Handbuch zum Datenbank-Designer

Version 4,5

HS Harz
Fachbereich
Automatisierung und Informatik

von

Dipl. Inf., Dipl.-Ing. (FH) Michael Wilhelm

http://mwilhelm.hs-harz.de
mwilhelm@hs-harz.de

Version 03. März 2013

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

Das vorliegende Programm soll den Datenbankentwurf vereinfachen. Dazu werden die Elemente (Entities, Relationen etc), grafisch dargestellt. Mit einfachen Befehlen können die Elemente erzeugt und bearbeitet werden.

Es wird vorausgesetzt, dass Grundbegriffe in der Datenbanktechnologie vorhanden sind.

1.1 Eigenschaften des Programms

Die Abbildung 1 zeigt den Aufbau des Designers. Es hat die normale Menü- und Schalterstruktur. Alle Datenbankelemente werden in einem Baum links dargestellt. Die Breite dieses Baumes kann mit der blauen Linie (Splitter) verändert werden. Im rechten Hauptteil werden die Entities und Beziehungen dargestellt.

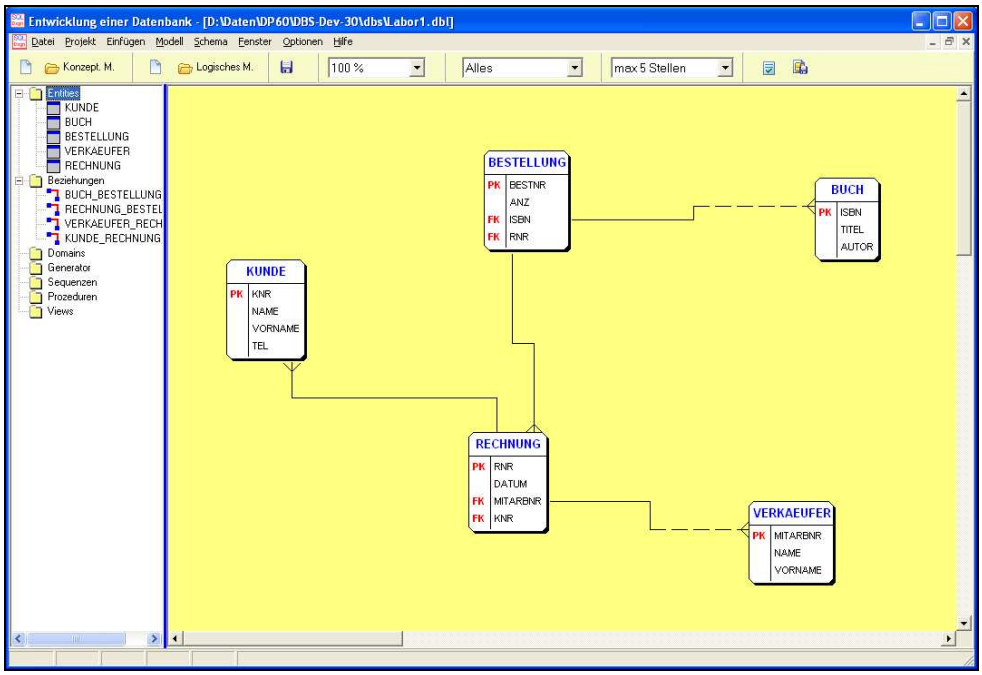

Abbildung 1 Überblick des Programms

1.2 Die Eigenschaften in Kurzform:

- Verwendung des konzeptionelles Modell
- Verwendung des logischen Modell
- Automatischer Transfer vom konzeptionellen zum logischen Modell
- Definition der Datenbanktypen (logische Modell)
- Vordefinierte Datenbanken
- Freie Definition einer Datenbank

  - Namen der Datentypen
  - Zuordnung zu den Grunddatentyp
  - SQL-Syntax definieren (Kommentare, Eckige Klammer etc.)
  - Counter definieren (Generator, Sequenz, Autonumber)
  - Liste der reservierten Wörter
- Erzeugung und Verwaltung von Entities
- Erzeugung und Verwaltung von Attributen
- Erzeugung und Verwaltung von Relationen
- Erzeugung und Verwaltung von Generatoren
- Erzeugung und Verwaltung von Sequenzen
- Generierung eines SQL-Scriptes
- Vielfältige Farb- und Schriftwahl
- Anzeige der Relationen mit Krähenfuss oder Punktnotation
- Export der Daten nach Winword
- Export der ERM-Grafik in die Zwischenablage

1.3 Wichtige neue Eigenschaften der Version 4

Allgemeines:

- Es kann eine zusätzliche Instanz aufgerufen werden (ALT-Tab Umschaltung)
- **Kopieren von Entities über die Zwischenablage möglich**
- Export in die Zwischenablage als Bild mit Randdefinition
- Überprüfung der Datenbank auf doppelte Namen, Checkbedingungen
- Check-Attribut in Entity-Einst eingetragen

Konzeptionellen Modell:

- **Es kann nun auch ein Defaultwert eingegeben werden**
- **Es kann nun auch eine Check-Bedingung eingegeben w**erden
- Die Länge der Kurznamen wird auf 8 Zeichen begrenzt
- Es gibt einen Test der Relationsnamen FK_???? auf Länge <=31 Zeichen
- **Es gibt nun auch eine Weak-Relation**
- Die Anzeige erfolgt in der Chen-Notation und Martin-Notation
- **Es gibt Multi-Attribute in den Entities**
- **Es gibt Ternäre Relationen**
- **Es gibt auch eine Self-Relation**
- **Rauten-Darstellung der Relation**
- **Die Relationen können Attribute haben**

Logische Modell:

- Es gibt nun auch ein Weak-Entity
- Die Anzeige erfolgt nun auch in der Martin-Notation

1.4 Installation

Das Programm bzw. die Zip-Dateien können in jedes beliebige Verzeichnis kopiert werden. Beim ersten Aufruf wird das Unterverzeichnis „dbs“ erzeugt. In diesem werden zum Einen die Projekte gespeichert, und zum Anderen die Datenbank-Definitionsdateien (*.def).

Interne Daten werden in der Registry abgespeichert.

Eintrag:

HKEY_CURRENT_USER\Software\WILHELM\DESIGNER

1.5 Ablauf bei der Erstellung eines konzeptionellen Modells

- Erzeugen eines neuen Fensters (STRG+M)
- Speichern unter einem Dateinamen
- Einstellen der Projekteinstellungen (Projekt | Eigenschaften)
- Erstellen der Entities (STRG+E)
- Eintragen der Attribute, ohne der/des Fremdschlüssels
- Festlegen der Primärschlüssel
- Erstellen der Beziehungen (STRG+R)
- Zusätzliche Eintragungen (Check, Unique, Weak-Entities)
- Eintragen der Texte (Beschreibungen etc.)
- Aufruf der Übertragung vom konzeptionellen zum logischen Modell (F9)

1.6 Ablauf bei der Erstellung eines logischen Modells

- Erzeugen eines neuen Fensters (STRG+N), nur wenn es kein konzeptionelles Modell gab.
- Einstellen der Projekteinstellungen (Darstellungen der Beziehungen, Texte)
- Erstellen der Entities (STRG+E)
- Eintragen der Attribute, ohne der/des Fremdschlüssels
- Festlegen der Primärschlüssel
- Erstellen der Beziehungen (STRG+R)
- Zusätzliche Eintragungen (Check, Unique, Weak-Entities, Generator, Sequenz)
- Eintragen der Texte (Beschreibungen etc.)
- Aufruf der Generierung zum SQL-Skript (F9)

2 Beispiele

Dieses Kapitel beschreibt die Vorgehensweise zur Erzeugung einer Datenbank. Es wird unter dem Namen „Handbuch Bsp1.dbk“ mitgeliefert.

Hinweis:

Das konzeptionelle Modell erlaubt alle möglichen Beziehungstypen zwischen zwei Entitäten. Dazu gehört auch die n:m-Beziehung.
Mit dem Schritt zum logischen Modell müssen alle netzförmigen Beziehungen (n:m etc.) in hierarchische Beziehungen umgewandelt werden. Dabei wird jeweils ein neues Entity erzeugt. Der Name wird aus den beiden Entitäten gebildet.

2.1 Entwicklung eines konzeptionelles ER-Modells

Das folgende Beispiel entwickelt eine einfache Kundendatenbank: „Handbuch Bsp1.DBK“

2.1.1 Entitäten

- Kunde
- Ort
- Artikel

2.1.2 Beziehungen:

Kunde zu Ort	cm:1
Kunde zu Artikel	cm:cm

2.1.3 Neues Projekt

Mit der Taste „STRG+M“ wird ein neues Projekt angelegt. Es erscheint der Rahmen mit den Einträgen im Baum und der leeren Diagrammfläche. Im nächsten Schritt müssen nun die Entitäten angelegt werden.

Im ersten Schritt sollte die Definition der Primarykeys definiert werden. Dazu ruft man den Menüeintrag „PROJEKT|Eigenschaften“ auf.

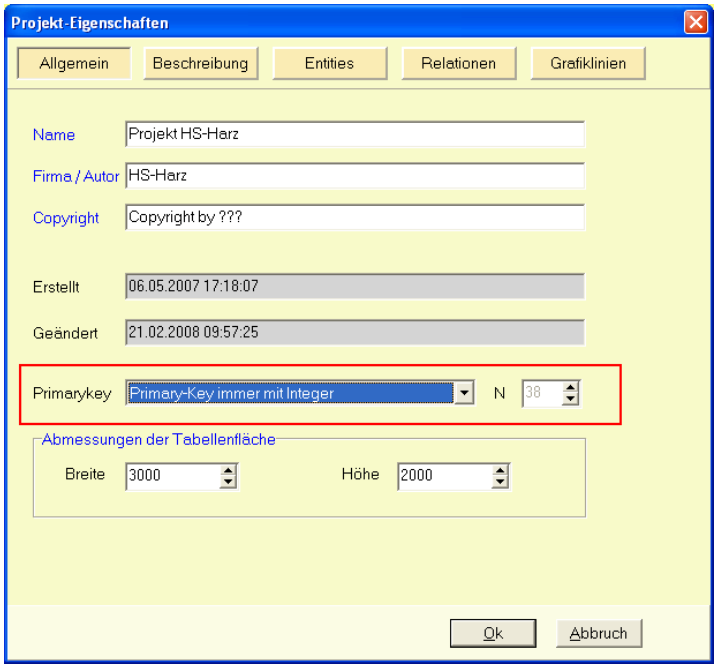

Abbildung 2 Projekteigenschaften (Integer vs. Max Numeric)

In der Comboliste kann man entscheiden, ob man der Datentyp eines Primarykeys ein Integer oder ein Numeric/Number-Datentyp ist.

Hinweis:
Fremdschlüsselattribute sollten nicht angelegt werden. Diese werden automatisch beim Erzeugen des logischen Modells eingetragen.

2.1.4 Entitiy anlegen

Mit dem Tastendruck „STRG+E“ werden die jeweiligen Entities angelegt.

Zur erst wird nach dem Namen gefragt, dieser darf nur aus Buchstaben, Zahlen und dem Underline („_“) bestehen. Es darf kein Leerzeichen verwendet werden. Am Anfang muss auch ein Buchstabe stehen.

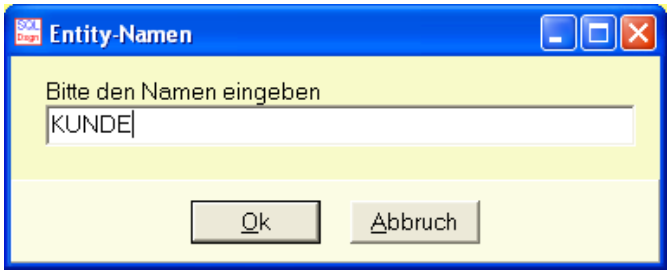

Abbildung 3 Erzeugen eines Entities

Nach der Eingabe der Namen, erscheinen die Entities in der Grafik

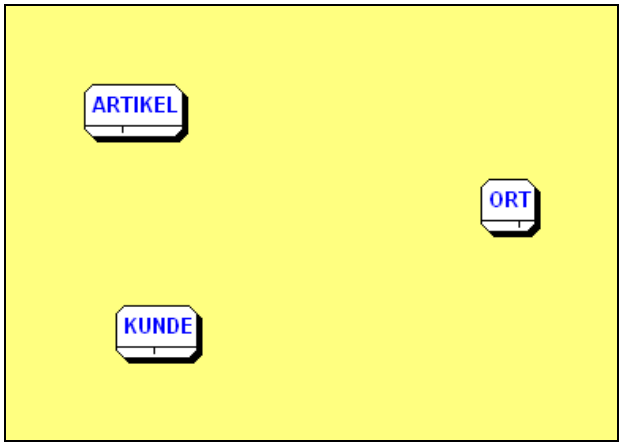

Abbildung 4 Neue Entities

Mit einem Doppelklick kann das Entitiy „Kunde“ bearbeitet werden. Es erscheint die Abbildung 5. Hier sollten die Einträge ausgefüllt werden. Des Weiteren ist es ratsam, die nähere Beschreibung des Entities im Register „Beschreibung“ vorzunehmen. Im zweiten Register, siehe Abbildung 6, werden dann die Attribute eingetragen.

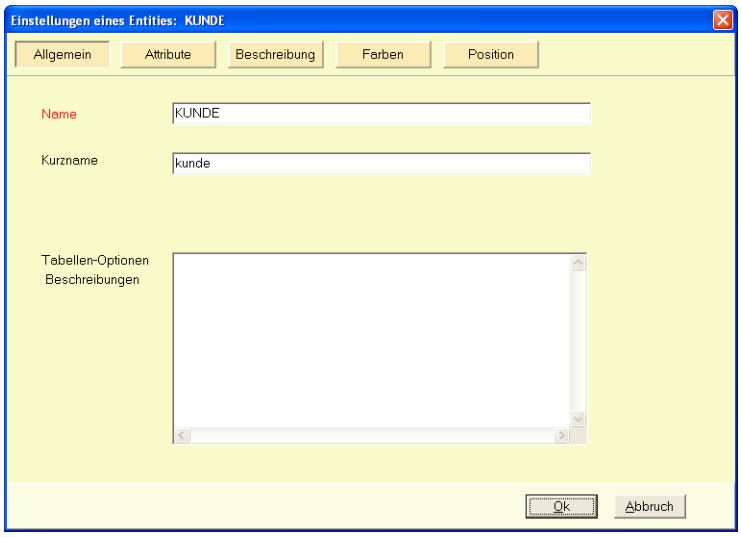

Abbildung 5 Haupt-Eigenschaften eines Entities

Das zweite Register verwaltet die Attribute. Mit den unteren Schaltern kann man diese bearbeiten, löschen und erzeugen. Die Schalter rechts an der Seite verschieben die Reihenfolge.

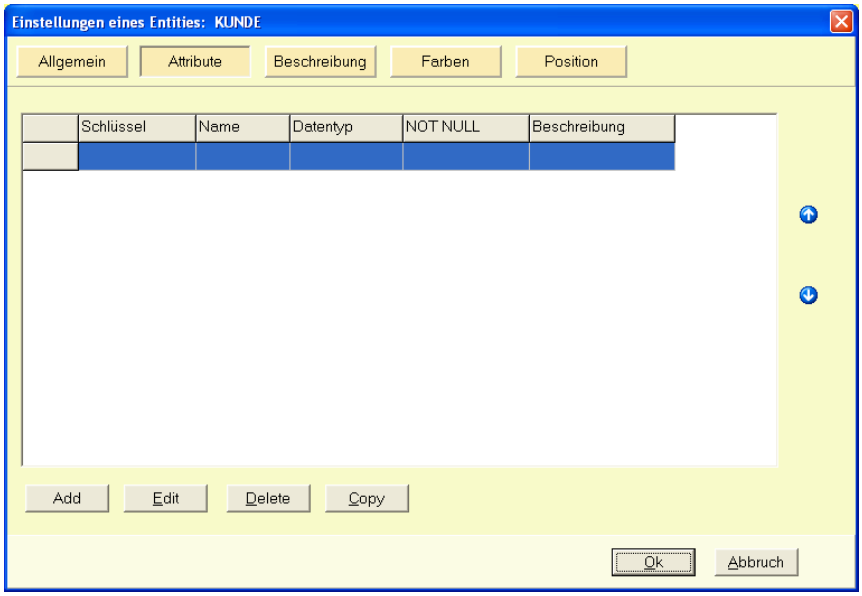

Abbildung 6 Bearbeiten, Erzeugen, Löschen von Attributen

2.1.4.1 Attribute des Entity „Kunde“

- KNr
- KName

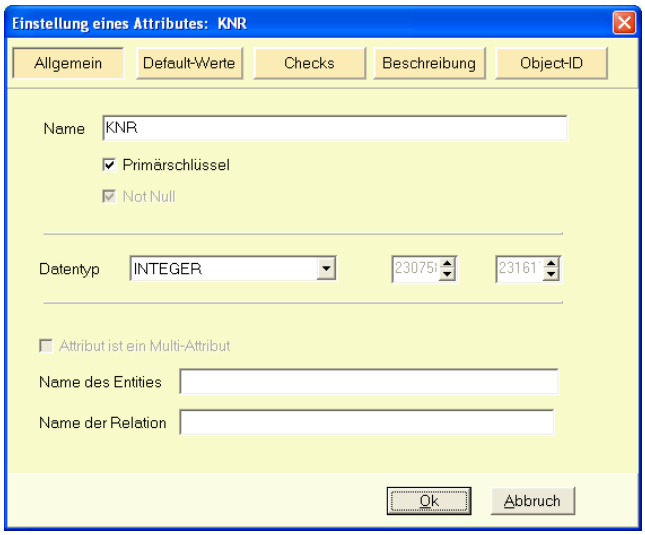

Abbildung 7 Kundennummer einfügen

Wenn man das Kontrollfeld „Primarykey“ anklickt, so wird automatisch beim ersten Mal der Datentyp „Integer“ bzw. Numerik/Number ausgewählt. Man diesen Datentyp aber danach auch wieder ändern.

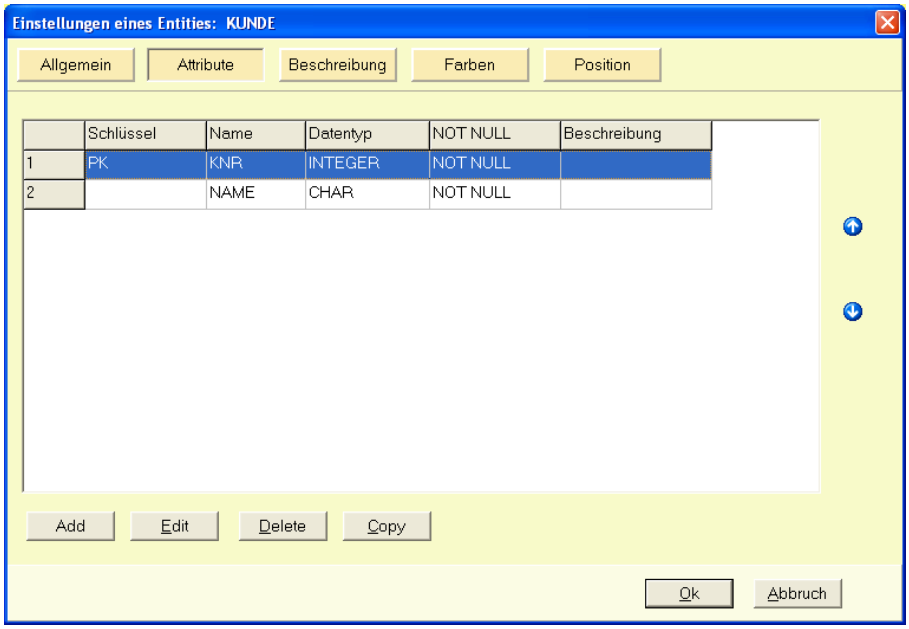

Abbildung 8 Attribute des Entities Kunde

Hinweis:
Der Ort wird als Fremdschlüssel später hinzugefügt.

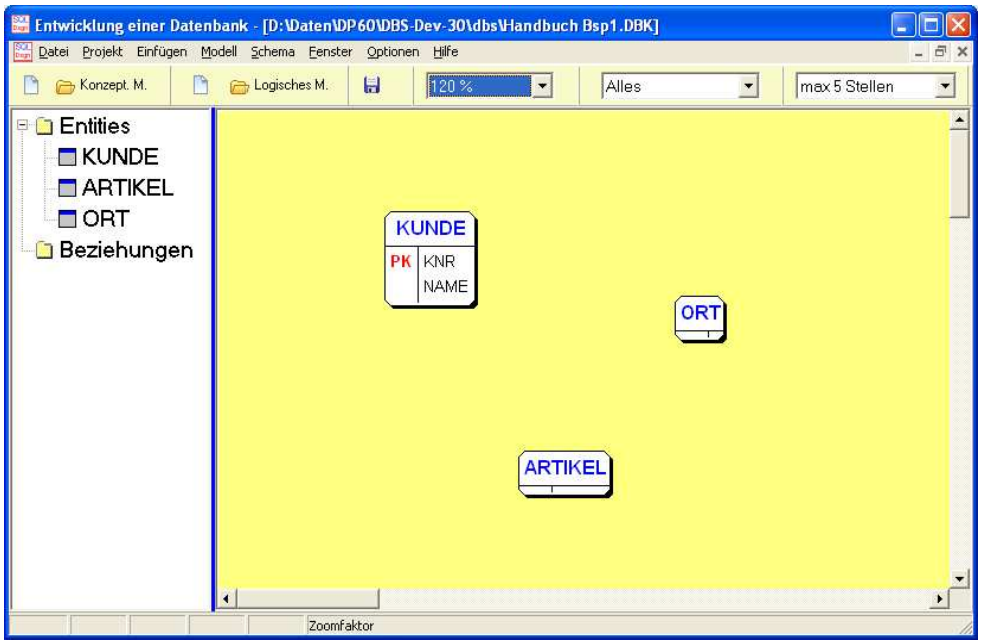

Abbildung 9 Entitäten mit Attributen des Kunden

2.1.4.2 Attribute des Entity „Ort“

- ONr
- PLZ
- Name

Da die Postleitzahl ein Zeichenfeld sein muss, wird hier ein separater Schlüssel definiert.

2.1.4.3 Attribute des Entity „Artikel“

- ANr
- Name
- Preis

Wenn alle Entitäten eingegeben wurden, hat das ER-Modell folgendes Aussehen:

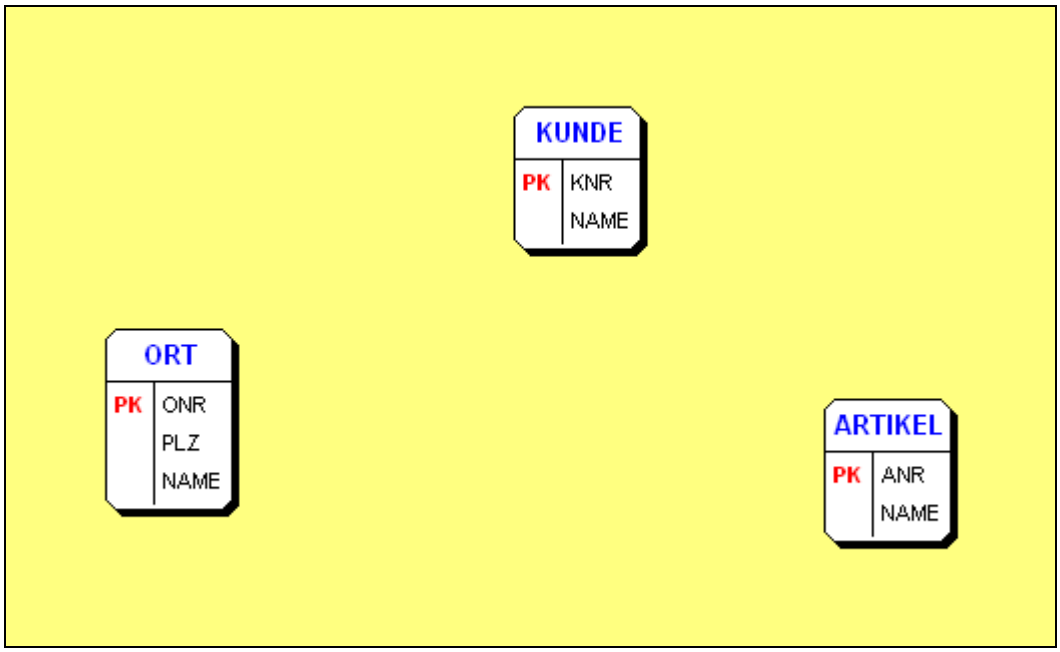

Abbildung 10 ER-Modell des ersten Beispiels

Beim konzeptionellen Modell ist es noch nicht vorgesehen, das ein Attribut nur eindeutige Werte – Unique – zugewiesen werden darf. Bei Bedarf muss man dieses im logischen Modell nachholen. Das Entity „Ort“ hat zwar das Attribut „PLZ“, dieses kann aber mehrfach vergeben werden.

2.1.5 Beziehungen anlegen

Dazu ruft man aus dem Hauptmenü „Einfügen“ die Funktion „Beziehung“ auf. Es erscheint folgendes Fenster:

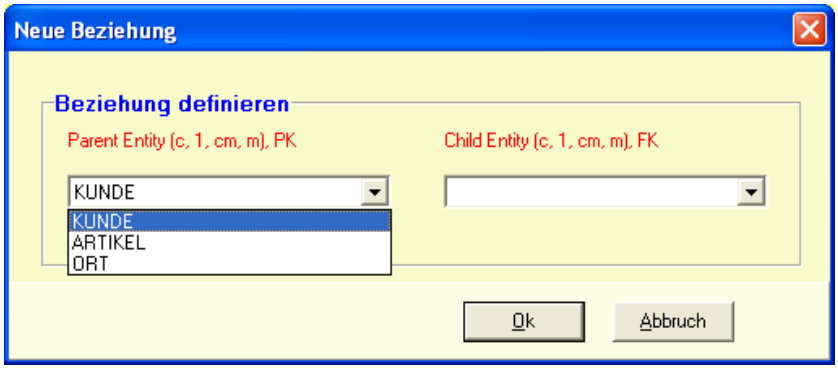

Abbildung 11 Erstellen der Beziehung Ort zu Kunde

In beiden Listen sind alle Entities eingetragen. In der linken Liste wählt man den Parent-Entity aus. Das wäre hier in diesem Beispiel „Ort“. In der rechten Liste wird die Beziehung eingetragen. Hier also „Kunde“.

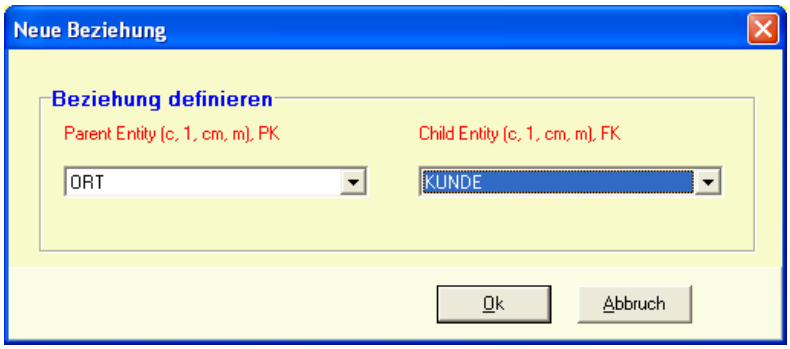

Abbildung 12 Korrekte Einstellung für das erste Beispiel

Die Abbildung 13 zeigt die endgültigen Eingaben. Mit dem Schalter „Ok“ wird das Beziehungsfenster zur Definition der Beziehung angezeigt.

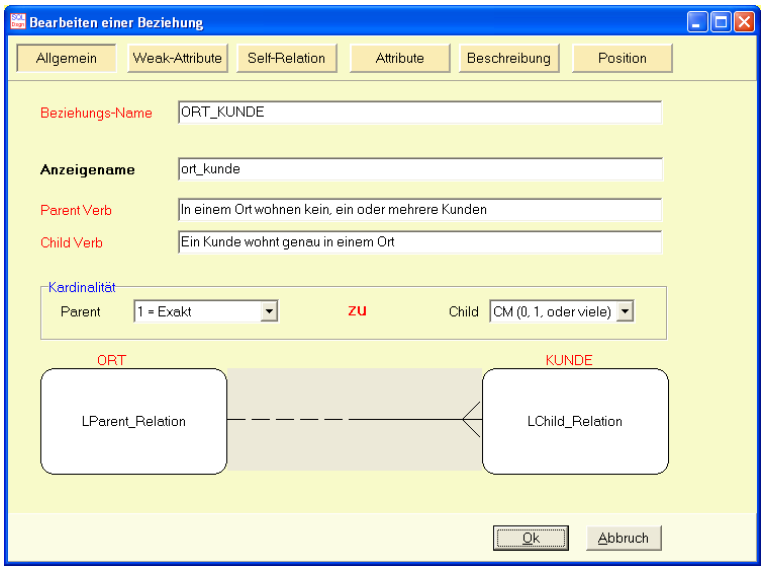

Abbildung 13 Definition der Beziehung Ort zu Kunde

Im ersten Register werden die Grundeinstellungen (Name, Beziehungstyp) eingetragen. Wichtig sind hier die Einträge „Parent-Verb“ und „Child-Verb“. Sie zeigen den Charakter der Beziehung an. Das Parent-Verb beschreibt die Beziehung durch Wörter wie „Kunde hat“ etc.

Im zweiten Register kann man weitere Beschreibungstexte eintragen

Die zweite Beziehung betrifft die Entitäten Kunde und Artikel. Hier ist es eine cm:n-Beziehung.

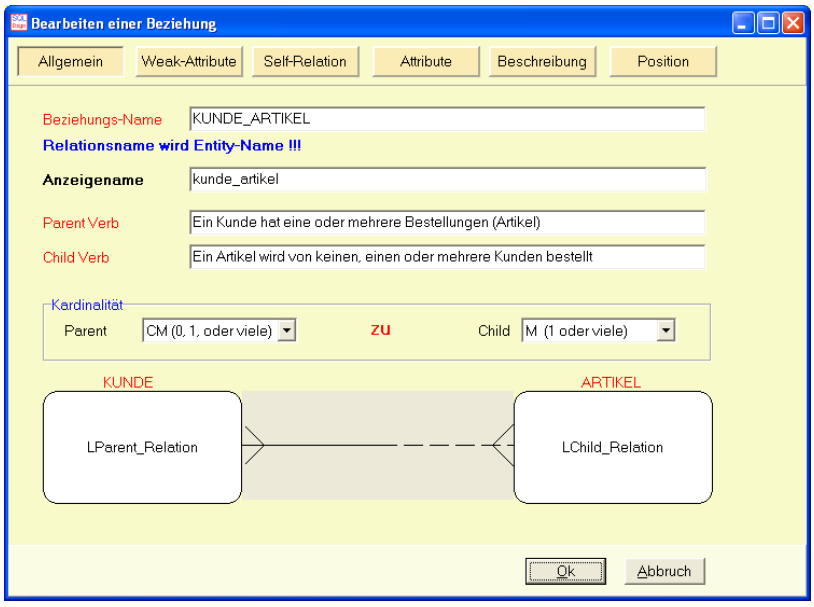

Abbildung 14 Zweite Beziehung im ersten Beispiel

Nun hat man das fertige konzeptionelle Modell:

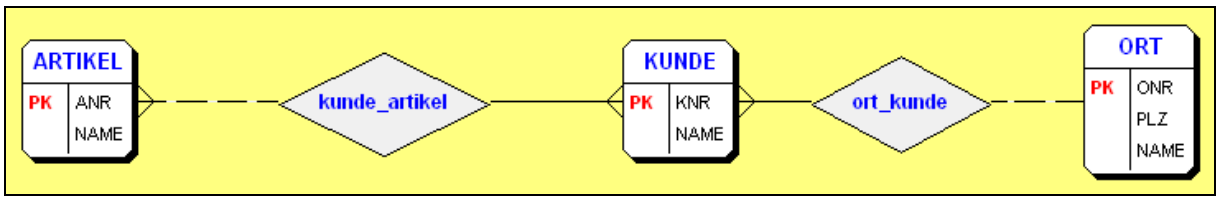

Abbildung 15 Konzeptionelles Modell des ersten Beispiels

2.1.6 Weitere Schritte

Dieses Kapitel zeigt weitere mögliche Schritte im konzeptionellen Modell. Zum Einen kann man für verschiedene Attribute Defaultwerte und Checkbedingungen definieren.

2.1.6.1 Default-Werte definieren

Mit einem Doppelklick auf ein Entity erhält man das normale Dialogfenster.

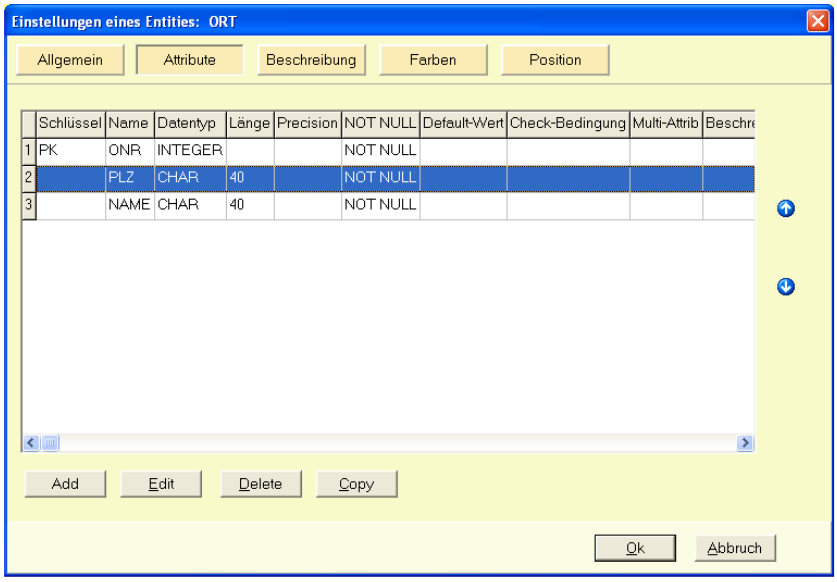

Abbildung 16 Attribute des Entity Ort

Mit einem Doppelklick auf das Attribut PLZ erscheint das Attributdialog. Hier kann man die Länge der Postleitzahl zum Einen etwas verkleinern, Länge auf 5 setzen, und zum Anderen die wichtigste Postleitzahl definieren. Dazu klickt man den Schalter „Default-Werte an.

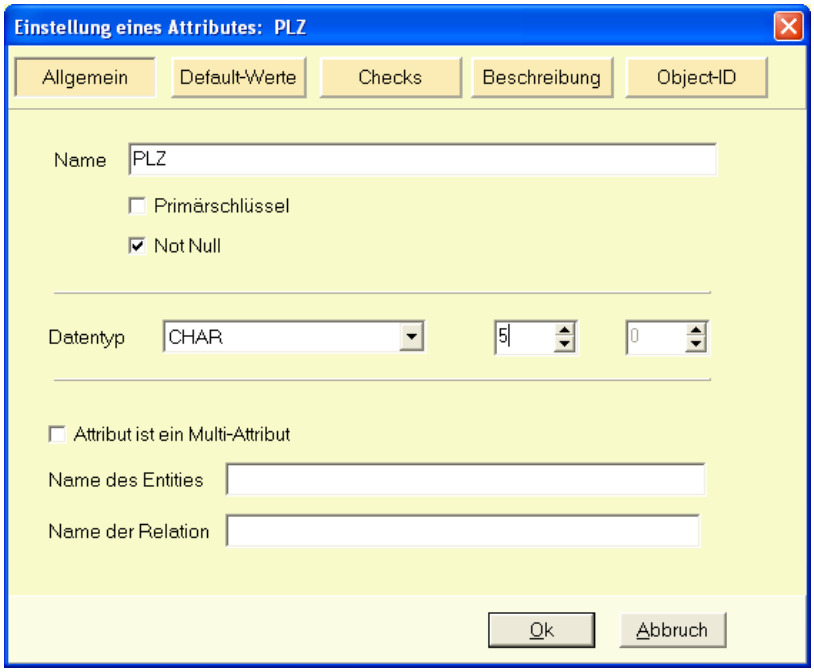

Abbildung 17 Postleitzahl auf Länge 5

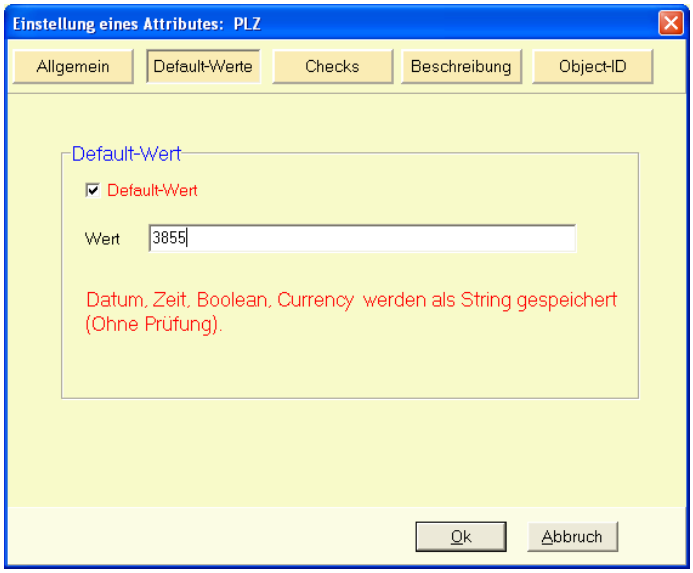

Abbildung 18 Postleitzahl als Default-Wert

Für alle Werte in Wernigerode benötigt man nun nicht mehr die Eingabe der Postleitzahl.

2.1.6.2 Unique-Attribute definieren

Für manche Attribute ist es sinnvoll, eindeutige Werte zu definieren (Beispiele Artikelname, Autoname). Dazu gibt es bei den Attribut-Registern im zweiten Register ein Eintrag. Man aktiviert das Kontrollfeld „Unique“ und trägt einen passenden Name in das Eingabefeld.

