
instructions
kunde master of auftrag
end
```

FORMAT04

```
database versand
screen
{

--kunde-------------------------------------------------------------------------

Firma     [f001                ]   Adresse           [f004                ]
Plz       [f006 ]                  Ort               [f007           ]
                                   Bundesland        [a ]
--auftrag----------- Kundennummer [f000       ] ---------------------------------

Offen?              [b]            Auftragsdatum     [f010      ]
Lieferdatum         [f013      ]   Liefergewicht     [f014      ]
Zustellgebuehr      [f015    ]     Zahldatum         [f016      ]

--posten---------- Auftragsnummer [f009       ] ---------------------------------

Postennummer        [f017  ]       Herstellercode    [c ]
Artikelnummer       [f018  ]       Artikel           [lo01                ]
Menge               [f019  ]       Gesamtpreis       [f020      ]
}
end
tables
    kunde
    auftrag
    posten
    artikel
attributes
f000 = * kunde.kunden_nr
     =   auftrag.kunden_nr, reverse;
f001 = kunde.firma,
       comments = "Bei Neuaufnahme: falls 'GmbH', bitte mit eingeben!";
f004 = kunde.adresse1, required;
f006 = kunde.plz;
f007 = kunde.ort;
a    = kunde.bundesland, upshift, default="BY"
f009 = * auftrag.auftrags_nr
     =   posten.auftrags_nr, reverse;
b    = auftrag.offen,
       include = (j, n),
       comments = "'j' oder 'n' eingeben";
f010 = auftrag.auftragsdatum, default = today;
f013 = auftrag.lieferdatum, verify;
f014 = auftrag.liefergewicht;
f015 = auftrag.zustellgebuehr;
f016 = auftrag.zahldatum;
f017 = posten.posten_nr;
c    = posten.herstellercode;
f018 = posten.artikel_nr,
       lookup lo01 = artikel.bezeichnung
       joining artikel.artikel_nr;
```

```
f019 = posten.menge, right,
       include = (1 to 50),
       comments = "Mindestens 1, maximal 50";
f020 = posten.gesamtpreis;
instructions
kunde master of auftrag;
auftrag master of posten;
end
```

FORMAT05

```
database versand
screen
{
 \gp-------------------------------------------------------------q\g
 \g|\g                        FORMAT05                            \g|\g
 \g|\g                        --------                            \g|\g
 \g|-------------------------------------------------------------|\g
 \g|\g                                                            \g|\g
 \g|\g                                                            \g|\g
 \g|\g                                                            \g|\g
 \g|\g   Kundennummer         [f000       ]                       \g|\g
 \g|\g                                                            \g|\g
 \g|\g                                                            \g|\g
 \g|\g   Firma                [f001                 ]             \g|\g
 \g|\g                                                            \g|\g
 \g|\g   Vorname              [f002           ]                   \g|\g
 \g|\g   Nachname             [f003           ]                   \g|\g
 \g|\g                                                            \g|\g
 \gb-------------------------------------------------------------d\g


}
end
tables
k=kunde
attributes
f000 = k.kunden_nr, reverse;
f001 = k.firma;
f002 = k.vorname;
f003 = k.nachname;
end
```

muster

```
{ Datei: muster.per }
database versand
screen
{
================================================================================
================================================================================
KUNDENDATEN:

   Kundennummer: [k1         ]

         Firma: [k4                  ]
       Vorname: [k2        ]              Nachname: [k3             ]

       Adresse: [k5                  ]
                [k6                  ]

           Plz: [k7  ]   Ort: [k8             ]     Bundesland: [k9]
       Telefon: [k10          ]


================================================================================
================================================================================
}
screen
{
================================================================================
KUNDENNUMMER: [k1         ]           FIRMA: [k4                  ]
AUFTRAGSDATEN:
 Auftragsnummer: [au11        ]    Auftragsdatum: [au12         ]
  Artikelnummer: [p13        ]        Hersteller: [p16]
   Beschreibung: [ar14           ]                [m17           ]
    Stueckliste: [ar16           ]
                                             Menge: [p18        ]
                                             Preis: [ar15             ]
                                       Gesamtpreis: [p19              ]

LIEFERHINWEIS:
  Fremde Nummer: [au20      ]      Liefergebuehr: [d1        ]

          offen: [a]          Auftragsgesamtsumme: [d2        ]
    Lieferdatum: [au21       ]
      Zahldatum: [au22       ]
  Lieferhinweis: [au23                                      ]
}
end
tables
kunde posten artikel
auftrag hersteller
attributes

k1   = * kunde.kunden_nr
         = auftrag.kunden_nr;
k2   = vorname,
       comments = "Bei Bedarf Vornamen eingeben";
k3   = nachname;
k4   = firma, reverse;
k5   = adresse1;
```

