Grundlegende Begriffe

Tabelle

Spalte-1	Spalte-2	Spalte-3	Spalte-4

Eine Tabelle besteht aus 1 bis n Spalten und 0 bis n Zeilen.

Alternative Begriffe:

Tabelle	Spalte	Zeile
Table	Column	Row, Record
	Feld	Satz, Datensatz

Eine Tabellenspalte definiert sich durch Eigenschaften:
- Name
- Datentyp
- Maximale Gesamtlänge
- Maximale Anzahl Nachkommastellen
- Inhalt erforderlich (Ja/Nein)
- Besondere Prüfbedingungen

Name

Jede Spalte muss durch einen eindeutigen Namen gekennzeichnet sein. Ein Spaltenname darf innerhalb einer Tabelle nur genau einmal vorkommen. Die einzelnen Spalten müssen über den Namen eindeutig identifizierbar sein. Aus Kompatibilitätsgründen sollten Sie bei der Namensvergabe folgende Regeln beachten:

- Das erste Zeichen ist ein Buchstabe.
- Alle weiteren Zeichen sind Buchstaben (A..Z ohne Sonderzeichen) oder Ziffern (0..9)
- Die Gesamtlänge des Namens darf 10 Stellen nicht überschreiten.

Groß- und Kleinbuchstaben in Spaltennamen werden als gleich interpretiert. Der Spaltenname TELEFON ist z.B. mit dem Spaltennamen Telefon identisch.

Datentyp

Je nach Art der aufzunehmenden Information werden unterschiedliche Datentypen benutzt.

Datentyp	Beispiele
Beliebige Zeichen	Namen, Bezeichnungen, Mehrzeilige Texte, Festgelegte Kennzeichen etc.
Numerische Inhalte	Preise, Mengen, Werte, Fortlaufende Nummern, Zeitdauer in Minuten oder Sekunden etc.
Datum und Uhrzeit	Jahr, Monat, Tag, Stunde, Minute, Sekunde
Große binäre Daten	Bilder, Photos, Zeichnungen etc.

Leider werden von Datenbanksystem zu Datenbanksystem häufig unterschiedliche Datentypen verwendet.
Ein hohes Maß an Kompatibilität erreichen wir durch eine Selbstbeschränkung auf folgende Datentypen:

Datenbanksystem	Beliebige Zeichen	Numerische Inhalte	Datum und Uhrzeit	Große binäre Daten
ORACLE	VARCHAR2	NUMBER	DATE	BLOB
INTERBASE	VARCHAR	DECIMAL	TIMESTAMP	BLOB
SYBASE	VARCHAR	DECIMAL	TIMESTAMP	LONG BINARY

Maximale Gesamtlänge und maximale Anzahl der Nachkommastellen

Abhängig vom Datentyp sind zusätzliche Angaben erforderlich.

Datentyp	Maximale Gesamtlänge	Maximale Anzahl Nachkommastellen
VARCHAR / VARCHAR2	1..4000	-/-
DECIMAL / NUMBER	1..14	0..4
TIMESTAMP / DATE	-/-	-/-
BLOB / LONG BINARY	-/-	-/-

Bei Daten vom Typ DECIMAL bzw. NUMBER ist die maximale Anzahl der Nachkommastellen in der maximalen Gesamtlänge enthalten. Werte dieses Typs können negative und positive Wertebereiche umfassen. Bei der Berechnung der Längenangaben ist das Vorzeichen bzw. das Dezimalzeichen nicht relevant.

Längenangaben erfolgen in runden Klammern direkt hinter der Angabe des Datentyps. Bei numerischen Werten wird die Angabe der Nachkommastellen von der Angabe der Gesamtlänge durch ein Komma getrennt.

Beispiele:

Wertebereich	ORACLE	INTERBASE / SYBASE
Name mit max. 40 Zeichen	VARCHAR2(40)	VARCHAR(40)
-999 bis +999	NUMBER(3,0)	DECIMAL(3,0)
-99,99 bis 99,99	NUMBER(4,2)	DECIMAL(4,2)
-9,9999 bis 9,9999	NUMBER(5,4)	DECIMAL(5,4)

Viele Datenbanksysteme bieten einen erhöhten Wertebereich für den numerischen Datentyp DECIMAL bzw. NUMBER. Bei einer Beschränkung auf maximal 14 Stellen, davon maximal 4 Nachkommastellen, ist eine Austauschbarkeit mit anderen Datenbanksystemen äußerst wahrscheinlich.

In ORACLE ist die maximale Gesamtlänge des Datentyps VARCHAR2 auf 4000 Zeichen beschränkt.
Aus Gründen der Kompatibilität bzw. Übertragbarkeit sollte diese Grenze generell, d.h. auch bei Verwendung anderer Datenbanksysteme, eingehalten werden.

Moderne Datenbanksysteme speichern Daten des Typs VARCHAR bzw. VARCHAR2 nur in der tatsächlich benötigten Länge und belegen damit nur den unbedingt erforderlichen Speicherplatz auf der Festplatte.

Für Daten des Typs DECIMAL bzw. NUMBER werden, abhängig von der maximalen Gesamtlänge, entweder 1, 2, 4 oder 8 Byte Speicherplatz benötigt.

Zur Speicherung von Daten des Typs TIMESTAMP bzw. DATE werden regelmäßig 8 Byte reserviert.

Die Länge von großen binären Daten (BLOB bzw. LONG BINARY) kann häufig mehrere Gigabyte umfassen.

Übungsaufgabe

Erstellen Sie eine Tabelle zur Speicherung von Vereinsmitgliedern.

Erforderliche Daten:
- Vorname (max. 40 Stellen)
- Familienname (max. 40 Stellen)
- Geburtsdatum
- Jahresbeitrag (max. 999,99 €)

Stellen Sie das Ergebnis in Form einer Tabelle dar:

Spaltenname	Spaltendefinition ORACLE	Spaltendefinition INTERBASE / SYBASE

Musterlösung

Spaltenname	Spaltendefinition ORACLE	Spaltendefinition INTERBASE / SYBASE
VORNAME	VARCHAR2(40)	VARCHAR(40)
FAMNAME	VARCHAR2(40)	VARCHAR(40)
GEBDATUM	DATE	TIMESTAMP
BEITRAG	NUMBER(5,2)	DECIMAL(5,2)

Tabelle mit Beispieldaten

VORNAME	FAMNAME	GEBDATUM	BEITRAG
Walter	Meier	18.09.1960 00:00:00	120,75
Jutta	Schulze	20.05.1968 00:00:00	120,75
Thorsten	Kleine	05.01.1988 00:00:00	80,5

Sieht doch schon ziemlich gut aus.

