

Fachbereich
Automatisierung und Informatik

▲ Hochschule Harz
Wernigerode

**Kurzhandbuch zum
Datenbank-Designer
und der
FBConsole**

Version 4,74

von

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

Version 17. Februar 2016

Dipl. Inf., Dipl.-Ing. (FH) Michael Wilhelm
Friedrichstraße 57 - 59
38855 Wernigerode

Raum: 2.202

Tel.: 03943/659-338

Fax: 03943/659-399

Email: mwilhelm@hs-harz.de

Web: <http://www.miwilhelm.de>

<http://www.miwilhelm.de/datenbanken/dbspaket/index.html>

<http://www.miwilhelm.de/datenbanken/designer/index.html>

Inhaltsverzeichnis

1	Zip-Datei.....	5
1.1	Installation	5
1.2	Inhalt der ZIP-Datei	5
2	Datenbank-Designer	6
2.1	Die Eigenschaften in Kurzform:	6
2.2	Menüfunktionen	7
2.2.1	Menü Datei	7
2.2.2	Menü Projekt	8
2.2.3	Menü Einfügen	8
2.2.4	Menü Modell	8
2.2.5	Menü Schema	8
2.3	Ablauf bei der Erstellung eines konzeptionellen Modells	9
2.3.1	Neues Entity	9
2.4	Ablauf bei der Erstellung eines logischen Modells	12
2.4.1	Auswahl des logischen Modell	12
2.4.2	Neues Entity	13
2.5	Neue Beziehung	15
2.6	Erstellen der Skripte	19
2.7	Sequenz	20
2.7.1	Generator oder Sequenz definieren	20
2.7.2	Sequenzbefehle mit SQL:	20
3	FBConsole.....	21
3.1	Neue Datenbank erstellen	22
3.2	Daten eintragen	24
3.2.1	SQL-Befehle zum Eintragen:	24
3.3	Testabfragen	25
4	SQL-Anweisungen	27
4.1	Aufbau der SQL-Sprache	27
4.2	SELECT-Anweisung	27
4.3	DISTINCT	28
4.4	BETWEEN	28
4.5	LIKE	28
4.6	Join	29
4.6.1	Inner-Join	30
4.6.2	Left-Join	30
4.6.3	Right-Join	30
4.6.4	Outer-Join	31
4.7	SubSelect, SubQuery	31
4.8	Gruppenfunktionen	33
4.8.1	Syntax der GROUP BY-Klausel	34
4.8.2	Richtlinien für die Verwendung von Gruppenfunktionen	34

4.8.3	SUM-Funktion	34
4.8.4	AVG-Funktion	35
4.8.5	MIN-Funktion	35
4.8.6	MAX-Funktion	36
4.8.7	COUNT-Funktion	36
4.8.8	Group by having	37
4.9	Codierungsfehler	37
4.10	Die INSERT INTO Anweisung	38
4.11	Die UPDATE Anweisung	38
4.12	Die DELETE FROM Anweisung	38
5	DDL-Befehle.....	39
5.1	Die CREATE TABLE Anweisung	39
5.2	Computed By	39
5.2.1	Syntax von Computed by	39
6	Literatur und Links.....	40
7	Indexverzeichnis	42

Abbildungsverzeichnis

Abbildung 1	Überblick des Programms	6
Abbildung 2	Eintragen des Namens für das Entity	9
Abbildung 3	Primärschlüssel Kundennr	9
Abbildung 4	Attribut Gehalt mit genau zwei Nachkommastellen	10
Abbildung 5	Attribut Gehalt mit dem Defaultwert	10
Abbildung 6	Attribut Gehalt mit einer Prüfbedingung	11
Abbildung 7	Attribute eines Kundens	11
Abbildung 8	Auswahl des logischen Modells	12
Abbildung 9	Eintragen des Namens für das Entity	13
Abbildung 10	Primärschlüssel Kundennr	13
Abbildung 11	Attribut Gehalt mit genau zwei Nachkommastellen	14
Abbildung 12	Attribut Gehalt mit dem Defaultwert	14
Abbildung 13	Attribut Gehalt mit einer Prüfbedingung	15
Abbildung 14	Attribute eines Kundens	15
Abbildung 15	Neue Beziehung, rechts ist das abhängige Entity	16
Abbildung 16	Eintragen der Texte	16
Abbildung 17	Attribut "Foreign-Key" eintragen	17
Abbildung 18	Name des "Foreign-Key" eintragen	17
Abbildung 19	Beziehungsdialog mit allen Daten	18
Abbildung 20	Beziehung im ERM-Abbildung	18
Abbildung 21	Dialog zum Erzeugen der SQL-Skripte	19
Abbildung 22	Hauptfenster der FBConsole	21
Abbildung 23	Erstellen der Datenbank "Firma.fdb"	22
Abbildung 24	Eintragen der SQL-Skripte	23
Abbildung 25	Anzeige der Datenbank-Tabellen	23
Abbildung 26	Daten mittels „Insert-Into“ eintragen	24
Abbildung 27	Abfrage bezüglich der Artikel	25
Abbildung 28	Abfrage bezüglich der Kunden	25
Abbildung 29	Abfrage bezüglich der Bestellungen	26
Abbildung 30	Abfrage bezüglich der Bestellungen	26
Abbildung 31	Subselect-Beispiel	32
Abbildung 32	Gruppenfunktionen mit der Tabelle Employee	33
Abbildung 33.:	Die UPDATE Anweisung	38