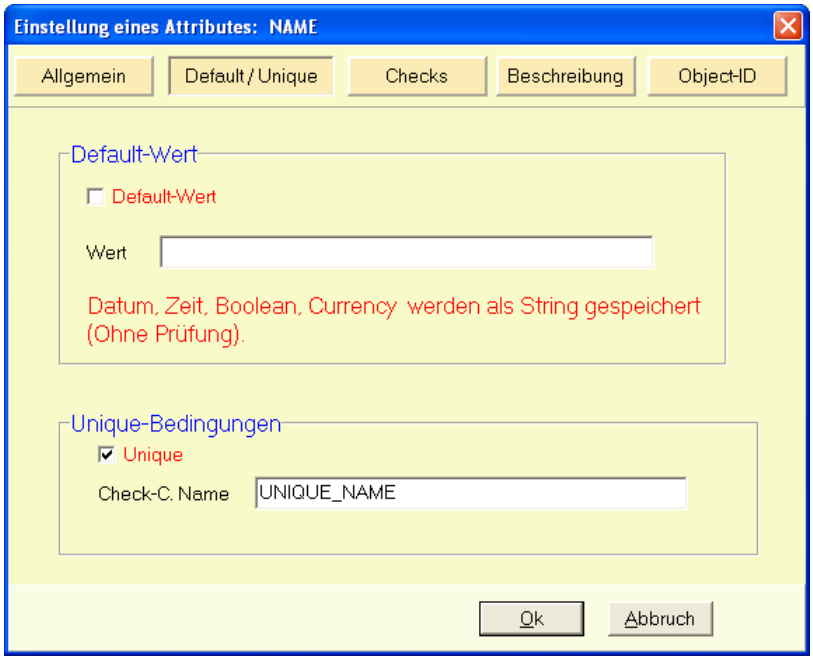

Abbildung 19 Unique-Constraint für den Ortsnamen einfügen

2.1.7 Umwandlung in ein logisches Modell

Mit dem Menü „Schema“ und dem Eintrag „Generieren logisches Modell“ wird ein neues Fenster erzeugt, in dem das logische Modell automatisch eingetragen wird. Vor diesem Schritt wird noch die Zieldatenbank abgefragt.

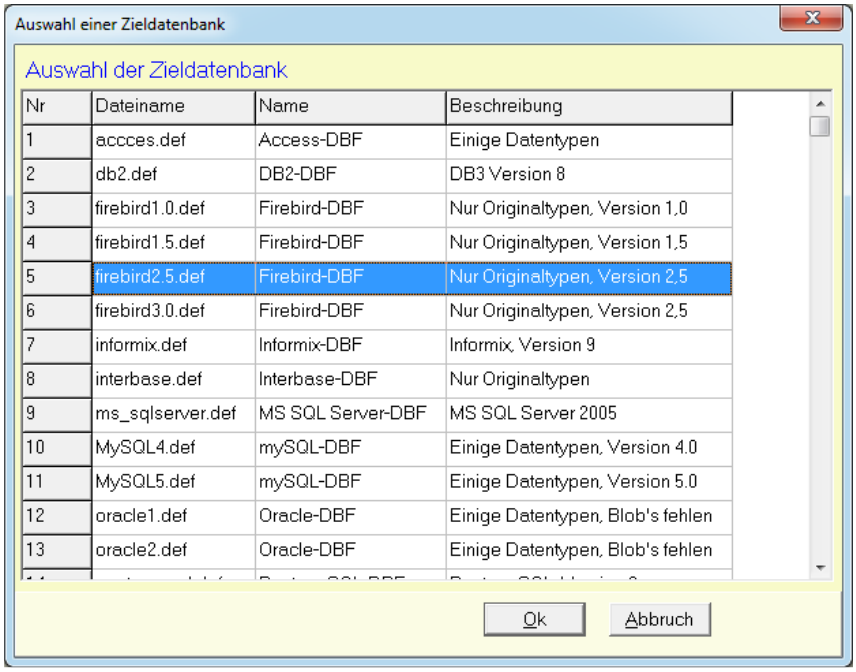

Abbildung 20 Zieldatenbank

Doppelklick oder der Schalter „Ok“ erzeugt dann das logische Modell in einem neuem Fenster.

Abbildung 21 Portierungsmeldung

Die nächste Abbildung zeigt das logische Modell:

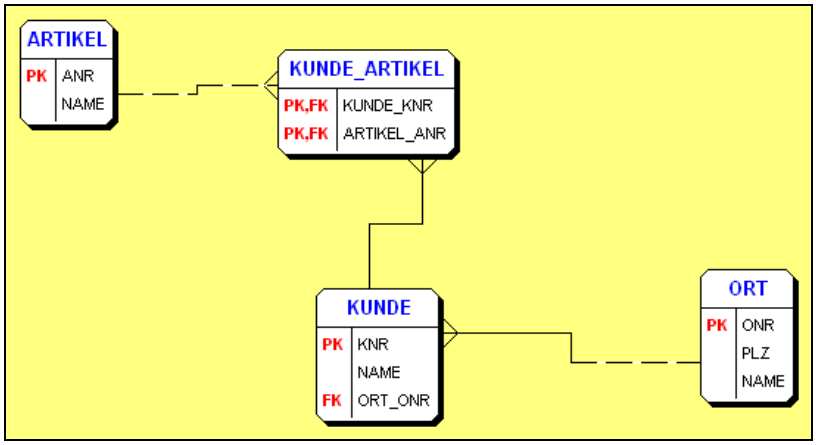

Abbildung 22 Das logische Modell des ersten Beispiels

Die cm:m-Beziehung wurde durch eine zusätzliche Entität in ein hierarchisches Modell überführt. Des Weiteren wurde das Attribut „ORT_ONR“ als Fremdschlüssel in das Entity „Kunde“ eingetragen.

2.1.8 Weitere Schritte

2.1.8.1 Check-Constrainst definieren

Check-Bedingungen werden benutzt, um zusätzliche Überprüfungen VOR dem Eintragen in die Datenbanken vorzunehmen.

Beispiel:

Das Entity Kunde hat noch das Attribut „GebDatum“.

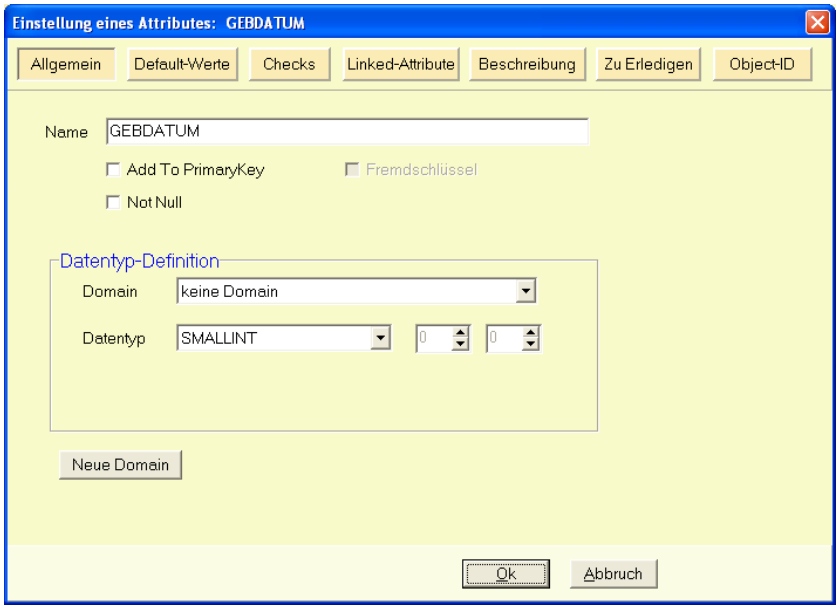

Abbildung 23 Attribut GebDatum

Die obige Abbildung zeigt die Standardeingaben eines Attributs.

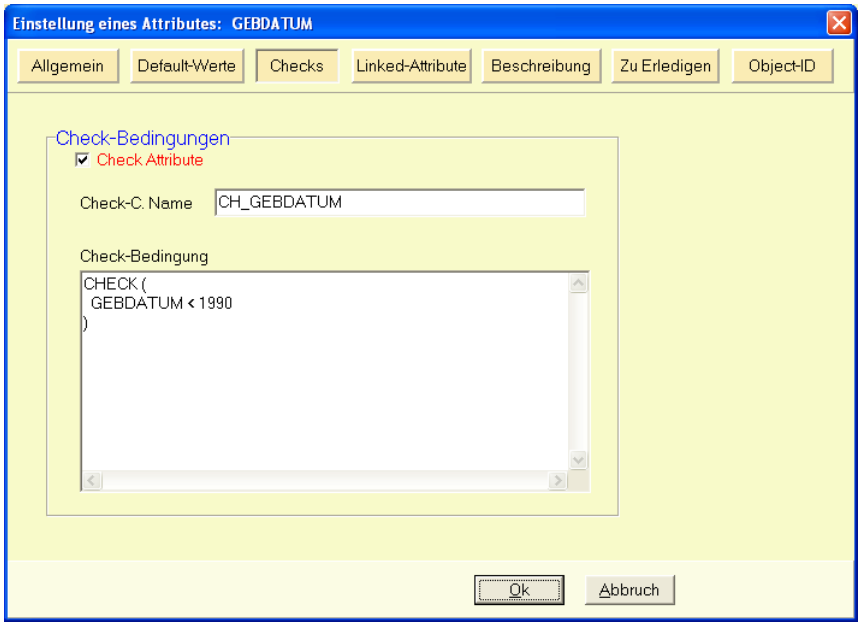

Abbildung 24 Check-Bedingung

Alternativ kann man auch das aktuelle Jahr ermitteln und dann 16 bzw. 18 Jahre abziehen.

Für Firebird wäre das diese Syntax:

CAST('NOW' AS DATE)	liefert das aktuelle Datum
CAST('NOW' AS TIMESTAMP)	liefert das aktuelle Datum, Zeit
CAST('NOW' AS TIME)	liefert die aktuelle Zeit

EXTRACT(year FROM column)	Auslesen des Jahres

Bedingung GebDatum nur als Jahr:

```
CHECK (
  GEBDATUM   <  EXTRACT(year FROM  CAST('NOW' AS DATE))-17
)
```

Bedingung GebDatum als Datum:

```
CHECK (
  EXTRACT(year  FROM  GEBDATUM)  <  EXTRACT(year  FROM  CAST('NOW'  AS
DATE))-17
)
```

oder mit Monaten etwas genauer

```
CHECK (
  EXTRACT(year FROM GEBDATUM)*12 + EXTRACT(month FROM GEBDATUM)

     <
        EXTRACT(year FROM  CAST('NOW' AS DATE))*12 +
        EXTRACT(month FROM  CAST('NOW' AS DATE))
)
```

Weiteres Beispiel:

Man fügt das Attribut „Preis“ zum Entity „Artikel“. Dann sollte dieser nicht negativ sein.

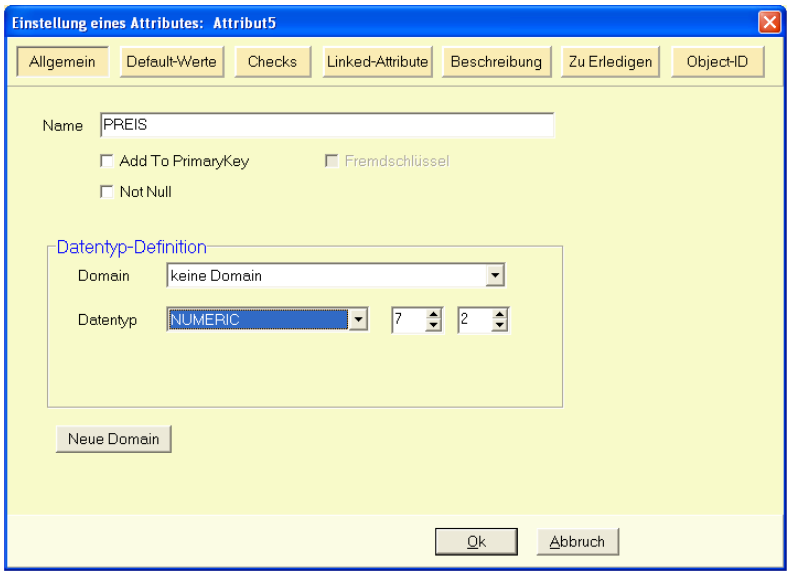

Abbildung 25 Erzeugen des Attributs Preis

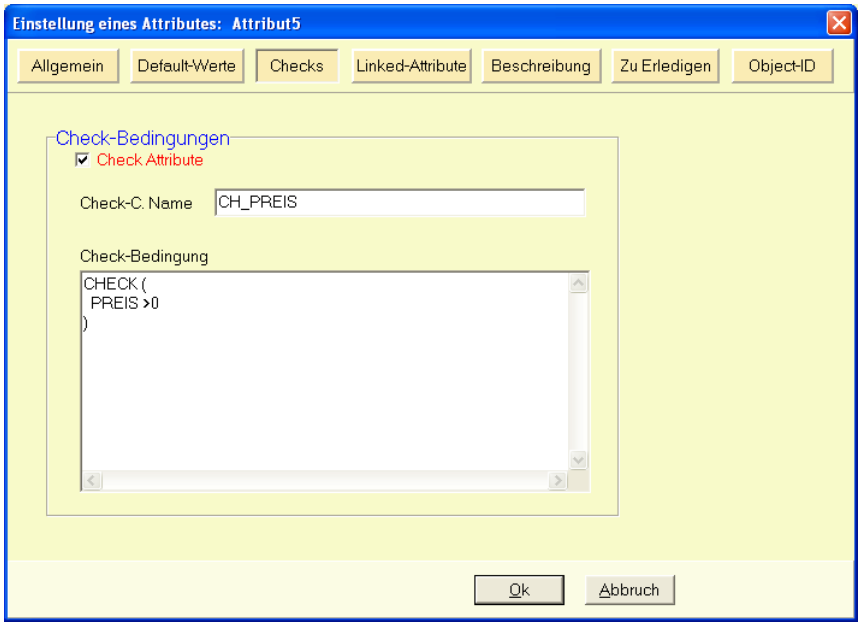

Abbildung 26 Check-Bedingung des Preises

2.1.8.2 Generator oder Sequenz definieren

Einige Datenbanken verwenden Generatoren, andere Sequenzen. Access wiederum benutzt einen Autoincrement-Datentyp. Ziel eines Generators ist die automatische Erzeugung eindeutiger Schlüssel. Man könnte dies auch so definieren:

```
SELECT MAX(empno)
FROM employee;
```

Diese Nummer wird nun um eins erhöht und als neue empno verwendet. Dieses funktioniert im Single-User-Betrieb vollkommen. Nur im Netzwerk und im Multi-User-Betrieb können zwei oder drei neue Mitarbeiter die gleiche Nummer erhalten. Mit Hilfe eines Generators bzw. einer Sequenz wird eine Nummer nur einmal vergeben.

Nachteil:

Es kann zu Lücken kommen, wenn ein Abbruch der Transaktion, Rollback, vorgenommen wird.

Generator definieren:

Um einen Generatoren zu erzeugen, ruft man das Menü „Einfügen“ mit dem Eintrag „Generator“ auf.

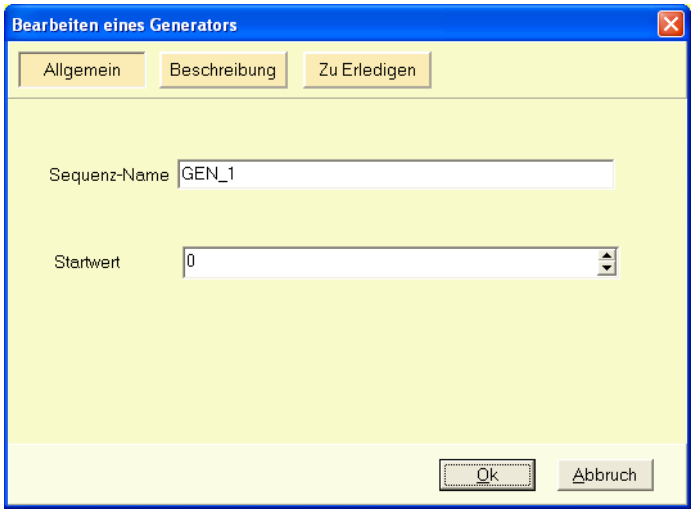

Abbildung 27 Definition eines Generators

In Firebird bzw. Interbase kann man neben dem Namen nur den Startwert definieren. Der erste richtige Wert ist dann um eins erhöht.

Beispielskript:

```
CREATE GENERATOR GEN_KUNDE;
SET GENERATOR GEN_KUNDE TO 0;
```

Sequenz definieren (z. B. Oracle):

Um einen Sequenz zu erzeugen, ruft man das Menü „Einfügen“ mit dem Eintrag „Sequenz“ auf.

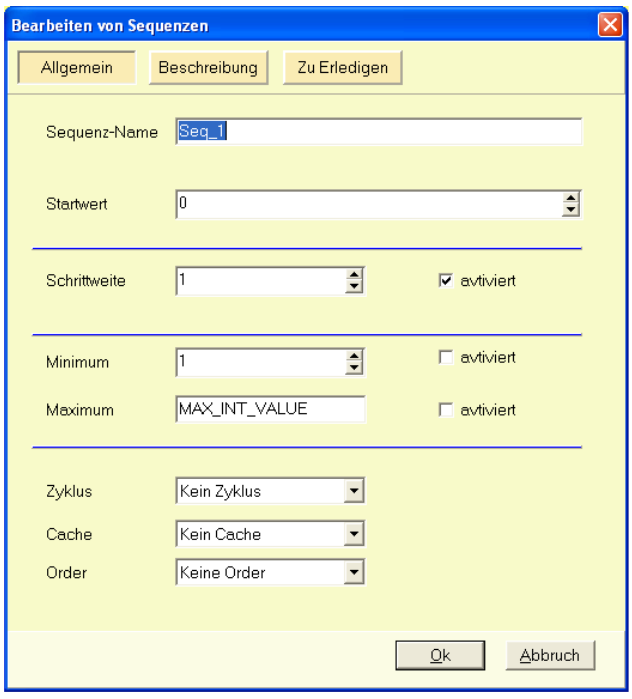

Abbildung 28 Definition einer Sequenz

In Oracle hat man detailliertere Optionen. Zusätzlich definiert man die Schrittweite, das Minimum und Maximum. Die Einstellung Zyklus stellt sicher, dass man bei einem Erreichen des Maximalwerts wieder von vorne anfängt.

Beispielskript:

```
CREATE SEQUENZ SEQ_1
 INCREMENT BY 1
 START WITH 0
 NOMINVALUE
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER;
```

Einige Datenbanken verwenden Sequenzen (Oracle, PostgreSQL, Informix). Die Definition ist aber nicht genormt. Deshalb können die obigen Teile separat definiert werden.

2.1.8.3 Ergebnis

Die untere Abbildung zeigt das vollständige und erweiterte ER-Modell.

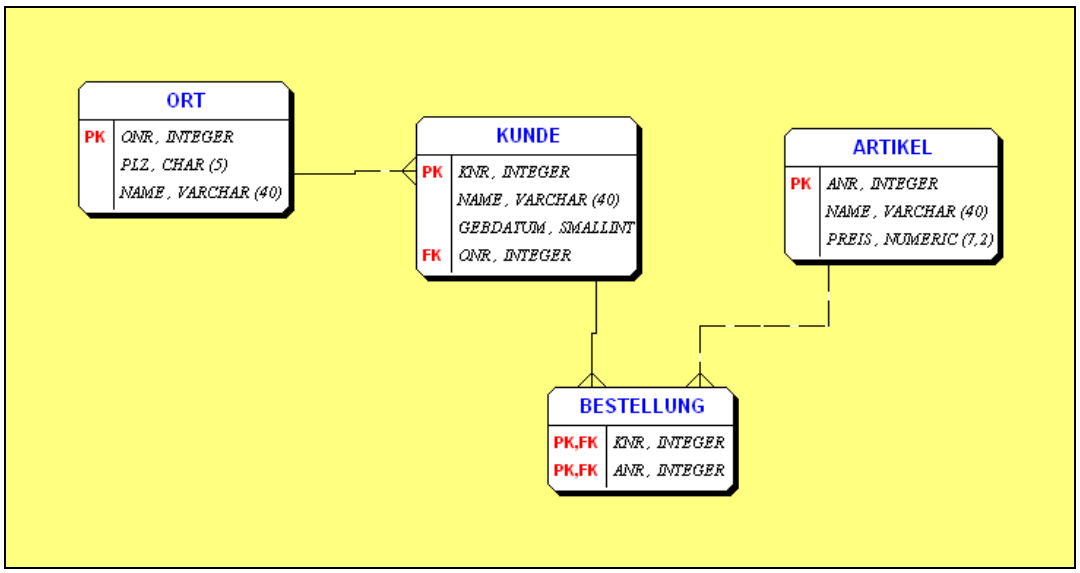

Abbildung 29 Logisches ER-Modell des ersten Beispiels

Mit dem Menü „Schema“, Eintrag „Generierung Datenbank-Skript“ kann man nun das SQL-Skript erstellen (F9).

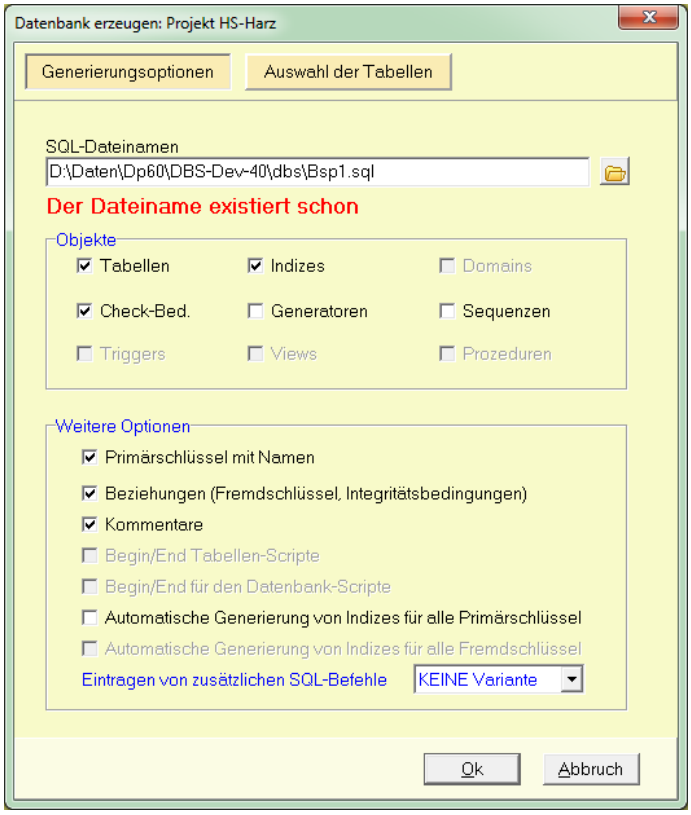

Abbildung 30 Erzeugen eines SQL-Skriptes

Ergebnis mit SQL-Befehlen:

```
/*==============================================================*/
/* Project Filename: C:\Daten\Handbuch Bsp1.sql                 */
/* Project Name:     Beispiel 1                                 */
/* Author:           HS-Harz                                    */
/* DBMS:             Firebird-DBF                               */
/* Copyright:        Copyright by Michael Wilhelm               */
/* Generated on:     06.05.2007 23:29:06                        */
/*==============================================================*/

/*==============================================================*/
/*  Tables                                                      */
/*==============================================================*/
CREATE TABLE KUNDE (
  KNR  INTEGER  NOT NULL,
  NAME  VARCHAR(40)  NOT NULL,
  GEBDATUM  SMALLINT,
  ONR  INTEGER  NOT NULL,

  PRIMARY KEY (KNR)
);


CREATE TABLE ARTIKEL (
  ANR  INTEGER  NOT NULL,
  NAME  VARCHAR(40)  UNIQUE  NOT NULL,
  PREIS  NUMERIC(7,2),

  PRIMARY KEY (ANR)
);
```

```
CREATE TABLE ORT (
  ONR  INTEGER  NOT NULL,
  PLZ  CHAR(5)  DEFAULT '38855'  NOT NULL,
  NAME  VARCHAR(40)  NOT NULL,

  PRIMARY KEY (ONR)
);


CREATE TABLE BESTELLUNG (
  KNR  INTEGER  NOT NULL,
  ANR  INTEGER  NOT NULL,

  PRIMARY KEY (KNR, ANR )
);


/*==============================================================*/
/*  Foreign Keys                                                */
/*==============================================================*/
ALTER TABLE KUNDE
   ADD CONSTRAINT FK_ORT_KUNDE FOREIGN KEY (ONR) REFERENCES ORT(ONR);


ALTER TABLE BESTELLUNG
   ADD CONSTRAINT FK_KUNDE__kunde_artikel FOREIGN KEY (KNR) REFERENCES
KUNDE(KNR);


ALTER TABLE BESTELLUNG
   ADD CONSTRAINT FK_ARTIKEL__kunde_artikel FOREIGN KEY (ANR) REFERENCES
ARTIKEL(ANR);


/*==============================================================*/
/*  Check-Constraints                                           */
/*==============================================================*/
ALTER TABLE KUNDE
     ADD CONSTRAINT CH_GEBDATUM CHECK (
  GEBDATUM < 1990
);


ALTER TABLE ARTIKEL
     ADD CONSTRAINT CH_PREIS CHECK (
  PREIS >0
);
```

2.2 Entwicklung eines konzeptionelles ER-Modells

Das folgende Beispiel entwickelt eine einfache Firmendatenbank:, bei der Multiattribute im Entity „Abteilung“ definiert werden.
Dateiname: Handbuch Bsp2_MultiAttribute.DBK

2.2.1 Entitäten

- Mitarbeiter
- Abteilung

2.2.2 Beziehungen:

Abteilung zu Mitarbeiter 1:cm

2.2.3 Neues Projekt

Mit der Taste „STRG+M“ wird ein neues Projekt angelegt. Es erscheint der Rahmen mit den Einträgen im Baum und der leeren Diagrammfläche. Im nächsten Schritt müssen nun die Entitäten angelegt werden. Vorab sollte die Definition der Primarykeys definiert werden. Dazu ruft man den Menüeintrag „PROJEKT|Eigenschaften“ auf (siehe erste Beispiel).

Hinweis:
Fremdschlüsselattribute sollten nicht angelegt werden. Diese werden automatisch beim Erzeugen des logischen Modells eingetragen.

2.2.4 Entitiy anlegen

Mit dem Tastendruck „STRG+E“ werden die jeweiligen Entities angelegt.

Zur erst wird nach dem Namen gefragt, dieser darf nur aus Buchstaben, Zahlen und dem Underline („_“) bestehen. Es darf kein Leerzeichen verwendet werden. Am Anfang muss auch ein Buchstabe stehen.

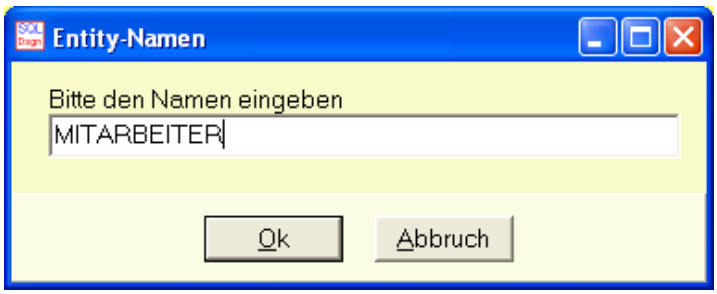

Abbildung 31 Erzeugen eines Entities

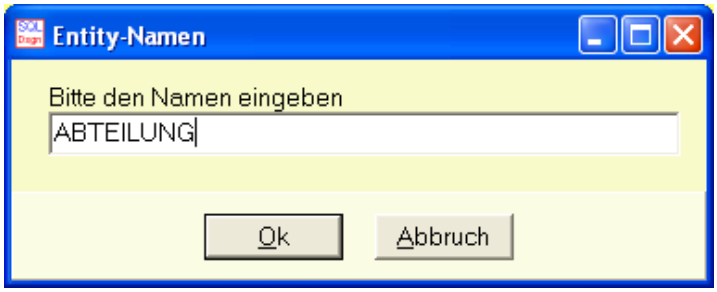

Abbildung 32 Erzeugen eines Entities

Nach der Eingabe der Namen, erscheinen die Entities in der Grafik

Abbildung 33 Neue Entities

2.2.4.1 Attribute des Entity „Mitarbeiter

- MITNR INTEGER
- NAME VCHAR(50)
- VORNAME VCHAR(50)
- STRASSE VCHAR(50)
- PLZ CHAR(5)
- ORT VCHAR(50)

Mit einem Doppelklick kann das Entitiy „Mitarbeiter“ bearbeitet werden. Hier sollten die Einträge ausgefüllt werden. Des Weiteren ist es ratsam, die nähere Beschreibung des Entities im Register „Beschreibung“ vorzunehmen. Die fertigen Attribute sind in der unteren Abbildung dargestellt.

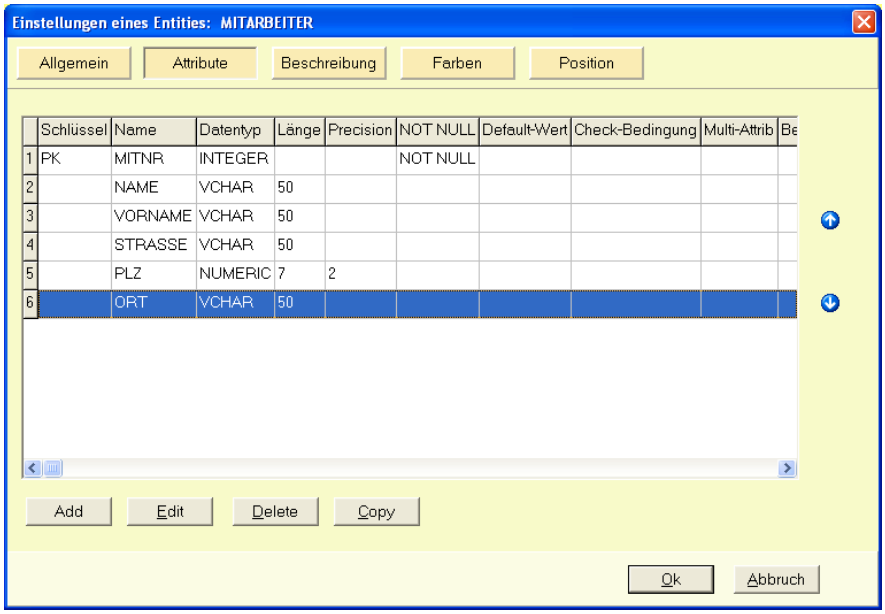

Abbildung 34 Haupt-Eigenschaften eines Entities

2.2.4.2 Attribute des Entity „Abteilung

- NR INTEGER
- NAME VCHAR(50)
- STANDORTE VCHAR(50) MultiAttribut !!!!!

Mit einem Doppelklick kann das Entitiy „Abteilung“ bearbeitet werden. Hier sollten die Einträge ausgefüllt werden. Des Weiteren ist es ratsam, die nähere Beschreibung des Entities im Register „Beschreibung“ vorzunehmen. Die fertigen Attribute sind in der unteren Abbildung dargestellt. Wichtig dabei ist, das das Attribut Standorte als Multiattribut definiert wird. Damit verstößt es gegen die erste Normalform. Im logischen Modell wird dieser Verstoß behoben.

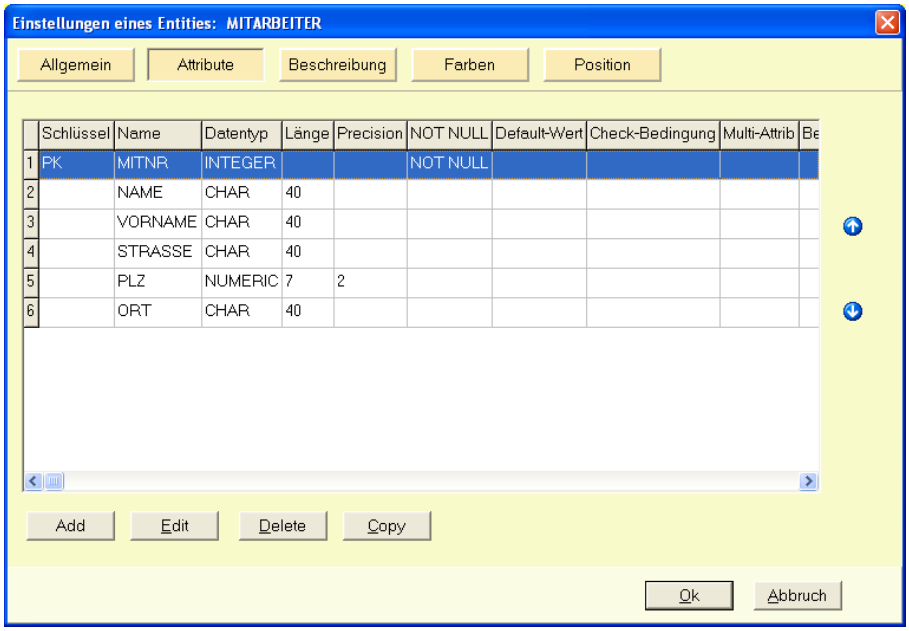

Abbildung 35 Haupt-Eigenschaften eines Entities

Das Attribut „Standorte“ wird als Multiattribut definiert. Dazu wird die Checkbox gesetzt und es müssen die beiden Namen (Entity und Relation) eingetragen werden.

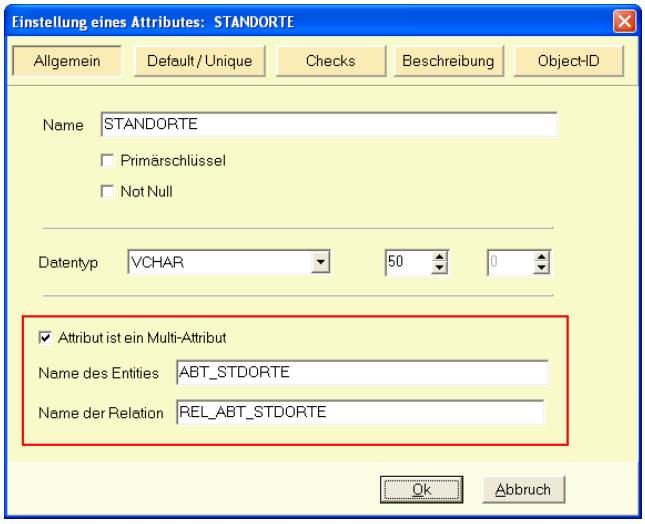

Abbildung 36 Attribut als Multiattribut

Nach diesen Vorbereitungen wird die Beziehung mittels STRG+R erstellt.

Abbildung 37 Auswahl der Entities der Relation

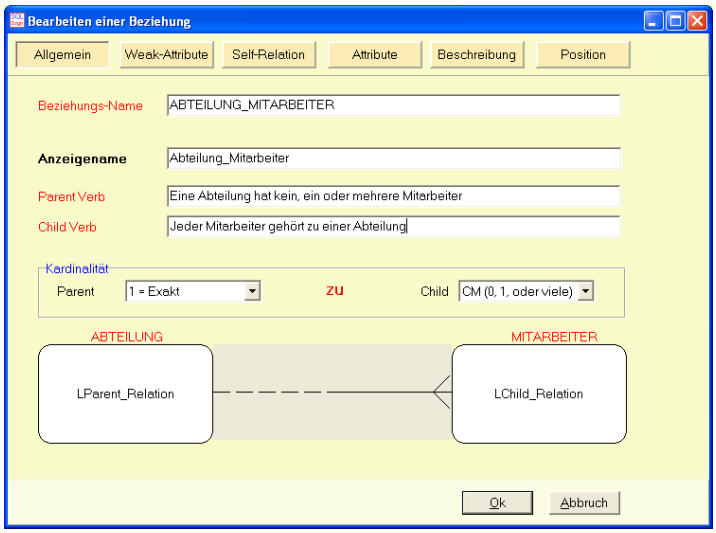

Abbildung 38 Relation Mitarbeiter zu Abteilung

Nach der Eingabe der Relation sieht das konzeptionelle ER-Modell folgendermaßen aus:

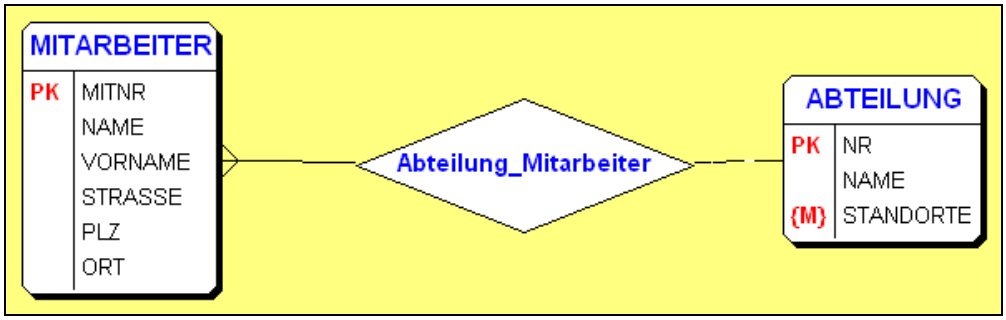

Abbildung 39 Standorte als Multi-Attribute

Interessant ist nun die Umsetzung in das logische Modell. Hier wird nun eine zusätzliche Entität in das Modell eingefügt.

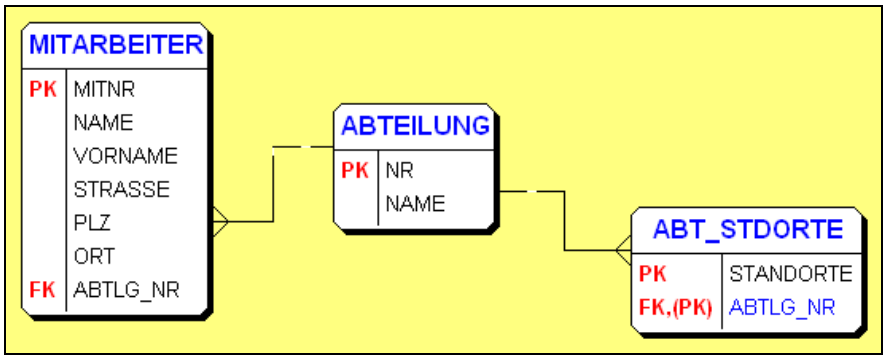

Abbildung 40 Multi-Attribute umgesetzt im logischen Modell

Die Standorte werden als „Weak-Entity“ eingetragen, das heißt, dass der Fremdschlüssel zusätzlich den Primärschlüssel definiert.