Eine wichtige Anforderung an Tabellen haben wir zwar bereits zufällig erfüllt, aber bisher nicht bewusst berücksichtigt.

Alle Sätze innerhalb einer Tabelle müssen aufgrund ihrer Inhalte eindeutig identifizierbar sein.

Vorausgesetzt, Thorsten Kleine hat eine Zwillingsschwester, dann könnte die Tabelle folgenden Inhalt aufweisen:

VORNAME	FAMNAME	GEBDATUM	BEITRAG
Walter	Meier	18.09.1960 00:00:00	120,75
Jutta	Schulze	20.05.1968 00:00:00	120,75
Thorsten	Kleine	05.01.1988 00:00:00	80,5
Monika	Kleine	05.01.1988 00:00:00	80,5

Auch hier ist die oben genannte Forderung erfüllt, da sich die Datensätze für Thorsten Kleine und seine Zwillingsschwester Monika zumindest im Vornamen unterscheiden.

Es könnte aber auch sein, dass es irgendwo auf dieser Welt einen zweiten Walter Meier oder eine zweite Jutta Schulze mit identischem Geburtsdatum gibt, die dann auch noch, zu allem Überfluss, Mitglied in unserem Verein werden möchten.

VORNAME	FAMNAME	GEBDATUM	BEITRAG
Walter	Meier	18.09.1960 00:00:00	120,75
Walter	Meier	18.09.1960 00:00:00	120,75
Jutta	Schulze	20.05.1968 00:00:00	120,75
Jutta	Schulze	20.05.1968 00:00:00	120,75
Thorsten	Kleine	05.01.1988 00:00:00	80,5
Monika	Kleine	05.01.1988 00:00:00	80,5

Wenn jetzt z.B. eine neue Spalte eingefügt werden soll, aus der hervorgeht, ob der Beitrag für das aktuelle Jahr gezahlt wurde oder nicht, dann haben wir ein Problem! Wie interpretieren wir dann die nachfolgende Aussage?

Walter Meier, geboren am 18.09.1960, hat seinen Jahresbeitrag von 120,75 € gezahlt.

Wenn wir selbst schon nicht in der Lage sind, zu bestimmen, um welchen Walter Meier es sich handelt, wie soll es dann eine Maschine bzw. ein Computer können? Wir müssen also dafür sorgen, dass solche Konstellationen ausgeschlossen sind.

Deshalb muss für jede Tabelle ein **Primärschlüssel** definiert werden. Ein Primärschlüssel kann aus einer einzigen Spalte bestehen oder auch aus mehreren Spalten zusammengesetzt werden. Das Datenbanksystem sorgt dann dafür, dass doppelte Sätze mit identischen Primärschlüsselinhalten nicht vorkommen können.

In unserem Fall bietet sich eine Mitgliedsnummer als Primärschlüssel an. Eine solche fortlaufende Nummer kann von den meisten Datenbanksystemen übrigens automatisch vergeben werden.

Zusätzlich zum Primärschlüssel erweitern wir unsere Tabelle um das Zahlungskennzeichen (J=Ja,N=Nein) und um eine Angabe zum biologischen Geschlecht (W=Weiblich,M=Männlich).

Tabellendefinition mit Primärschlüssel (hervorgehoben) sowie Geschlechtsangabe und Zahlungskennzeichen

Spaltenname	Spaltendefinition ORACLE	Spaltendefinition INTERBASE / SYBASE
MITGLIED	NUMBER(4,0)	DECIMAL(4,0)
GESCHL	VARCHAR2(1)	VARCHAR(1)
VORNAME	VARCHAR2(40)	VARCHAR(40)
FAMNAME	VARCHAR2(40)	VARCHAR(40)
GEBDATUM	DATE	TIMESTAMP
BEITRAG	NUMBER(5,2)	DECIMAL(5,2)
GEZAHLT	VARCHAR2(1)	VARCHAR(1)

Tabelle mit Beispieldaten

MITGLIED	GESCHL	VORNAME	FAMNAME	GEBDATUM	BEITRAG	GEZAHLT
1	M	Walter	Meier	18.09.1960 00:00:00	120,75	J
2	M	Walter	Meier	18.09.1960 00:00:00	120,75	N
3	W	Jutta	Schulze	20.05.1968 00:00:00	120,75	J
4	W	Jutta	Schulze	20.05.1968 00:00:00	120,75	J
5	M	Thorsten	Kleine	05.01.1988 00:00:00	80,5	J
6	W	Monika	Kleine	05.01.1988 00:00:00	80,5	J

Eine Aussage über die Zahlungsmoral unserer Mitglieder ist jetzt eindeutig möglich.

Das Vereinsmitglied Walter Meier mit der Mitgliedsnummer 1 hat gezahlt.
Das Vereinsmitglied Walter Meier mit der Mitgliedsnummer 2 hat nicht gezahlt.

Das Geschlechtskennzeichen (M/W) kann die eine oder andere Verunsicherung beseitigen.
Die folgende Frage kann dann ebenso eindeutig beantwortet werden.

Ist Andrea Benelli weiblich oder männlich?

Inhalt erforderlich (Ja/Nein)

Gehen wir zunächst davon aus, dass alle Spalten einen Inhalt aufweisen müssen. Wenn das der Fall ist, muss die Spaltendefinition um einen Zusatz **NOT NULL** erweitert werden. Die Angabe **NOT NULL** ist für alle Spalten des Primärschlüssels ohnehin unbedingt erforderlich.

NULL ist ein spezieller Wert in Datenbanksystemen, der anzeigt, ob eine Spalte einen Inhalt hat oder nicht.

Der numerische Wert 0 oder auch eine leere Zeichenfolge '' sind nicht mit dem Wert **NULL** identisch.
Dazu später mehr.

Besondere Prüfbedingungen

Den Wertebereich der einzelnen Spalten haben wir durch die Angabe des Datentyps und der maximalen Gesamtlänge sowie der maximal möglichen Nachkommastellen bereits festgelegt. Der Wertebereich der Spalte BEITRAG erlaubt aber auch die Angabe von negativen Beträgen. Außerdem ist es möglich, in den Spalten GESCHL und GEZAHLT ein beliebiges Zeichen einzugeben.

Durch die Angabe einer **CHECK**-Bedingung kann der mögliche Wertebereich zusätzlich eingeschränkt werden.
Dazu später mehr.