```
k6   = adresse2;
k7   = plz, autonext;
k8   = ort;
k9   = bundesland, upshift, autonext,
       include = ("HH","BY","BW","BE"),
       default = "BY" ,
       comments = "Eingabe erlaubt:'HH','BY','BW','BE'- Standardwert: 'BY'";
k10  = telefon, picture = "######/########";
au11 = * auftrag.auftrags_nr
         = posten.auftrags_nr;
au12 = auftragsdatum, default = today, format = "dd.mm.yy";
p13  = posten.artikel_nr;
       = * artikel.artikel_nr, noentry, noupdate, queryclear;
p16  =  posten.herstellercode,
           lookup m17 = hersteller.herstellername
           joining *hersteller.herstellercode, upshift, autonext;
        = * artikel.herstellercode, noentry, noupdate,
            upshift, autonext, queryclear;
ar14 = artikel.bezeichnung, noentry, noupdate;
ar15 = artikel.preis, noentry, noupdate;
ar16 = artikel.stueckliste, noentry, noupdate;
p18  = posten.menge, include = (1 to 50),
       comments = "Menge zwischen 1 und 50" ;
p19  = posten.gesamtpreis;
au20 = fremd_nr, required,
       comments = "Falls keine fremde Nummer, Name des Anrufers" ;
a    = offen, autonext;
au21 = lieferdatum, default = today, format = "dd.mm.yy";
au22 = zahldatum, format = "dd.mm.yy";
au23 = lieferhinweis;
d1   = displayonly type money;
d2   = displayonly type money;

instructions
kunde master of auftrag;
auftrag master of posten;
composites <posten.artikel_nr, posten.herstellercode>
           *<artikel.artikel_nr, artikel.herstellercode>
before editadd editupdate of auftrag
   nextfield = au20
before editadd editupdate of posten
   nextfield = p13
after editadd editupdate of menge
   let p19 = p18 * ar15
   nextfield = au11
after add update query of posten
   if (total of p19) <= 100 then
      let d1 = 7.50
      else
      let d1 = (total of p19) * .04
   let d2 = (total of p19) + d1
after display of auftrag
   let d1 = 0
   let d2 = 0
end
```

Listenprogramme

auftrag1

```
{ Datei: auftrag1.ace - Quellprogramm fuer 1. Auftragsliste }

database
   versand
end

output
   page length 72                {Standardlaenge von 'lpr'     }
   report to "aufliste1"
end

select
   auftrag.auftrags_nr aufnr,
   auftragsdatum, kunden_nr,
   fremd_nr, lieferdatum, zustellgebuehr,
   zahldatum,
   posten.auftrags_nr, artikel_nr, herstellercode,
   menge, gesamtpreis
   from auftrag, posten
   where auftrag.auftrags_nr = posten.auftrags_nr
   order by aufnr
end

format

   before group of aufnr
   print "Auftragsnummer:  ", aufnr using "#####",
      " fuer Kundennummer:  ", kunden_nr using "#####"
   print 3 spaces, "Fremdnummer:  ", fremd_nr,
      "Auftragsdatum:  ", auftragsdatum
   skip 1 line
   print "Art.nr", column 20,
      "Herst.", column 28, "Mge", column 42, "Preis"
```

```
    on every row
    print artikel_nr using "###", column 20,
       herstellercode, column 28, menge using "###",
       column 38, gesamtpreis using "$$$,$$$.&&"
    after group of aufnr
    skip 1 line
    print 5 spaces, "Gesamtpreis des Auftrags:    ",
       group total of gesamtpreis using "$$,$$$,$$$.&&"
    skip 3 lines
end
```

auftrag2