1 Zip-Datei

Dieses Script bietet einen Einstieg in die Benutzung von Datenbanken unter Winform / WPF. Dabei wird ein Gesamtpaket vom Datenbank-Designer, über eine Datenbank-Console bis hin zu grafischen Programme bereitgestellt.

1.1 Installation

- Entzippen der Datei in einen beliebigen Ordner

1.2 Inhalt der ZIP-Datei

Ordner Designer

- Datenbank-Designer
 - Programm „DBW_Designer.exe“
 - Datenbank-Designer-Handbuch.pdf
 - Ordner „dbs“
 - Ordner „def“

Ordner FBConsole

- Firebird-Employee-Datenbank
- Datenbank_Employee.pdf
- dbs_Firebird_Oracle.pdf
- Tabelle der Firebird-Beispieldatenbank
 - Employee.fdb
 - Firma.fdb
 - HANDBUCHBEISPIEL6.FDB
 - HANDBUCHBEISPIEL7.FDB
 - Pundt_Student.fdb
- Firebird-Datenbank-Konsole, nur Programm
 - FbConsole.exe
- DLL's
 - fbembed.dll
 - FirebirdSql.Data.FirebirdClient.dll
 - icudt30.dll
 - icuin30.dll
 - icuuc30.dll

Ordner IBOConsole

- IBOConsole.exe
- Programmdateien von IBOConsole

2 Datenbank-Designer

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.