Jetzt geht's zur Sache

Eine Datenbank enthält normalerweise mehrere Tabellen. Damit diese Tabellen eindeutig unterscheidbar sind, muss jede Tabelle mit einem Namen versehen werden. Die Konventionen für Tabellennamen sind mit denen für Spaltennamen identisch. Ich selbst beschränke mich generell auf feste dreistellige Namen, die in unserer Datenbankanwendung entsprechend strukturiert interpretiert und behandelt werden. Für unsere erste Tabelle verwenden wir den Namen VMG (Vereinsmitglieder).

Datenbanksysteme verwenden eine festgelegte Kommandosprache, mit der alle Operationen, nach einem für das Datenbanksystem verständlichen Schema, formuliert werden.

Die Syntax zum Anlegen einer Tabelle ist wie folgt definiert:

```
CREATE TABLE tblname (colname datatype[(maxlen[,decimals])] [NOT NULL] [CHECK (…)],
                      colname datatype[(maxlen[,decimals])] [NOT NULL] [CHECK (…)],
                      …
                      PRIMARY KEY (colname[,colname,…]));
```

Die Elemente in Kleinbuchstaben sind durch die tatsächlich benötigten Angaben zu ersetzen. Alle Angaben in eckigen Klammern sind optional und können, teilweise allerdings abhängig vom Datentyp, unterbleiben.

Unsere Tabelle der Vereinsmitglieder legen wir in ORACLE mit folgendem Kommando an:

```
CREATE TABLE VMG (MITGLIED NUMBER(4,0) NOT NULL,
                  GESCHL   VARCHAR2(1) NOT NULL,
                  VORNAME  VARCHAR2(40) NOT NULL,
                  FAMNAME  VARCHAR2(40) NOT NULL,
                  GEBDATUM DATE NOT NULL,
                  BEITRAG  NUMBER(5,2) NOT NULL,
                  GEZAHLT  VARCHAR2(1) NOT NULL,
                  PRIMARY KEY (MITGLIED));
```

Das Kommando ist, abgesehen von den Datentypen, in INTERBASE bzw. SYBASE identisch.

```
CREATE TABLE VMG (MITGLIED DECIMAL(4,0) NOT NULL,
                  GESCHL   VARCHAR(1) NOT NULL,
                  VORNAME  VARCHAR(40) NOT NULL,
                  FAMNAME  VARCHAR(40) NOT NULL,
                  GEBDATUM TIMESTAMP NOT NULL,
                  BEITRAG  DECIMAL(5,2) NOT NULL,
                  GEZAHLT  VARCHAR(1) NOT NULL,
                  PRIMARY KEY (MITGLIED));
```

Anmerkung: Mit diesem Kommando wird zunächst nur die Struktur der Tabelle festgelegt.
Datensätze sind nach Ausführung dieses Kommandos noch nicht vorhanden.
Anweisungen werden generell mit einem Semikolon abgeschlossen.

Eine Tabelle kann mit folgendem Kommando (Syntax) wieder gelöscht werden:

```
DROP TABLE tblname;
```

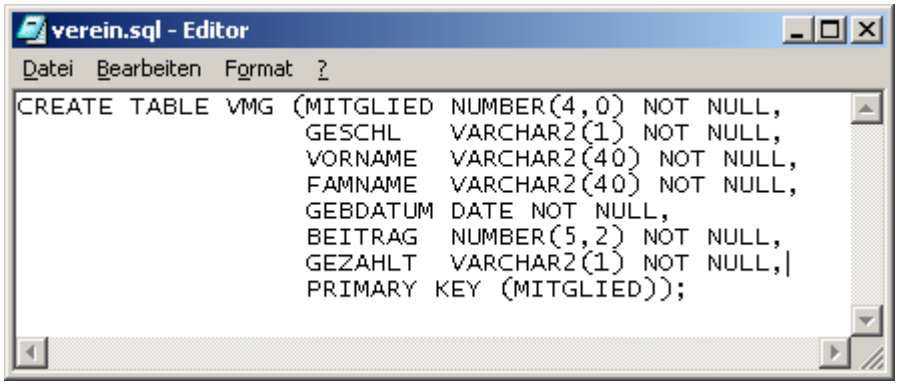
Unsere Tabelle VMG kann also mit dem Kommando

```
DROP TABLE VMG;
```

aus der Datenbank entfernt bzw. gelöscht werden.

Wichtiger Hinweis

Die erforderlichen Anweisungen zur Generierung einer Tabellenstruktur sollten in einer Textdatei (hier: verein.sql) gespeichert werden. Der Inhalt dieser Datei wird dann vom Datenbanksystem zeilenweise abgearbeitet.



Erste Schritte mit ORACLE

Wir aktivieren das Oracle-Dienstprogramm Sql-Plus und melden uns als Benutzer beim Datenbanksystem an.

Eine Scriptdatei wird mit folgender Kommandosyntax ausgeführt:

```
START dateiname;
```

Hier z.B.:

```
START C:\TEMP\VEREIN.SQL;
```

Groß- bzw. Kleinbuchstaben in der Kommandosyntax werden von den Datenbanksystemen ignoriert.
Die nachfolgende Anweisung wird also vom Datenbanksystem als identisch mit der obigen betrachtet.

```
Start C:\Temp\Verein.sql;
```

Das Datenbanksystem verarbeitet die Anweisungen zeilenweise. Nach Abarbeitung der gesamten Scriptdatei (hier nur ein Kommando) sollten Sie eine Mitteilung erhalten, dass Ihre Tabelle erfolgreich angelegt wurde.

Sehen wir uns einmal an, was zum jetzigen Zeitpunkt in der Tabelle gespeichert ist. Dazu benutzen wir die Anweisung **SELECT**. Die Syntax in der einfachsten Form der Select-Anweisung lautet:

```
SELECT * FROM tblname;
```

In unserem Fall also:

```
SELECT * FROM VMG;
```

Wie bereits vorab erläutert, wird mit der Anweisung **CREATE TABLE** nur die Struktur der Tabelle festgelegt. Deshalb sind zum jetzigen Zeitpunkt keine Datensätze in der Tabelle gespeichert.

Neue Datensätze werden mit dem Kommando **INSERT** in die Tabelle eingefügt. Die Syntax lautet:

```
INSERT INTO tblname VALUES(coldata[,coldata,…]);
```

Die Inhalte sämtlicher Spalten sind in der Reihenfolge gemäß der Anweisung CREATE TABLE,
durch Komma getrennt, anzugeben.