```
{ Datei: auftrag2.ace - Quellprogramm fuer 2. Auftragsliste }

database
   versand
end

define
   variable startdatum date
   variable enddatum date
end

input
   prompt for startdatum using
      "Bitte geben Sie das Startdatum fuer die Liste ein:  "
   prompt for enddatum using
      "Bitte geben Sie das Enddatum fuer die Liste ein:    "
end

output
   left margin 0
   page length 72                {Standardlaenge von 'lpr'     }
   report to "aufliste3"
end

select
   kunde.kunden_nr, vorname, nachname, firma,

   auftrag.auftrags_nr aufnr, auftrag.kunden_nr,
      auftragsdatum, month(auftragsdatum) months,
      day(auftragsdatum) tage, year(auftragsdatum) jahre,

   posten.auftrags_nr, menge, gesamtpreis

   from  kunde, auftrag, posten

   where kunde.kunden_nr = auftrag.kunden_nr
      and auftrag.auftrags_nr = posten.auftrags_nr and
      auftragsdatum between $startdatum and $enddatum

   order by jahre, months, tage, firma, aufnr
end

format

   first page header
   print column 10,

      "=================================================="
```

```
   print column 10,
      "                    TAGEWEISE AUFTRAGSLISTE"
   print column 10,
      "=================================================="
   skip 1 line
   print column 15, "VOM:    ", startdatum
      using "dd/mm/yy",
      column 35, "BIS ZUM: ", enddatum
      using "dd/mm/yy"
   print column 15, "Liste vom:  ",
      today using "dd mmm, yyyy"
   skip 2 lines
   print column 2, "AUFTR. VOM", column 15,
      "FIRMA", column 35, "NAME",
      column 55, "NUMMER", column 66, "UMSATZ"

   before group of tage
   skip 2 lines

   after group of aufnr
   print column 2, auftragsdatum, column 15,
      firma clipped, column 35, vorname clipped,
      1 space, nachname clipped, column 55,
      aufnr using "####", column 60,
      group total of gesamtpreis using "$$,$$$,$$$.&&"

   after group of tage
   skip 1 line
   print column 21, "Auftragssumme fuer den Tag:  ",
      group total of gesamtpreis using "$$$$,$$$,$$$.&&"
   skip 1 line
   print column 15,
      "=================================================="

   on last row
   skip 1 line
   print column 15,
      "=================================================="
   skip 2 lines
   print "Gesamtauftragssumme:  ", total of
      gesamtpreis using "$$$$,$$$,$$$.&&"

   page trailer
   print column 28, pageno using "Seite <<vr1.vl1><<vr1.vl1><<vr1.vl1><"
end
```

kliste1

```
{ Datei: kliste1.ace - Quellprogramm fuer 1. Kundenliste }

database
   versand
end

output
   left margin 2
end

select
   kunden_nr,
   vorname,
   nachname,
   firma,
   ort,
   bundesland,
   plz,
   telefon

   from kunde

   order by ort
end

format
   first page header
   print column 33, "KUNDENLISTE"
   print column 33, "-----------"
   skip 2 lines
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
   skip 1 line

   page header
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
   skip 1 line

   on every row
   print kunden_nr using "####",
```

```
        column 9, vorname clipped, 1 space, nachname clipped,
        column 32, plz,
        column 40, ort clipped, ", " , bundesland,
        column 62, telefon

    on last row
    skip 1 line
    print  "ANZAHL DER KUNDEN:",
        column 30, count using "##"
end
```

kliste2

```
{ Datei: kliste2.ace - Quellprogramm fuer 2. Kundenliste }

database
   versand
end
define
   variable diesland char(2)
end
input
   prompt for diesland using
   "Bitte Bundesland der Kunden angeben (in GROSSBUCHSTABEN):
end
output
   left margin 0
end
select
   kunden_nr,
   vorname,
   nachname,
   firma,
   ort,
   bundesland,
   plz,
   telefon
   from kunde
   where bundesland matches $diesland
   order by plz, nachname
end
format
   first page header
   print column 33, "KUNDENLISTE"
   print column 33, "-----------"
   skip 2 lines
   print "Liste fuer Bundesland ", diesland
   skip 2 lines
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
   skip 1 line
   page header
   print "NUMMER",
      column 9, "NAME",
      column 32, "PLZ",
      column 40, "ORT",
      column 62, "TELEFON"
```

```
    skip 1 line
    on every row
    print kunden_nr using "####",
       column 9, vorname clipped, 1 space, nachname clipped,
       column 32, plz,
       column 40, ort clipped, ", " , bundesland,
       column 62, telefon
    on last row
    skip 2 lines
    print "Anzahl der Kunden in ",diesland, " ist ",
       count using "<<vr1.vl1><<vr1.vl1><<vr1.vl1><&"
end
```

LISTE01