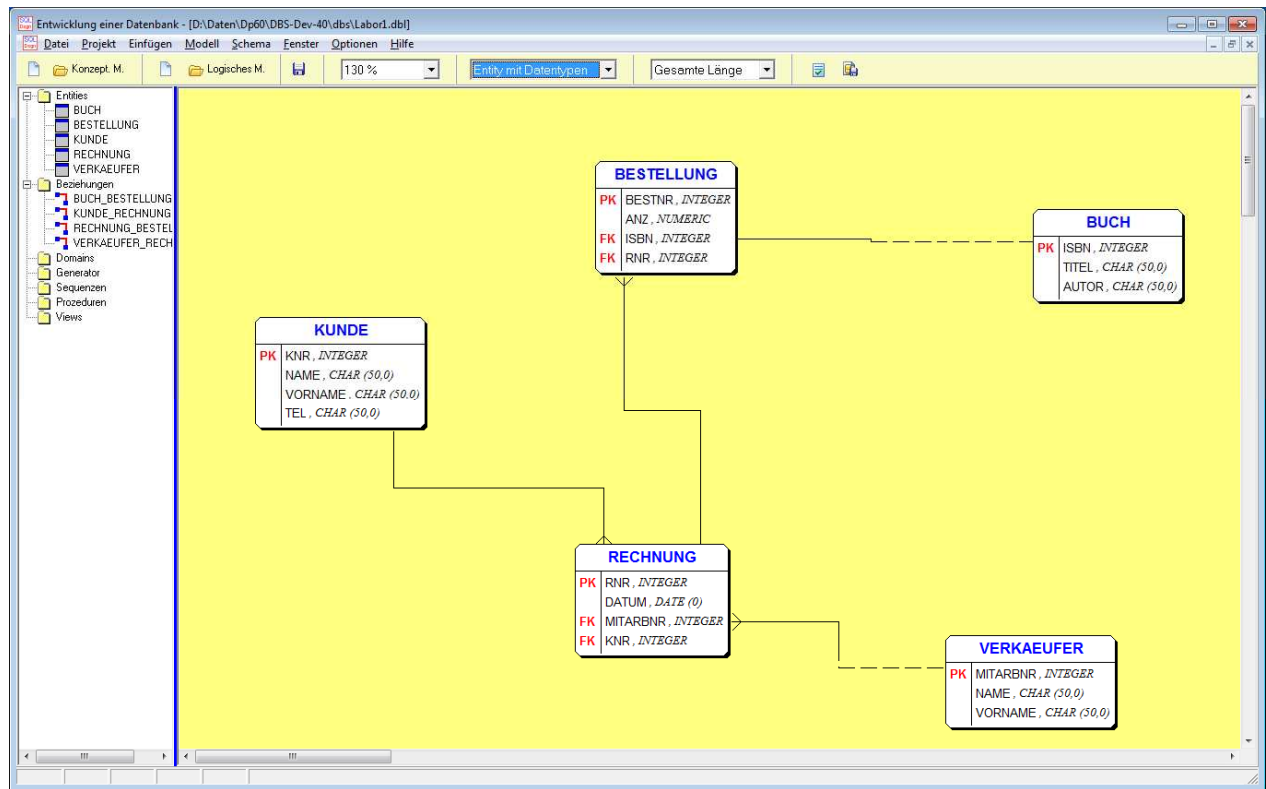


Abbildung 1 Überblick des Programms

2.1 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, Autonumeric)
 - 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-Skriptes
- Vielfältige Farb- und Schriftwahl
- Anzeige der Relationen mit Krähenfuss oder Punktnotation
- Export der Daten nach Winword
- Export der ERM-Grafik in die Zwischenablage

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
- Es gibt **zusätzliche Anweisungen**, die für Initialisierungen benutzt werden können.
 - Automatische Eingabe von festen Werten in Tabellen

Konzeptionelles Modell:

- Es kann nun auch ein Defaultwert eingegeben werden
- Es kann nun auch eine Check-Bedingung eingegeben werden
- Die Länge der Kurznamen wird auf 8 Zeichen begrenzt
- Es gibt einen Test der Relationsnamen FK_???? auf Länge <=31 Zeichen
- Es gibt 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

Logisches Modell:

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

Dieses Kapitel beschreibt kurz die wichtigsten Funktionen.

2.2 Menüfunktionen

2.2.1 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 logisches Projekt(STRG+O)
Öffnen	Öffnet ein vorhandenes konzeptionelles Projekt (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.

2.2.2 Menü Projekt

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

2.2.3 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)
Sequenz	Fügt eine Sequenz für ein Attribut in das Projekt

2.2.4 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)
Entity-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.
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.

2.2.5 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.