Dabei werden die notwendigen Konstanten, abhängig vom Datentyp, wie folgt formuliert:

VARCHAR/VARCHAR2

Daten dieses Typs werden beidseitig durch einfache Hochkomma begrenzt.
z.B.: 'Fritz' oder 'Zwei Worte'

DECIMAL/NUMBER

Numerische Daten werden durch Ziffern, einen evtl. Dezimalpunkt und/oder
ein evtl. linksbündig angeordnetes, negatives Vorzeichen repräsentiert.
z.B.: 12 oder 1.5 oder –0.5 oder –20

TIMESTAMP / DATE

Leider werden konstante Datum- und Zeitangaben in den verschiedenen Datenbanksystemen unterschiedlich gehandhabt:

In ORACLE umgehen wir die vorhandenen Abhängigkeiten von verschiedenen Konfigurationseinstellungen durch den Einsatz der Konvertierungsfunktion **TO_DATE**.

Beispiele:

Datum	TO_DATE('15.03.2006','DD.MM.YYYY')
Datum und Uhrzeit	TO_DATE('15.03.2006 17:38:45','DD.MM.YYYY HH24:MI:SS')

In INTERBASE werden folgende Standardformate verwendet:

Datum	'DD-MON-YYYY'
Datum und Uhrzeit	'DD-MON-YYYY HH24:MI:SS'

MON ist eine dreistellige englischsprachige Abkürzung der Monatsnamen:
JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC

Beispiele: '15-MAR-2006' oder '15-MAR-2006 17:38:45'

In SYBASE-Datenbanken sind Datumsangaben gemäß der Spezifikation in den Ländereinstellungen des Betriebssystems vorzunehmen. Ein generell verwendbares Format ist nicht vorhanden.

BLOB / LONG BINARY

Daten dieses Typs können nicht als Konstanten angegeben werden.

Einfügen des ersten Datensatzes

Oracle:

Zunächst setzen wir einen notwendigen Konfigurationsparameter (mehr dazu später):

```
SET AUTOCOMMIT OFF;
```

danach:

```
INSERT INTO VMG VALUES(1,'M','Walter','Meier',TO_DATE('18.09.1960','DD.MM.YYYY'),120.75,'J');
```

Interbase:

```
INSERT INTO VMG VALUES(1,'M','Walter','Meier','18-SEP-1960',120.75,'J');
```

Zur Kontrolle:

```
SELECT * FROM VMG;
```

Anmerkungen Oracle Sql-Plus: Die Spaltennamen in den Überschriften werden bei der Ausgabe teilweise abgeschnitten. Dies ist ausschließlich ein Darstellungsproblem in Sql-Plus. Zeilenumbrüche in der Ausgabe können mit folgenden Kommando wirksam verhindert werden:

```
SET LINESIZE 1024;
```

Beenden einer Transaktion

Modifikationen durch Einfüge-, Änderungs- oder Löschaktionen werden erst nach Ausführung des Kommandos **COMMIT** permanent in der Datenbank gespeichert.

Mit dem Kommando **ROLLBACK** können alle Einfüge-, Änderungs- oder Löschaktionen seit dem letzten COMMIT bzw. ROLLBACK zurückgesetzt werden.

Da wir den ersten eingefügten Datensatz permanent in der Datenbank speichern möchten, setzen wir das nachfolgende Kommando ab:

```
COMMIT;
```

Übungsaufgabe

Erzeugen Sie eine neue Scriptdatei unter dem Namen Mitglieder.sql und fügen Sie die fehlenden Insert-Anweisungen für die Vereinsmitglieder 2 bis 6 ein. Im Anschluss an die Einfügeoperationen setzen Sie das Kommando ROLLBACK ein und überprüfen mittels o.g. SELECT-Anweisung den aktuellen Datenbestand in der Tabelle VMG. Führen Sie anschließend die Scriptdatei in Sql-Plus aus.

Musterlösung

Datei Mitglieder.sql

```
INSERT INTO VMG VALUES(2,'M','Walter','Meier',TO_DATE('18.09.1960','DD.MM.YYYY'),120.75,'N');
INSERT INTO VMG VALUES(3,'W','Jutta','Schulze',TO_DATE('20.05.1968','DD.MM.YYYY'),120.75,'J');
INSERT INTO VMG VALUES(4,'W','Jutta','Schulze',TO_DATE('20.05.1968','DD.MM.YYYY'),120.75,'J');
INSERT INTO VMG VALUES(5,'M','Thorsten','Kleine',TO_DATE('05.01.1988','DD.MM.YYYY'),80.5,'J');
INSERT INTO VMG VALUES(6,'W','Monika','Kleine',TO_DATE('05.01.1988','DD.MM.YYYY'),80.5,'J');
ROLLBACK;
SELECT * FROM VMG;
```

Ausführen Scriptdatei

```
START C:\TEMP\MITGLIEDER.SQL;
```

Oops! Was ist aus unseren Insert-Anweisungen geworden?

Alle Änderungen nach dem letzten **COMMIT** wurden durch die Anweisung **ROLLBACK** zurückgesetzt.
Wir sehen also nur den bereits per COMMIT permanent gespeicherten Satz mit der Mitgliedsnummer 1.

Ersetzen wir die Anweisung ROLLBACK in der Scriptdatei durch COMMIT und führen das Script erneut aus.

Datei Mitglieder.sql

```
INSERT INTO VMG VALUES(2,'M','Walter','Meier',TO_DATE('18.09.1960','DD.MM.YYYY'),120.75,'N');
INSERT INTO VMG VALUES(3,'W','Jutta','Schulze',TO_DATE('20.05.1968','DD.MM.YYYY'),120.75,'J');
INSERT INTO VMG VALUES(4,'W','Jutta','Schulze',TO_DATE('20.05.1968','DD.MM.YYYY'),120.75,'J');
INSERT INTO VMG VALUES(5,'M','Thorsten','Kleine',TO_DATE('05.01.1988','DD.MM.YYYY'),80.5,'J');
INSERT INTO VMG VALUES(6,'W','Monika','Kleine',TO_DATE('05.01.1988','DD.MM.YYYY'),80.5,'J');
COMMIT;
SELECT * FROM VMG;
```

Ausführen Scriptdatei

```
START C:\TEMP\MITGLIEDER.SQL;
```

Jetzt sind auch die Daten für die Vereinsmitglieder 2 bis 6 permanent in der Datenbank gespeichert.