2.3 Entwicklung eines konzeptionelles ER-Modells

Das folgende Beispiel entwickelt eine einfache Kundendatenbank:, bei der die Relation ein Attribut besitzt.
Dateiname: Handbuch Bsp3 Kunden_Bestellung_Rel_Attrib.DBK

2.3.1 Entitäten

- Kunden
- Bestellung

2.3.2 Beziehungen:

Kunden zu Bestelllung m:cm

2.3.3 Neues Projekt

Mit der Taste „STRG+M“ wird ein neues Projekt angelegt. Es erscheint der Rahmen mit den Einträgen im Baum und der leeren Diagrammfläche. Im nächsten Schritt müssen nun die Entitäten angelegt werden. Vorab sollte die Definition der Primarykeys definiert werden. Dazu ruft man den Menüeintrag „PROJEKT|Eigenschaften“ auf (siehe erste Beispiel).

Hinweis:
Fremdschlüsselattribute sollten nicht angelegt werden. Diese werden automatisch beim Erzeugen des logischen Modells eingetragen.

2.3.4 Entitiy anlegen

Mit dem Tastendruck „STRG+E“ werden die jeweiligen Entities angelegt.

Nach der Eingabe der Namen, erscheinen die Entities in der Grafik

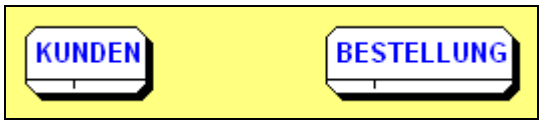

Abbildung 41 Neue Entities

2.3.4.1 Attribute des Entity „Kunden

- KUNDENNR INTEGER
- NAME VCHAR(50)

- VORNAME VCHAR(50)
- STRASSE VCHAR(50)
- PLZ CHAR(5)
- ORT VCHAR(50)

Die untere Abbildung zeigt alle wichtigen Daten des Entity „Kunden“

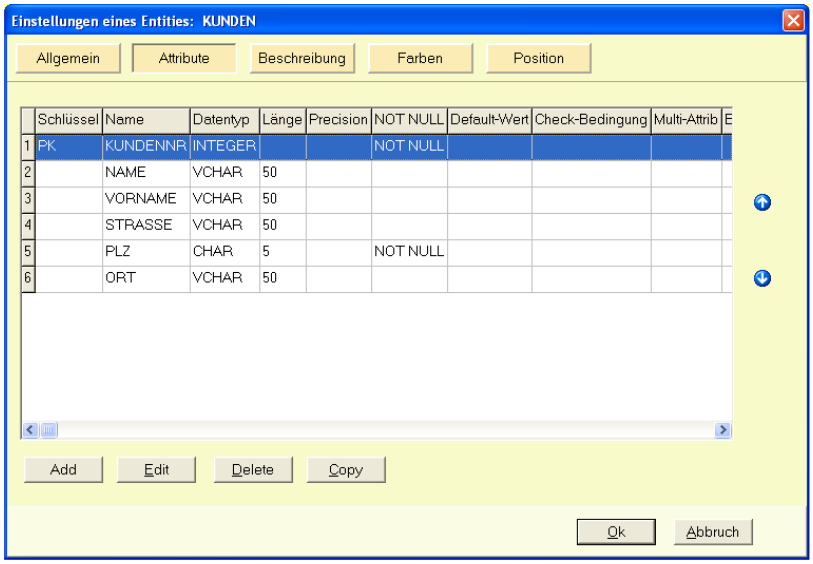

Abbildung 42 Haupt-Eigenschaften des Entities Kunden

2.3.4.2 Attribute des Entity „Bestellung

- ARTIKELNR INTEGER
- MENGE INTEGER

Die untere Abbildung zeigt alle wichtigen Daten des Entity „Bestellung“

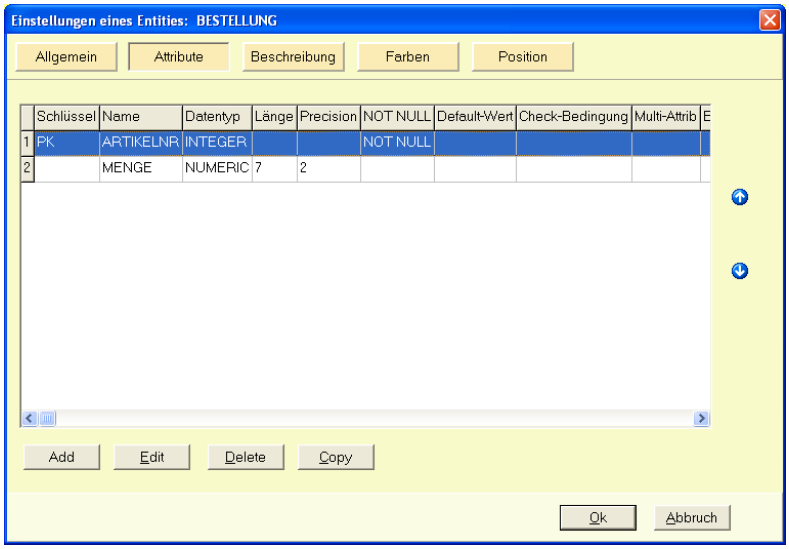

Abbildung 43 Haupt-Eigenschaften eines Entities

Nach diesen Vorbereitungen wird die Beziehung mittels STRG+R erstellt.

Abbildung 44 Auswahl der Entities der Relation

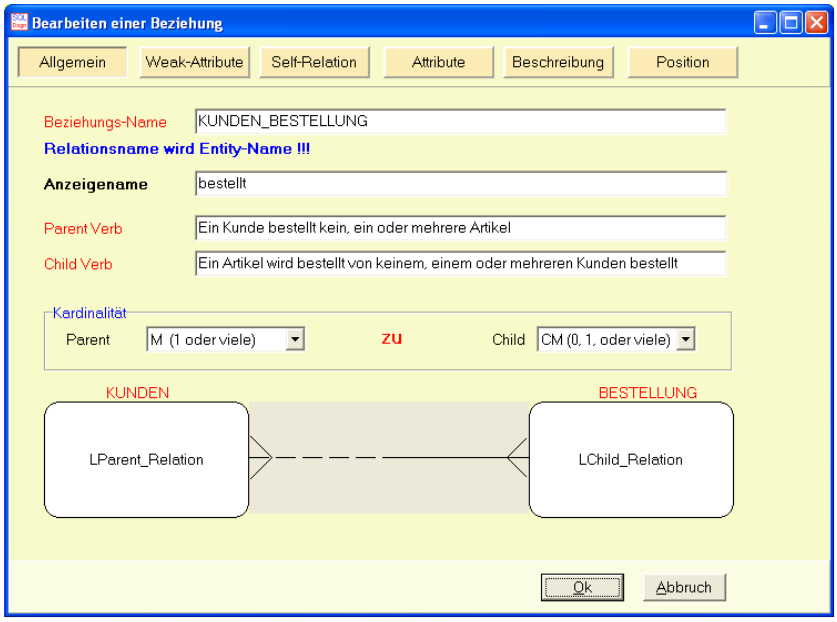

Abbildung 45 Relation Kunden zu Bestellung

Im zweiten Register wird nun das Attribut Datum eingetragen.

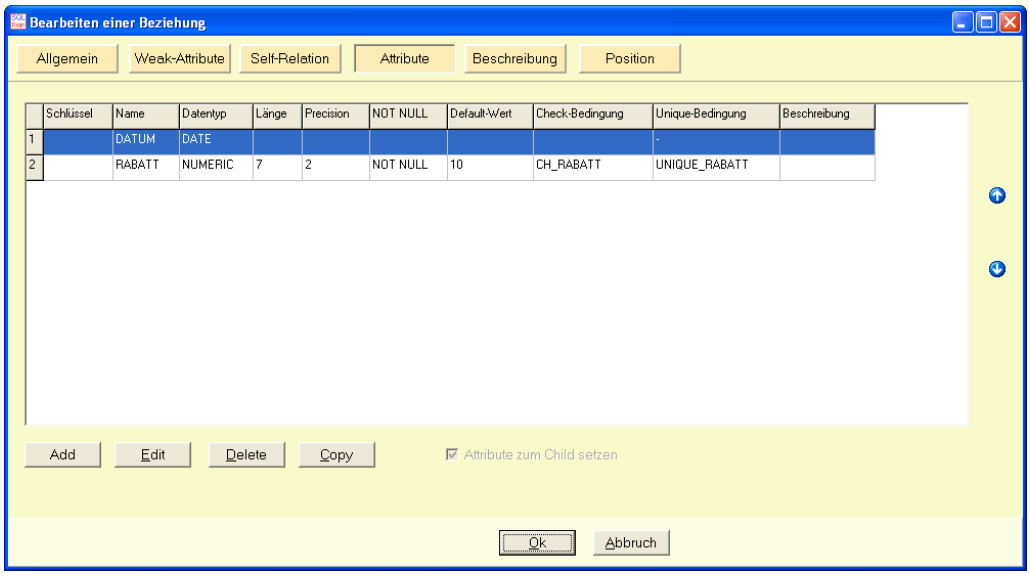

Abbildung 46 Attribut Datum und Rabatt in einer Relation

Im Datum wird das Bestelldatum gespeichert, das Rabattattribut speichert optionale Rabatt Nach der Eingabe der Relation sieht das konzeptionelle ER-Modell folgendermaßen aus:

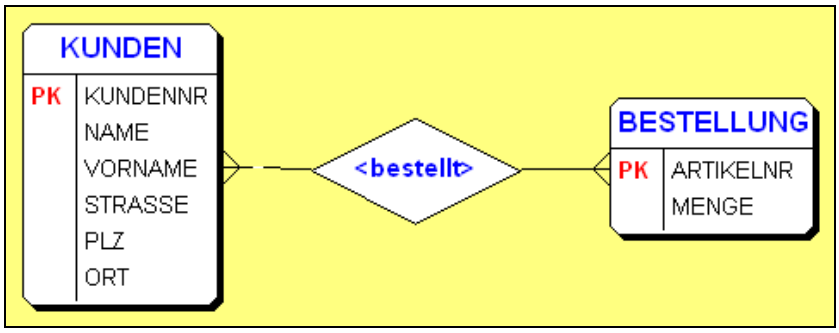

Abbildung 47 Standorte als Multi-Attribute

Im ER-Modell ist durch die Klammerung <> ersichtlich, dass diese Relation Attribute hat. Genaueres erfährt man mit dem Mauskursor. Man geht mit dem Mauskursor über die Relation, das erscheint ein Hinweis, Hint, der die Beziehung beschreibt.

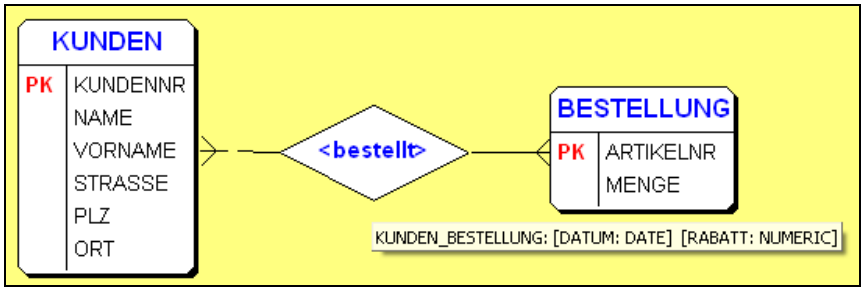

Abbildung 48 Hinweis, Hint, einer Relation

Interessant ist nun die Umsetzung in das logische Modell. Hier wird nun eine zusätzliche Entität in das Modell eingefügt.

Abbildung 49 Relationen-Attribute umgesetzt im logischen Modell

Bei einer einfachen Beziehung werden die Attribute im „Child“ eingetragen.

2.4 Entwicklung eines konzeptionelles ER-Modells

Das folgende Beispiel entwickelt eine einfache Lieferanten-Datenbank:, bei der eine ternäre Relation eingesetzt wird.
Dateiname: Handbuch Bsp4 Ternäre Beziehung.DBK

Eine Beziehung zwischen zwei Attributen ist die Regel. Manchmal benötigt man aber eine Beziehung, bei der drei oder mehrere Entities vertreten sind. Diese Sonderform ist nun im Designer als ternäre-Beziehung realisiert. Höherwertige Beziehung können nachprogrammiert werden. Ziel ist es, im konzeptionellen Modell schon diese Beziehung deutlich zu darzustellen, ohne die komplexere Darstellung im logischen Modell zu betrachten.

2.4.1 Entitäten

- Lieferant
- Teil
- Projekt

2.4.2 Beziehungen:

Alle Beziehungen als 1:cm

2.4.3 Neues Projekt

Mit der Taste „STRG+M“ wird ein neues Projekt angelegt. Es erscheint der Rahmen mit den Einträgen im Baum und der leeren Diagrammfläche. Im nächsten Schritt müssen nun die Entitäten angelegt werden. Vorab sollte die Definition der Primarykeys definiert werden. Dazu ruft man den Menüeintrag „PROJEKT|Eigenschaften“ auf (siehe erste Beispiel).

Hinweis:
Fremdschlüsselattribute sollten nicht angelegt werden. Diese werden automatisch beim Erzeugen des logischen Modells eingetragen.

2.4.4 Entitiy anlegen

Mit dem Tastendruck „STRG+E“ werden die jeweiligen Entities angelegt.

Nach der Eingabe der Namen, erscheinen die Entities in der Grafik

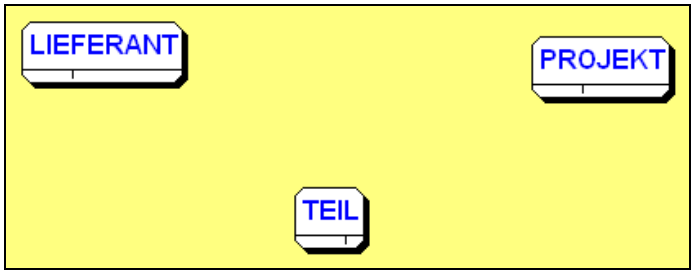

Abbildung 50 Neue Entities

2.4.4.1 Attribute des Entity „Lieferant

- L_NR INTEGER
- LNAME VCHAR(50)

Die untere Abbildung zeigt alle wichtigen Daten des Entity „Lieferant“

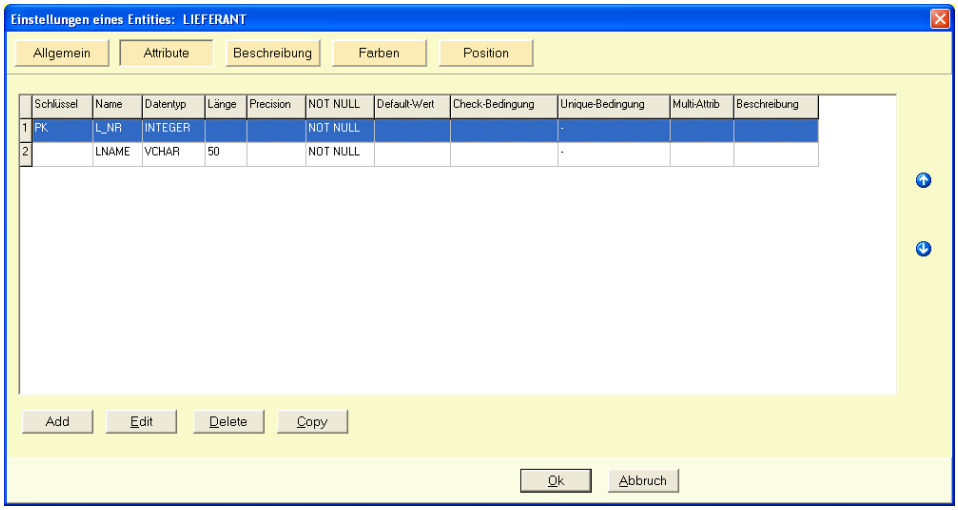

Abbildung 51 Haupt-Eigenschaften des Entities Lieferant

2.4.4.2 Attribute des Entity „Teil“

- T_NR INTEGER
- TNAME VCHAR(50)

Die untere Abbildung zeigt alle wichtigen Daten des Entity „Teil“

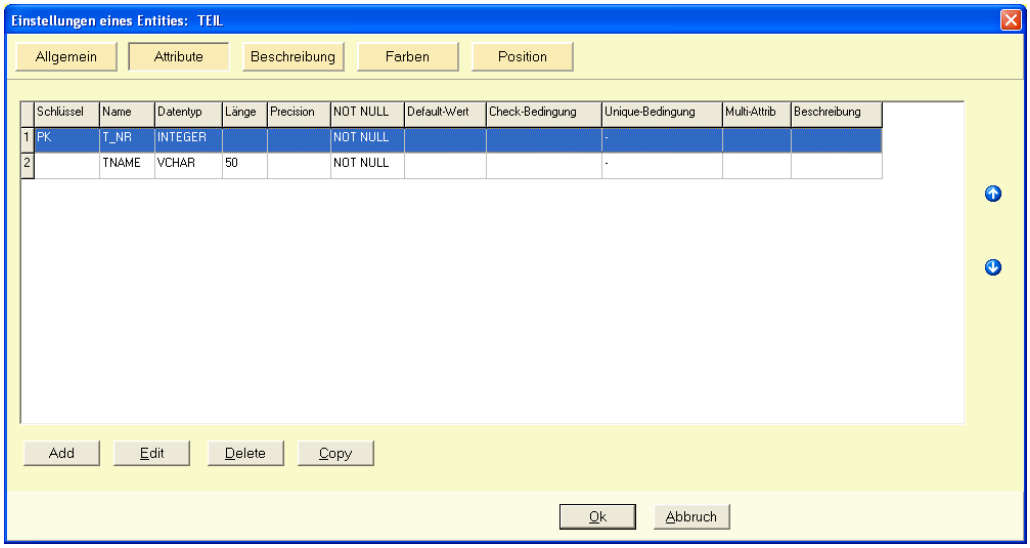

Abbildung 52 Haupt-Eigenschaften des Entities Teil

2.4.4.3 Attribute des Entity „Projekt“

- P_NR INTEGER
- PNAME VCHAR(50)

Die untere Abbildung zeigt alle wichtigen Daten des Entity „Projekt“

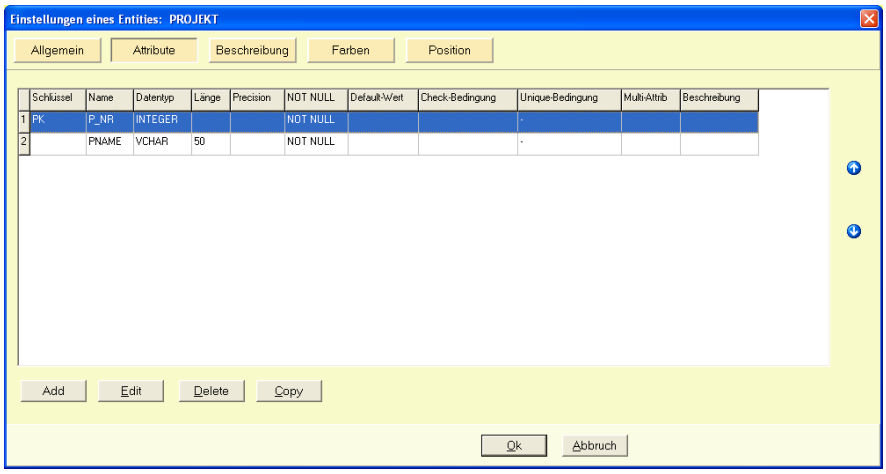

Abbildung 53 Haupt-Eigenschaften des Entities Teil

Nach diesen Vorbereitungen wird die ternäre Beziehung mittels STRG+T oder Menü „Einfügen“, Eintrag „Ternäre Beziehung“ erstellt.

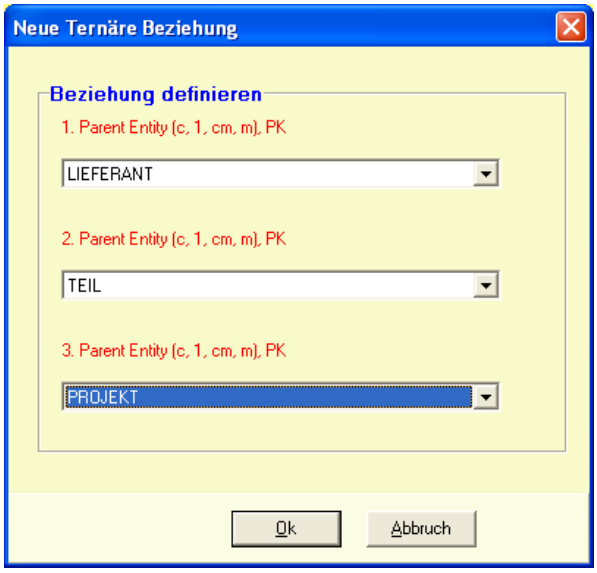

Abbildung 54 Auswahl der Entities der ternären Relation

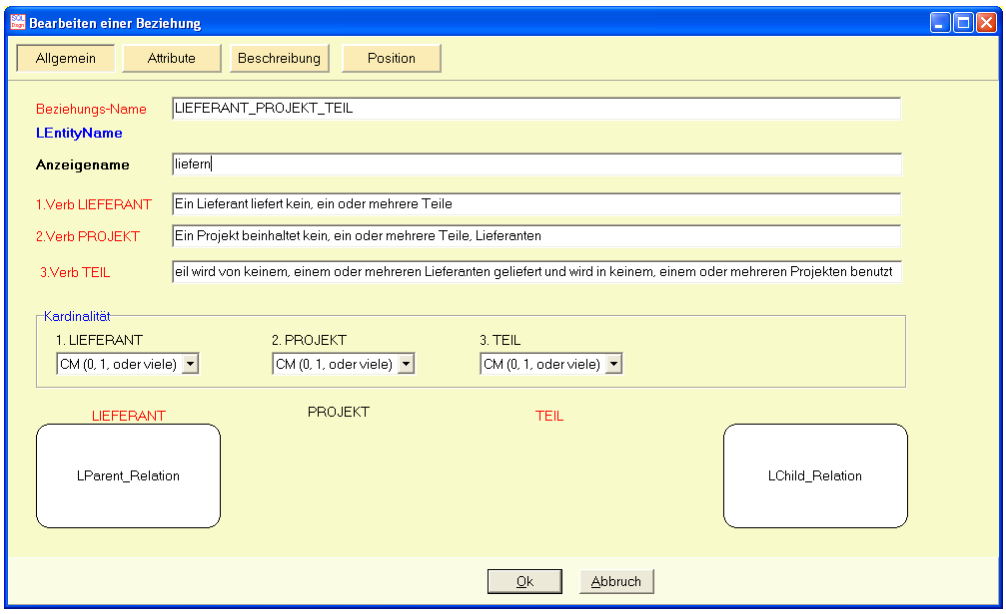

Abbildung 55 Relation Kunden zu Bestellung

Beziehungsverben:

- Ein Lieferant liefert kein, ein oder mehrere Teile
- Ein Projekt beinhaltet kein, ein oder mehrere Teile, Lieferanten
- Ein Teil wird von keinem, einem oder mehreren Lieferanten geliefert und wird in keinem, einem oder mehreren Projekten benutzt

Im zweiten Register werden nun die beiden Attribute Datum und Menge eingetragen.

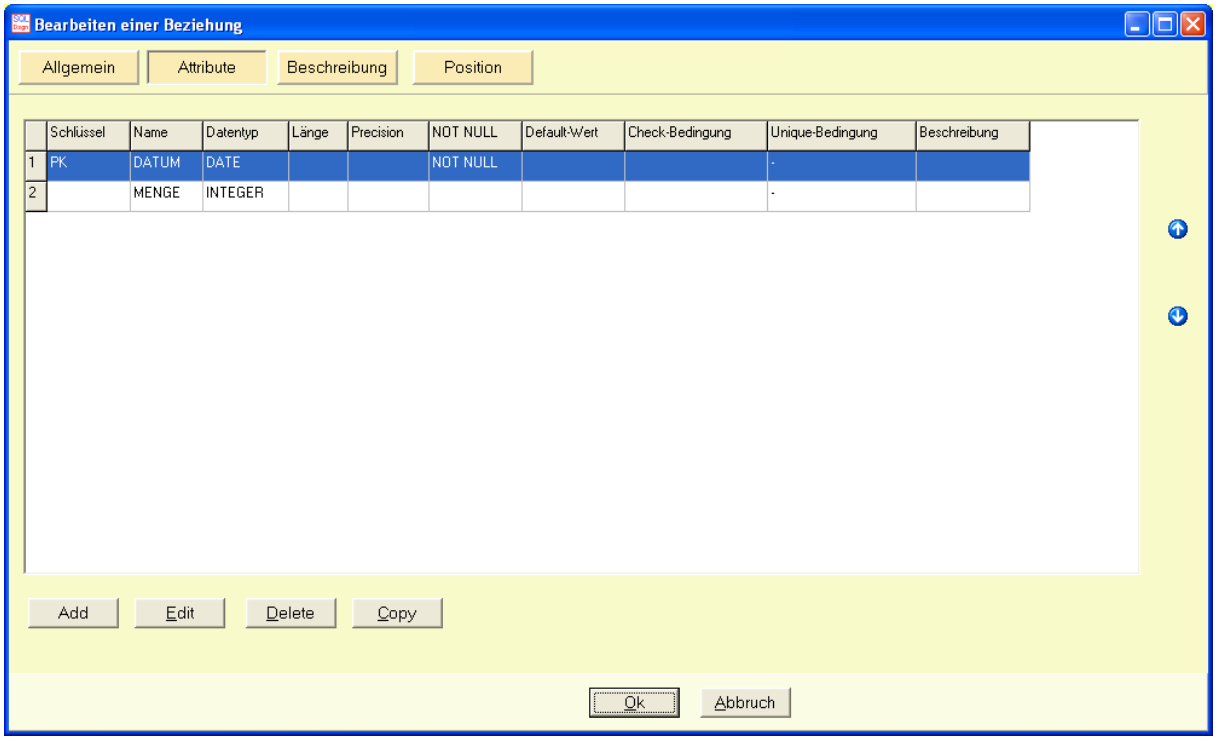

Abbildung 56 Attribut Datum und Menge in einer ternären Relation

Nach der Eingabe der Relation sieht das konzeptionelle ER-Modell folgendermaßen aus:

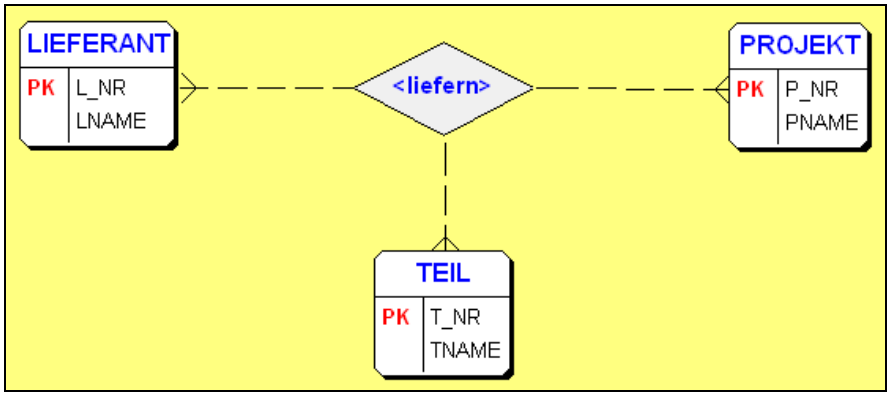

Abbildung 57 ER-Modell einer ternären Beziehung

Im ER-Modell ist durch die Klammerung <> ersichtlich, dass diese Relation Attribute hat. Genaueres erfährt man mit dem Mauskursor. Man geht mit dem Mauskursor über die Relation, das erscheint ein Hinweis, Hint, der die Beziehung beschreibt.

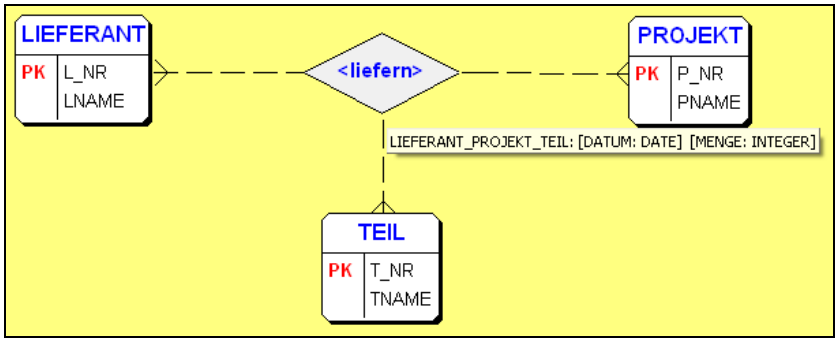

Abbildung 58 Hinweis, Hint, einer Relation

Interessant ist nun die Umsetzung in das logische Modell. Hier wird nun eine zusätzliche Entität in das Modell eingefügt.

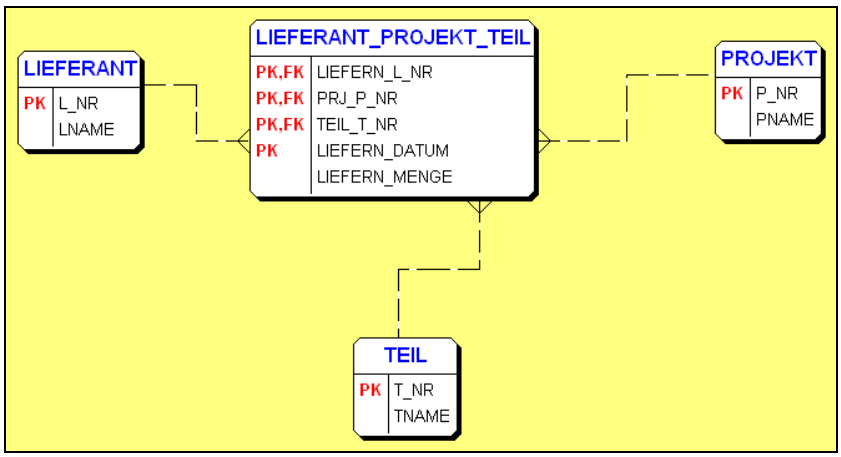

Abbildung 59 Relationen-Attribute umgesetzt im logischen Modell

Es wird eine zusätzliche Entität eingefügt, die aber alle Primärschlüssel als Fremdschlüssel einfügt. Zusätzlich werden die Attribute der ternären Relation in das neue Entity eingetragen.

2.5 Beispiel5, Entwicklung eines logischen ER-Modell

Dieses Beispiel zeigt die Entwicklung eines relativ einfachen logischen Modells. Die Vorgaben werden vorab aufgelistet, so dass man übungshalber dieses Modell auch ohne diese Vorlage entwickeln kann.

2.5.1 Aufgabenstellung des fünften Beispiels

Entwickelt werden soll ein Modell zur Modellierung einer Studentendatenbank. Folgende Entities wurden definiert:

- Student
- Fachbereich
- Vorlesung

Teilaufgaben:

- Definieren der Attribute
- Definieren der Beziehungen (es sollten zwei unterschiedliche Arten sein)
- Einbau von Prüfbedingungen
- Generieren des SQL-Skriptes

2.5.2 Neues Projekt

Mit der Taste „STRG+N“ wird ein neues Projekt angelegt. Es erscheint ein Dialogfenster, in dem man die Zieldatenbank auswählt (Abbildung 60).

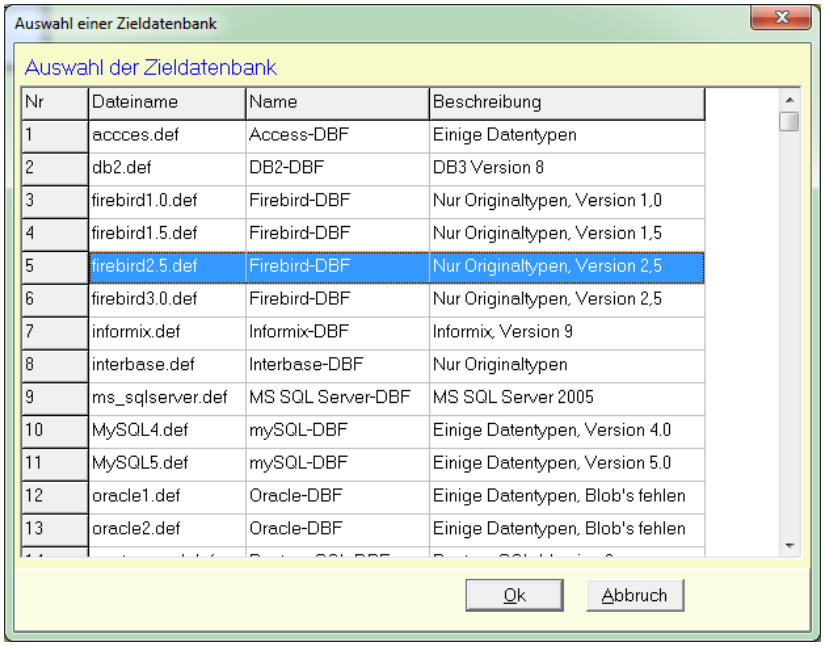

Abbildung 60 Auswahl der Datenbank

Im nächsten Schritt ruft man im Menü „Projekt“ den Eintrag „Eigenschaften“ auf. Im Dialogfenster kann man die notwendigen Einträge vornehmen. Besonders erwähnenswert ist die Auswahl der möglichen Darstellungen.

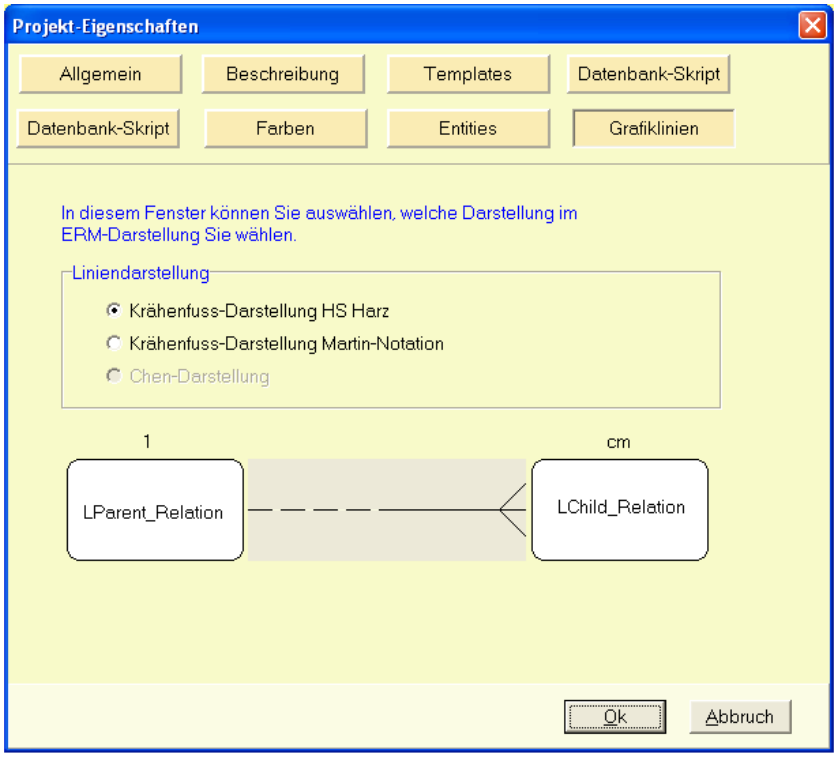

Abbildung 61 Auswahl der Darstellungen

Im ersten Schritt sollte die Definition der Primarykeys definiert werden. Dazu ruft man den Menüeintrag „PROJEKT|Eigenschaften“ auf.

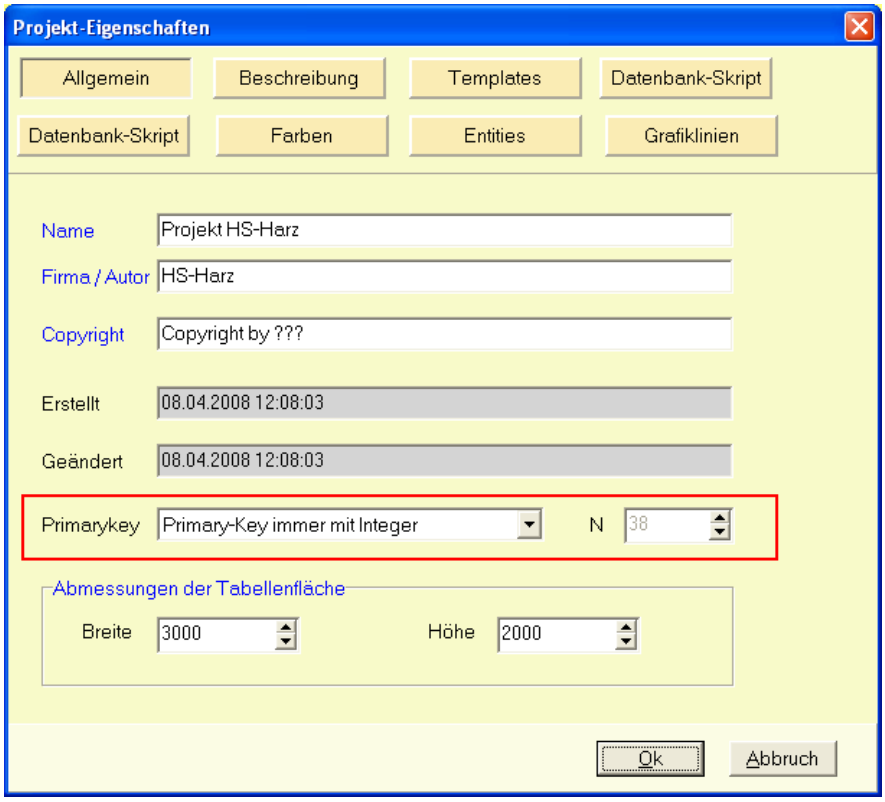

Abbildung 62 Projekteigenschaften (Integer vs. Max Numeric)

In der Comboliste kann man entscheiden, ob man der Datentyp eines Primarykeys ein Integer oder ein Numeric/Number-Datentyp ist.