```
{ Datei: LISTE01.ace }

database versand
end

define variable firmenname char(20)
end

input prompt for firmenname
using "Geben Sie den Namen der Firma ein: "
end

output report to printer
       page length 72          {Standardlaenge von 'lpr'}
end

select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
where
   firma = $firmenname
end

format

page header
print "K u n d e n d a t e n:"
skip 2 line
print "Kundennummer:", column 20, kunden_nr using "<<vr1.vl1><<vr1.vl1><"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
end
```

LISTE02

```
{ Datei: LISTE02.ace }

database versand
end

output report to printer
       page length 72           {Standardlaenge von 'lpr'}
end

select
   kunden_nr, firma, nachname, adresse1, plz, ort
from
   kunde
end

format

page header
print column 30, "Kunden-Liste"
skip 1 line
print "Stand: ", today
print "-------------------------------------------------------------"
skip 2 line

page trailer
print "-------------------------------------------------------------"
print "Kunden-Liste", column 54, "Seite: ", pageno using "##"

on every row
need 8 lines
print "K u n d e n d a t e n:"
skip 1 line
print "Kundennummer:", column 20, kunden_nr using "<<VR1.VL1><<VR1.VL1><"
print "Firma:",        column 20, firma
print "Nachname:",     column 20, nachname
print "Adresse:",      column 20, adresse1
print "Postleitzahl:", column 20, plz
print "Ort:",          column 20, ort
skip 2 line
end
```

LISTE03

```
{ Datei: LISTE03.ace }

database versand
end

output report to printer
        page length 72          {Standardlaenge von 'lpr'}
end

select
   nachname, firma, ort
from
   kunde
order by
   ort
end

format

page header
print column 18, "Kunden-Liste"
print column 18, "============"
skip 2 lines

before group of ort
print "STADT: ", ort
skip 1 line

on every row
print nachname, column 18, firma, column 40, ort
after group of ort
skip 2 lines
end
```

LISTE04

```
{ Datei: LISTE04.ace }

database versand
end

define variable prozent decimal(3,2)
end

input prompt for prozent
      using "Die Artikel sollen erhoeht werden um Prozent: "
end

output report to printer
       page length 72          {Standardlaenge von 'lpr'}
end

select
   herstellercode, bezeichnung, preis
from
   artikel
end

format

page header
print "PROZENTSATZ: ", prozent using "##.##"
let prozent = prozent / 100 + 1
skip 1 line
print      "HERSTELLER",
column 12, "BEZEICHNUNG",
column 32, "PREIS",
column 43, "PLUS",
column 49, "NEUPREIS"
skip 1 line

on every row
print herstellercode,
column 12, bezeichnung clipped,
column 30, preis using "#####.##",
column 40, preis * prozent - preis using "#####.##",
column 50, preis * prozent using "#####.##"
end
```

LISTE05

```
{ Datei: LISTE05.ace }

database versand end

output report to printer
        page length 72          {Standardlaenge von 'lpr'}
end

select
   kunde.kunden_nr knr, firma,
   auftrag.auftrags_nr anr
from
   kunde, auftrag
where
   kunde.kunden_nr = auftrag.kunden_nr
order by
   knr
end

format

first page header
print            "KUNDENNUMMER",
      column 20, "FIRMA",
      column 37, "AUFTRAGSNUMMER"
skip 2 lines

on every row
print column 10, knr using "###",
      column 20, firma clipped,
      column 47, anr using "####"

after group of knr
print "-------------------------------------------------"
print column 30, "ANZAHL AUFTRAEGE ",
      column 49,  group count using "##"
print "-------------------------------------------------"

on last row
skip 2 lines
print "================================================="
print column 23, "GESAMT-ANZAHL AUFTRAEGE ",
      column 49 , count using "##"
print "================================================="
end
```

post1

```
{ Datei: post1.ace - Quellprogramm fuer 1. Anschriftenliste

Dieses Listenprogramm gibt eine Anschriftenliste aus. Diese ist
nach den Postleitzahlen und den Nachnamen geordnet. Die PRINT-
Anweisung im FORMAT-Abschnitt erlaubt es, mehrere Felder in
einer Zeile auszugeben.
Wenn es fuer ein Feld in einer Anschrift keine Daten gibt, dann
stehen an der entsprechenden Stelle Leerzeichen.