Ändern eines vorhandenen Satzes

Die Spalteninhalte von vorhandenen Datensätzen können mit der Anweisung UPDATE geändert werden.

```
UPDATE tblname SET colname=coldata[,colname=coldata,…] WHERE keyname=keydata;
```

Wir möchten den Inhalt der Spalte GEZAHLT für Vereinsmitglied 2 von 'N' auf 'J' ändern und die Änderung sofort permanent in der Datenbank speichern.

```
UPDATE VMG SET GEZAHLT='J' WHERE MITGLIED=2;
COMMIT;
SELECT * FROM VMG;
```

Wichtiger Hinweis: Vergessen Sie nicht die WHERE-Bedingung.
Ohne WHERE-Bedingung wird die Aktion für alle Datensätze durchgeführt.
Beispiel:

```
UPDATE VMG SET GEZAHLT='N';
SELECT * FROM VMG;
ROLLBACK;
SELECT * FROM VMG;
```

Gut, dass es die Anweisung ROLLBACK gibt.

Übungsaufgabe

Ändern Sie den Jahresbeitrag für Mitglied 4 auf 80,5 € und korrigieren Sie das Geburtsdatum auf den 20.05.1986.

Musterlösung

```
UPDATE VMG SET BEITRAG=80.5,GEBDATUM=TO_DATE('20.05.1986','DD.MM.YYYY') WHERE MITGLIED=4;
COMMIT;
SELECT * FROM VMG;
```

Löschen eines vorhandenen Satzes

Ein Datensatz wird mit der Anweisung DELETE aus der Tabelle entfernt.

```
DELETE FROM tblname WHERE keyname=keydata;
```

Die Mitglieder 1 und 3 sind aus dem Verein ausgeschieden und sollen gelöscht werden.

```
DELETE FROM VMG WHERE MITGLIED=1;
DELETE FROM VMG WHERE MITGLIED=3;
COMMIT;
SELECT * FROM VMG;
```

Auch hier gilt: **Vergessen Sie nicht die WHERE-Bedingung (siehe UPDATE)**

Übungsaufgabe

Das Straßenverkehrsamt benötigt eine Tabelle der zugelassenen Kraftfahrzeuge.

Die Tabelle KFZ soll über folgende Spalten verfügen:

Name	Datentyp	Max. Länge	Nachkommastellen	Kommentar
AMTLKENN	Beliebige Zeichen	10	-/-	Amtl. Kennzeichen
ANREDE	Beliebige Zeichen	1	-/-	H=Herr,F=Frau
HALTER	Beliebige Zeichen	40	-/-	Vor- und Zuname des Halters
ERSTZUL	Datum und Uhrzeit	-/-	-/-	Datum der Erszulassung
STEUER	Numerischer Inhalt	6	2	Jahressteuerbetrag in €

Keine der genannten Spalten darf leer sein.

Wählen Sie einen geeigneten Primärindex.

Beispieldaten:

AMTLKENN	ANREDE	HALTER	ERSTZUL	STEUER
BI-X 1234	F	Eleonore Wagner	17.03.1998	169,78
BI-DK 765	H	Wolfgang Protzig	24.08.2001	440,00
BI-S 336	F	Maria Huber	03.10.1993	241,50

Erzeugen Sie eine Scriptdatei für ORACLE (UEB1OR.SQL) und eine Scriptdatei für INTERBASE (UEB1IB.SQL).

Inhalt:
- Erzeugen der Tabellenstruktur
- Einfügen der oben genannten Beispielsätze
- Permanentes Speichern der Datensätze in der Datenbank
- Anzeigen der vorhandenen Sätze

Führen Sie die Scriptdateien in Oracle und Interbase aus.

Musterlösung

Datei UEB1OR.SQL

```
CREATE TABLE KFZ (AMTLKENN VARCHAR2(10) NOT NULL,
                  ANREDE   VARCHAR2(1) NOT NULL,
                  HALTER   VARCHAR2(40) NOT NULL,
                  ERSTZUL  DATE NOT NULL,
                  STEUER   NUMBER(6,2) NOT NULL,
                  PRIMARY KEY (AMTLKENN));

INSERT INTO KFZ VALUES('BI-X 1234','F','Eleonore Wagner',TO_DATE('17.03.1998','DD.MM.YYYY'),169.78);
INSERT INTO KFZ VALUES('BI-DK 765','H','Wolfgang Protzig',TO_DATE('24.08.2001','DD.MM.YYYY'),440);
INSERT INTO KFZ VALUES('BI-S 336','F','Maria Huber',TO_DATE('03.10.1993','DD.MM.YYYY'),241.5);

COMMIT;

SELECT * FROM KFZ;
```

Datei UEB1IB.SQL

```
CREATE TABLE KFZ (AMTLKENN VARCHAR(10) NOT NULL,
                  ANREDE   VARCHAR(1) NOT NULL,
                  HALTER   VARCHAR(40) NOT NULL,
                  ERSTZUL  TIMESTAMP NOT NULL,
                  STEUER   DECIMAL(6,2) NOT NULL,
                  PRIMARY KEY (AMTLKENN));

INSERT INTO KFZ VALUES('BI-X 1234','F','Eleonore Wagner','17-MAR-1998',169.78);
INSERT INTO KFZ VALUES('BI-DK 765','H','Wolfgang Protzig','24-AUG-2001',440);
INSERT INTO KFZ VALUES('BI-S 336','F','Maria Huber','03-OCT-1993',241.5);

COMMIT;

SELECT * FROM KFZ;
```

Ausführung des Scripts in ORACLE

```
START C:\TEMP\UEB1OR.SQL;
```

Übungsaufgabe

Gehen Sie davon aus, dass die Daten aus o.g. Insert-Anweisungen den Inhalt der Tabelle KFZ darstellen.

Welche Anweisungen werden von einer Interbase-Datenbank als fehlerhaft abgewiesen? **Mit Begründung!**
Nach fehlerfreien Anweisungen wird ein ROLLBACK ausgeführt, d.h. die Dateninhalte verändern sich nicht.