2.5.3 Entities anlegen

Mit dem Tastendruck „STRG+E“ werden die jeweiligen Entities angelegt.

Hinweis:
Dabei sollten die Fremdschlüssel mit anderen Entities noch nicht eingefügt werden!!!

Zur erst wird nach dem Namen gefragt, dieser darf nur aus Buchstaben, Zahlen und dem Underline („_“) bestehen. Es darf kein Leerzeichen verwendet werden. Am Anfang muss auch ein Buchstabe stehen.

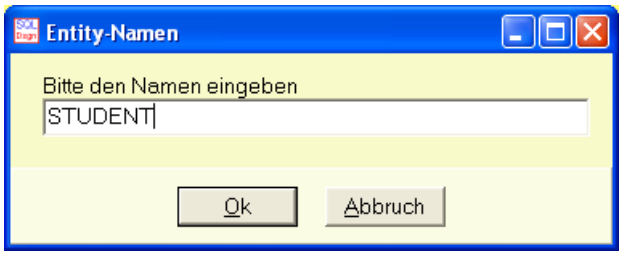

Abbildung 63 Erzeugen eines Entities

Nach der Eingabe der Namen, erscheint das Entity in der Grafik. Nun werden alle weiteren Entitäten eingetragen (siehe Abbildung 64).

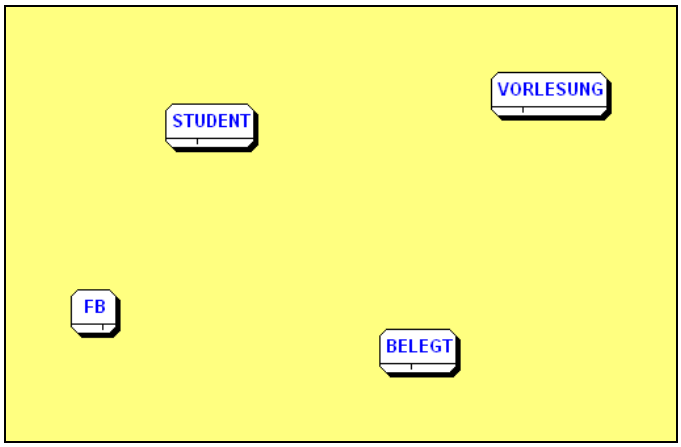

Abbildung 64 Entites des zweiten Beispiels

Mit einem Doppelklick werden nun die einzelnen Entitäten bearbeiten und die Attribute eingetragen (Abbildung 65). Hier sollten die Einträge ausgefüllt werden. Des Weiteren ist es ratsam, die nähere Beschreibung des Entities im Register „Beschreibung“ vorzunehmen. Im zweiten Register, siehe Abbildung 65, werden dann die Attribute eingetragen.

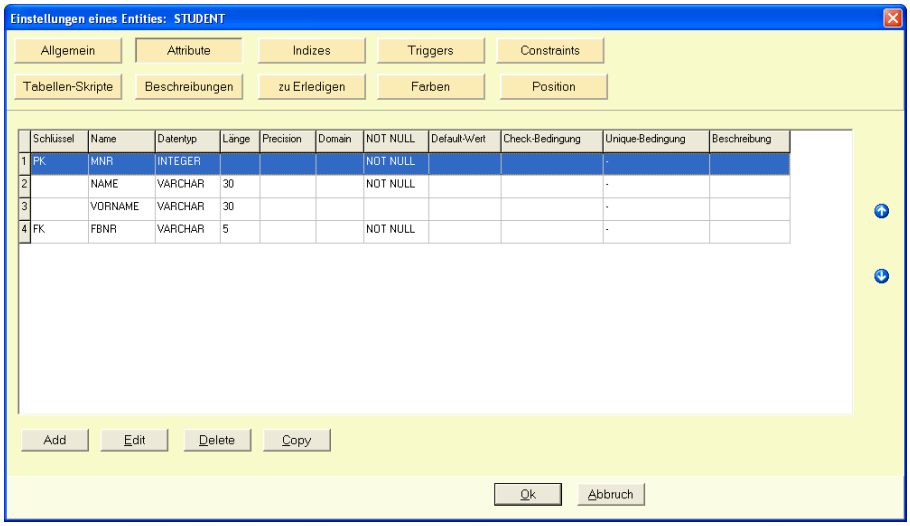

Abbildung 65 Haupt-Eigenschaften eines Entities

Das zweite Register verwaltet die Attribute. Mit den unteren Schaltern kann man diese bearbeiten, löschen und erzeugen. Die Schalter rechts an der Seite verschieben die Reihenfolge.

Am Schluss könnte das Projekt dann so aussehen:

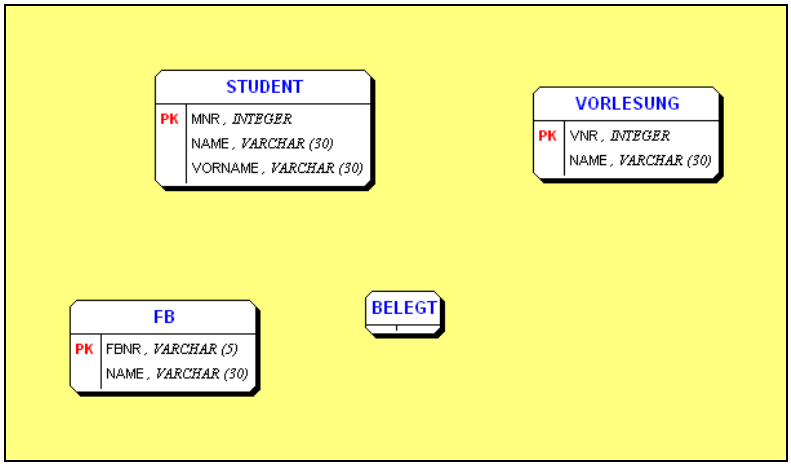

Abbildung 66 Fertige Entities des fünften Beispiels

Bemerkungen:

- Die Beziehung „Student“ zu „FB“ ist eine normale 1:cm-Beziehung.
- Die Beziehung „Student“ zu „Vorlesung“ ist eine m:m-Beziehung, deshalb musste ein drittes Entity „belegt“ eingeführt werden.

Im nächsten Abschnitt werden die Beziehungen eingetragen.

2.5.4 Beziehungen anlegen

Dazu ruft man aus dem Hauptmenü „Einfügen“ die Funktion „Beziehung“ auf. Es erscheint folgendes Fenster:

Abbildung 67 Erstellen einer Beziehung

In beiden Listen sind alle Entities eingetragen. In der linken Liste wählt man den Parent-Entity aus. Das wäre hier in diesem Beispiel „FB“. In der rechten Liste wird die Beziehung eingetragen. Hier also „Student“.

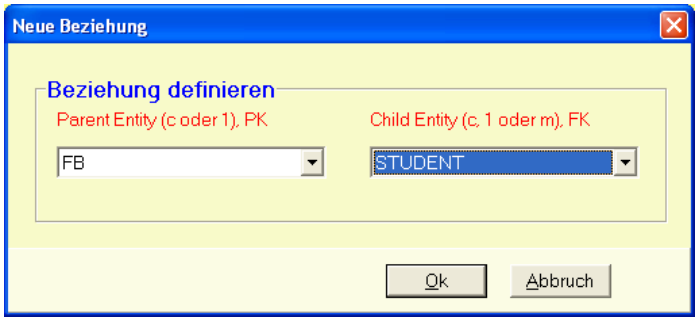

Abbildung 68 Korrekte Einstellung für das Beispiel

Nach dem Betätigen des Schalters „Ok“ erscheint der Relationsdialog, indem die Beziehungsdaten eingetragen werden. Danach wird der Primarykey „FBNR“ automatisch als Fremdschlüssel in das Entity „Student“ eingefügt. Die Beziehungsverben sollten vervollständigt werden.

Die Abbildung 69 zeigt die endgültigen Eingaben. Mit dem Schalter „Ok“ wird die Beziehung eingefügt. Danach erscheint das eigentliche Fenster zur Definition bzw. zur Bearbeitung einer Beziehung.

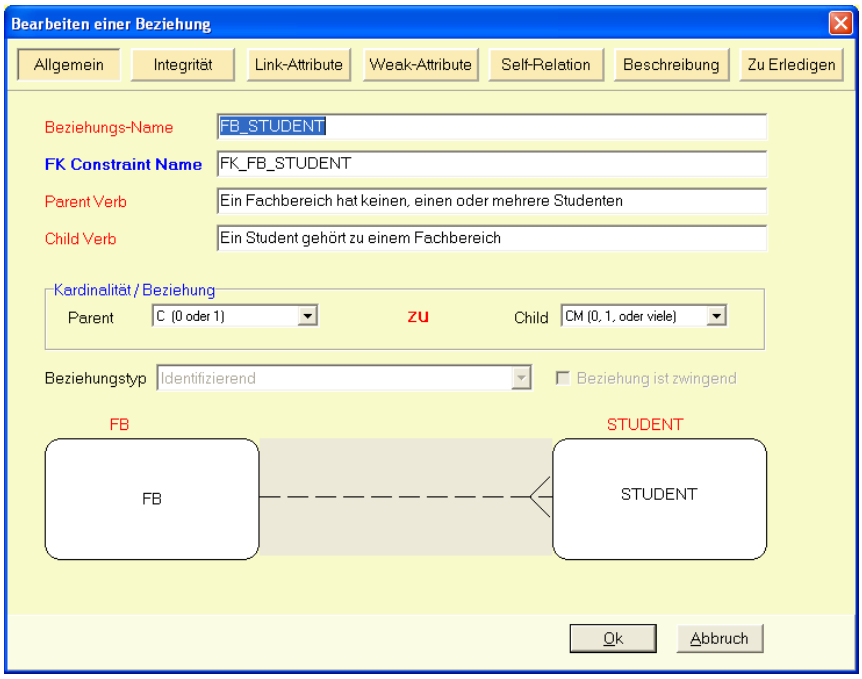

Abbildung 69 Definition der Beziehung

Beziehungstexte:

- Ein Fachbereich hat keinen, einen oder mehrere Studenten
- Ein Student gehört zu einem Fachbereich

Im ersten Register werden die Grundeinstellungen (Name, Beziehungstyp) eingetragen. Wichtig sind hier die Einträge „Parent-Verb“ und „Child-Verb“. Sie zeigen den Charakter der Beziehung an. Das Parent-Verb beschreibt die Beziehung durch Wörter wie „Kunde hat“ etc.

Das zweite Register wird zurzeit nicht unterstützt.

Im dritten Register werden die Attribute verknüpft, mit denen die Beziehung definiert wird. Wie oben erwähnt, sind keine Fremdattribute im Child-Entity („Emp“) eingetragen. Wechselt man in das dritte Register, wird man einmal gefragt, ob alle Primärattribute an das Child übertragen werden sollen (Abbildung 70).

Sinnvoll ist es, diese Frage zu bejahen. Dann werden alle Primär-Attribute in das Child-Entity eingefügt. Diese Attribute erhalten dann die Eigenschaft „Fremdschlüssel“. Die Abbildung 71 zeigt die fertige Liste. Sie können aber auch die Attribute manuell verknüpfen. Dazu markiert man in der linken und in der rechten Tabelle jeweils das gewünschte Attribut. Mit dem Schalter „Connect“ werden diese in die obere Tabelle eingefügt. Mit dem Schalter „DisConnect“ wird die ausgewählte Verbindung wieder gelöst. Die Datentypen der beiden Attributen müssen natürlich exakt übereinstimmen und vorher in den Entities eingetragen worden sein.

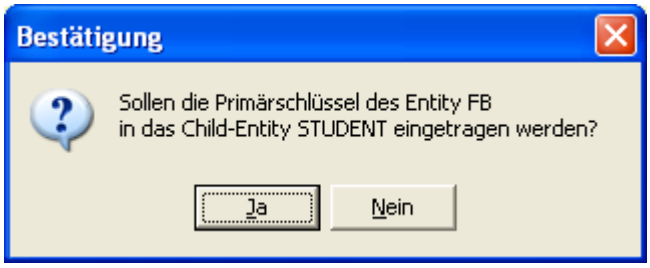

Abbildung 70 Übertragen der Primär-Attribute

Die untere Abbildung zeigt die Beziehung, nachdem die Primär-Attribute in das Child-Entity übertragen wurden.

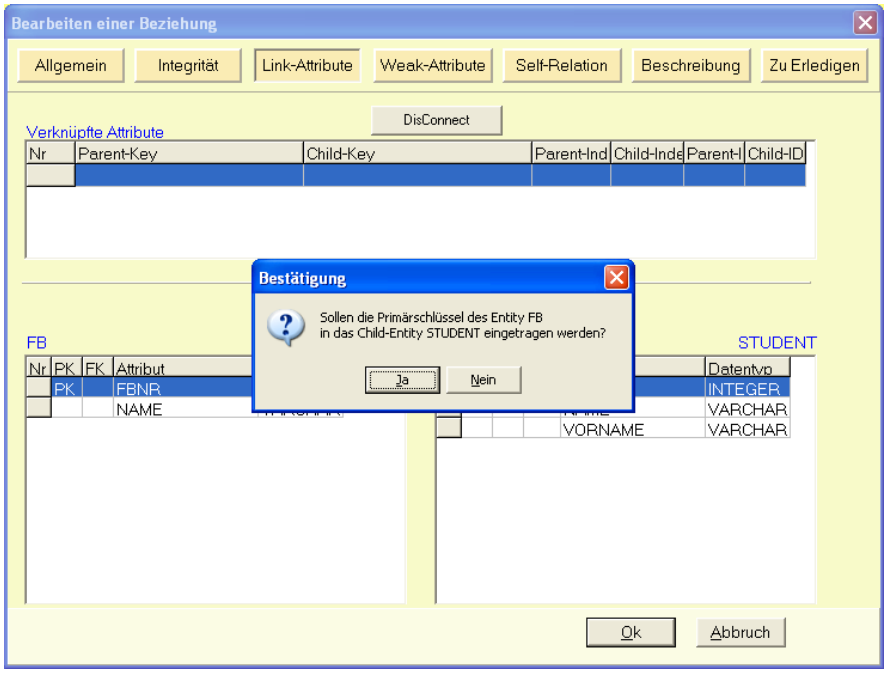

Abbildung 71 Übertragen eines Fremdschlüssels

Das vierte Register „Weak-Attribute“ dient der Definition einer Beziehung einer Weak-Entity (siehe Kapitel 4.1, Seite 100).

Mit dem Betätigen des Schalters „Ok“ wird die Beziehung eingetragen und angezeigt. Mit der Taste F6 erhält man eine Liste aller Beziehungen.

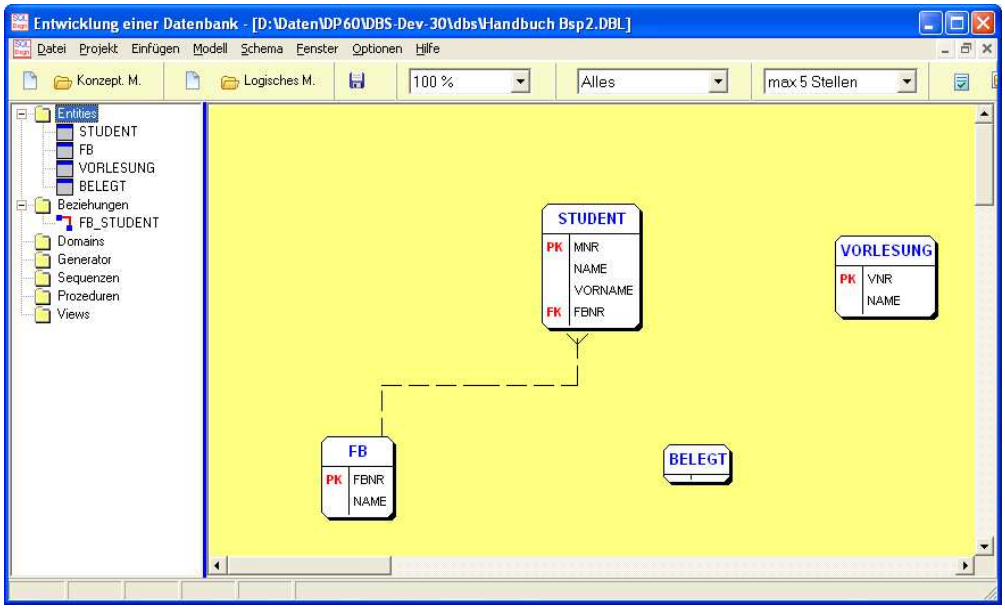

Abbildung 72 Erste Beziehung im fünften Beispiel

2.5.5 Beziehung der Vorlesungen

Die Beziehung der Vorlesungen entspricht einer cm:cn-Beziehung. Dazu existiert das neue Entity „Belegt“. Es müssen nun zwei Relationen eingetragen werden.

a) Student zu Belegt
b) Vorlesungen zu Belegt

Abbildung 73 Beziehung Student zu Belegt

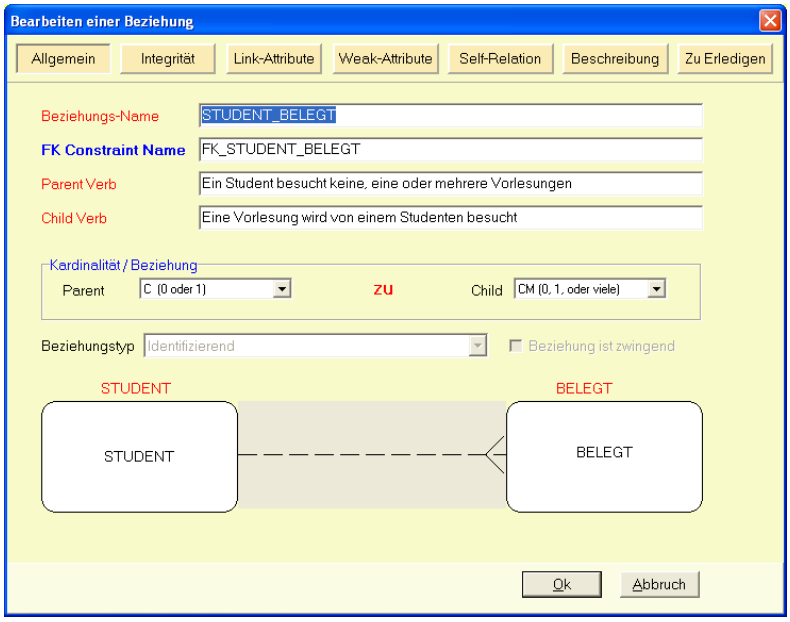

Abbildung 74 Beziehung „Student“ zu „Belegt Vorlesung“

Das nächste Dialogfenster zeigt die korrekte Einstellung. Links das Parent-Entity, rechts das Child-Entity.

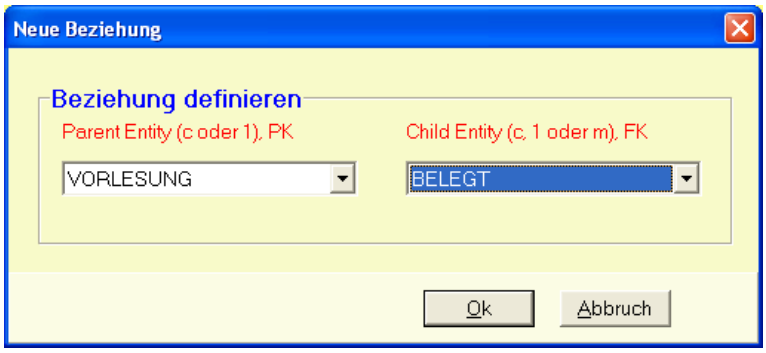

Abbildung 75 Aufruf Beziehung Vorlesung zu Belegt

Eingetragen werden dann die Beschreibungen und die Kardinalitäten.

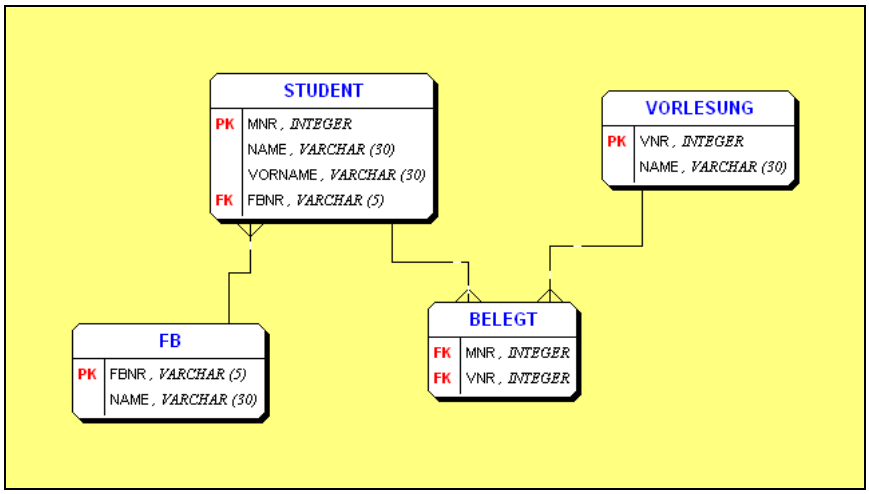

Abbildung 76 Eintrag der Beziehung "Vorlesung" zu "Belegt"

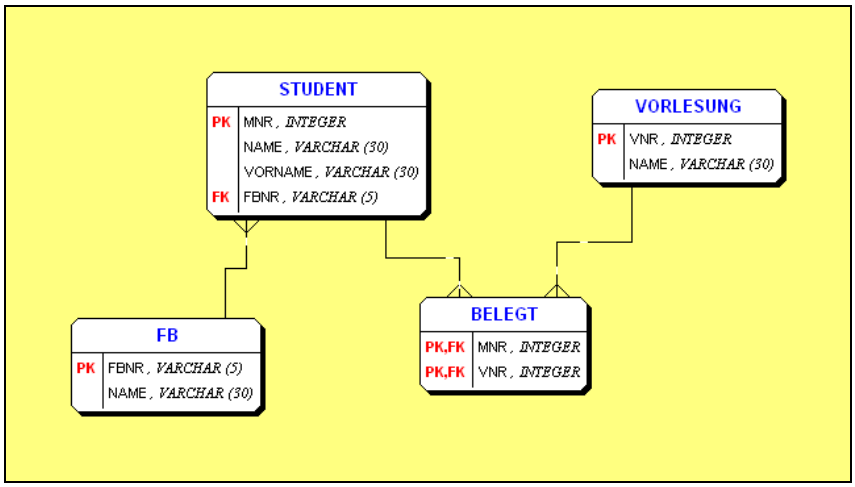

Abbildung 77 ER-Modell nach den Beziehungen der Hobbies

Im zusätzlichen Entity „Belegt“ wurden beide Attribute als Fremdschlüssel eingetragen. Sinnvoll ist es nun, diese auch als Primary-Key zu definieren.

2.5.6 Weitere Schritte

Als weitere Schritte wären Prüfbedingungen möglich. Dazu zählen:

- Check-Constraint (Prüfbedingung des aktuellen Wertes)
- Unique-Constraint (Eindeutigkeit eines normalen Attributs)

2.5.7 Generierung einer Datenbank

Mit dem Menü „Schema“ und dem Eintrag „Generierung Datenbank“ wird die Erzeugung des SQL-Scriptes angestoßen. Als Kurztaste existiert die Taste F9. Es erscheint folgendes Fenster (siehe Abbildung 78). In der obersten Zeile wird der Dateiname eingetragen. Der nächste Teil definiert die Objekte, die generiert werden sollen. Im letzten Teil sind zusätzliche Optionen vorhanden.

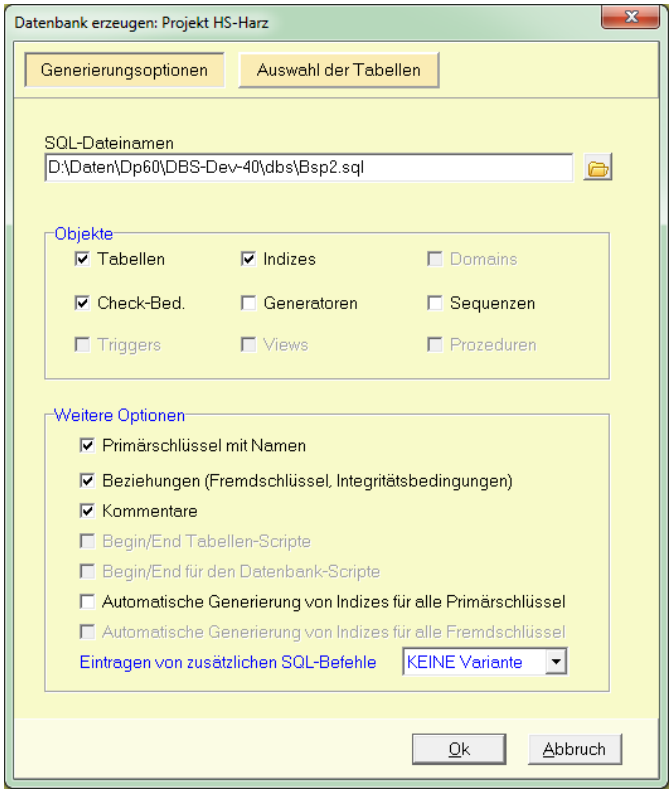

Abbildung 78 Erzeugung einer Datenbank

Mit dem Schalter „Ok“ wird das Script erzeugt und im Notepad dargestellt.

Ergebnis mit SQL-Befehlen:

```
/*==============================================================*/
/* Project Filename: C:\Daten\Handbuch Bsp2.sql                  */
/* Project Name:     Beispiel 2                                  */
/* Author:           Michael Wilhelm                             */
/* DBMS:             Firebird-DBF                                */
/* Copyright:        Copyright by Michael Wilhelm                */
/* Generated on:     09.05.2007 17:59:12                         */
/*==============================================================*/


/*==============================================================*/
/*  Tables                                                      */
/*==============================================================*/
CREATE TABLE STUDENT (
  MNR  INTEGER  NOT NULL,
  NAME  VARCHAR(30)  NOT NULL,
  VORNAME  VARCHAR(30),
  FBNR  VARCHAR(5)  NOT NULL,

  PRIMARY KEY (MNR)
);

CREATE TABLE FB (
  FBNR  VARCHAR(5)  NOT NULL,
  NAME  VARCHAR(30)  NOT NULL,

  PRIMARY KEY (FBNR)
);

CREATE TABLE VORLESUNG (
  VNR  INTEGER  NOT NULL,
  NAME  VARCHAR(30)  NOT NULL,

  PRIMARY KEY (VNR)
);

CREATE TABLE BELEGT (
  MNR  INTEGER  NOT NULL,
  VNR  INTEGER  NOT NULL,

  PRIMARY KEY (MNR, VNR )
);



/*==============================================================*/
/*  Foreign Keys                                                */
/*==============================================================*/
ALTER TABLE STUDENT
 ADD CONSTRAINT FK_FB_STUDENT FOREIGN KEY (FBNR) REFERENCES FB(FBNR);

ALTER TABLE BELEGT
 ADD CONSTRAINT FK_STUDENT_BELEGT FOREIGN KEY (MNR)
    REFERENCES STUDENT(MNR);

ALTER TABLE BELEGT
 ADD CONSTRAINT FK_VORLESUNG_BELEGT FOREIGN KEY (VNR)
    REFERENCES VORLESUNG(VNR);
```

2.5.8 Erzeugen der Datenbank

Mit Aufruf einer Datenbank-Konsole à la IBOConsole, Workbench, wird nun die Datenbank erzeugt.

Beispiel mit IBOConsole:

1. Schritt: Aufruf des Programms

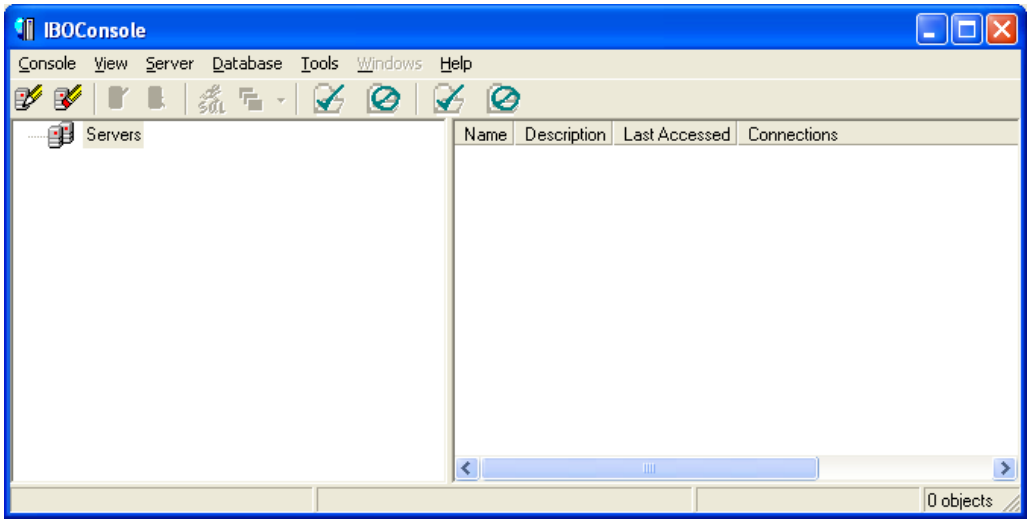

Abbildung 79 Start der IBO-Console

2. Schritt: Doppelklick auf den Eintrag „Servers“

Eintrag: „Local Server“
Eingabe des Usernamens: „sysdba“
Eingabe des Passworts: „masterkey“

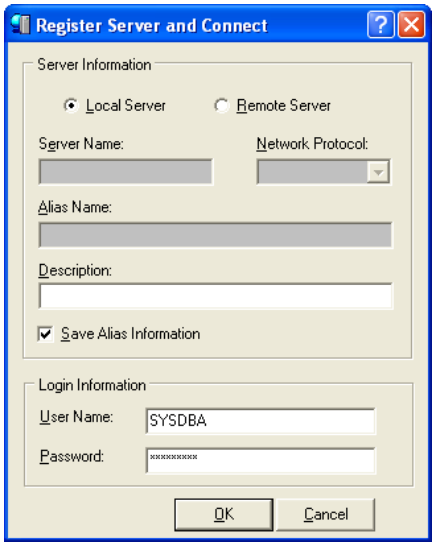

Abbildung 80 Anmelden an die Datenbank

3. Schritt: Menü Database, Eintrag „Create Database“

Datenbankname: c:\daten\bsp5.fdb

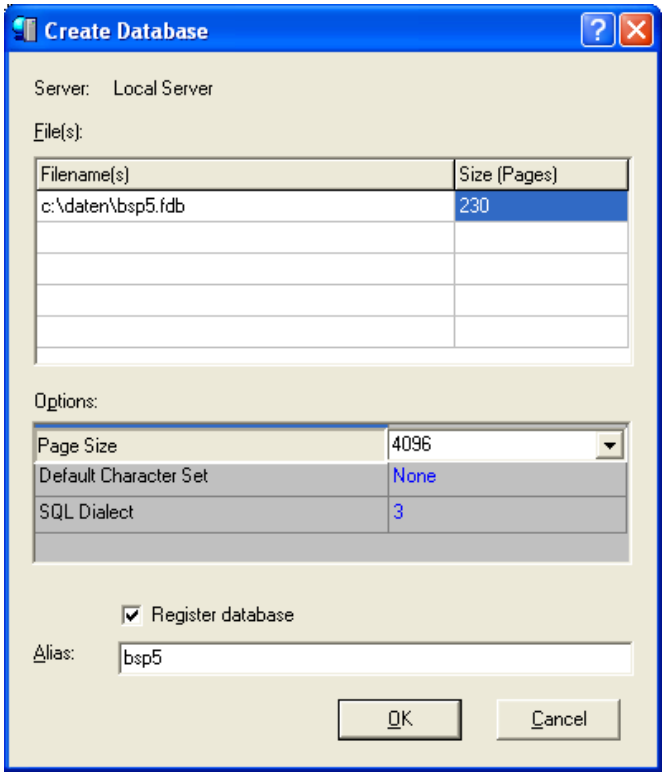

Abbildung 81 Eintrag für eine neue Datenbank

In der linken Liste steht der Dateinamen mit voller Pfadangabe. Rechts daneben wird die Anzahl der Seiten eingetragen. Hier muss mindestens eine 230 als Wert eingetragen werden. In den Abschnitt „Options“ wird die Clustergröße und der SQL-Dialekt definiert. Ganz unten wird der Alias-Namen eingetragen. Mit diesen kann man per Programm eine Datenbank ansprechen, ohne dass man weiss, in welchem Pfad die Datenbank sich befindet. Diese Eigenschaft ist aber nur für Programmentwickler interessant.

Danach sollte ein neuer Eintrag in der Datenbank vorhanden sein.

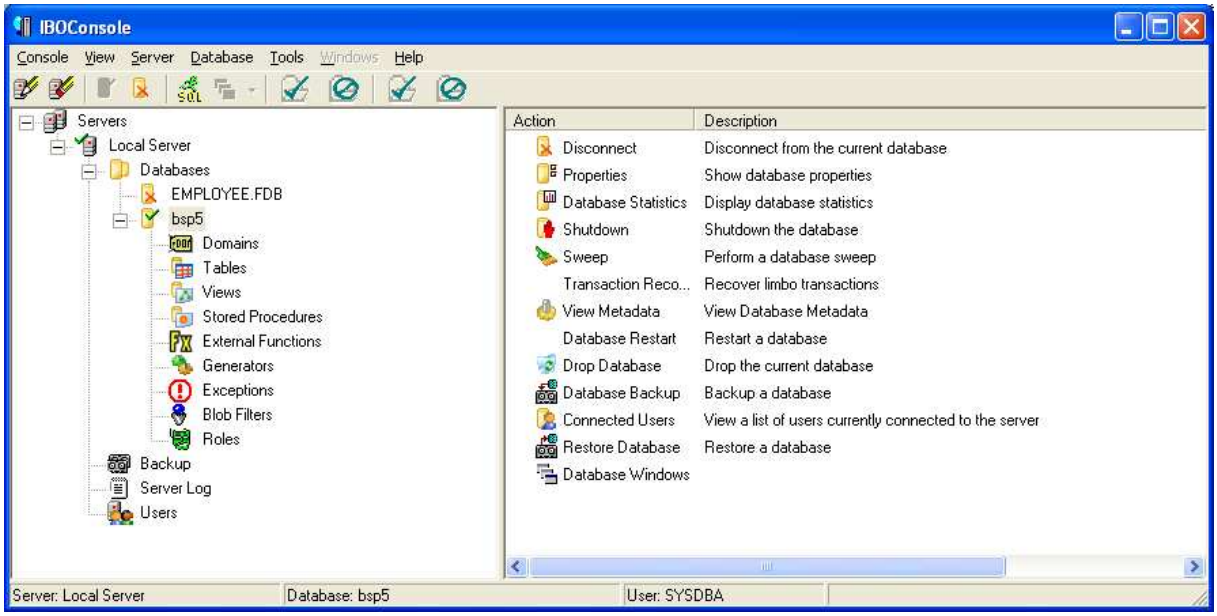

Abbildung 82 neue Datenbank des fünften Beispiels

Mit dem Menü „Tools“, Eintrag „Interactive SQL“ oder dem entsprechendem Symbol öffnet sich der SQL-Editor.

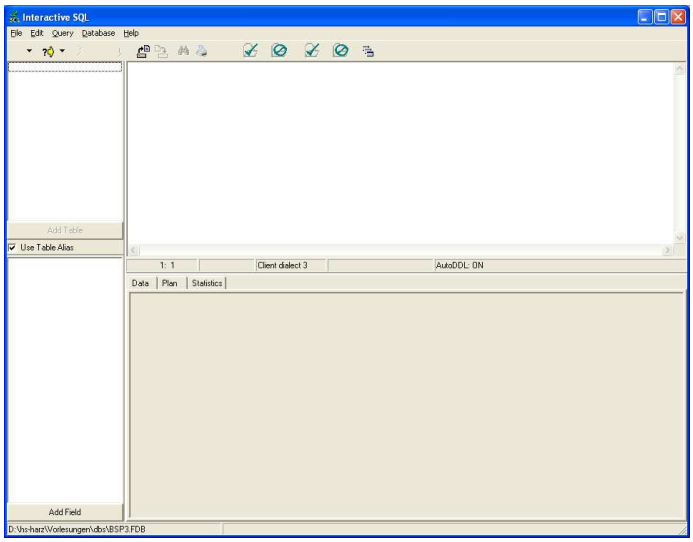

Abbildung 83 SQL-Editor für das fünfte Beispiel

Man klickt in das rechte obere Fenster und fügt mit dem „Paste“-Befehl, STRG+V, den SQL-Code ein.

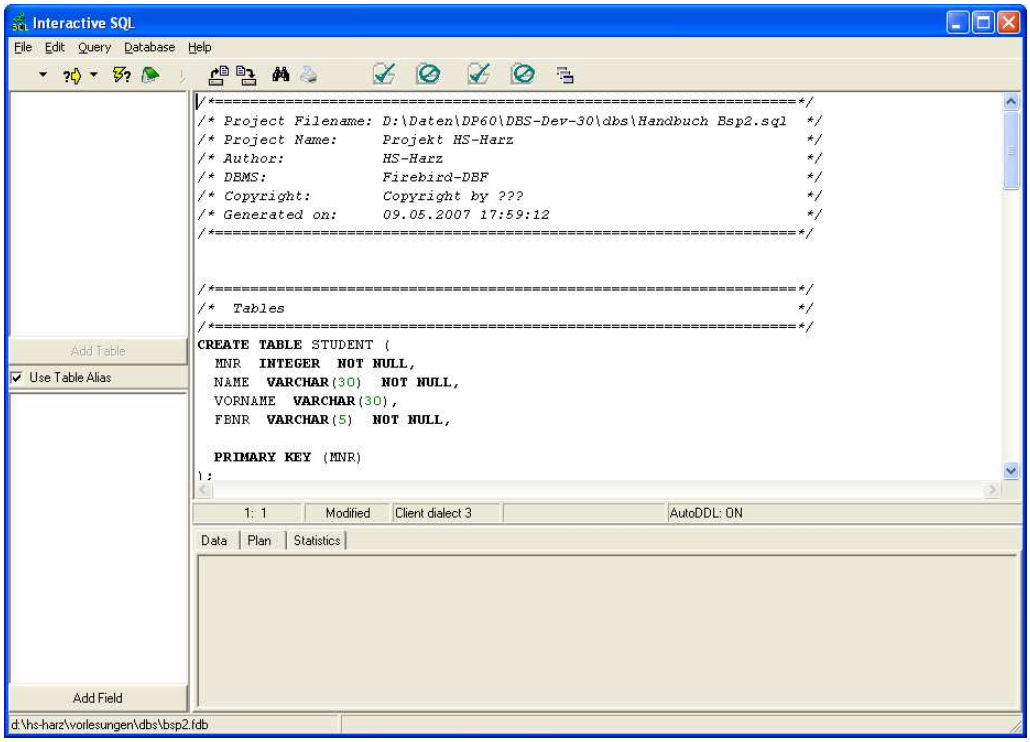

Abbildung 84 SQL-Befehle für das fünfte Beispiel

Mit der Taste „Strg+E“ bzw. Menü Query, Eintrag „Execute“ werden die Befehle ausgeführt.