Zwischen den Feldern einer Anschrift sind mehr Leerzeichen als
noetig. Im Listen-Quellprogramm 'post2.ace' werden solche
Leerzeichen abgeschnitten.
}

database
   versand
end

select *
   from kunde
   order by plz, nachname
end

format
   on every row
   print vorname, nachname
   print firma
   print adresse1
   print adresse2
   print plz, 2 spaces, ort, ", " , bundesland
   skip 2 lines
end
```

post2

```
{ Datei: post2.ace - Quellprogramm fuer 2. Anschriftenliste

Dieses Listenprogramm gibt wie 'post1.ace' eine Anschriftenliste
aus. Es ist im Vergleich zu diesem erweitert:

-  Die Raender und die Seitenlaenge werden abweichend vom
   Standard gesetzt und die Ausgabe erfolgt in eine
   Datei (OUTPUT-Abschnitt).

-  Im FORMAT-Abschnitt gibt es IF-Anweisungen und ueberfluessige
   Leerzeichen am Ende von Feldern werden abgeschnitten.
}

database
   versand
end
output
   top margin 0
   bottom margin 0
   left margin 0
   page length 9
   report to "anschriften2"
end
select
   vorname, nachname,
   firma,
   adresse1,
   adresse2,
   ort, bundesland, plz
   from kunde
   order by plz, nachname
end
format
   on every row
   if (ort is not null) and
      (bundesland is not null) then
   begin
      print vorname clipped, 1 space, nachname
      print firma
      print adresse1
      if (adresse2 is not null) then
         print adresse2
      print plz, 2 spaces, ort clipped,
         ", " , bundesland
      skip to top of page
   end
end
```

post3

```
{ Datei: post3.ace - Quellprogramm fuer 3. Anschriftenliste

Dieses Listenprogramm schreibt eine Anschriftenliste in einer,
zwei oder in drei Spalten in eine Datei. Es speichert jeweils
bis zu drei Anschriften in drei Zeichenketten; wenn diese ent-
sprechend gefuellt sind, gibt das Programm sie aus. Nach dem
Aufruf des Programms, legt der Benutzer die Anzahl der Spalten
fest.
}

database
   versand
end

define
   variable name     char(75)     {fuer Vor- und Nachnamen       }
   variable plobl    char(75)     {fuer PLZ, Ort, B.land         }
   variable z_kette1 char(80)     {Zeichenkette fuer 1. Zeile    }
   variable z_kette2 char(80)     {Zeichenkette fuer 2. Zeile    }
   variable z_kette3 char(80)     {Zeichenkette fuer 3. Zeile    }
   variable start    smallint     {Start- und Endposition einer }
   variable ende     smallint     {Adresse in Zeichenkette       }
   variable a_breite smallint     {Anschriftenbreite             }
   variable platz    smallint     {Anschriftenzwischenraum       }
   variable sp_anz   smallint     {Anzahl der Listenspalten      }
   variable i        smallint     {Zaehler fuer Listenspalte     }
end

input
   prompt for sp_anz using
   "Wieviel Spalten soll die Liste haben? [1-3] "
end

output
   top margin 0
   bottom margin 0
   left margin 0
   page length 72                 {Standardlaenge von 'lpr'      }
   report to "anschriften3"
end

select *
   from kunde

   order by plz
end
```

```
format
   first page header                {In diesem Abschnitt wird
                                     nichts ausgegeben. Das Pro-
                                     gramm versieht nur Variable
                                     mit Werten.                 }
   let i = 1                        {i initialisieren           }
   let a_breite = 72/sp_anz         {legt die Anschriftenbreite
                                     fest                        }
   let platz = 8/sp_anz             {legt den Platz zwischen zwei
                                     Anschriftenspalten fest     }

   on every row
   let name = vorname clipped, 1 space, nachname
   let plobl = plz, 2 spaces, ort clipped,
               ", ", bundesland

   let ende = (i * a_breite)        {Aktuelle Endposition be-   }
              + platz               {rechnen                    }
   let start = ende - a_breite      {Aktuelle Startposition be- }
                                     {rechnen                    }
   let z_kette1[start, ende]        {Namen an richtige Stelle in }
      = name                        {die Zeichenkette schreiben }
   let z_kette2[start, ende]        {Str. an richtige Stelle in }
      = adresse1                    {die Zeichenkette schreiben }
   let z_kette3[start, ende]        {Ort an richtige Stelle in  }
      = plobl                       {die Zeichenkette schreiben }
   if i = sp_anz then               {Falls alle Spalten in den  }
   begin                            {Zeichenketten, schreibe    }
      print z_kette1 clipped        {die Namen,                 }
      print z_kette2 clipped        {die Strassen und           }
      print z_kette3 clipped        {die Orte.                  }
      skip 1 line
      let z_kette1 = " "            {Zeichenketten mit Leerzeichen}
      let z_kette2 = " "            {belegen                    }
      let z_kette3 = " "
      let i  = 1                    {Aktuellen Spaltenzaehler fuer}
   end                              {Anschriften zuruecksetzen  }
   else
      let i = i + 1