1	`INSERT INTO KFZ VALUES('BI-Y 1234','F','Eleonore Wagner','17-MAR-1998',169.78);`
2	`INSERT INTO KFZ VALUES('BI-DK 765','H','Michael Berg','14-APR-1996',218.5);`
3	`INSERT INTO KFZ VALUES('BI-AD 163','Herr','Klaus Meier','04-OCT-2000',128.75);`
4	`UPDATE KFZ SET ANREDE='' WHERE AMTLKENN='BI-S 336';`
5	`UPDATE KFZ SET ANREDE='H';`
6	`UPDATE KFZ SET AMTLKENN='BI-S 337' WHERE AMTLKENN='BI-S 336';`
7	`INSERT INTO KFZ VALUES('BI-AD 245','X','Klaus Meier','04-OCT-2000',128.75);`
8	`INSERT INTO KFZ VALUES('BI-VH 512','F','Vera Stahnke','04-DEC-2010',-125.5);`
9	`INSERT INTO KFZ VALUES('BI-SD 3412','F','Agathe Kleinemeier','19-JAN-1998');`
10	`UPDATE KFZ SET STEUER=220,5 WHERE AMTLKENN='BI-DK 765';`
11	`UPDATE KFZ SET ERSTZUL='24-JUL-2001 15:30:00' WHERE AMTLKENN='BI-DK 765';`
12	`UPDATE KFZ SET HALTER='Gisela' WHERE AMTLKENN='BI-DK 765';`
13	`DELETE FROM KFZ;`
14	`UPDATE KFZ SET AMTLKENN='BI-X 1234' WHERE AMTLKENN='BI-S 336';`
15	`DELETE FROM KFZ WHERE AMTLKENN='Bi-Dk 765';`
16	`update kfz set steuer=0 where amtlkenn='Bi-Dk 765';`
17	`Update Kfz set Steuer=0 where AmtlKenn='BI-DK 765';`
18	`UPDATE KFZ SET STEUER=12000 WHERE AMTLKENN='BI-S 336';`
19	`UPDATE KFZ SET STEUER=120.225 WHERE AMTLKENN='BI-S 336';`
20	`INSERT INTO KFZ VALUES('BI-TK 137','H','Hubert Wiebe','04-DEZ-2000',130);`

Musterlösung

Falsch:	2	Primärindex bereits vorhanden
	3	'Herr' in Spalte Anrede zu lang, da nur max. 1 Zeichen zulässig
	9	Unzulässig, da nicht alle Spalten angegeben
	10	Dezimalkomma statt Punkt unzulässig
	14	Primärschlüsselverletzung
	20	Monatskürzel DEZ nicht definiert
Richtig:	1	O.k.
	4	Leere Zeichenkette '' ist in Interbase ungleich NULL, in Oracle gleich NULL
	5	Anrede 'H' steht jetzt in jedem Datensatz
	6	Auch der Primärindex kann geändert werden
	7	Anrede 'X' enstpricht den Spaltenkonventionen
	8	Steuer –125,5 entspricht den Spaltenkonventionen
	11	Datum mit Uhrzeit entspricht den Spaltenkonventionen
	12	Vorname ohne Nachname entspricht den Spaltenkonventionen
	13	Alle Sätze wurden gelöscht.
	15	Formal richtig, aber kein Satz betroffen
	16	Formal richtig, aber kein Satz betroffen
	17	Steuer 0 ist nicht NULL
	18	Absolut falsch! Wird aber von Interbase akzeptiert! (Oracle meldet Fehler)
	19	Steuer wird auf 2 Nachkommastellen kaufmännisch gerundet

Hinweis: Führen Sie die Anweisungen 4 und 18 in Oracle aus.

Check-Bedingungen

Zu jeder Spaltendefinition kann eine Check-Bedingung hinterlegt werden.

Beispiele Wertebereichsprüfungen

```
Syntax:  CHECK (colname {<=|<|=|<>|>|>=} coldata)
Beispiel: CHECK (STEUER>=0)

Syntax:  CHECK (colname BETWEEN minvalue AND maxvalue)
Beispiel: CHECK (STEUER BETWEEN 0 AND 9999.99)

Syntax:  CHECK (colname IN (coldata[,coldata...]))
Beispiel: CHECK (ANREDE IN ('F','H'))

Syntax:  CHECK (colname = UPPER(colname))
Beispiel: CHECK (AMTLKENN = UPPER(AMTLKENN))
```

Beispiel für Prüfung, ob Datum ohne Uhrzeit angegeben

```
Oracle:    Syntax:  CHECK (colname = TRUNC(colname))
           Beispiel: CHECK (ERSTZUL = TRUNC(ERSTZUL))

Interbase: Syntax:  CHECK (colname = CAST(colname AS DATE))
           Beispiel: CHECK (ERSTZUL = CAST(ERSTZUL AS DATE))
```

Anmerkungen: Check-Bedingungen können eine prozedurale Prüfung von Dateninhalten häufig nicht ersetzen. Zur generellen Einschränkung von Wertebereichen (z.B. Datum statt Datum und Uhrzeit) sind sie aber dennoch sinnvoll einsetzbar.

Übungsaufgabe

Ändern Sie die Scriptdateien UEB1OR.SQL und UEB1IB.SQL. Löschen Sie als Erstes die Tabelle KFZ. Die Spalte AMTLKENN darf keine Kleinbuchstaben beinhalten. Als Anrede darf nur ’H’ oder ’F’ möglich sein. Der Halternamе (HALTER) darf nicht leer sein. Die Spalte ERSTZUL darf nur das Datum, nicht die Uhrzeit beinhalten. Die Spalte STEUER darf nur Werte zwischen 0 und 9999,99 beinhalten. Führen Sie die Scriptdateien aus.

Musterlösung

Datei UEB1OR.SQL

```
DROP TABLE KFZ;

CREATE TABLE KFZ (AMTLKENN VARCHAR2(10) NOT NULL CHECK(AMTLKENN=UPPER(AMTLKENN)),
                  ANREDE   VARCHAR2(1) NOT NULL CHECK(ANREDE IN ('H','F')),
                  HALTER   VARCHAR2(40) NOT NULL,
                  ERSTZUL  DATE NOT NULL CHECK(ERSTZUL=TRUNC(ERSTZUL)),
                  STEUER   NUMBER(6,2) NOT NULL CHECK(STEUER>=0),
                  PRIMARY KEY (AMTLKENN));

INSERT INTO KFZ VALUES('BI-X 1234','F','Eleonore Wagner',TO_DATE('17.03.1998','DD.MM.YYYY'),169.78);
INSERT INTO KFZ VALUES('BI-DK 765','H','Wolfgang Protzig',TO_DATE('24.08.2001','DD.MM.YYYY'),440);
INSERT INTO KFZ VALUES('BI-S 336','F','Maria Huber',TO_DATE('03.10.1993','DD.MM.YYYY'),241.5);

COMMIT;

SELECT * FROM KFZ;
```