Die Syntax wurde mit einigen Datenbanken getestet. Bei den Zusatzoptionen wird folgende Reihenfolge definiert:

- DEFAULT
- NOT NULL
- UNIQUE

Bei Fehlern muss man diese Reihenfolge ggfs. anpassen.

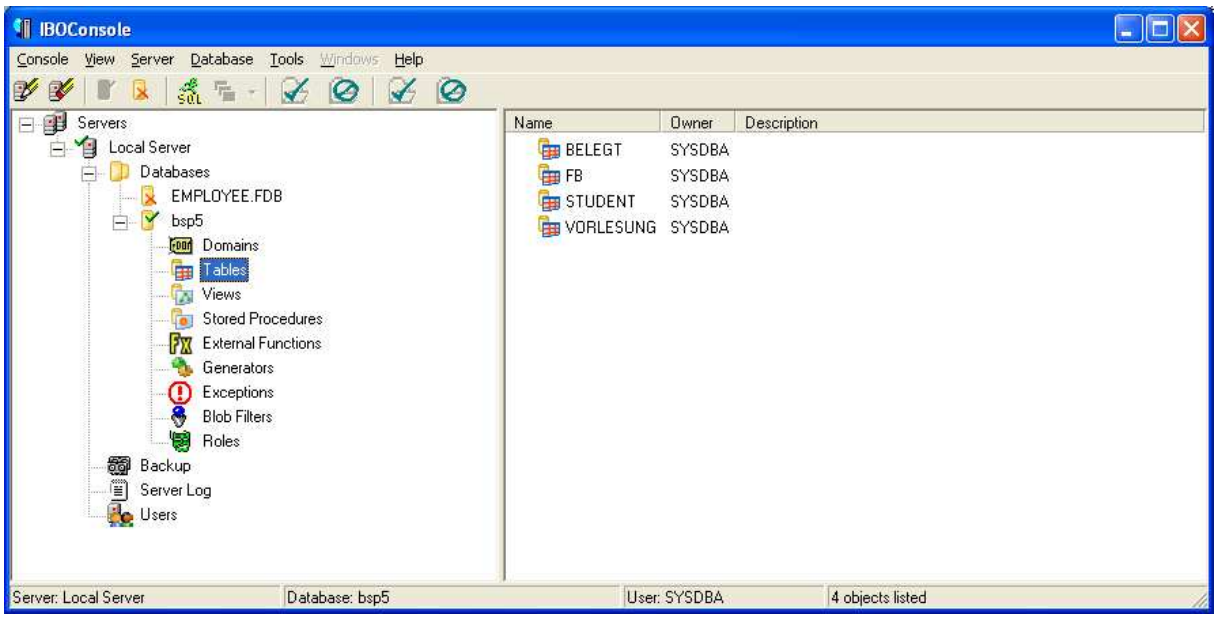

Abbildung 85 Erzeugte Tabellen des fünften Beispiels

Im ersten Schritt müssen nun die Fachbereiche eingetragen werden (warum?)

Antwort:
Die Abhängigkeiten der Entities in den Relationen bestimmt die Reihenfolge des Eintragens bzw. des Löschens.

2.5.8.1 Daten für die Tabelle „FB“

```
INSERT INTO FB (fbnr, name)
VALUES ('AI', 'Automatisierung und Informatik');

INSERT INTO FB (fbnr, name)
VALUES ('VW', 'Verwaltungswissenschaft');

INSERT INTO FB (fbnr, name)
VALUES ('W', 'Wirtschaft');
```

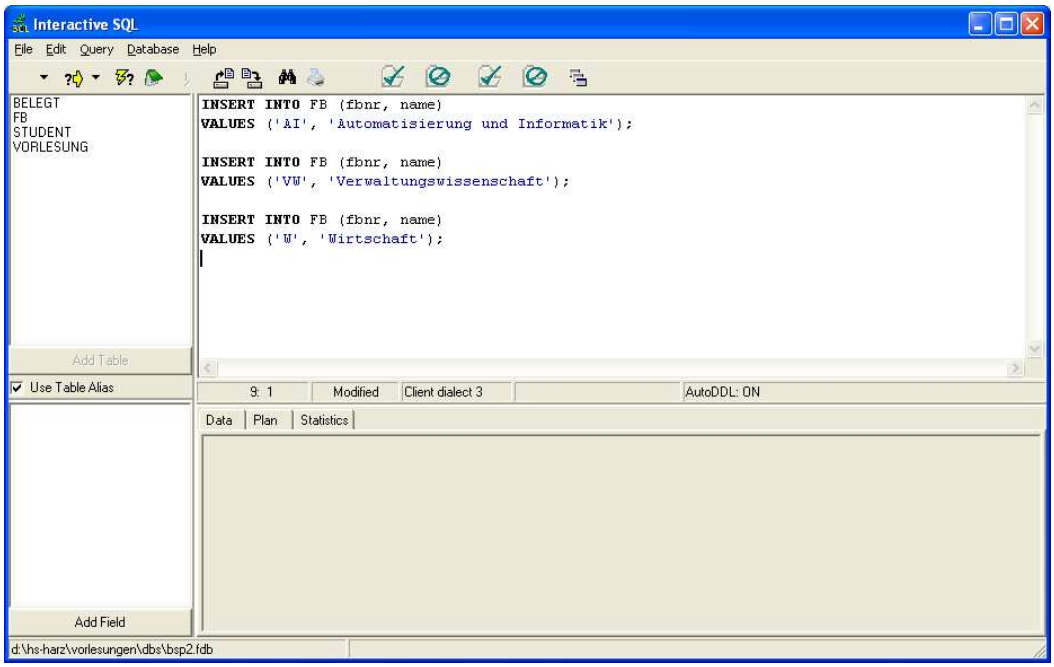

Abbildung 86 Daten für die Fachbereiche

Die Insert-Befehle werden nun in den SQL-Editor eingetragen und mit Strg+E in die Datenbank eingefügt.

Eine Überprüfung zeigt die korrekte Eingabe.

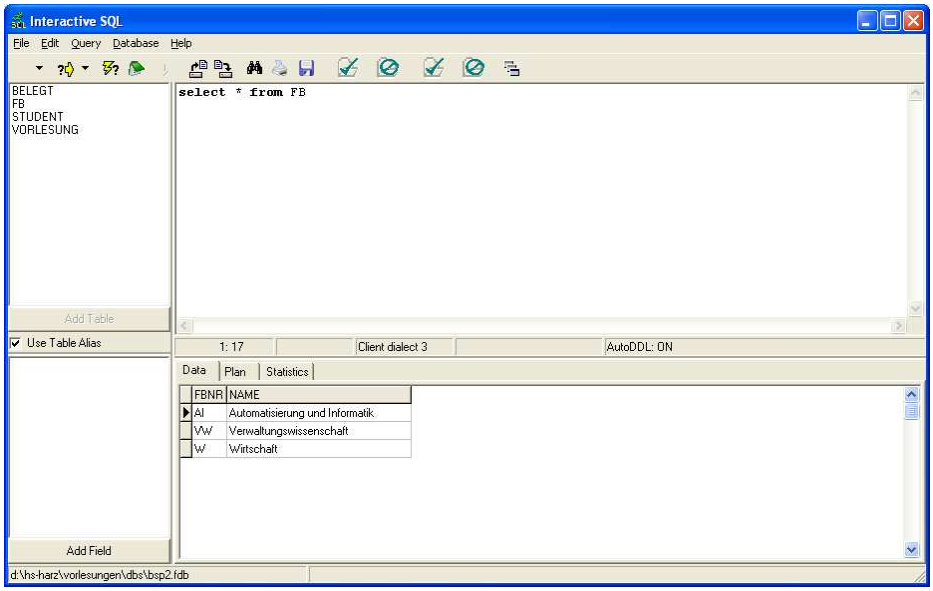

Abbildung 87 Abfrage der Daten für die Abteilung

2.5.8.2 Daten für die Tabelle „Vorlesung“

Daten:

```
INSERT INTO Vorlesung (vnr, name)
VALUES (1,'GDI1');

INSERT INTO Vorlesung (vnr, name)
VALUES (2,'GDI2');

INSERT INTO Vorlesung (vnr, name)
VALUES (3,'GDI3');

INSERT INTO Vorlesung (vnr, name)
VALUES (4,'BS');

INSERT INTO Vorlesung (vnr, name)
VALUES (15,'GUI');

INSERT INTO Vorlesung (vnr, name)
VALUES (21,'MFC');

INSERT INTO Vorlesung (vnr, name)
VALUES (7,'DBS');
```

2.5.8.3 Daten für die Tabelle „Student“

Hier wird als Fremdschlüssel die Fachbereichsnummer mit übergeben.

Daten:

```
INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10001,'Müller', 'Hans', 'AI');

INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10002,'Meyer', 'Ute', 'AI');

INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10003,'Schulze', 'Robert', 'AI');

INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10005,'Robertson', 'Bill', 'VW');

INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10007,'Anderson', 'Andrea', 'VW');

INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10015,'Steiner', 'Ralf', 'W');

INSERT INTO Student (mnr, name, vorname, fbnr)
VALUES (10017,'Steffenson', 'Gerd', 'W');
```

2.5.8.4 Daten für die Tabelle „Belegt“

Daten:

```
INSERT INTO belegt (mnr, vnr)
VALUES (10001, 1);

INSERT INTO belegt (mnr, vnr)
VALUES (10003, 1);

INSERT INTO belegt (mnr, vnr)
VALUES (10001, 2);

INSERT INTO belegt (mnr, vnr)
VALUES (10001, 4);

INSERT INTO belegt (mnr, vnr)
VALUES (10005, 15);

INSERT INTO belegt (mnr, vnr)
VALUES (10003, 2);

INSERT INTO belegt (mnr, vnr)
VALUES (10007, 7);
```

```
INSERT INTO belegt (mnr, vnr)
VALUES (10007, 2);

INSERT INTO belegt (mnr, vnr)
VALUES (10007, 15);
```

2.5.8.5 Update-Befehle

Einfacher Update-Befehl:

```
UPDATE Student
SET Name='Hochstetter'
WHERE ( MNR = 10003);
```

Ändert des Namens eines Studenten mit der Matrikelnummer 10003

Kompexere Update-Befehle:

```
UPDATE Student
SET Name='Hochstetter',    FBNR='W'
WHERE ( MNR = 10003);
```

Ändern des Namens eines Studenten mit der Matrikelnummer 10003 und der Wechsel zum Fachbereich W[1].

```
UPDATE Student
SET Name='Kiesewetter',    FBNR='VW'
```

Dieser Befehl setzt alle Namen auf „Kiesewetter“ und löscht zwei Fachbereiche !

2.5.8.6 Delete-Befehle

```
DELETE FROM Student
WHERE MNR=10005;
```

Exmatrikulation des Studenten 10005

```
DELETE FROM Student
```

Löschen der gesamten Tabelle.

Hinweis:
Das Löschen geht nur, wenn keine weitere Beziehung vorhanden ist.

[1] Aus Sicht des Fachbereichs AI natürlich nicht zu unterstützen

2.6 Beispiel3, Entwicklung eines logischen ER-Modell

Dieses Beispiel zeigt die Entwicklung eines komplexen logischen Modells. Die Vorgaben werden vorab aufgelistet, so dass man übungshalber dieses Modell auch ohne diese Vorlage entwickeln kann.

2.6.1 Aufgabenstellung des sechsten Beispiels

Entwickelt werden soll ein Modell zur Modellierung einer Firma. Folgende Entities wurden definiert:

- Emp
- Dept
- Kid
- Hobbies

Teilaufgaben:

- Definieren der Attribute
- Definieren der Beziehungen (es sollten drei unterschiedliche Arten sein)
- Einbau von Prüfbedingungen
- Generieren des SQL-Skriptes

2.6.2 Neues Projekt

Mit der Taste „STRG+N“ wird ein neues Projekt angelegt. Es erscheint ein Dialogfenster, in dem man die Zieldatenbank auswählt (Abbildung 88).

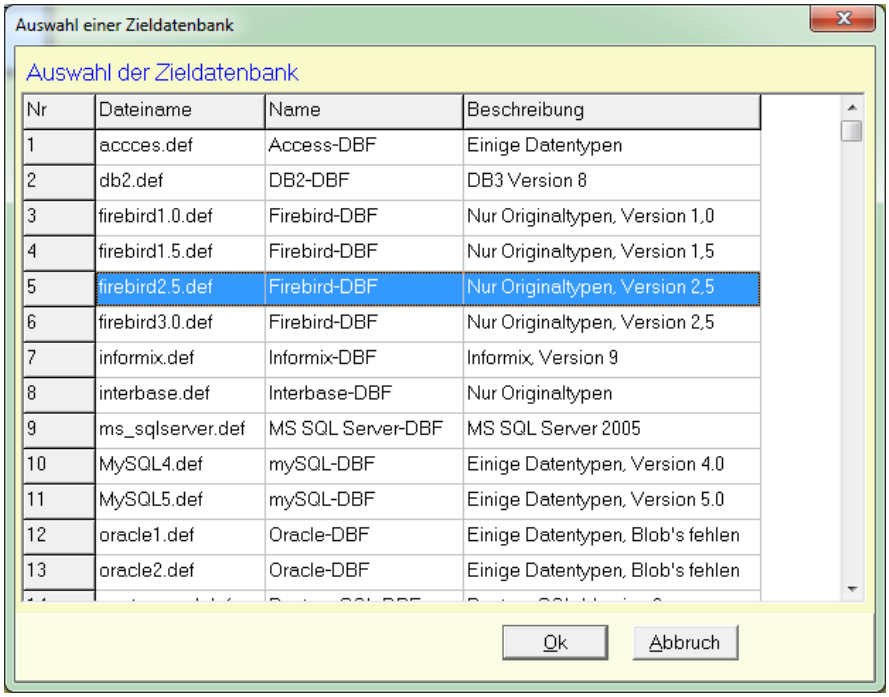

Abbildung 88 Auswahl der Datenbank

Im nächsten Schritt ruft man im Menü „Projekt“ den Eintrag „Eigenschaften“ auf. Im Dialogfenster kann man die notwendigen Einträge vornehmen. Besonders erwähnenswert ist die Auswahl der möglichen Darstellungen.

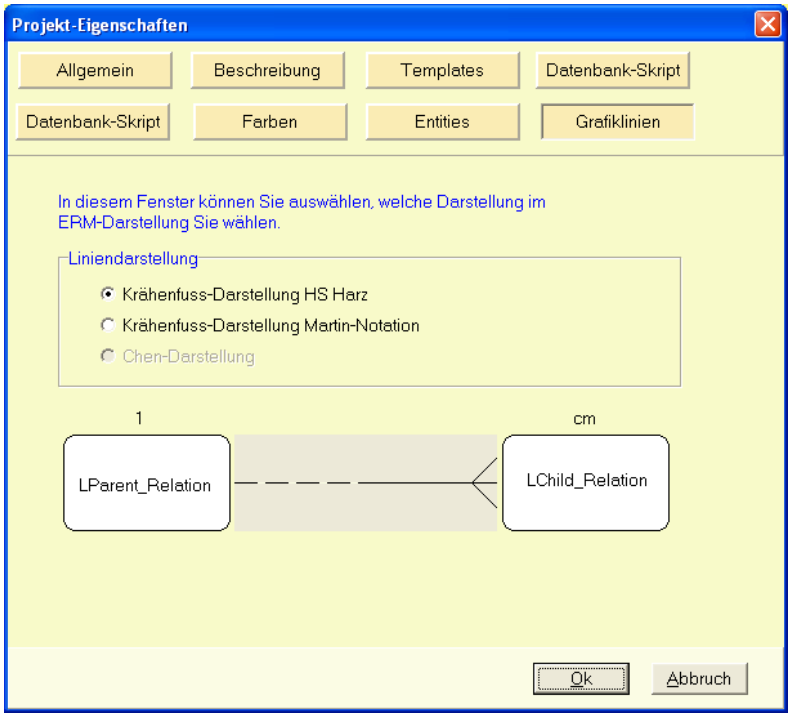

Abbildung 89 Auswahl der Darstellungen

2.6.3 Entities anlegen

Mit dem Tastendruck „STRG+E“ werden die jeweiligen Entities angelegt.

Hinweis:
Dabei sollten die Fremdschlüssel mit anderen Entities noch nicht eingefügt werden!!!

Zur erst wird nach dem Namen gefragt, dieser darf nur aus Buchstaben, Zahlen und dem Underline („_“) bestehen. Es darf kein Leerzeichen verwendet werden. Am Anfang muss auch ein Buchstabe stehen.

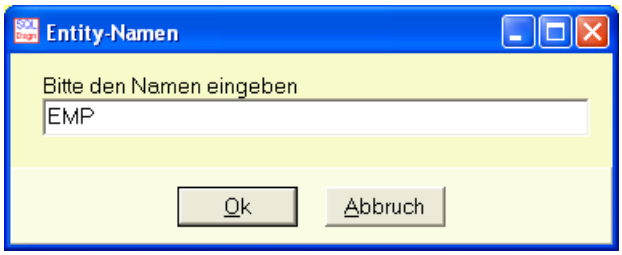

Abbildung 90 Erzeugen eines Entities

Nach der Eingabe des Names, erscheint das Entity in der Grafik. Nun werden alle weiteren Entitäten eingetragen (siehe Abbildung 91).

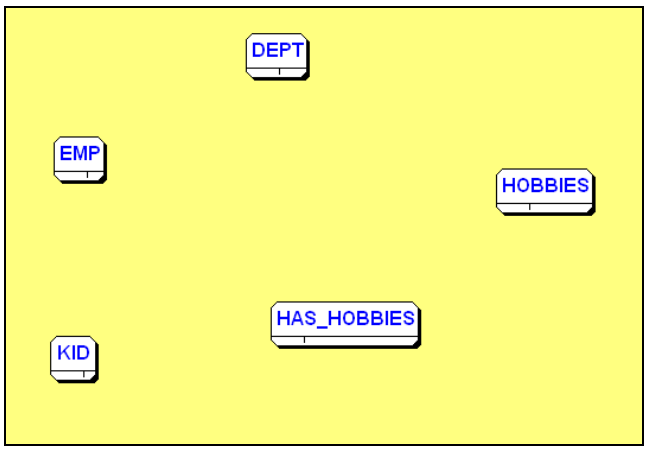

Abbildung 91 Entites des sechsten Beispiels

Mit einem Doppelklick werden nun die einzelnen Entitäten bearbeiten und die Attribute eingetragen (Abbildung 92). Hier sollten die Einträge ausgefüllt werden. Des Weiteren ist es ratsam, die nähere Beschreibung des Entities im Register „Beschreibung“ vorzunehmen. Im zweiten Register, Abbildung 93, werden dann die Attribute eingetragen.

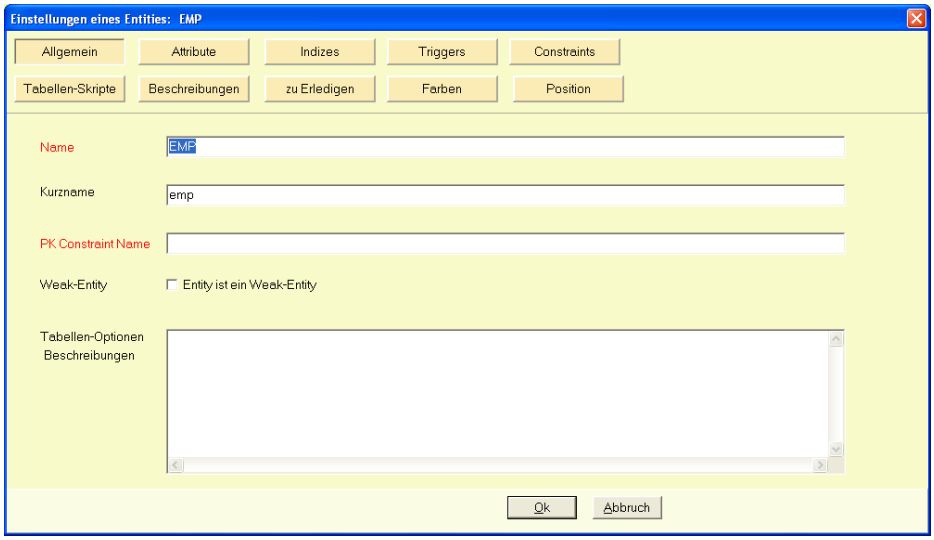

Abbildung 92 Haupt-Eigenschaften eines Entities

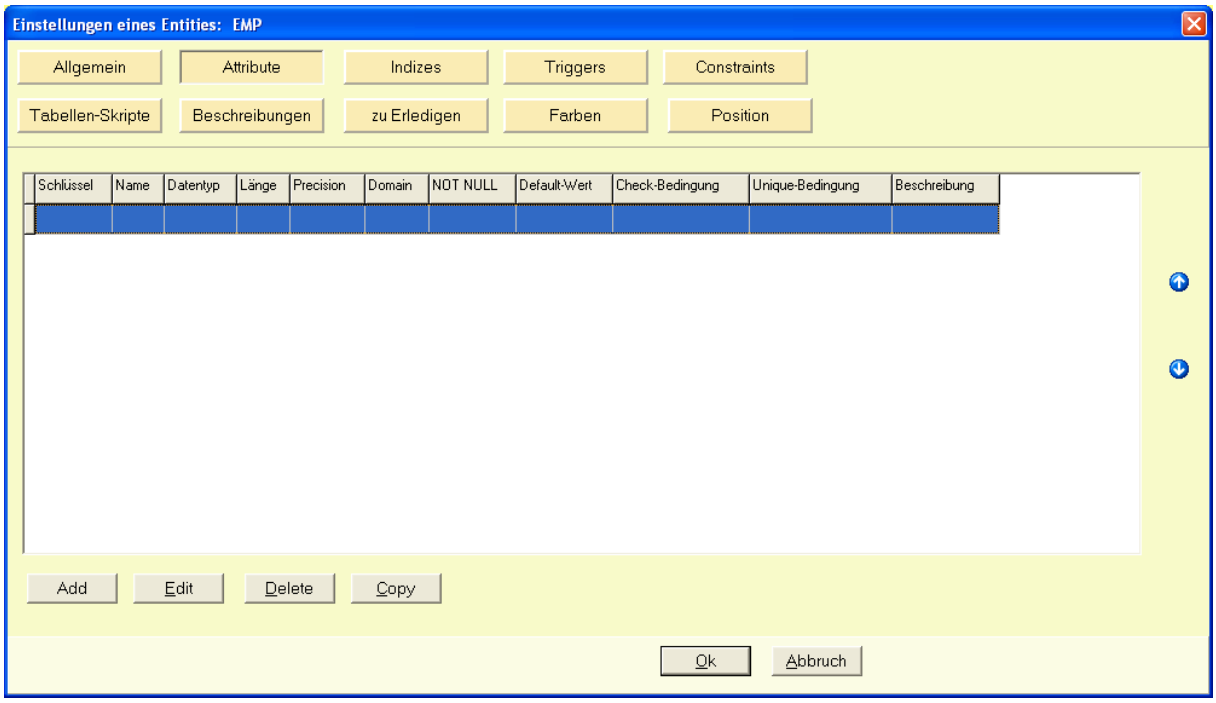

Abbildung 93 Bearbeiten, Erzeugen, Löschen von Attributen

Das zweite Register verwaltet die Attribute. Mit den unteren Schaltern kann man diese bearbeiten, löschen und erzeugen. Die Schalter rechts an der Seite verschieben die Reihenfolge.

Am Schluss könnte das Projekt dann so aussehen:

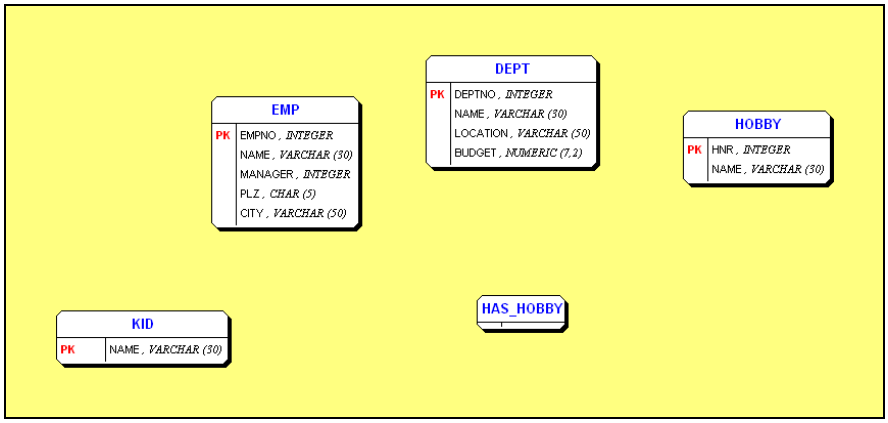

Abbildung 94 Fertige Entities des sechsten Beispiels

Bemerkungen:

- Die Beziehung „Emp“ zu „Dept“ ist eine normale 1:cm-Beziehung.
- Die Beziehung „Emp“ zu „Hobby“ ist eine m:m-Beziehung, deshalb musste ein drittes Entity „Has_Hobby“ eingeführt werden.
- Das Entity „Kid“ ist ein schwaches Entity. Die Namen können nicht einzeln als Primärschlüssel dienen. Es sei denn, man führt eine Nummer ein oder beschränkt die Auswahl auf einheitliche Kindernamen. Dann darf man aber einige Bewerber nicht als Mitarbeiter einstellen. Die hier vorgestellte Lösung verwendet die Weak-Beziehung.
- Des Weiteren müssen die Attribute „Empno“, „Deptno“ und „HNr“ automatisch hochgezählt werden. Dies wird exemplarisch am Beispiel der „Empno“ vorgeführt.
- Mitarbeiter haben auch Vorgesetzte, so dass im Entity ein Attribut Manager existiert. Dies ist eine 1:cm-Beziehung mit dem Attribut „EmpNo“.

Im nächsten Abschnitt werden die Beziehungen eingetragen.

2.6.4 Beziehungen anlegen

Dazu ruft man aus dem Hauptmenü „Einfügen“ die Funktion „Beziehung“ auf. Es erscheint folgendes Fenster:

Abbildung 95 Erstellen einer Beziehung

In beiden Listen sind alle Entities eingetragen. In der linken Liste wählt man den Parent-Entity aus. Das wäre hier in diesem Beispiel „Dept“. In der rechten Liste wird die Beziehung eingetragen. Hier also „Emp“.

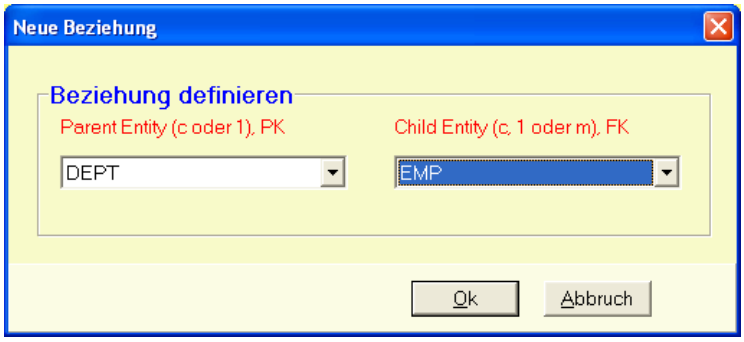

Abbildung 96 Korrekte Einstellung für das Beispiel

Nun muss die korrekte Beziehung eingetragen werden. Danach wird der Primarykey „Deptno“ automatisch als Fremdschlüssel in das Entity „Emp“ eingefügt. Die Beziehungsverben sollten vervollständigt werden.

Die Abbildung 97 zeigt die endgültigen Eingaben. Mit dem Schalter „Ok“ wird die Beziehung eingefügt. Danach erscheint das eigentliche Fenster zur Definition bzw. zur Bearbeitung einer Beziehung.

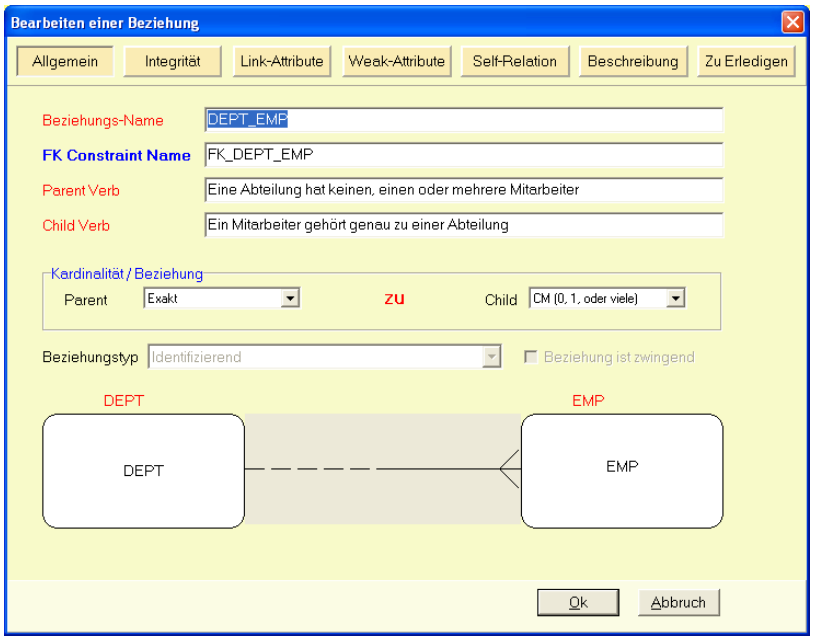

Abbildung 97 Definition der Beziehung

Beziehungstexte:
Eine Abteilung hat keinen, einen oder mehrere Mitarbeiter
Ein Mitarbeiter gehört genau zu einer Abteilung

Im ersten Register werden die Grundeinstellungen (Name, Beziehungstyp) eingetragen. Wichtig sind hier die Einträge „Parent-Verb“ und „Child-Verb“. Sie zeigen den Charakter der Beziehung an. Das Parent-Verb beschreibt die Beziehung durch Wörter wie „Kunde hat“ etc.

Das zweite Register wird zurzeit nicht unterstützt.

Im dritten Register werden die Attribute verknüpft, mit denen die Beziehung definiert wird. Wie oben erwähnt, sind keine Fremdattribute im Child-Entity („Emp“) eingetragen. Wechselt man in das dritte Register, wird man einmal gefragt, ob alle Primärattribute an das Child übertragen werden sollen (Abbildung 98).

Sinnvoll ist es, diese Frage zu bejahen. Dann werden alle Primär-Attribute in das Child-Entity eingefügt. Diese Attribute erhalten dann die Eigenschaft „Fremdschlüssel“. Die Abbildung 98 zeigt die fertige Liste. Sie können aber auch die Attribute manuell verknüpfen. Dazu markiert man in der linken und in der rechten Tabelle jeweils das gewünschte Attribut. Mit dem Schalter „Connect“ werden diese in die obere Tabelle eingefügt. Mit dem Schalter „DisConnect“ wird die ausgewählte Verbindung wieder gelöst. Die Datentypen der beiden Attributen müssen natürlich exakt übereinstimmen.

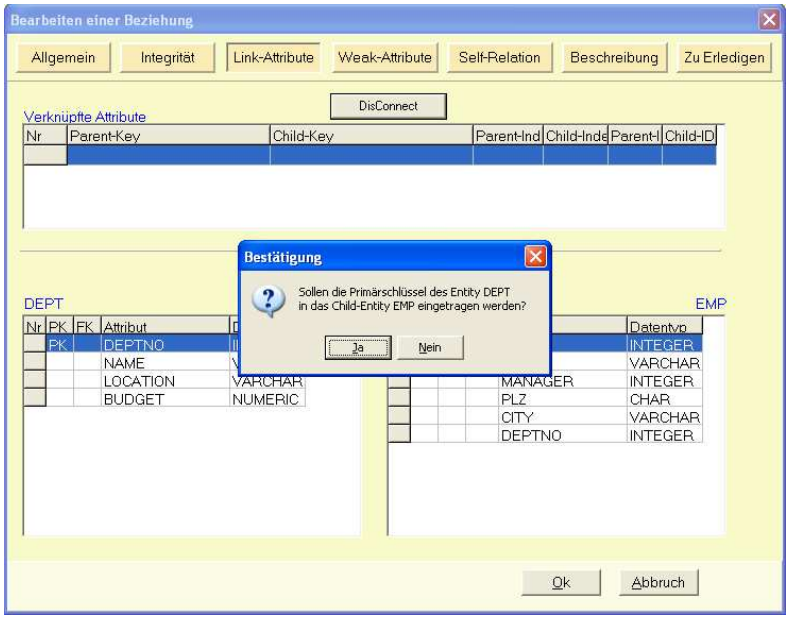

Abbildung 98 Übertragen der Primär-Attribute

Die untere Abbildung zeigt die Beziehung, nachdem die Primär-Attribute in das Child-Entity übertragen wurden.

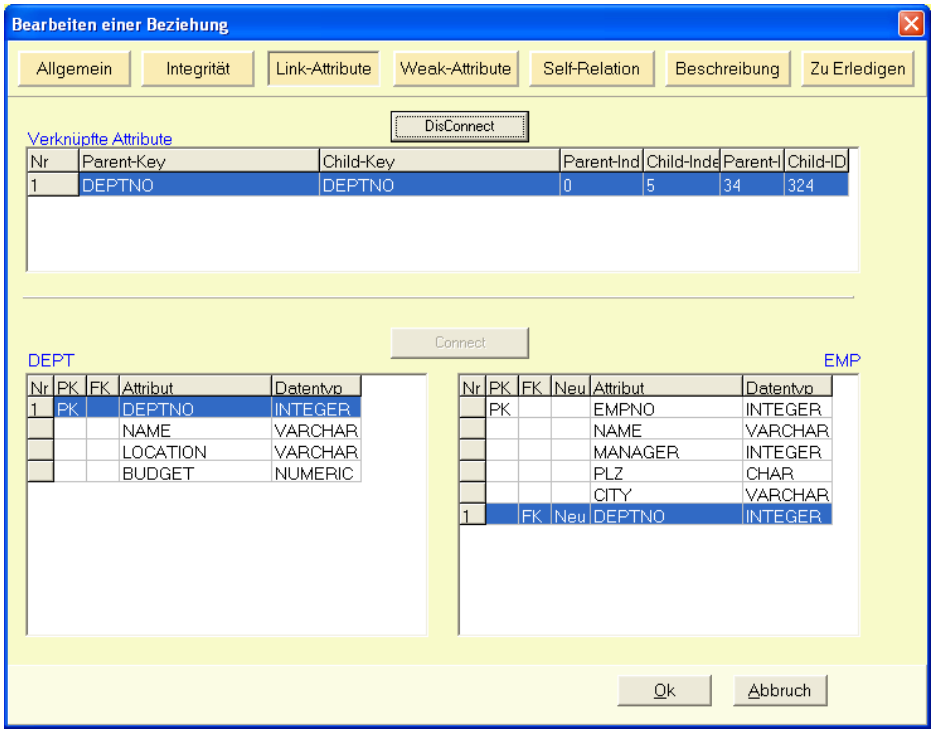

Abbildung 99 Übertragen eines Fremdschlüssels

Das vierte Register „Weak-Attribute“ dient der Definition einer Beziehung einer Weak-Entity (siehe Kapitel 4.1, Seite 100).

Mit dem Betätigen des Schalters „Ok“ wird die Beziehung eingetragen und angezeigt. Mit der Taste F6 erhält man eine Liste aller Beziehungen.