   on last row
   if i > 1 then                    {Die uebriggebliebenen An-  }
   begin                            {schriften werden ausgegeben }
      print z_kette1 clipped
      print z_kette2 clipped
      print z_kette3 clipped
   end
end
```

Literatur

[1] INFORMIX (SINIX)
SQL
Beschreibung

[2] Betriebssystem SINIX
INFORMIX-SQL
Nachschlagen

[3] Betriebssystem SINIX
Fehlermeldungen für
INFORMIX-Produkte

[4] Betriebssystem SINIX
INFORMIX-ONLINE
Datenbank-Server
Administrator-Handbuch

[5] Betriebssystem SINIX
Buch 2
Menüs

Bestellen von Handbüchern

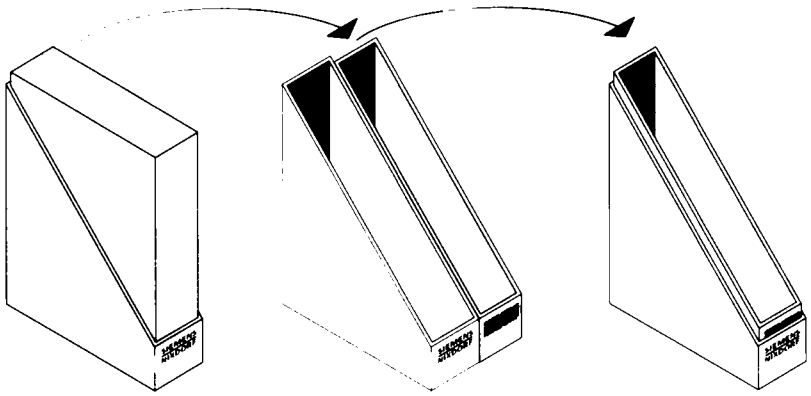
Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriftenverzeichnis Datentechnik*. Dort ist auch der Bestellvorgang erklärt. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen Datentechnik*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an eine Geschäftsstelle unseres Hauses.

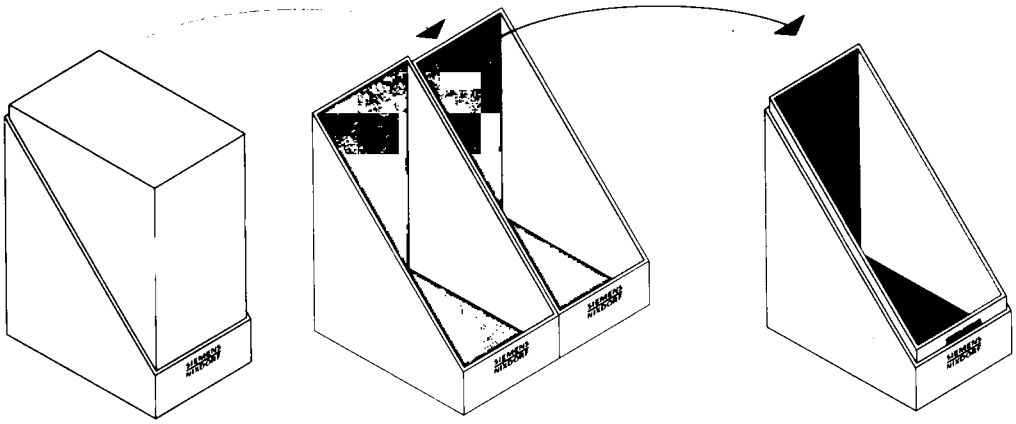
Stichwörter

Sammelboxen

Für Handbücher des vorliegenden Formates bieten wir zweiteilige Sammelboxen in zweierlei Größen an. Der Bestellvorgang entspricht dem für Handbücher.



Breite: ca. 5 cm
Bestellnummer: U3775-J-Z18-1



Breite: ca. 10 cm
Bestellnummer: U3776-J-Z18-1

92 7236

Herausgegeben von/Published by
Siemens Nixdorf Informationssysteme AG
Postfach 21 60, W-4790 Paderborn
Postfach 83 09 51, W-8000 München 83

Bestell-Nr./Order No. ***U3512-J-Z95-2***
Printed in the Federal Republic of Germany
8230 AG 11905 (10290)