Datei UEB1IB.SQL

```
DROP TABLE KFZ;

CREATE TABLE KFZ (AMTLKENN VARCHAR(10) NOT NULL CHECK(AMTLKENN=UPPER(AMTLKENN)),
                  ANREDE   VARCHAR(1) NOT NULL CHECK(ANREDE IN ('H','F')),
                  HALTER   VARCHAR(40) NOT NULL CHECK(HALTER<>''),
                  ERSTZUL  TIMESTAMP NOT NULL CHECK(ERSTZUL=CAST(ERSTZUL AS DATE)),
                  STEUER   DECIMAL(6,2) NOT NULL CHECK(STEUER BETWEEN 0 AND 9999.99),
                  PRIMARY KEY (AMTLKENN));

INSERT INTO KFZ VALUES('BI-X 1234','F','Eleonore Wagner','17-MAR-1998',169.78);
INSERT INTO KFZ VALUES('BI-DK 765','H','Wolfgang Protzig','24-AUG-2001',440);
INSERT INTO KFZ VALUES('BI-S 336','F','Maria Huber','03-OCT-1993',241.5);

COMMIT;

SELECT * FROM KFZ;
```

Check-Bedingungen für Spalten ohne den Zusatz NOT NULL

Check-Bedingungen für Spalten, die NULL beinhalten können, werden um eine Klausel erweitert:

```
OR colname IS NULL
```

Beispiele (Annahme: Tabelle KFZ enthält keine Spaltendefinition mit dem Zusatz NOT NULL)

```
CHECK(AMTLKENN=UPPER(AMTLKENN) OR AMTLKENN IS NULL),
CHECK(ANREDE IN ('H','F') OR ANREDE IS NULL),
CHECK(HALTER<>'' OR HALTER IS NULL),
CHECK(ERSTZUL=TRUNC(ERSTZUL) OR ERSTZUL IS NULL),
CHECK(ERSTZUL=CAST(ERSTZUL AS DATE) OR ERSTZUL IS NULL),
CHECK(STEUER>=0 OR STEUER IS NULL),
CHECK(STEUER BETWEEN 0 AND 9999.99 OR STEUER IS NULL),
```

Übungsaufgabe

Wiederholen Sie die Anweisungen (INSERT, UPDATE und DELETE) gemäß Seite 10 sowohl in Interbase wie auch in Oracle.

Experimentieren Sie mit den Anweisungen
- INSERT
- UPDATE
- DELETE
- COMMIT
- ROLLBACK

und versuchen Sie, die Konsistenz- und Prüfmechanismen bewusst auszuhebeln.

Zusätzliche Schlüssel (Indizes)

Indizes können sowohl zur Steigerung der Auswertegeschwindigkeit wie auch zur Konsistenzsicherung genutzt werden.

Nehmen wir an, die Tabelle der Kraftfahrzeuge (KFZ) beinhaltet mehrere 100.000 Datensätze. Wenn häufiger nach dem Datum der ersten Zulassung (ERSTZUL) gesucht wird, kann die Selektionsgeschwindigkeit durch das Erzeugen eines Indexes über die Spalte ERSTZUL vermutlich drastisch erhöht werden.

Ein Index wird in der Datenbank gespeichert und belegt daher Speicherplatz. Bei jeder INSERT- UPDATE- oder DELETE-Anweisung müssen Indizes durch die Datenbank aktualisiert werden, wodurch die Geschwindigkeit beim Ausführen von Modifikationen natürlich erheblich reduziert wird. Deshalb ist das Anlegen eines Indexes nur für Spalten zu empfehlen, die häufig als Suchkriterium genutzt werden. Außerdem ist das Anlegen eines Suchindexes nur sinnvoll, wenn eine nennenswerte Anzahl von Sätzen (mind. 1000, abhängig von der Datenbank) in der Tabelle gespeichert sind.

Ein Index kann aber auch als zusätzlicher Primärschlüssel genutzt werden, um doppelte Einträge für eine oder mehrere Spalten wirksam zu unterbinden.

Syntax

Anlegen: `CREATE [UNIQUE] INDEX indname ON tblname (colname[,colname...]);`
Löschen: `DROP INDEX indname;`

Anmerkungen:
- Bei Verwendung des Zusatzes **UNIQUE** verhält sich der Schlüssel wie ein zusätzlicher Primärindex.
- Der Indexname (indname) muss innerhalb der gesamten Datenbank eindeutig sein. Es ist daher zu empfehlen, den Namen der Tabelle dem eigentlichen Indexnamen voranzustellen.

Beispiel (Suchindex über Spalte ERSTZUL)

```
CREATE INDEX KFZ_SEARCH_1 ON KFZ(ERSTZUL);
DROP INDEX KFZ_SEARCH_1;
```

Empfehlungen zur Namensvergabe:

Der Indexname wird meinerseits nach folgendem Schema gebildet:

```
tblname_{SEARCH|UNIQUE}_lfdnr

z.B.: KFZ_SEARCH_1 (falls ohne UNIQUE-Klausel)
      KFZ_SEARCH_2 (falls ohne UNIQUE-Klausel)
      KFZ_UNIQUE_1 (falls mit UNIQUE-Klausel)
      KFZ_UNIQUE_2 (falls mit UNIQUE-Klausel)
```

Übungsaufgabe

Ändern Sie die Scriptdateien UEB1OR.SQL und UEB1IB.SQL.

Fügen Sie einen Suchindex für die Spalte ERSTZUL im Anschluss an die Tabellendefinition ein.

Als Mangel hat sich das Fehlen einer Fahrgestellnummer herausgestellt. Fügen Sie eine Spalte FGESTNR vor der Spalte ERSTZUL in die Tabellendefinition ein. Die Fahrgestellnummer kann maximal 20 beliebige Zeichen umfassen und darf nicht leer sein.

Eine Fahrgestellnummer darf in der gesamten Tabelle KFZ nur genau einmal vorkommen. Sorgen Sie dafür, dass doppelte Fahrgestellnummern vom Datenbanksystem als fehlerhaft abgewiesen werden.

Ändern Sie die vorhandenen INSERT-Anweisungen.

Führen Sie die beiden Scriptdateien im jeweiligen Datenbanksystem aus.