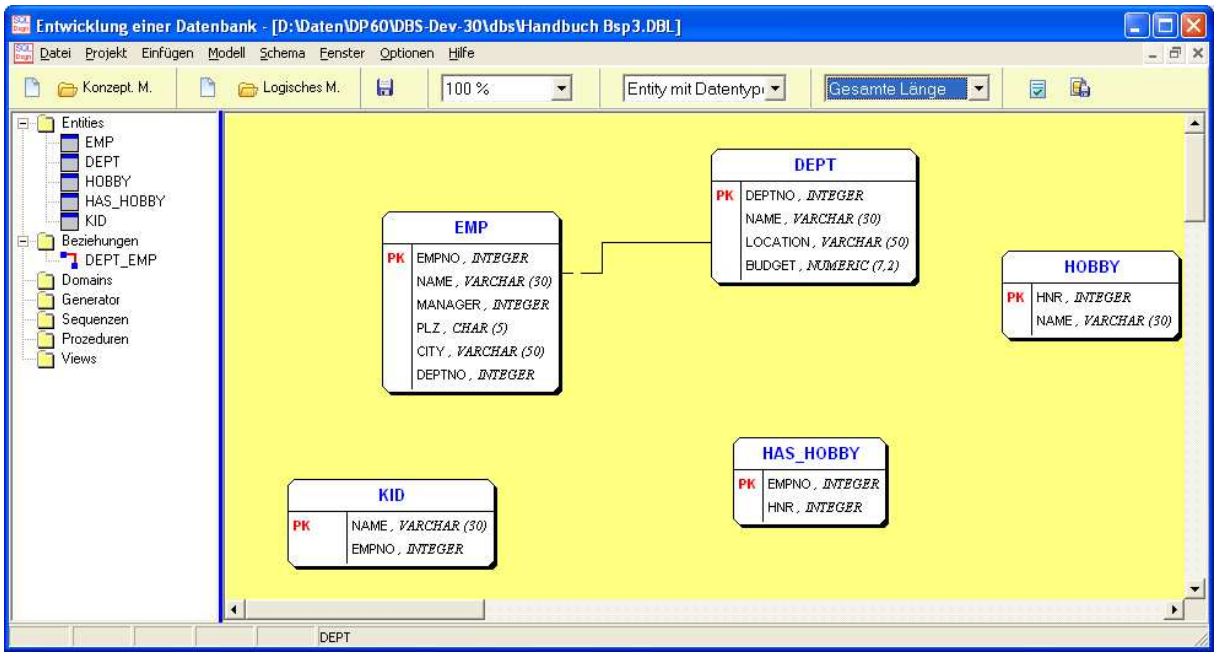

Abbildung 100 Erste Beziehung im sechsten Beispiel

2.6.5 Beziehung der Hobbies

Die Beziehung der Hobbies entspricht einer cm:cn-Beziehung. Dazu existiert das neue Entity „Has_Hobbies“. Es müssen nun zwei Relationen eingetragen werden.

a) Emp zu Has_Hobbies
b) Hobbies zu Has_Hobbies

Abbildung 101 Beziehung Emp zu Has_Hobby

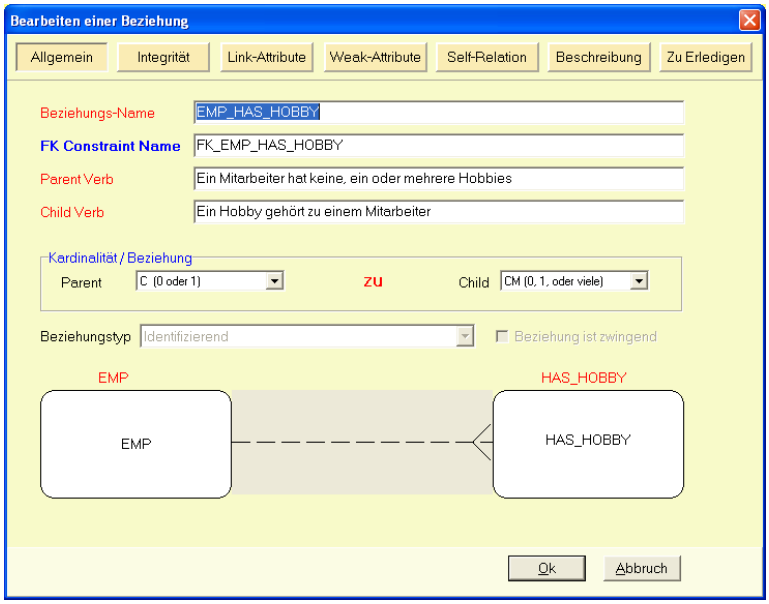

Abbildung 102 Beziehung „Emp“ zu „Has_Hobby“

Das nächste Dialogfenster zeigt die korrekte Einstellung. Links das Parent-Entity, rechts das Child-Entity.

Abbildung 103 Aufruf Beziehung Emp zu Hobbies

Eingetragen werden dann die Beschreibungen und die Kardinalitäten.

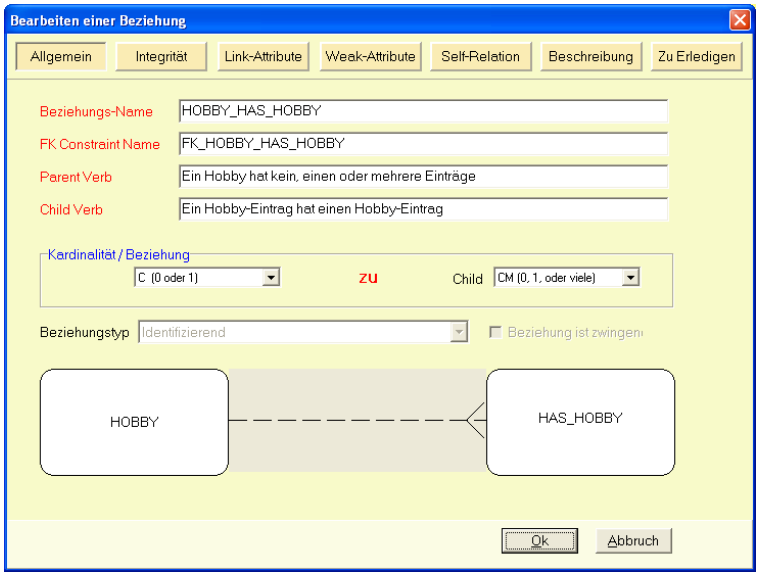

Abbildung 104 Eintrag der Beziehung "Emp" zu "Hobbies"

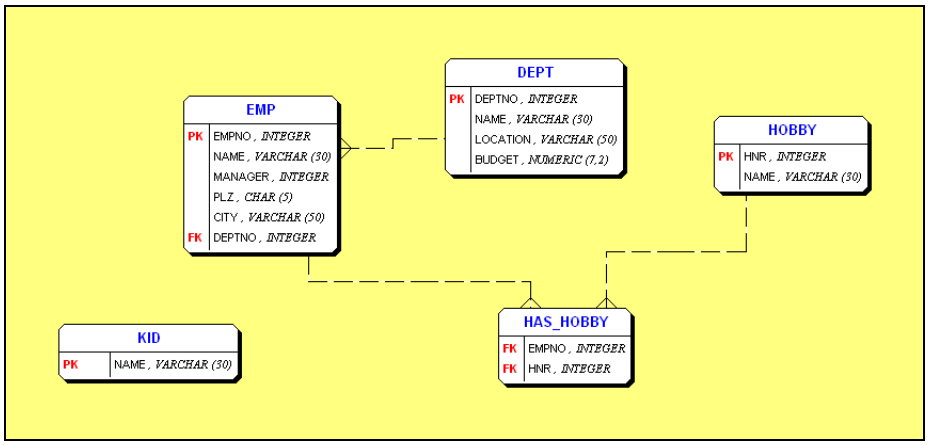

Abbildung 105 ER-Modell nach den Beziehungen der Hobbies

Im zusätzlichen Entity „Has_Hobby“ wurden beide Fremdschlüssel eingetragen. Sinnvoll ist es nun, diese als Primary-Key zu definieren.

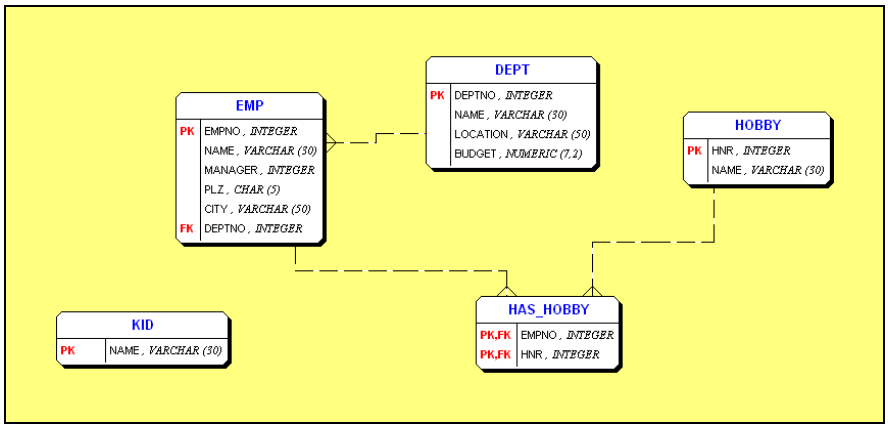

Abbildung 106 ER-Modell mit Primary-Key (Has_Hobby)

2.6.6 Self-Relation

Das Entity „Emp“ hat eine spezielle Beziehung. Das Attribut „Manager“ verweist auf das Attribut „EmpNo“ im gleichen Entity. Das bedeutet zum Einen, dass Manager Mitarbeiter sein müssen, und zum Anderen dass in der Tabelle fast jeder Mitarbeiter einen Vorgesetzten hat.

Der Aufruf der Beziehung liefert:

Abbildung 107 Erzeugen einer Self-Relation

Beziehungstexte:

- Ein Mitarbeiter kann kein oder einen Vorgesetzten haben
- Ein Vorgesetzter hat keinen, einen oder mehrere Mitarbeiter

Parent-Attribut: Emp
Child-Attribut: Manager

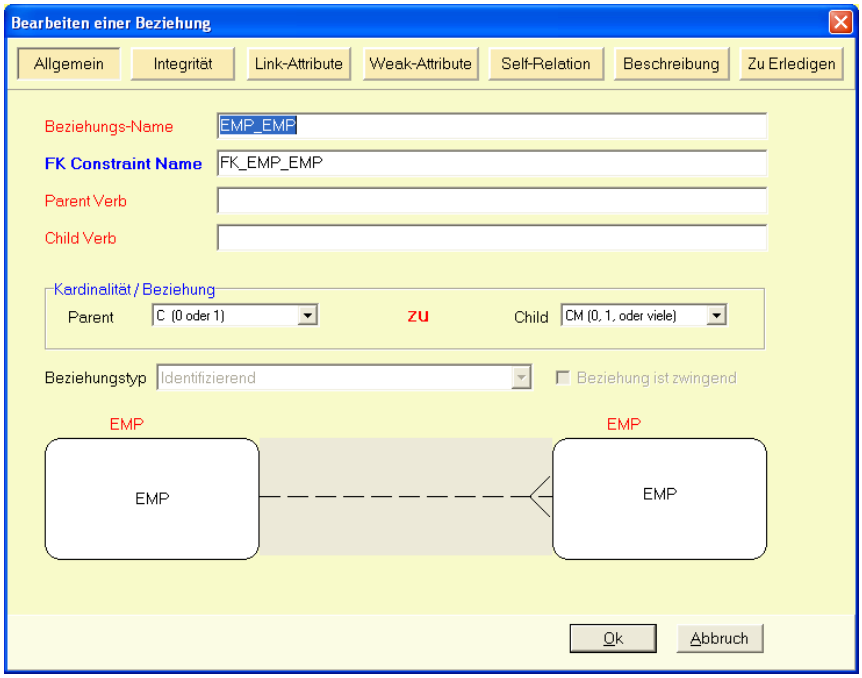

Abbildung 108 Self-Relation

Hinweis:
Das Fremdschlüssel-Attribut muss schon im Entity sein. Ansonsten würde es doppelt erscheinen.

- Aufruf des dritten Registers
- Abfrage **verneinen**
- Links anklicken Emp
- Rechts anklicken Manager
- Schalter „Connect“ betätigen

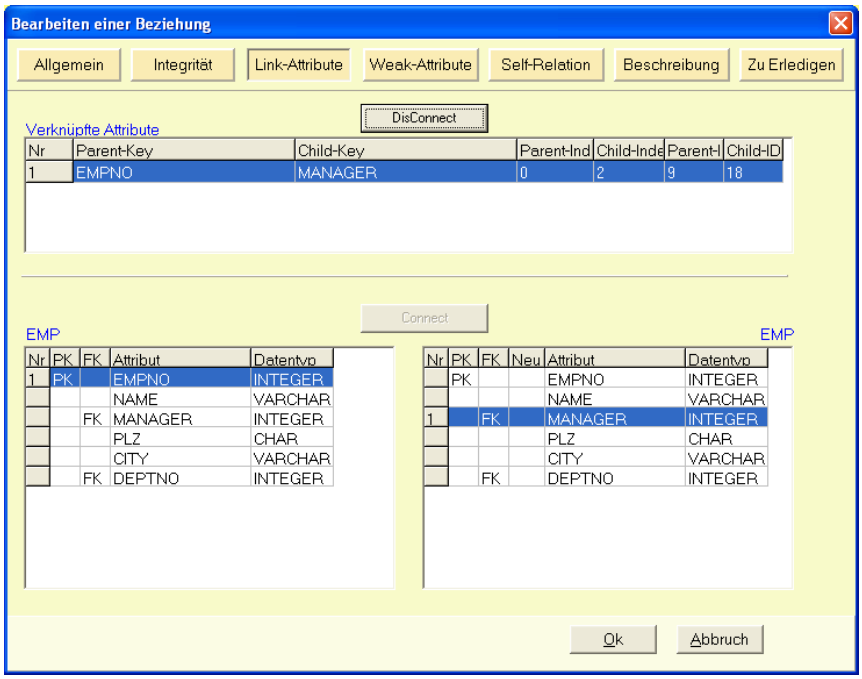

Abbildung 109 EmpNo zu Manager definieren

Im fünften Register kann man die Darstellung der Self-Relation beeinflussen.

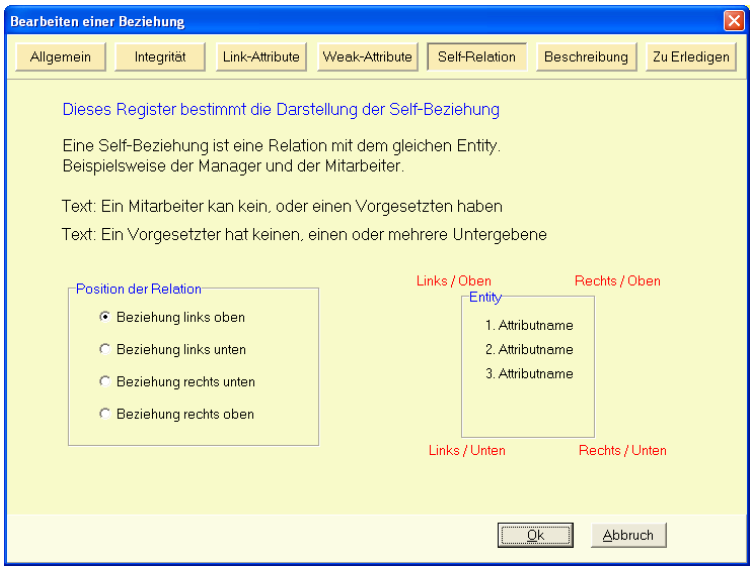

Abbildung 110 Einstellungen für die Darstellung der Self-Beziehung

In der ersten Position wird diese links oben, wie in Abbildung 111, gezeichnet. Bei der nächsten Option wird diese dann links unten dargestellt. Und so weiter.

Das nächste Bild zeigt das neue ER-Modell.

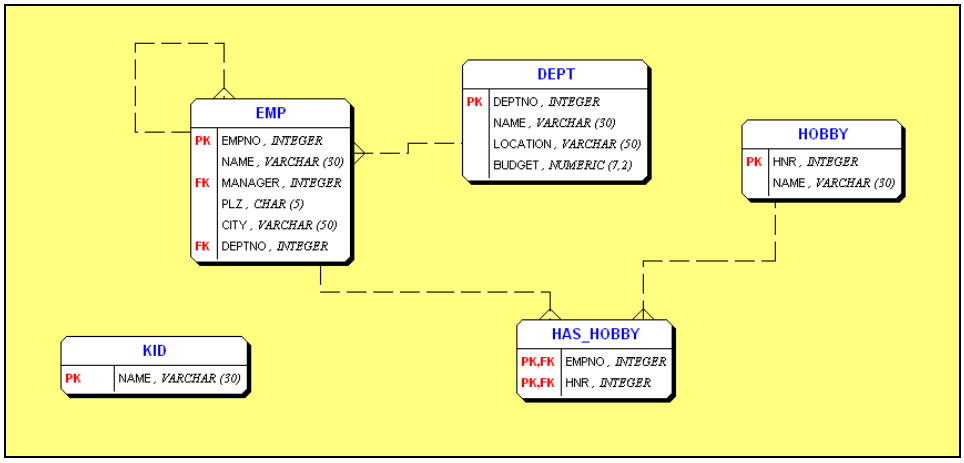

Abbildung 111 Darstellung der Self-Relation

Der letzte Punkt ist die Beziehung zwischen dem Entity „Emp“ und „Kid“.

2.6.7 Weak-Relation

Nach dem Aufruf der Beziehung werden folgende Eingaben getätigt:

- Eintragen der Beziehung (1:cm)
- Automatische Übertragung des Fremdschlüssels
- Im vierten Register die Eigenschaft „Weak-Entity“ aktivieren

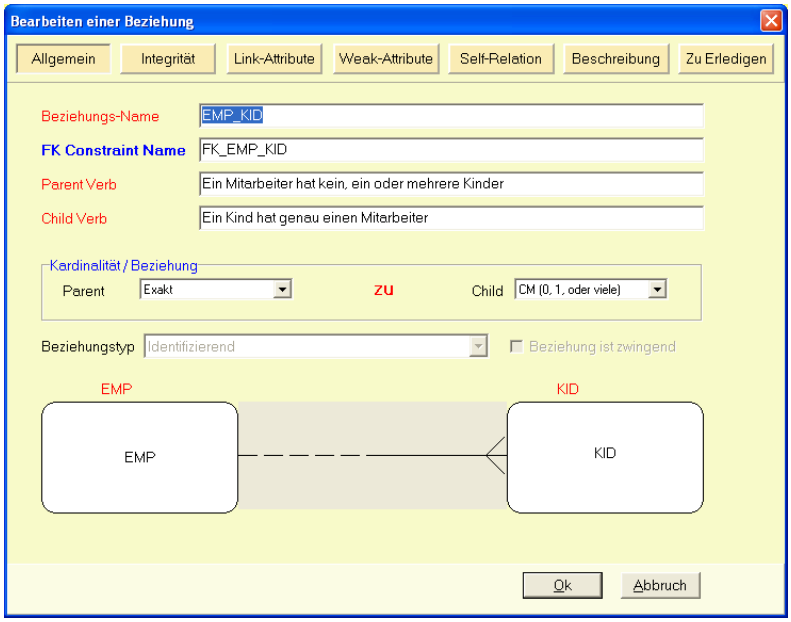

Abbildung 112 Beschreibung des Weak-Entities

Automatische Beziehung erstellen:

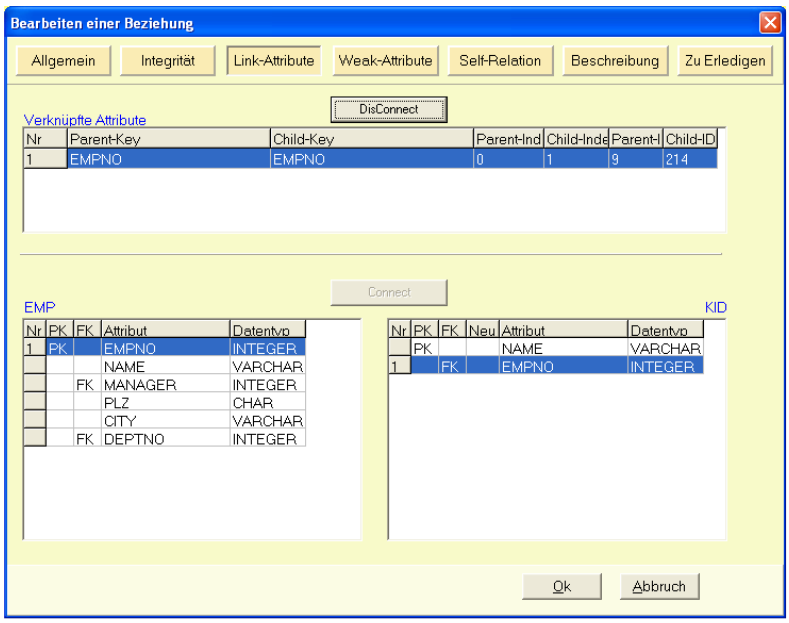

Abbildung 113 Eintragen des Primarykeys als Fremdschlüssels

Aktivieren des Weak-Entity:

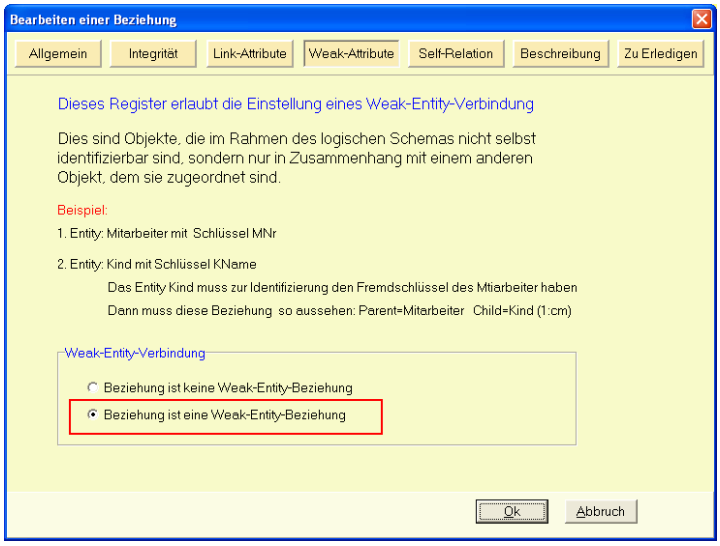

Abbildung 114 Aktivieren des Weak-Entity

Damit ist das ER-Modell fertig:

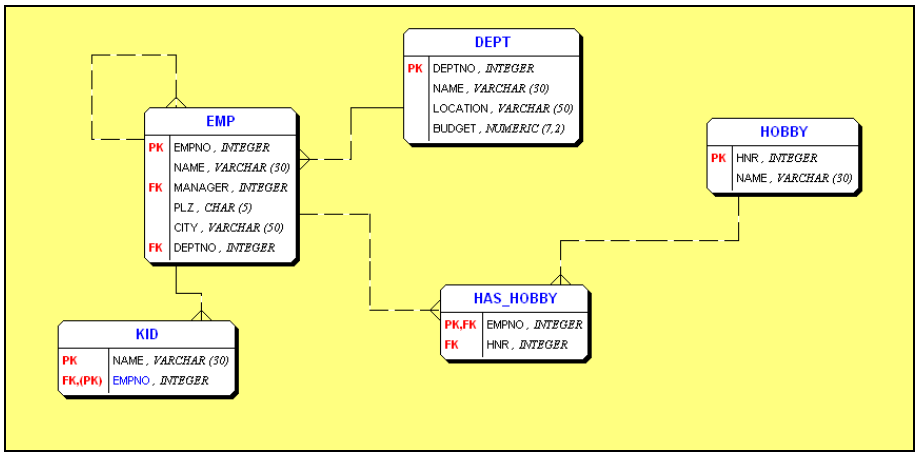

Abbildung 115 ER-Modell des sechsten Beispiels

Im Entity „Kid“ wurde zusätzlich im Attribut „Empno“ die Eigenschaft „PK“ eingetragen. Zur Unterscheidung gegenüber dem normalen Fall, siehe „Has_Hobby“, wird die Bezeichung „PK“ in Klammern gesetzt und das Attribut mit blauer Schriftfarbe gezeichnet. In der allgemeinen Notation wird ein Weak-Entity mit einem Rechteck mit doppelten Rand gezeichnet.

2.6.8 Weitere Schritte

Als weiteren Schritt wird nun eine Prüfbedingung für das Attribut „Budget“ im Entity „Dept“ eingetragen. Dazu klickt man doppelt auf das Entity in der Grafik oder man klickt im Baum auf den Eintrag „Dept“.

Im Dialogfenster ruft man den Editor für das Attribut „Budget“ auf.

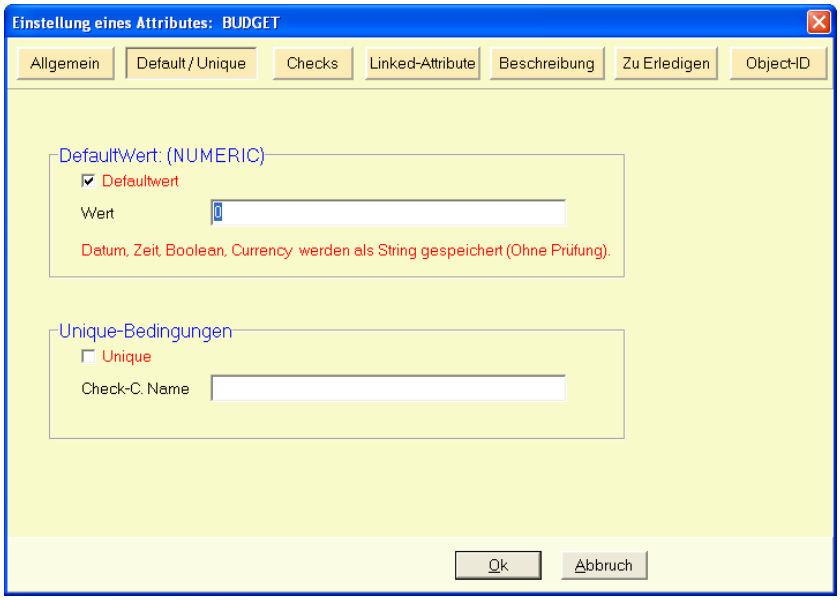

Abbildung 116 Defaultwert für Budget eingegeben

Im Register „Default-Werte“ trägt man eine Ziffer 0 ein, oder einen beliebigen passenden Wert. Im dritten Register aktiviert man das Kontrollfeld „Check-Attribute“.

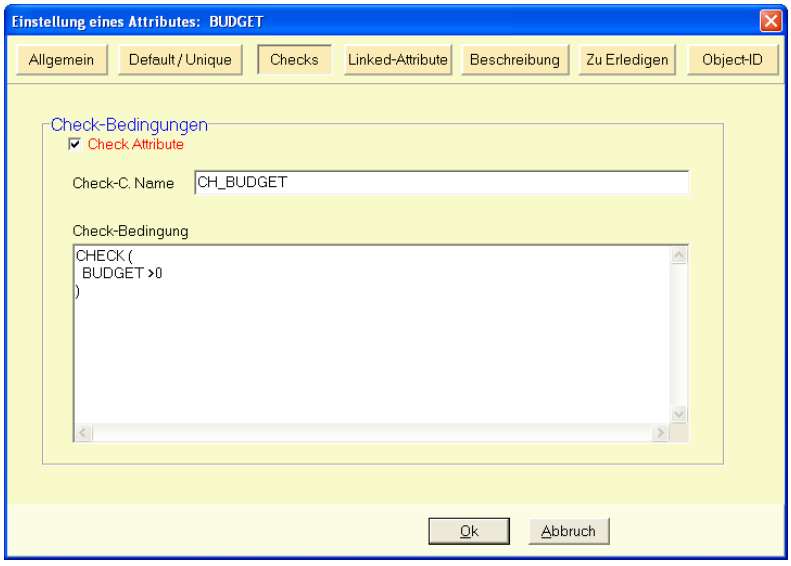

Abbildung 117 Check-Constraint für das Attribut Budget

Dann wird automatisch ein passender Rahmen eingetragen. Für dieses Beispiel reicht es aus, wenn das Budget größer Null sein soll.

Als letzter Schritt wird eine Unique-Bedingung für den Abteilungsnamen eingeben.

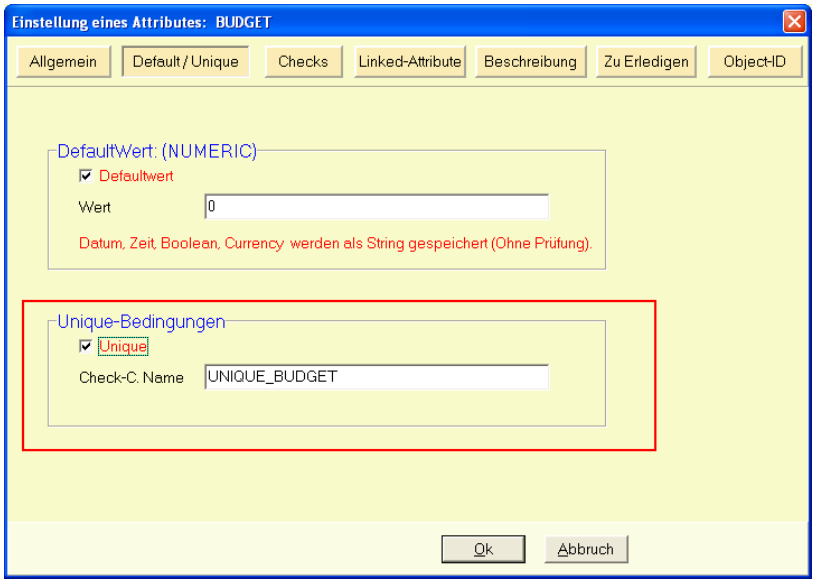

Abbildung 118 Unique-Bedingung für den Abteilungsnamen

Nach diesen Eingaben sind alle Vorbereitungen abgeschlossen und man kann das SQL-Skript erzeugen.

2.6.9 Generierung einer Datenbank

Mit dem Menü „Schema“ und dem Eintrag „Generierung Datenbank“ wird die Erzeugung des SQL-Scriptes angestoßen. Als Kurztaste existiert die Taste F9. Es erscheint folgendes Fenster (siehe Abbildung 119). In der obersten Zeile wird der Dateiname eingetragen. Der nächste Teil definiert die Objekte, die generiert werden sollen. Im letzten Teil sind zusätzliche Optionen vorhanden.

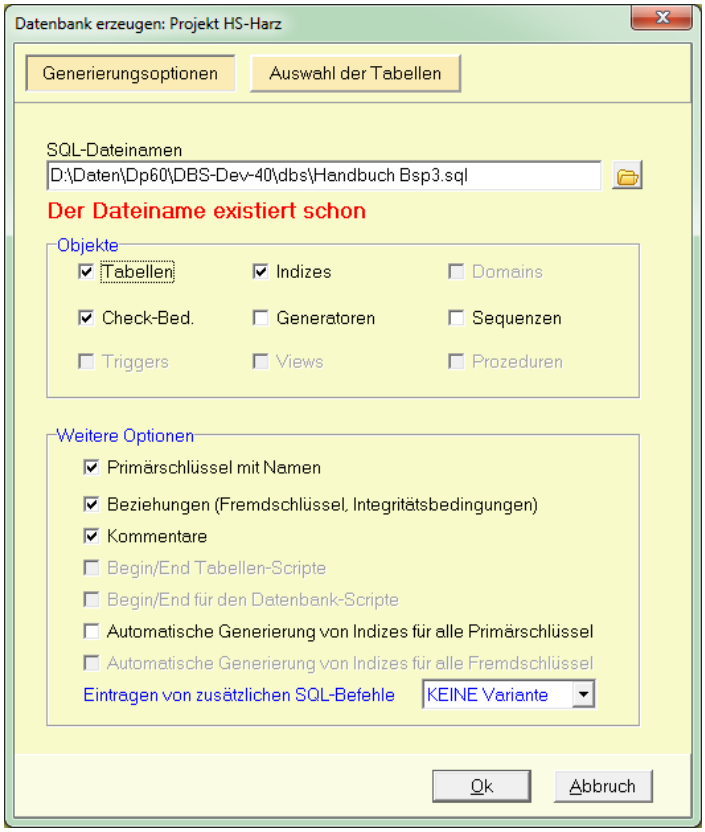

Abbildung 119 Erzeugung einer Datenbank

Mit dem Schalter „Ok“ wird das Script erzeugt und im Notepad dargestellt.

Ergebnis mit SQL-Befehlen:

```
/*==============================================================*/
/* Project Filename: C:\Daten\Handbuch Bsp2.sql                 */
/* Project Name:     Beispiel 3                                 */
/* Author:           Michael Wilhelm                            */
/* DBMS:             Firebird-DBF                               */
/* Copyright:        Copyright by Michael Wilhelm               */
/* Generated on:     09.05.2007 14:36:56                        */
/*==============================================================*/



/*==============================================================*/
/*  Tables                                                      */
/*==============================================================*/
CREATE TABLE EMP (
  EMPNO  INTEGER  NOT NULL,
  NAME  VARCHAR(30)  NOT NULL,
  MANAGER  INTEGER  NOT NULL,
  PLZ  CHAR(5),
  CITY  VARCHAR(50),
  DEPTNO  INTEGER  NOT NULL,

  PRIMARY KEY (EMPNO)
);


CREATE TABLE DEPT (
  DEPTNO  INTEGER  NOT NULL,
```

```
  NAME  VARCHAR(30)  NOT NULL  UNIQUE,
  LOCATION  VARCHAR(50),
  BUDGET  NUMERIC(7,2)  DEFAULT 0,

  PRIMARY KEY (DEPTNO)
);


CREATE TABLE HOBBY (
  HNR  INTEGER  NOT NULL,
  NAME  VARCHAR(30)  NOT NULL,

  PRIMARY KEY (HNR)
);


CREATE TABLE HAS_HOBBY (
  EMPNO  INTEGER  NOT NULL,
  HNR  INTEGER  NOT NULL,

  PRIMARY KEY (EMPNO)
);


/*
  Tabelle ist ein Weak-Entity
*/
CREATE TABLE KID (
  NAME  VARCHAR(30)  NOT NULL,
  EMPNO  INTEGER  NOT NULL,

  PRIMARY KEY (NAME, EMPNO )
);




/*================================================================*/
/*  Foreign Keys                                                  */
/*================================================================*/
ALTER TABLE EMP
 ADD CONSTRAINT FK_DEPT_EMP FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO);

ALTER TABLE HAS_HOBBY
 ADD CONSTRAINT FK_EMP_HAS_HOBBY FOREIGN KEY (EMPNO) REFERENCES EMP(EMPNO);

ALTER TABLE HAS_HOBBY
 ADD CONSTRAINT FK_HOBBY_HAS_HOBBY FOREIGN KEY (HNR) REFERENCES HOBBY(HNR);

ALTER TABLE KID
 ADD CONSTRAINT FK_EMP_KID FOREIGN KEY (EMPNO) REFERENCES EMP(EMPNO);

ALTER TABLE EMP
 ADD CONSTRAINT FK_EMP_EMP FOREIGN KEY (MANAGER) REFERENCES EMP(EMPNO);

/*================================================================*/
/*  Check-Constraints                                             */
/*================================================================*/
ALTER TABLE DEPT
     ADD CONSTRAINT CH_BUDGET CHECK (
  BUDGET >=0
);
```

2.6.10 Erzeugen der Datenbank

Mit Aufruf einer Datenbank-Konsole à la IBOConsole, Workbench, wird nun die Datenbank erzeugt.

Beispiel mit IBOConsole:

1. Schritt: Aufruf des Programms

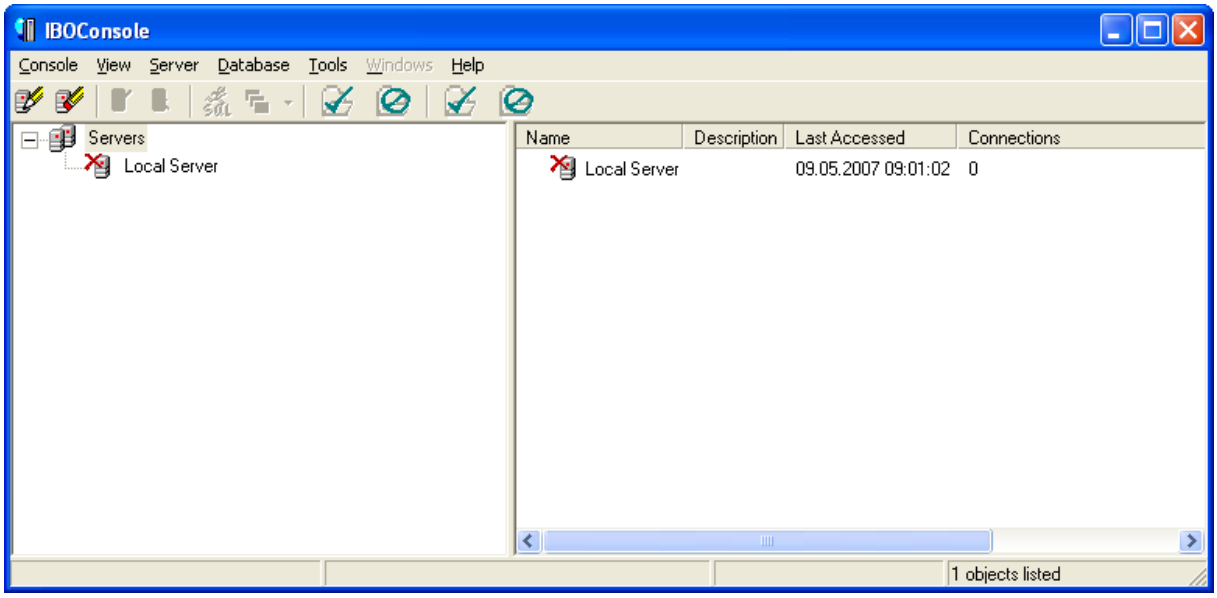

Abbildung 120 Start der IBO-Console

2. Schritt: Doppelklick auf den Eintrag „Local Server“

3. Schritt: Menü Database, Eintrag „Create Database“

Datenbankname: c:\daten\bsp6.fdb

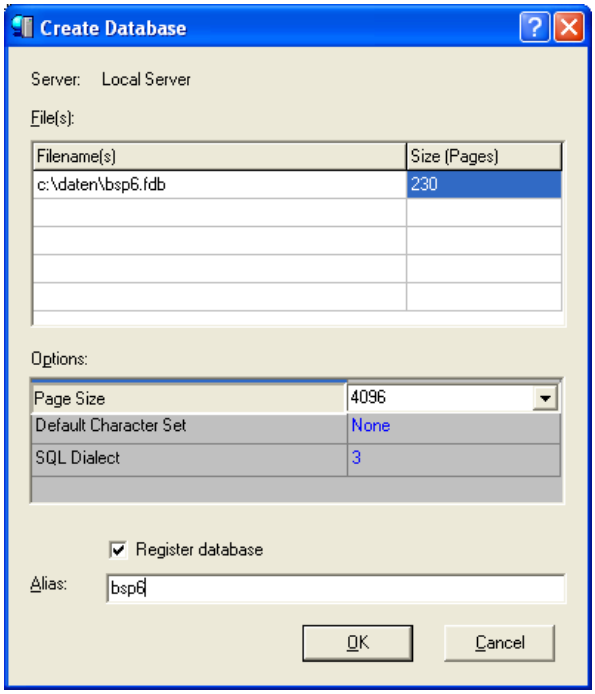

Abbildung 121 Eintrag für eine neue Datenbank

In der linken Liste steht der Dateinamen mit voller Pfadangabe. Rechts daneben wird die Anzahl der Seiten eingetragen. Hier muss mindestens eine 230 als Wert eingetragen werden. In den Abschnitt „Options“ wird die Clustergröße und der SQL-Dialekt definiert. Ganz unten wird der Alias-Namen eingetragen. Mit diesen kann man per Programm eine Datenbank ansprechen, ohne dass man weiss, in welchem Pfad die Datenbank sich befindet. Diese Eigenschaft ist aber nur für Programmentwickler interessant.

Danach sollte ein neuer Eintrag in der Datenbank vorhanden sein.

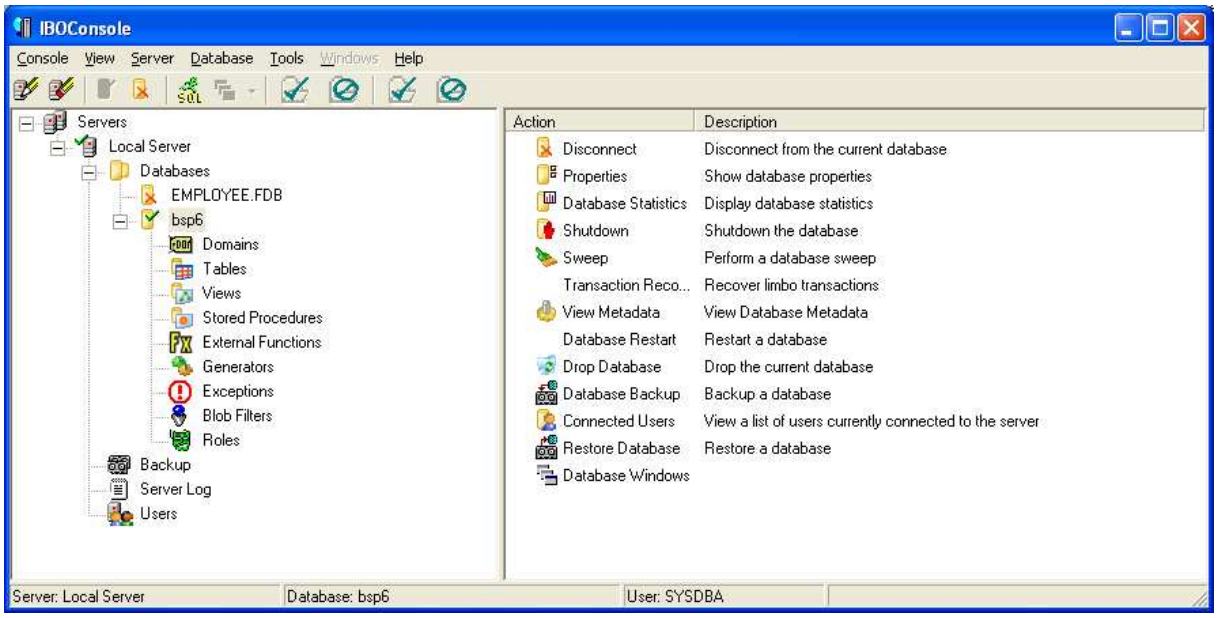

Abbildung 122 neue Datenbank

Mit dem Menü „Tools“, Eintrag „Interactive SQL“ oder dem entsprechendem Symbol öffent sich der SQL-Editor.

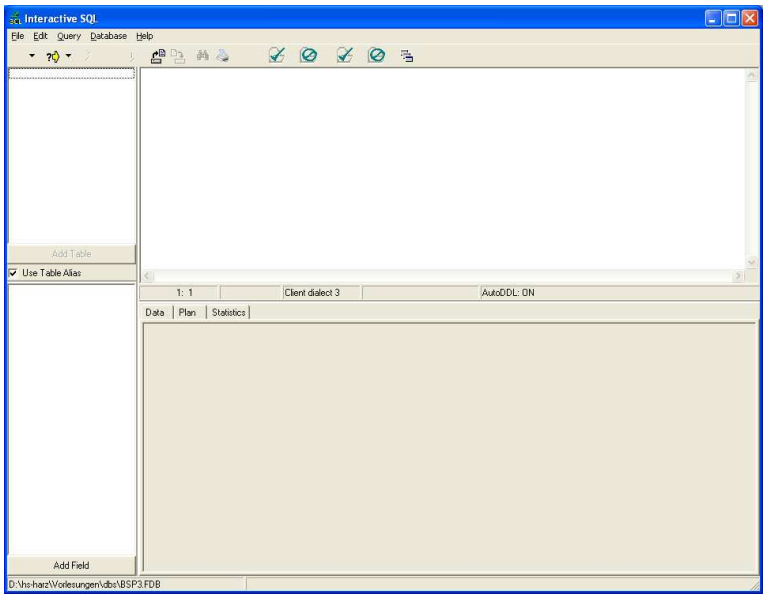

Abbildung 123 SQL-Editor für das sechste Beispiel

Man klickt in das rechte obere Fenster und fügt der „Paste“-Befehl, STRG+V, die SQL-Code ein.

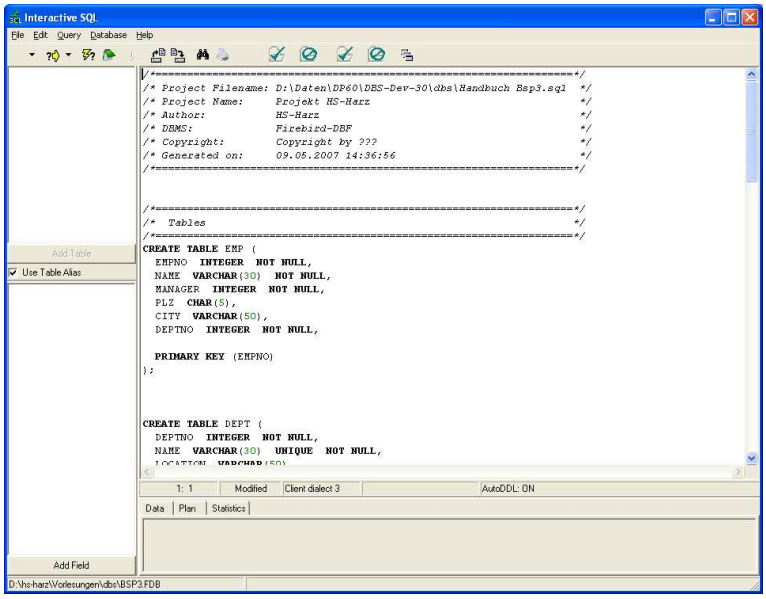

Abbildung 124 SQL-Befehle für das dritte Beispiel

Mit der Taste „Strg+E“ bzw. Menü Query, Eintrag „Execute“ werden die Befehle ausgeführt.

Die Syntax wurde mit einigen Datenbanken getestet. Bei den Zusatzoptionen wird folgende Reihenfolge definiert:

- DEFAULT
- NOT NULL
- UNIQUE

Bei Fehlern muss man diese Reihenfolge ggfs. anpassen.

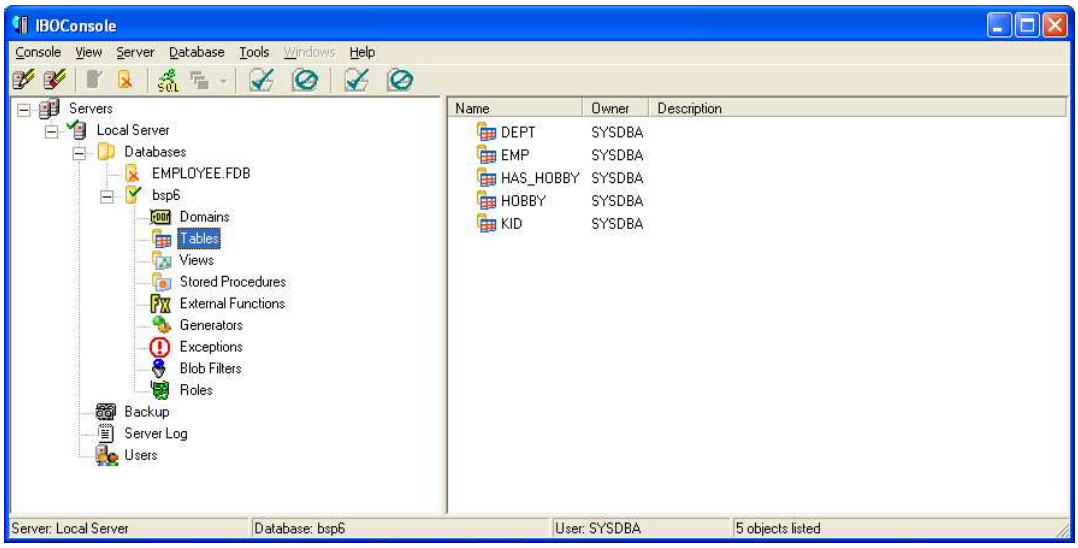

Abbildung 125 Erzeugte Tabellen des sechsten Beispiels

Im ersten Schritt müssen nun die Abteilungen eingetragen werden (warum?)

Antwort:
Die Abhängigkeiten der Entities in den Relationen bestimmt die Reihenfolge des Eintragens bzw. des Löschens.

2.6.10.1 Daten für die Tabelle „Dept“

```
INSERT INTO DEPT (deptno, name)
VALUES (1,'Sales');

INSERT INTO DEPT (deptno, name)
VALUES (2,'Fince');

INSERT INTO DEPT (deptno, name)
VALUES (3,'Marketing');
```

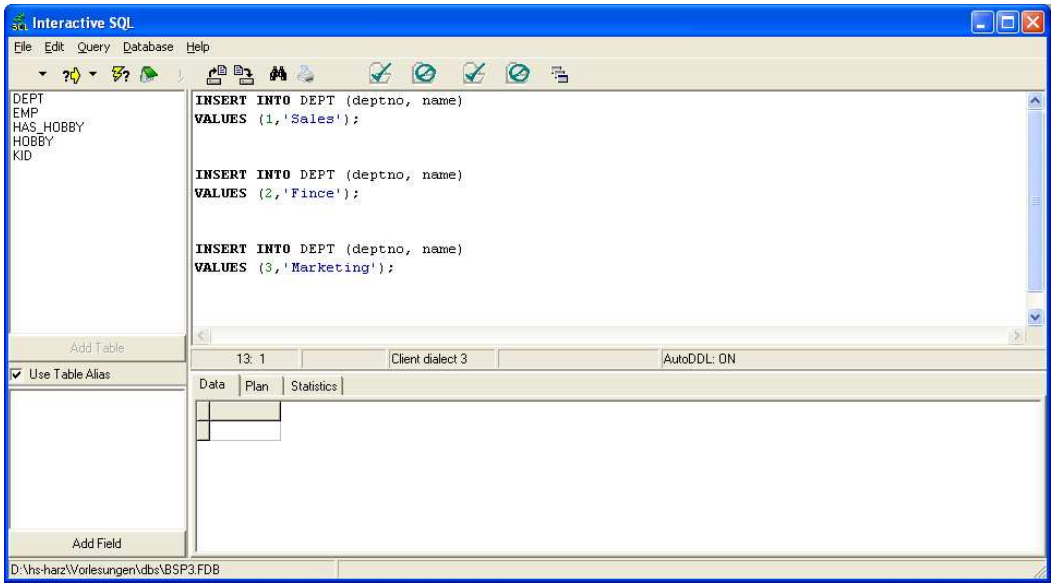

Abbildung 126 Daten für die Abteilung

Die Insert-Befehle werden nun in den SQL-Editor eingetragen und mit Strg+E in die Datenbank eingefügt.

Eine Überprüfung zeigt die korrekte Eingabe.

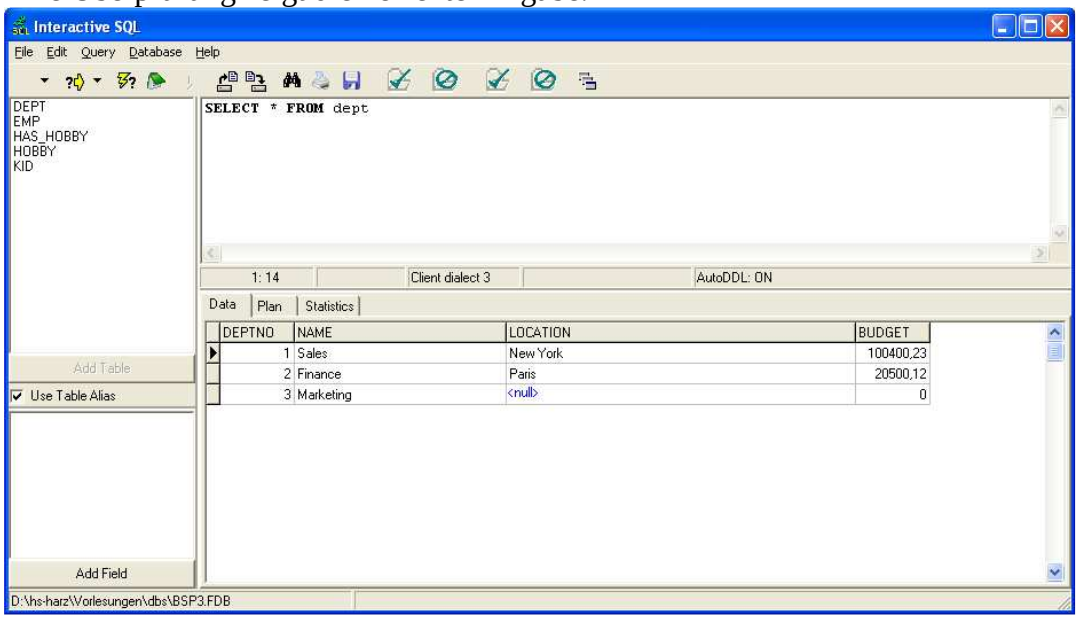

Abbildung 127 Abfrage der Daten für die Abteilung

Hinweis:

INSERT INTO DEPT (deptno, name, location, budget)
VALUES (4,'testabteilung', 'hogwarts', -888);

Die obige Anweisung wird von der Datenbank nicht akzeptiert, da das Budget kleiner als Null ist.

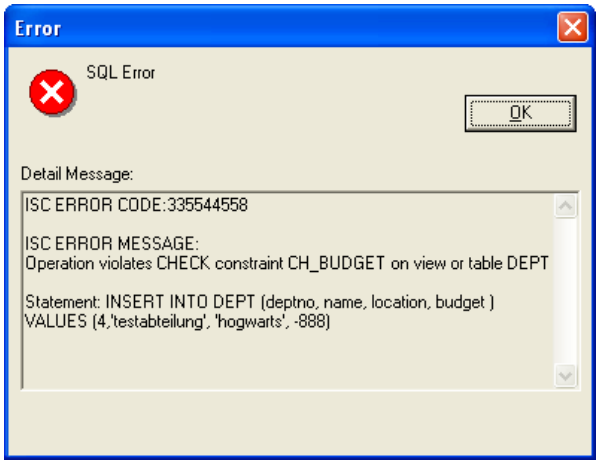

Abbildung 128 Test der Check-Bedingung Budget

Bei den Daten für die Mitarbeiter müssen die Manager mit eingegeben werden, da eine Fremdschlüsselbeziehung existiert. Deshalb müssen als erstes die Manager eingeben werden. Des Weiteren brauchen diese natürlich auch „Vorgesetzte“. Der einfachste Weg ist es, für Manager ohne Chef, die eigene Nummer zu einzutragen. Oder den Constraint bei diesen Mitarbeitern auszuschalten.

Alternativ, kann man die Fremdschlüssel-Beziehung auch in einem „Before-Insert“Trigger prüfen.

2.6.10.2 Daten für die Tabelle „Emp“

Daten:

```
INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (11,'müller', 11, '38855', 'Wernigerode',1);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (5,'schmidt', 5,'39114', 'Magdeburg',2);


INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (21,'schulze', 21, '06012', 'Halle (Saale)',3);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (1,'meier', 11, '38855', 'Wernigerode',1);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (2,'bernstein', 5, '39115', 'Magdeburg',1);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (4,'meyer', 21, '38855', 'Wernigerode',3);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (7,'schubert', 5, '39125', 'Magdeburg',2);
```

```
INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (44,'koch', 5, '38855', 'Wernigerode',3);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (27,'schulze', 21, '39125', 'Magdeburg',2);
```

2.7 Generator für die Mitarbeiternummer EMPNO

Siehe Kapitel 4.8, Seite 106

3 Funktionen

Dieses Kapitel beschreibt kurz die wichtigsten Funktionen.

3.1 Menüfunktionen

3.1.1 Menü Datei

Menüfunktionen	**Beschreibung**
Neues Projekt	Öffnet ein neues logisches Projekt(STRG+N)
Neues Projekt	Öffnet ein neues konzeptionelles Projekt(STRG+M)
Öffnen	Öffnet ein vorhandenes Projekt(STRG+O), logisches Modell
Öffnen	Öffnet ein vorhandenes Projekt(STRG+O), konzeptionelles Modell (STRG+K)
Projekt schließen	Schließt das aktuelle Projekt
Speichern	Speichert das Projekt. (STRG+S)
Speichern unter	Speichert das Projekt unter einem anderen Namen
Export als Bild	Das ER-Modell wird als Bild in die Zwischenablage kopiert
Export der Daten nach Winword	Alle Daten werden in eine neue WinWord-Datei eingetragen
Alte Dateien	Ein Untermenü zeigt die zuletzt bearbeiteten Dateien an.

3.1.2 Menü Projekt

Menüfunktionen	**Beschreibung**
Eigenschaften	Zeigt ein Dialogfenster, in dem man diverse Einträge vornehmen kann.

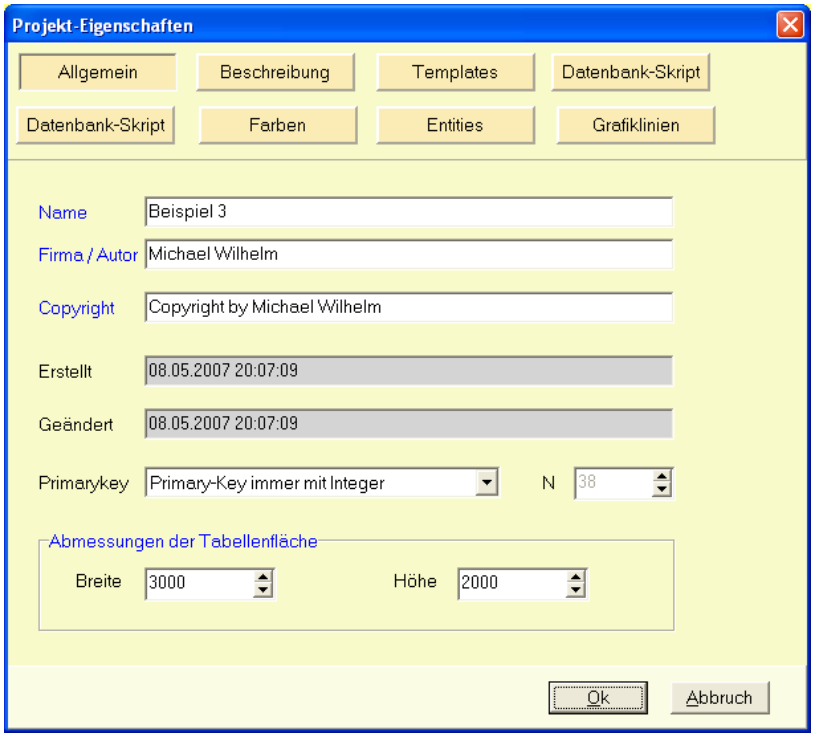

Abbildung 129 Projekt-Eigenschaften

In diesem Dialogfenster werden alle Schriften und Farben der Entities definiert. Die Farben können nachher individuell pro Entity geändert werden. Im Register „Grafiklinien2 hat man die Möglichkeit, die Darstellung der Beziehungen zu definieren.

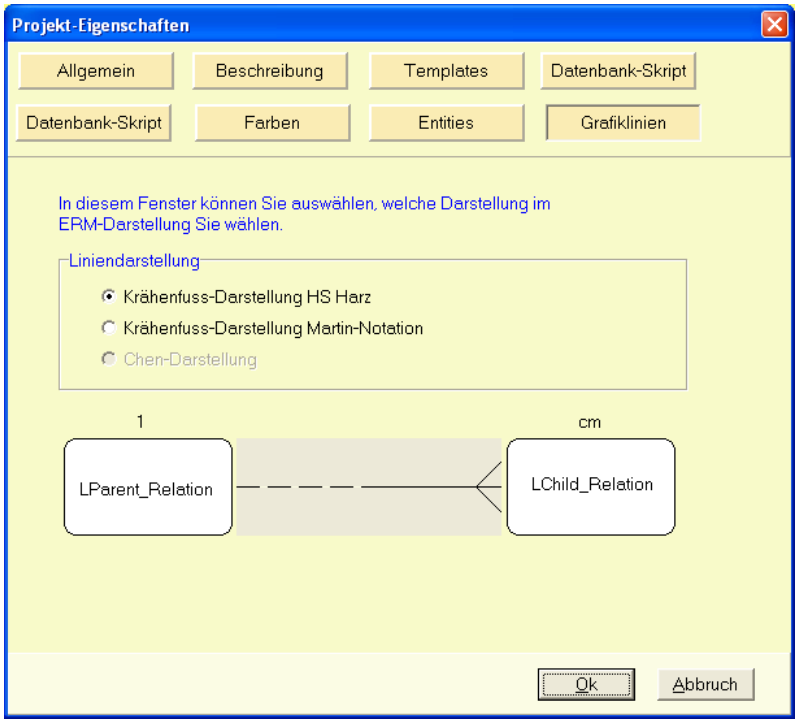

Abbildung 130 Grafiklinien

3.1.3 Menü Einfügen

Menüfunktionen	**Beschreibung**
Entity	Fügt ein neues Entity in das Projekt ein (STRG+E)
Beziehung	Fügt eine neue Beziehung in das Projekt ein (STRG+R)
Ternäre Beziehung	Fügt eine neue ternäre Beziehung in das Projekt ein (STRG+T)
Generator	Fügt einen Generator für ein Attribut in das Projekt
Sequenz	Fügt eine Sequenz für ein Attribut in das Projekt

3.1.4 Menü Modell

Menüfunktionen	**Beschreibung**
Eigenschaften	Zeigt ein Dialogfenster mit dem wichtigsten Eigenschaften des logischen Modells
Neues Entity	Fügt ein neues Entity in das Projekt ein (STRG+E)
Entitie-Liste	Es wird eine Liste aller Entities angezeigt, in der man Entities bearbeiten, erzeugen und löschen kann.
Neue Beziehung	Fügt eine neue Beziehung in das Projekt ein (STRG+R)
Liste aller Beziehungen	Es wird eine Liste aller Beziehungen angezeigt, in der man diese bearbeiten und löschen kann.
Neuen Generator	Fügt einen neuen Generator in das Projekt ein
Liste aller Generatoren	Es wird eine Liste aller Generatoren angezeigt, in der man diese bearbeiten und löschen kann.
Neue Sequenz	Fügt eine neue Sequenz in das Projekt ein
Liste aller Sequenzen	Es wird eine Liste aller Sequenzen angezeigt, in der man diese bearbeiten und löschen kann.

3.1.5 Menü Schema

Menüfunktionen	**Beschreibung**
Check Modell	Diese Funktion testet die Datenbank. Ist zurzeit noch nicht implementiert.
Generierung der Datenbank	Diese Funktion generiert ein SQL-Script. Im oberen Teil wird der Dateiname festgelegt, danach die Komponenten und am Schluss die zusätzlichen Optionen.

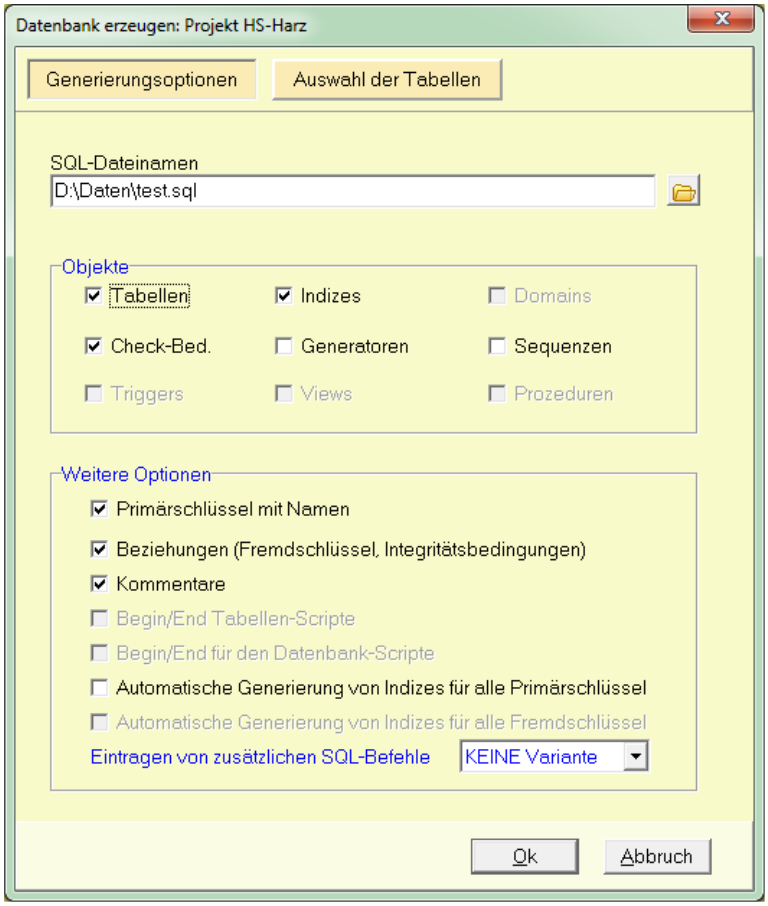

Abbildung 131 Generierung des SQL-Scriptes

Im zweiten Register können einzelne Tabellen ausgewählt werden.

3.1.6 Menü Optionen

Menüfunktionen	**Beschreibung**
Erzeugen eines Datenbankmodells	Mit dieser Funktion erstellt man die Definitionen einer beliebigen realtionalen Datenbank. Alle Datentypen sind frei wählbar.
Taschenrechner	Stellt den Windows-Taschenrechner zur Verfügung.
Dateiliste löschen	Löscht die Liste der alten Dateien.

3.2 Funktionen des Objekt-Baumes

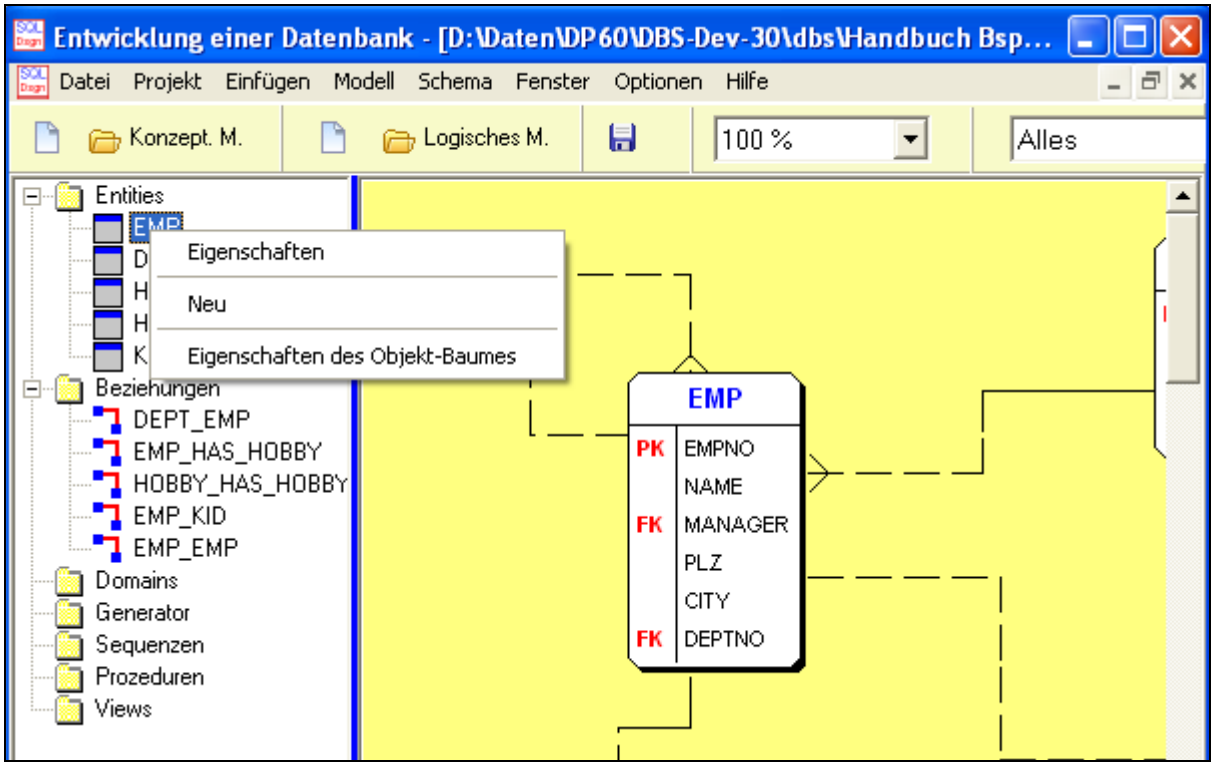

Abbildung 132 Funktionen des Objekt-Baumes

Um die Funktionen zu benutzen muss man zum Einen den Eintrag anklicken. Mit der rechten Maustaste erhält man Zugriff auf drei Einträge (Eigenschaften, Neu, Eigenschaften des Objektbaumes). Der Eintrag „Eigenschaften“ wird nur angezeigt, wenn man auf ein Objekt angeklickt hat. Die Funktion erlaubt das Erzeugen neuer Entities bzw. Relationen. Je nachdem, welches Element man angeklickt hat.

Mit dem letzten Punkt kann man die Schrift und Farbe des Baumes verändern.

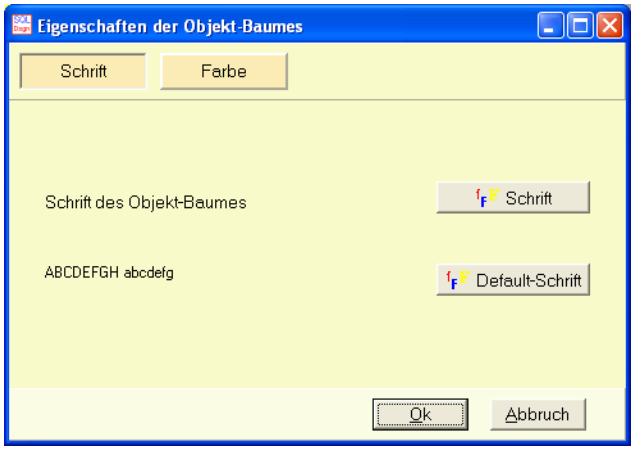

Abbildung 133 Eigenschaftsdialog des Baumes

3.3 Funktionen der Grafik

Die Grafik liefert zwei Funktionen. Zum Einen die Einstellung der Farbe und zum Anderen den Aufruf von Funktionen von Entities.

3.3.1 Funktionen des Grafik-Panels

Ein Doppelklick auf eine freie Stelle liefert die Abbildung 134. Hier kann man die Hintergrundfarbe und die Größe einstellen.

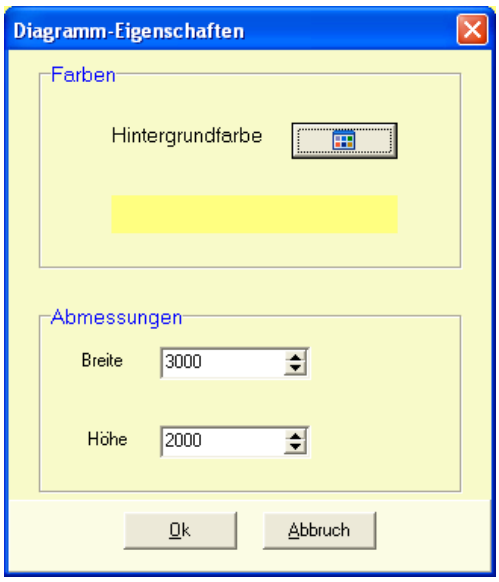

Abbildung 134 Eigenschaften des Grafikpanels

Mit der rechten Maustaste erscheint folgendes Menü. Hier kann man Entities und Beziehungen einfügen. Oder man wählt die zwei Beispiele, die vorgegeben sind.

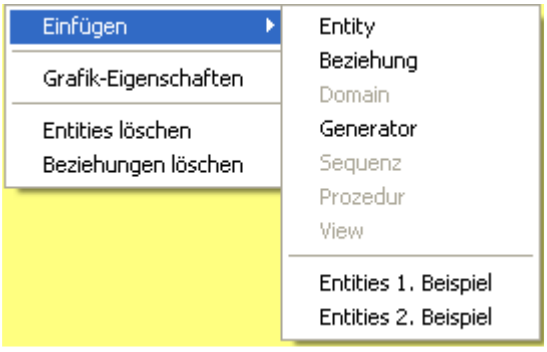

Abbildung 135 Popup-Funktionen des Grafik-Panels

3.3.2 Entity-Funktionen

Bewegt man den Kursor über die Entities und betätigt die rechte Maustaste, so erhält man die Funktionen (Eigenschaften und Farbe). Mit dem Eintrag „Eigenschaften“ wird das Dialogfenster Abbildung 136 angezeigt.

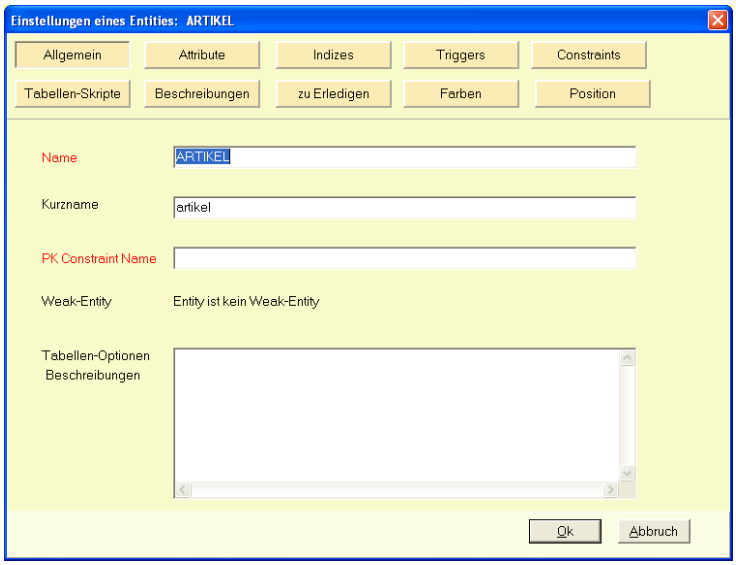

Abbildung 136 Eigenschaften eines Entities

In diesem Dialogfenster funktionieren folgende Register nicht:

- Indizes
- Triggers
- Constraints
- Tabellenscripte

Im Register „Farben“ kann man die verschiedenen Farben auswählen (siehe Abbildung 137).

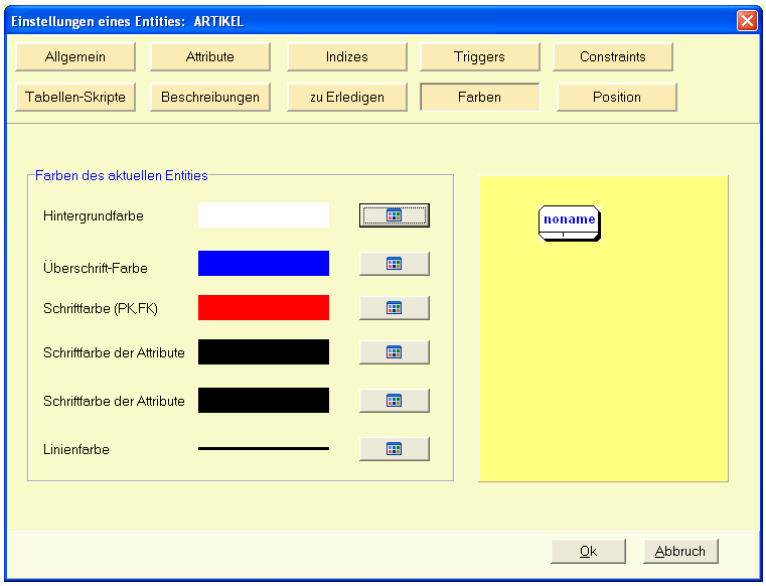

Abbildung 137 Register Farben in den Entity-Einstellungen

Hinweis:
Im Register „Attribute“ werden in einer Tabelle die vorhandenen angezeigt. Mit einem Doppelklick können diese bearbeitet werden.

3.4 Funktionen der Attribute

Aus dem Einstellungs-Dialogfenster „Entity“ kann man die einzelnen Attribute bearbeiten. Die Abbildung 138 zeigt das Fenster zur Bearbeitung der Attribute. Es werden zurzeit noch keine Domains unterstützt, so dass man alle Datentypen direkt eingeben muss. Das Register „Linked-Attributes“ hat noch keine Funktion. Im zweiten Register kann der Default-Wert eingetragen werden.

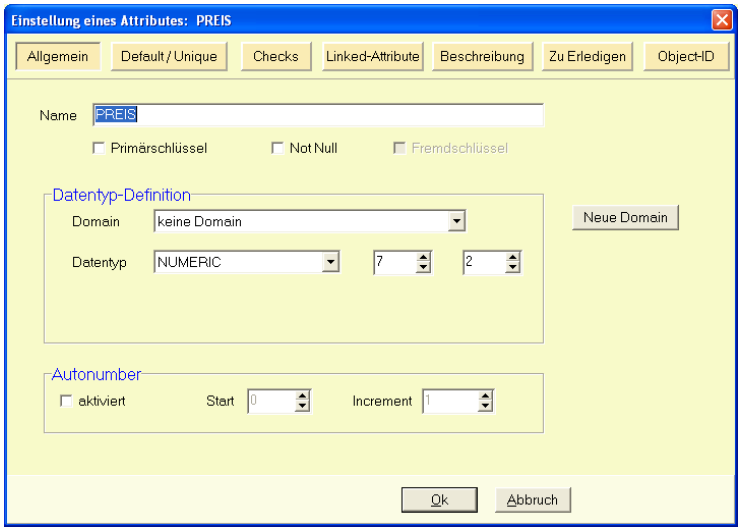

Abbildung 138 Einstellungen der Attribute. Primarykey, logische Modell

Klickt man das Kontrollfeld „Add to Primarykey“ an, so wird beim ersten Aufruf der Datentyp „Integer“ ausgewählt (siehe auch Abbildung 2, Seite 12, Projekteigenschaften (Integer vs. Max Numeric). In den weiteren Aufrufen bleibt der Datentyp erhalten.