Musterlösung

Datei UEB1OR.SQL

```
DROP TABLE KFZ;

CREATE TABLE KFZ (AMTLKENN VARCHAR2(10) NOT NULL CHECK(AMTLKENN=UPPER(AMTLKENN)),
                  ANREDE   VARCHAR2(1) NOT NULL CHECK(ANREDE IN ('H','F')),
                  HALTER   VARCHAR2(40) NOT NULL,
                  FGESTNR  VARCHAR2(20) NOT NULL,
                  ERSTZUL  DATE NOT NULL CHECK(ERSTZUL=TRUNC(ERSTZUL)),
                  STEUER   NUMBER(6,2) NOT NULL CHECK(STEUER>=0),
                  PRIMARY KEY (AMTLKENN));

CREATE INDEX KFZ_SEARCH_1 ON KFZ(ERSTZUL);
CREATE UNIQUE INDEX KFZ_UNIQUE_1 ON KFZ(FGESTNR);

INSERT INTO KFZ VALUES('BI-X 1234','F','Eleonore Wagner','A1',TO_DATE('17.03.1998','DD.MM.YYYY'),169.78);
INSERT INTO KFZ VALUES('BI-DK 765','H','Wolfgang Protzig','A2',TO_DATE('24.08.2001','DD.MM.YYYY'),440);
INSERT INTO KFZ VALUES('BI-S 336','F','Maria Huber','B3',TO_DATE('03.10.1993','DD.MM.YYYY'),241.5);

COMMIT;

SELECT * FROM KFZ;
```

Datei UEB1IB.SQL

```
DROP TABLE KFZ;

CREATE TABLE KFZ (AMTLKENN VARCHAR(10) NOT NULL CHECK(AMTLKENN=UPPER(AMTLKENN)),
                  ANREDE   VARCHAR(1) NOT NULL CHECK(ANREDE IN ('H','F')),
                  HALTER   VARCHAR(40) NOT NULL CHECK(HALTER<>''),
                  FGESTNR  VARCHAR(20) NOT NULL CHECK(FGESTNR<>''),
                  ERSTZUL  TIMESTAMP NOT NULL CHECK(ERSTZUL=CAST(ERSTZUL AS DATE)),
                  STEUER   DECIMAL(6,2) NOT NULL CHECK(STEUER BETWEEN 0 AND 9999.99),
                  PRIMARY KEY (AMTLKENN));

CREATE INDEX KFZ_SEARCH_1 ON KFZ(ERSTZUL);
CREATE UNIQUE INDEX KFZ_UNIQUE_1 ON KFZ(FGESTNR);

INSERT INTO KFZ VALUES('BI-X 1234','F','Eleonore Wagner','A1','17-MAR-1998',169.78);
INSERT INTO KFZ VALUES('BI-DK 765','H','Wolfgang Protzig','A2','24-AUG-2001',440);
INSERT INTO KFZ VALUES('BI-S 336','F','Maria Huber','B3','03-OCT-1993',241.5);

COMMIT;

SELECT * FROM KFZ;
```

Übungsaufgaben

Löschen Sie den Suchindex für die Spalte ERSTZUL.

Experimentieren Sie mit den Anweisungen
- INSERT
- UPDATE
- DELETE
- COMMIT
- ROLLBACK

und versuchen Sie, die Konsistenz- und Prüfmechanismen bewusst auszuhebeln.

Erzeugen Sie zwei Scriptdateien UEB2OR.SQL (Oracle) und UEB2IB.SQL (Interbase) und generieren Sie innerhalb dieser Scriptdateien eine Tabelle für Kunden mit dem Namen KND:
- KUNDNR (Numerisch, Wertebereich 1..999999, NULL nicht zulässig)
- KURZNAME (Max. 20 Großbuchstaben, NULL nicht zulässig)
- PKTOFIBU (Numerisch, Wertebereich 10000..69999, NULL nicht zulässig)
- LAND (Max. 3 Großbuchstaben, NULL zulässig)
- KRELIMIT (Numerisch, 0..99999999.99, NULL nicht zulässig)
- PRIVAT ('J'=Ja,'N'=Nein, NULL nicht zulässig)
- ANLDAT (Anlagedatum ohne Uhrzeit, NULL nicht zulässig)

Die Spalte KUNDNR bildet den Primärschlüssel.
Die Spalte KURZNAME soll als Suchkriterium indiziert werden.
Die Spalte PKTOFIBU (Personenkontonummer Finanzbuchhaltung) darf keine doppelten Inhalte aufweisen.

Musterlösung

Löschen Suchindex

```
DROP INDEX KFZ_SEARCH_1;
```

Datei UEB2OR.SQL

```
CREATE TABLE KND (KUNDNR   NUMBER(6,0) NOT NULL CHECK(KUNDNR>=1),
                  KURZNAME VARCHAR2(20) NOT NULL CHECK(KURZNAME=UPPER(KURZNAME)),
                  PKTOFIBU NUMBER(5,0) NOT NULL CHECK(PKTOFIBU BETWEEN 10000 AND 69999),
                  LAND     VARCHAR2(3) CHECK(LAND=UPPER(LAND) OR LAND IS NULL),
                  KRELIMIT NUMBER(10,2) NOT NULL CHECK(KRELIMIT>=0),
                  PRIVAT   VARCHAR2(1) NOT NULL CHECK(PRIVAT IN ('J','N')),
                  ANLDAT   DATE NOT NULL CHECK(ANLDAT=TRUNC(ANLDAT)),
                  PRIMARY KEY (KUNDNR));

CREATE INDEX KND_SEARCH_1 ON KND(KURZNAME);
CREATE UNIQUE INDEX KND_UNIQUE_1 ON KND(PKTOFIBU);
```

Datei UEB2IB.SQL

```
CREATE TABLE KND (KUNDNR   DECIMAL(6,0) NOT NULL CHECK(KUNDNR BETWEEN 1 AND 999999),
                  KURZNAME VARCHAR(20) NOT NULL CHECK(KURZNAME<>'' AND KURZNAME=UPPER(KURZNAME)),
                  PKTOFIBU DECIMAL(5,0) NOT NULL CHECK(PKTOFIBU BETWEEN 10000 AND 69999),
                  LAND     VARCHAR(3) CHECK(LAND<>'' AND LAND=UPPER(LAND) OR LAND IS NULL),
                  KRELIMIT DECIMAL(10,2) NOT NULL CHECK(KRELIMIT BETWEEN 0 AND 99999999.99),
                  PRIVAT   VARCHAR(1) NOT NULL CHECK(PRIVAT IN ('J','N')),
                  ANLDAT   TIMESTAMP NOT NULL CHECK(ANLDAT=CAST(ANLDAT AS DATE)),
                  PRIMARY KEY (KUNDNR));

CREATE INDEX KND_SEARCH_1 ON KND(KURZNAME);
CREATE UNIQUE INDEX KND_UNIQUE_1 ON KND(PKTOFIBU);
```

Anmerkungen zu Interbase

Die Prüfung auf eine leere Zeichenfolge '' für die Spalten KURZNAME und LAND wurde per Operator **AND** mit der UPPER-Bedingung verbunden. D.h.: Beide Bedingungen müssen erfüllt sein.

Übungsaufgabe

Experimentieren Sie mit den Anweisungen
- INSERT
- UPDATE
- DELETE
- COMMIT
- ROLLBACK

und versuchen Sie, die Konsistenz- und Prüfmechanismen bewusst auszuhebeln.