Im zweiten Register setzt man einen optionalen Defaultwert und einen Uniquewert. Die Unqiue-Bedingung definiert ähnlich einem Primarykey, dass im Wertebereich eines Attributs keine doppelten Tupel vorkommen. Sinnvoll zum Beispiel bei Abteilungs- oder Artikelnamen.

Das dritte Register setzt eine Prüfbedingung für das Attribut. Verletzt ein neuer Wert diese Bedingung, so werden alle Attribute einer Insert- oder Update-Anweisung nicht in die Datenbank eingetragen.

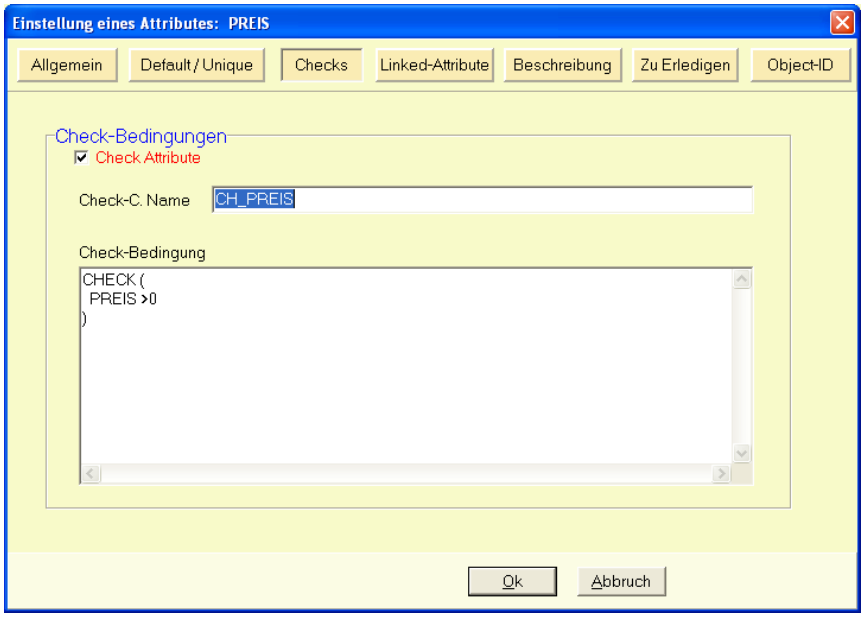

Abbildung 139 Check-Bedingung

Standardmäßig wird eine numerische Abfrage eingefügt. Die Zeichenmanipulations- oder Datumsfunktionen sind sehr speziell.

4 Spezielle Funktionen

4.1 Computed By

Mit Hilfe von „Computed By“ kann man berechnende Attribute in das Entity einfügen. Die Eingangsattribute müssen alle im Entity vorhanden sein.

Beispiel	**Eingangsparameter**	**Ausgangsparameter**
Monats- und Jahresgehalt	Gehalt	12 * Gehalt
Netto- und Bruttogehalt	Preis	1.19 * Preis
Gewässerstationen	Von und Bis	Länge = Bis-Von

4.1.1 Syntax von Computed by

Der erste Begriff zeigt den Namen, dann folgt der Begriff „Computed By“, am Schluss folgt die Formel:

```
   PREIS   COMPUTED BY  ( PREISCENT*0.01 ) ,
```

```
CREATE TABLE ARTIKEL (
  ANR  INTEGER  NOT NULL,
  NAME  VARCHAR(40)  NOT NULL  UNIQUE,
  PREISCENT  INTEGER  DEFAULT 0,
  PREIS_DOUBLE  DOUBLE PRECISION,
  PREIS_NUMERIC  NUMERIC(7,2),
  PREIS   COMPUTED BY  ( PREISCENT*0.01 ) ,

  CONSTRAINT PK_ANR  PRIMARY KEY (ANR)
);
```

Beispieldaten für „Handbuch_bsp7“:

```
INSERT INTO Artikel (ANR, NAME, PREISCENT, PREIS_DOUBLE, PREIS_NUMERIC)
VALUES (1,'Buch', 1245, 12.45, 12.45);

INSERT INTO Artikel (ANR, NAME, PREISCENT, PREIS_DOUBLE, PREIS_NUMERIC)
VALUES (2,'CD', 1210, 12.10, 12.10);

INSERT INTO Artikel (ANR, NAME, PREISCENT, PREIS_DOUBLE, PREIS_NUMERIC)
VALUES (3,'Computer', 14570, 145.70, 145.70);
```

Man sollte einen expliziten Datentyp immer mit angeben.

Definition mit Datentyp:

```
   PREIS   NUMERIC(7,2)  COMPUTED BY  ( PREISCENT*0.01 ) ,
```

4.2 Weak-Attribute

Siehe Kapitel 2.6.7, Seite 77

4.3 Self-Relation

Siehe Kapitel 2.6.6, Seite 75

4.4 Ternäre-Relation

Siehe Kapitel 2.4, Seite 39

4.5 Erstellung einer Datenbank-Typdatei

Dieses Kapitel zeigt die Definition einer neuen Typdatei für Datenbanken. Mit dieser kann man eine beliebige relationale Datenbank definieren. Dazu existieren Grundtypen (Integer, Char, Float, etc.), mit denen die echten Typen verbunden werden. Weitere Punkt sind die Definition der Syntax-Darstellung und die Generierung eines automatischen Counters.

4.5.1 Aufruf der Funktion

Aufruf im Menü Optionen
Eintrag: Erzeugen eines Datenbank-Modells

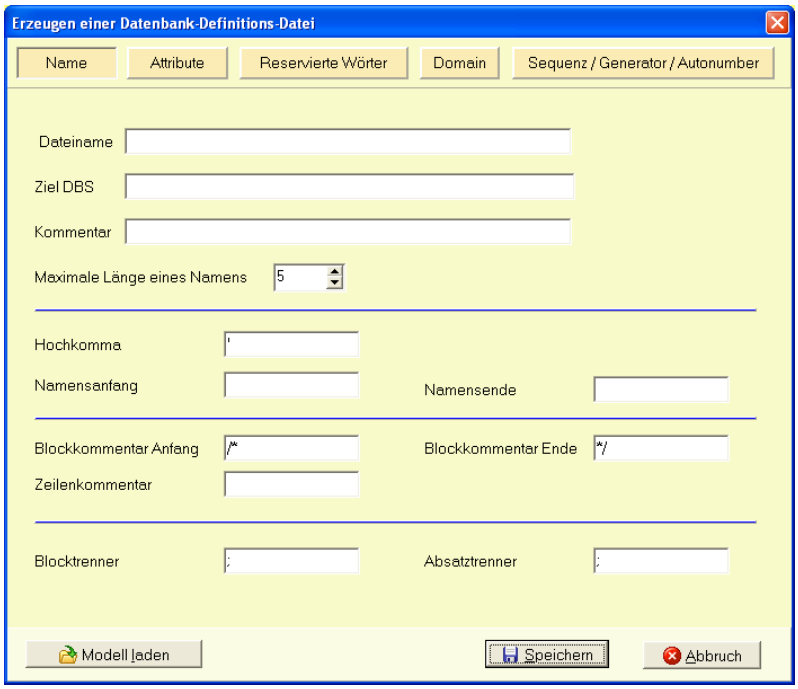

Abbildung 140 Dialogfenster zum Erstellen einer Datenbank-Definition

4.5.2 SQL-Syntax

Hochkomma:
Das Hochkomma können die einfachen und doppelten fungieren. Meistens werden die einfachen benutzt.

Namensanfang / Namensende:
Bei einige Datenbanken werden die Namen der Entites und Relationen in eckige Klanmmern eingegeben (MS SQL-Server, MS Access, NexusDB2). Mit diesem Trick kann man auch Leerzeichen in Namen verwenden.

Blockkommentar:
Die meistens verwenden die normalen Blockkommentare /* */. Einige verwenden aber nur Zeilenkommentare (DB2, MaxDB, mySQL).

Zeilenkommentar:
Die meistens verwenden nur die normalen Blockkommentare /* */. Einige verwenden aber auch Zeilenkommentare(DB2, dBase, dbIsam, FoxPro, Informix, MaxDB, Access, MS SQLK-Server, mySQL, NexusDB, Oracle, Paradox, PostgreSQL, SyBase). Dazu verwenden sie das Zeichen „-“.

Blocktrenner /Absatztrenner:
Der normalen Trenner ist das Semikolon. Einige wenige Datenbank verwenden hier folgende Definition:
#13#10GO#13#10

Dazu gehören: SyBase, MS SQL-Server, MaxDB

Mit den Einstellungen im ersten Register sollten alle Eventualitäten einstellbar sein.

Im ersten Register definiert man die allgemeinen Daten inklusive des Syntax. Im wichtigen zweiten Register definiert man die einzelnen Attribute mit ihren Grundtypen. Diese Typen dienen der Auswahl, welche grafischen Elemente bei der Auswahl aktiviert werden müssen. Das dritte Register zeigt die Begriffe, die in der Datenbank intern verwendet werden. Diese dürfen natürlich nicht als Namen für Entities oder Attribute dienen.

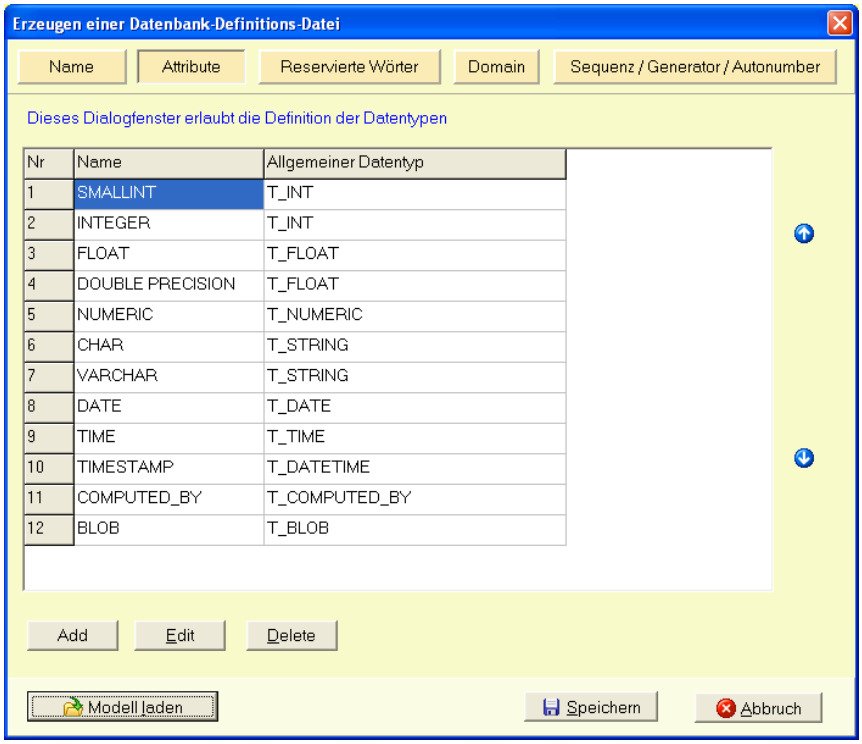

Abbildung 141 Datenbank-Definitionsdialog

Nicht jede Datenbank unterstützt Domains, also vordefinierte Typen mit optionalen Prüfbedingungen. Der Designer erlaubt aber trotz negativer Aussage, die Definition und Benutzung von Domains. Unterstützt die Datenbank keine Domains, werden die Domain-Typen durch die Definitionen ersetzt.

Ein etwas anderer Fall besteht bei der Definition der „Counter“. Die Datenbanken „Interbase“ und Firebird unterstützen den einfachen Generator. Oracle dagegen verwendet die Sequenz.

Man muss nun selber entscheiden, welche Definition bei der Zieldatenbank passender ist. Gegebenenfalls muss man das SQL-Skript anpassen.

Hinweis:

Die aktuelle Version 4,0 des Designers unterstützt noch nicht Domains und Trigger. Die Verhaltensweise wird aber dem obigen Text entsprechen!

Das letzte Register beinhaltet die Definition eines Counters. Auch hier ist die übliche Vielfalt zu bewundern. Access verwendet einen einfachen Zähler als Datentyp. Interbase und Firebird benutzen einen Generator. Oracle, Informix und Postgre verwenden die Sequenz. Hier aber auch in abgestufter Variante. Deshalb kann man alle Elemente einzeln aktivieren. Bei Problemen kann man ja das erstellte Skript immer noch abändern. Eine E-Mail wäre anr den Entwickler wäre wünschenswert.

Sybase und MS-SQL verwenden für Integer und Numeric-Datentypen einen internen Counter. Dieser hat einen Start- und Incrementwert. In unteren Beispiel wird der Startwert auf zwei gesetzt und der Incrementwert ist drei (IDENTITY).

Beispielsyntax für MS SQL-Server:

```
CREATE TABLE [KUNDEN] (
  [KUNDENNR]  INTEGER  IDENTITY(2,3)   NOT NULL,
  [NAME]  CHAR(50),
  [VORNAME]  CHAR(50),
  [STRASSE]  CHAR(50),
  [PLZ]  DECIMAL(5),
  [ORT]  CHAR(50),

  PRIMARY KEY ([KUNDENNR])
)
GO
```

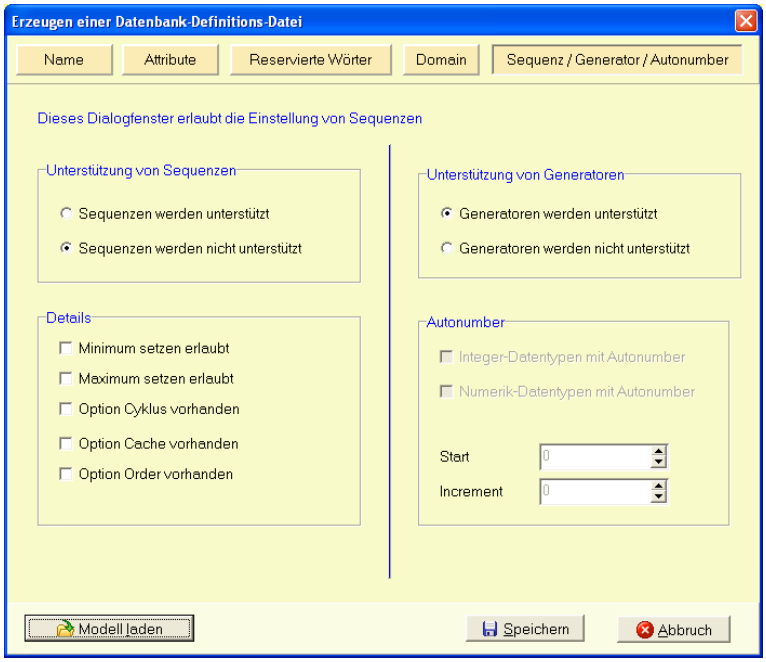

Abbildung 142 Sequenz, Generator definieren

4.6 Trigger in Oracle

Trigger sind Prozeduren, die mit DML-Operationen verknüpft sind:

- Insert
- Update
- Delete

Jede Prozedure kann man vorher oder nachher definieren:

- Before Insert
- After Insert
- Before Update
- After Update
- Before Delete
- After Delete

4.6.1 Syntax der Trigger

```
CREATE OR REPLACE TRIGGER Trigger_Name
  BEFORE INSERT OR UPDATE ON Tablename
    FOR EACH ROW
    BEGIN
        -- Anweisung
    END;
```

Im Anweisungblock hat man mit der “Variablen “:NEW” Zugriff auf den aktuellen Datensatz

Beispiel:

```
CREATE OR REPLACE TRIGGER Trigger_Kunden
  BEFORE INSERT OR UPDATE ON Kunde
    FOR EACH ROW
    BEGIN
        :NOW.MitarbeiterNr := 123;
        :NOW.Datum := NOW;
    END;
```

4.7 Generator / Sequenz

Häufig benötigt man einen ganzzahligen aufsteigenden Schlüssel für ein Feld. Nun kann man mit folgendem Code diesen bestimmen (Pseudocode und PL-SQL):

```
MaxNr =    SELECT max(MatrNr)
           FROM Student

INSERT INTO Student (MatrNr, Name)
    VALUES (MaxNr +1, "meier");
```

Diese Variante funktioniert leider nicht sicher in einem Netzwerk! Darum existieren Funktionen, die garantieren, dass ein Schlüssel immer nur einmalig vergeben wird.

4.7.1 Generator in der Datenbank Firebird

Beispiel:

Anlegen eines Studenten mittels Generator

```
CREATE GENERATOR GEN_NUMBER;
SET GENERATOR  GEN_ NUMBER   TO 999;

INSERT INTO EMP (empno, Name)
```

VALUES (GEN_ID(GEN_NUMBER,1), 'Bates', ' Norman ');

4.8 Generator für die Mitarbeiternummer EMPNO

4.8.1 Create Table

Im ersten Schritt muss man die Tabelle erstellen.

```
CREATE TABLE EMP (
  EMPNO  INTEGER  NOT NULL,
  NAME  VARCHAR(30)  NOT NULL,
  MANAGER  INTEGER  NOT NULL,
  PLZ  CHAR(5),
  CITY  VARCHAR(50),
  DEPTNO  INTEGER  NOT NULL,

  PRIMARY KEY (EMPNO)
);
```

4.8.2 Create Generator

Der zweite Schritt erstellt den Generator und setzt einen Anfangswert (Optional)

SQL-Code:

```
CREATE GENERATOR GEN_EMPNO;

SET GENERATOR  GEN_EMPNO   TO 1000;
```

Der erste Mitarbeiter hat dann die Nummer 1001. Der Generator funktioniert nicht für bestehende Daten.

Löschen eines Generators: DROP GENERATOR "GEN_EMPNO ";

Ablauf:

- Starten der IBO-Konsole:
- Aufruf des SQL-Editors
- Einfügen der beiden Befehle.
- Ausführen der Befehle

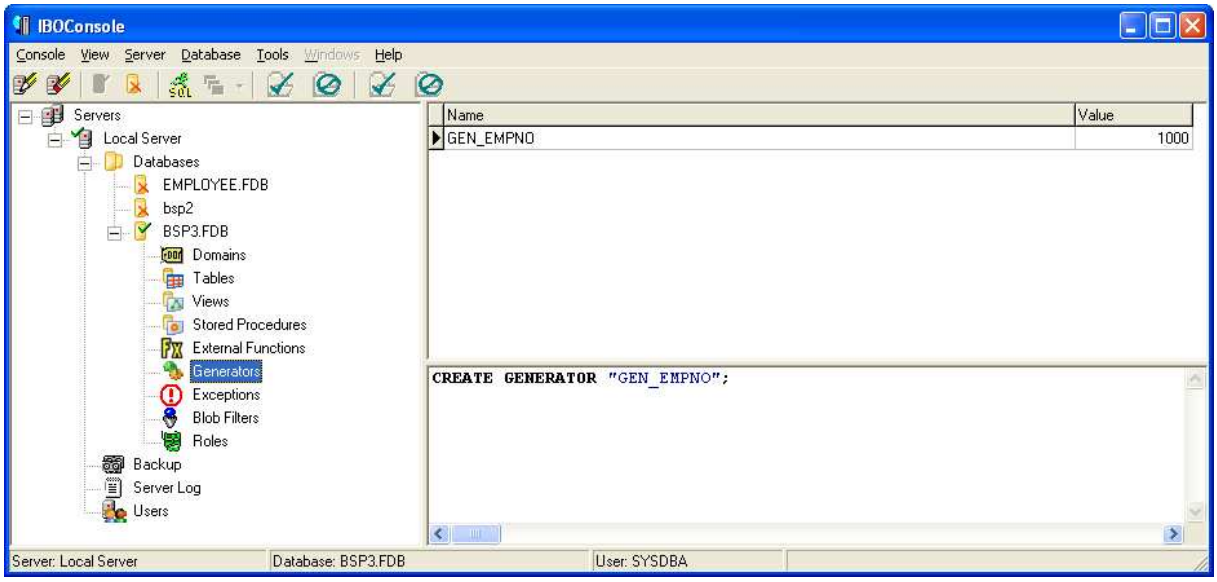

Abbildung 143 Generator erstellt

Im letzten Schritt muss der Generator mit einem Trigger verbunden werden:

- Aufruf der Tabelle EMP
- Anklicken des Register Trigger:

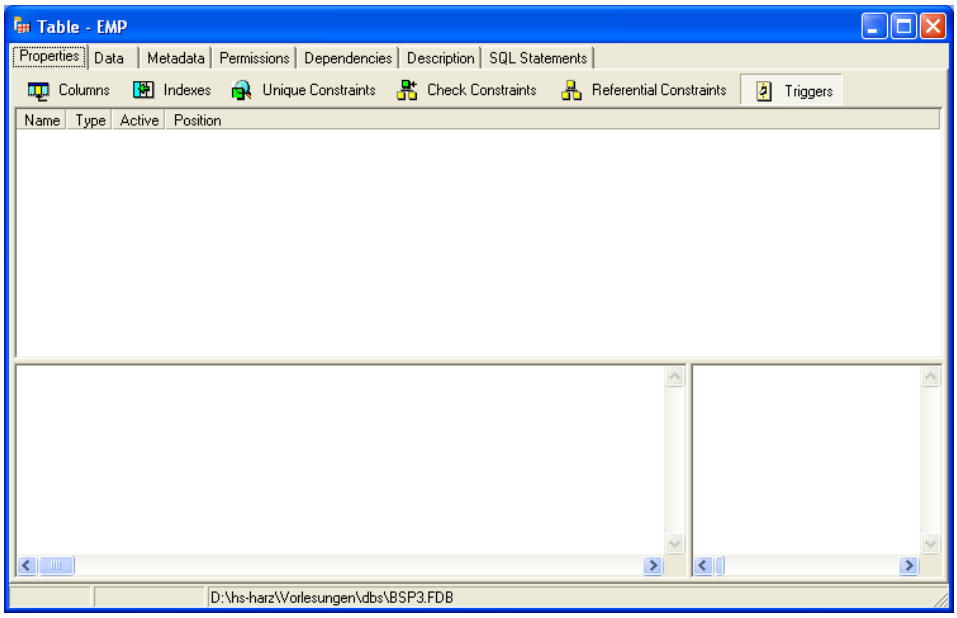

Abbildung 144 Trigger für die Tabelle EMP erstellen

Mit der rechten Maustaste den Eintrag „new“ wählen.

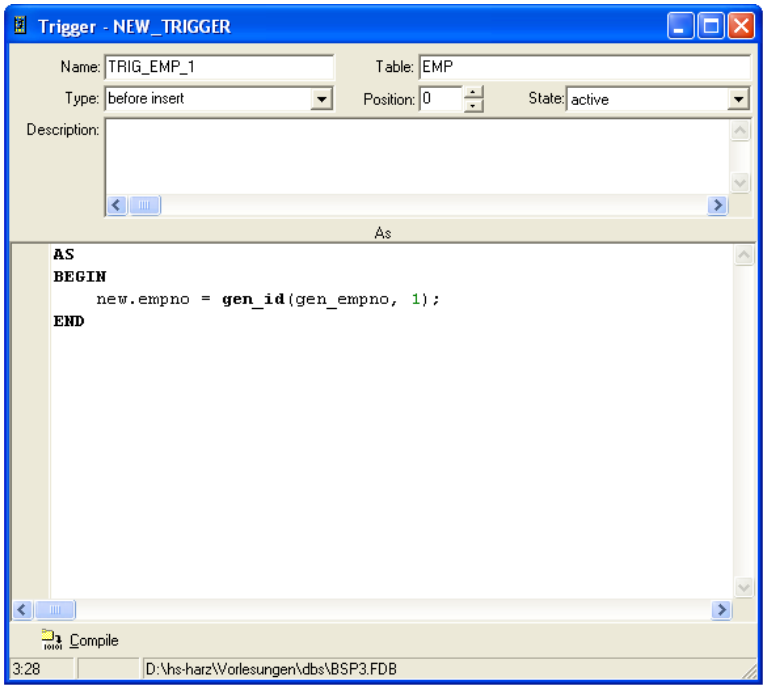

Abbildung 145 Generator im Trigger

Als Type wird natürlich „Insert Before“ ausgewählt. Die Position ist hier Null, da es der erste Trigger für diese Tabelle ist. Eingetragen wird der untere Quellcode:

```
AS
BEGIN
     new.empno = gen_id(gen_ empno, 1);
END
```

Der Begriff „new“ beinhaltet den Zugriff auf den neunen Mitarbeiter. Der Generator wird über die Funktion „Gen_id“ mit den beiden Parameter ausgeführt. Die eins zeigt den Sprung um jeweils eine Position.

Danach wird der Quellcode mit dem Schalter „Compile“ übersetzt (links unten).

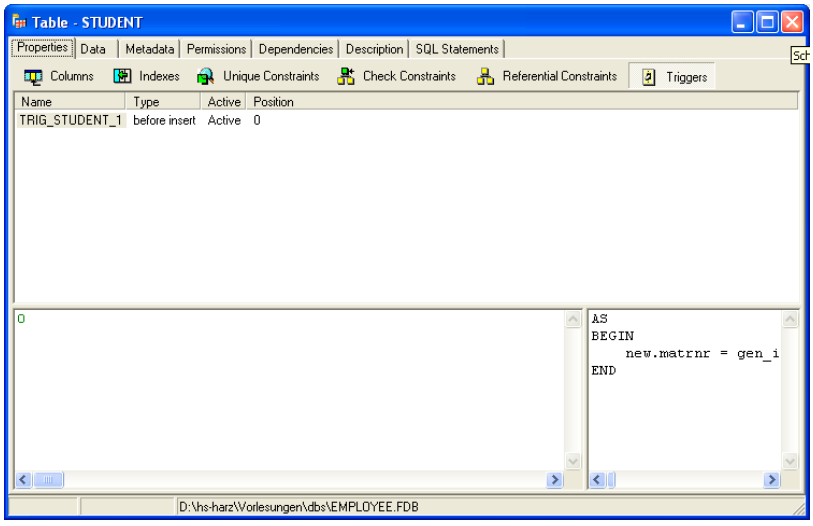

Abbildung 146 Fertiger Trigger

Es folgt der Test mit Insert-Befehlen:

```
INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (2222, 'Bischoff', 11, '39147', 'Erxleben',1);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES (3333,'Kardinal', 11, '38844', 'Halberstadt',1);
```

Ergebnis:

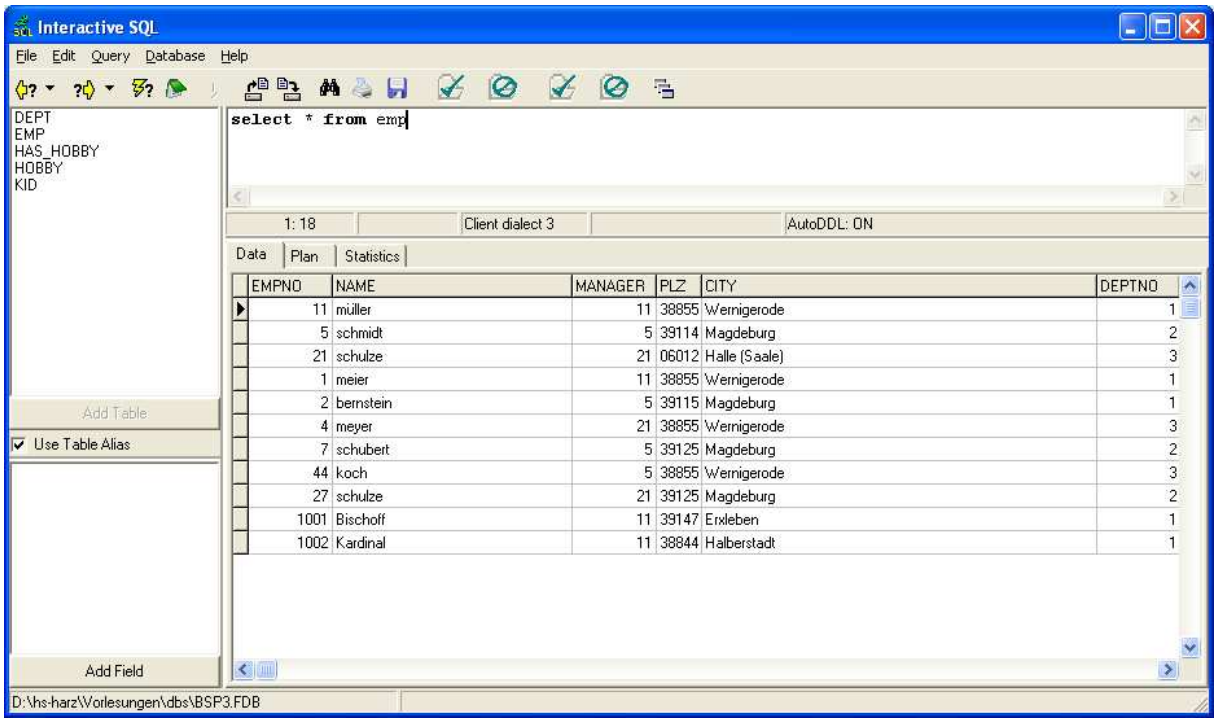

Abbildung 147 Ergebnis eines Insert-Befehls

Vereinfacht kann man dann auch dieses eingeben:

```
INSERT INTO EMP ( name, manager, plz, city, deptno)
VALUES ('Bischoff', 11, '39147', 'Erxleben',1);

INSERT INTO EMP (empno, name, manager, plz, city, deptno)
VALUES ('Kardinal', 11, '38844', 'Halberstadt',1);
```

Der folgende Code zeigt eine weitere Verwendung des obigen Triggers. Mit diesem Zusatz erhält jeder Mitarbeiter den Name „Müller“.

```
AS
BEGIN
    new.empno = gen_id(gen_empno, 1);
    new.name = 'Müller';
END
```

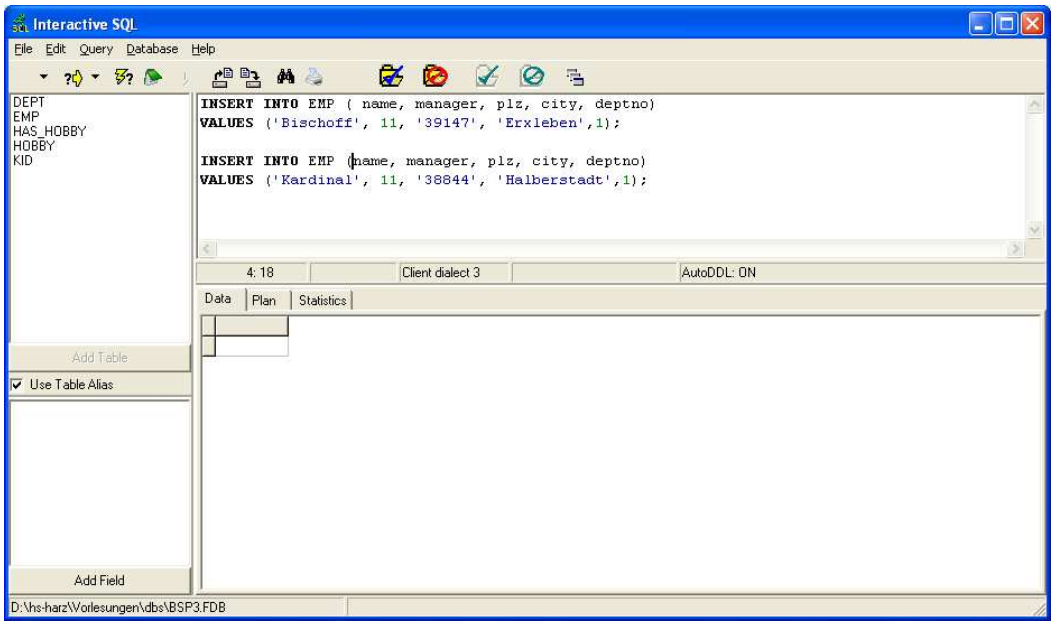

Abbildung 148 Test des erweiterten Triggers

Im obigen Beispiel wurde der Trigger um die Namensänderung erweitert. Als Ergebnis erhält man folgende Tabelle:

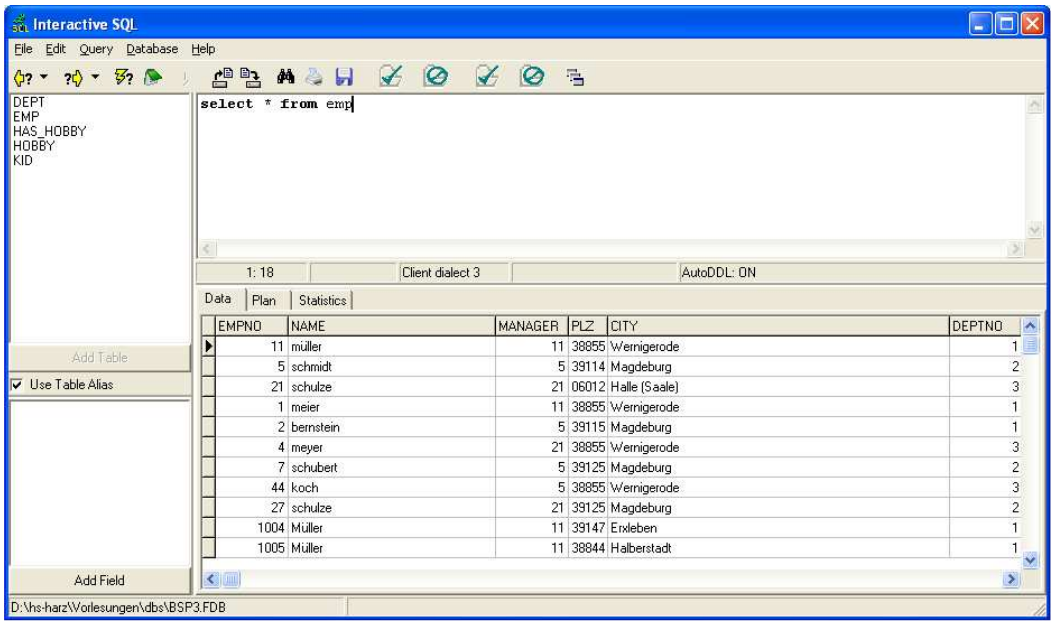

Abbildung 149 Ergebnis des Einfügens mittels Trigger

Obwohl die Namen eingeben wurden, setzte der Trigger die Namen auf Müller.

4.8.3 Sequenz in Oracle

Syntax:

```
CREATE SEQUENCE sequence
[INCREMENT BY n]
[START WITH n]
[{MAXVALUE n | NOMAXVALUE}]
[{MINVALUE n | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE n | NOCACHE}];
```

Beispiel:

Anlegen eines Studenten mittels Sequenz

```
CREATE SEQUENCE stud_matrnr
 INCREMENT BY 1
 START WITH 100
 MAXVALUE 100000
 NOCACHE
 NOCYCLE;
```

5 Literatur

[1] Ramez Elmasri, Shamkant B. Navathe: Grundlagen von Datenbanksystemen,
ISBN: 3-8273-7153-8

[2] /Codd-70/ Codd, E. F.:
A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol 13, No. 6, p. 377-387, 1970

[3] /Chen-76/ Chen, P. P.:
The Entity-Relationship Model-Toward a Unified View of Data, ACM Transactions on Database Systems, Vol. 1, p. 9-36, 1976

[4] /Date-85/Date, C. J.:
An Introduction to Database System, Vol. 1, Reading Massachusetts, 1985

[5] /Jack-89/ Jackson, G. A.:
Entwurf relationaler Datenbanken, Hanser Verlag München, Wien, 1989

[6] /Voss-91/ Vossen, G., Witt, K.-U.:
Entwicklungstendenzen bei Datenbanksystemen, Oldenbourg Verlag München, Wien, 1991

[7] /Wede-81/ Wedekind, H.:
Datenbanksysteme I, Bibliographisches Institut Mannheim, 1981

[8] /Trau-91/Trautloft, R., Lindner, U.:
Datenbanken - Entwurf und Anwendung, Verlag Technik GmbH Berlin, 1991

[9] /Schi-96/ Schicker, E.:
Datenbanken und SQL, B. G. Teubner Stuttgart, 1996

[10] /Sau-95/ Sauer, H.:
Relationale Datenbanken, Addison-Wesley, 1995

[11] /Mei-97/ Meier, A., Wüst, Th.:
Objektorientierte Datenbanken, dpunkt Verlag, 1997

[12] /Abb-99/ Abbey, Corey, Abramson:
Oracle 8i für Einsteiger, Hanser Verlag 1999

[13] /Pon-99/ Ponndorf, St., Matthäus, W.-G.:
Oracle 8i und Java, Addison-Wesley, 1999

[14] /Saa-99/ Saake, G., Heuer, A.:
Datenbanken – Implementierungstechniken, MITP Verlag GmbH, 1999

[15] /Käh-99/ Kähler, W.-M.:
Relationales und objektrelationales SQL, Vieweg & Sohn Verlagsgesellschaft mbH, 1999

[16] /Cul-99/ McCullough-Dieter, C.:
Oracle8i für Dummies, MITP-Verlag GmbH Bonn, 1999

[17] /Stü-00/ Stürner, G.:
Oracle 8i – Der objekt-relationale Datenbank Server, Verlag dbms publishing, 2000

[18] /Kof-01/ Kofler, M.:
MySQL, Addison-Wesley, 2001

[19] /Ric-01/ Riccardi, G.:
Datenbanksysteme mit Internet und Java-Applikationen, Addison-Wesley, 2001

[20] /Hoh-02/ Hohenstein, U., Pleßer, V.:
Oracle 9i, Effiziente Anwendungsentwicklung mit objektrelationalen Konzepten, dpunkt.verlag, 2002

[21] /Kuh-01/ Kuhlmann, G., Müllmerstadt, F.:
SQL, Der Schlüssel zu relationalen Datenbanken, Rowohlt Taschenbuch Verlag, 2001

6 Indexverzeichnis

A

B

E

F

G

H

I

K

L

M

O

R

S

W