2.3 Ablauf bei der Erstellung eines konzeptionellen Modells

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

2.3.1 Neues Entity

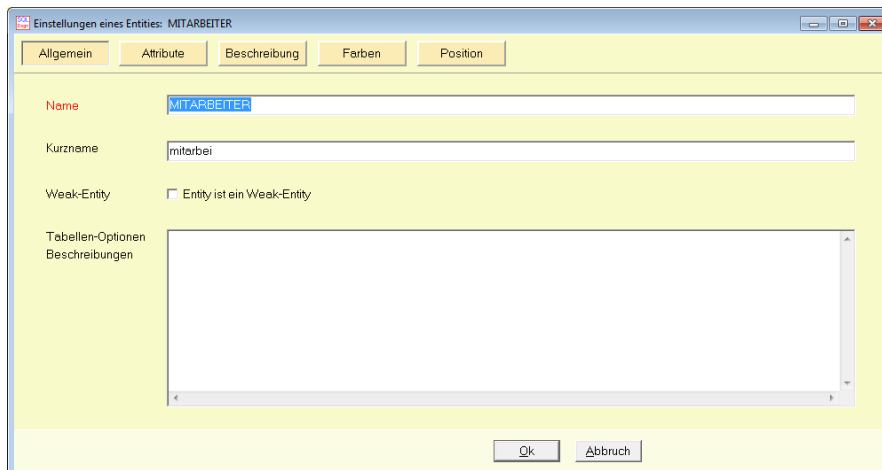


Abbildung 2 Eintragen des Namens für das Entity

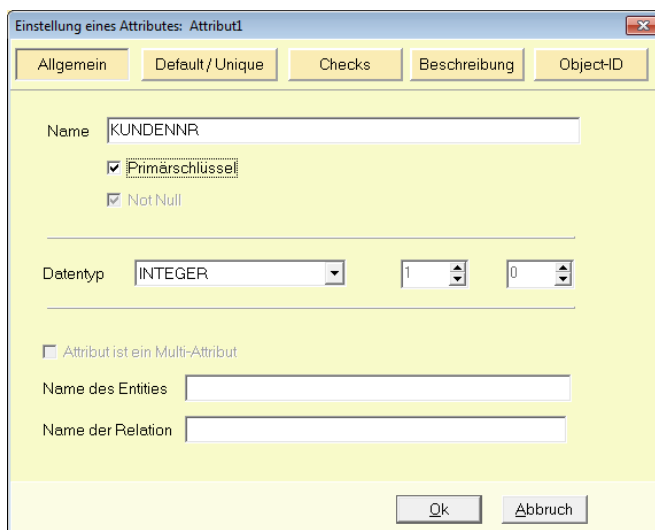


Abbildung 3 Primärschlüssel Kundennr

Einstellung eines Attributes: Attribut2

Allgemein Default / Unique Checks Beschreibung Object-ID

Name: GEHALT

☐ Primärschlüssel

☐ Not Null

Datentyp: NUMERIC 7 2

☐ Attribut ist ein Multi-Attribut

Name des Entities:

Name der Relation:

Ok Abbruch

Abbildung 4 Attribut Gehalt mit genau zwei Nachkommastellen

Einstellung eines Attributes: Attribut2

Allgemein Default / Unique Checks Beschreibung Object-ID

Default-Wert

☒ Default-Wert

Wert: 1000

Datum, Zeit, Boolean, Currency werden als String gespeichert (Ohne Prüfung).

Unique-Bedingungen

☐ Unique

Check-C. Name:

Ok Abbruch

Abbildung 5 Attribut Gehalt mit dem Defaultwert

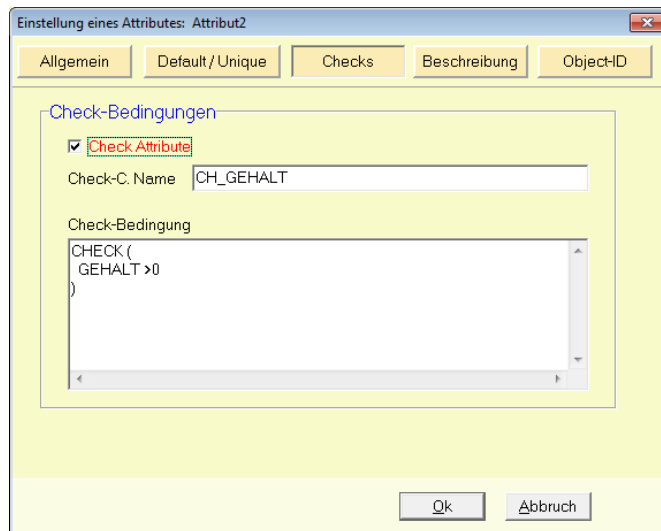


Abbildung 6 Attribut Gehalt mit einer Prüfbedingung

Es werden nur Gehälter größer Null eingetragen. Im Fehlerfall gibt es eine Fehlermeldung seitens des Datenbanksystems.

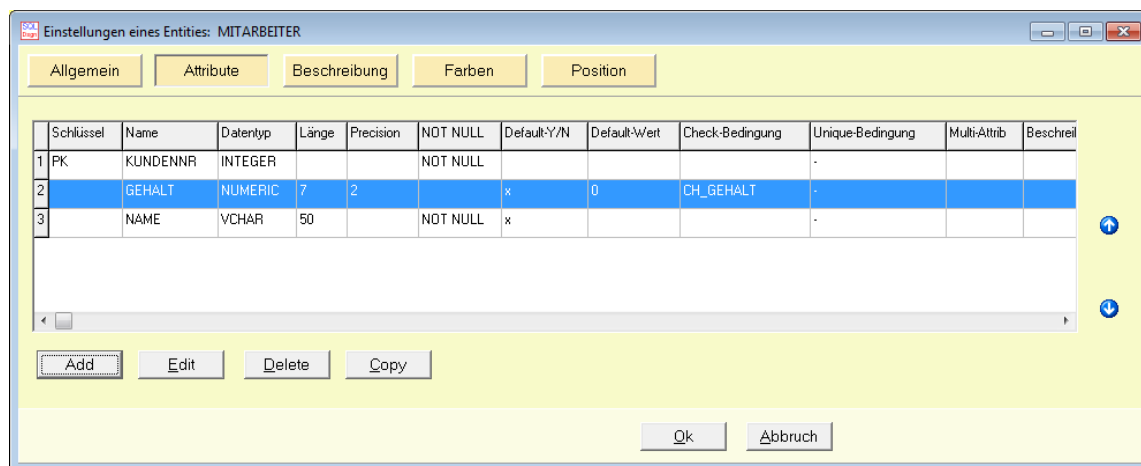


Abbildung 7 Attribute eines Kunden

2.4 Ablauf bei der Erstellung eines logischen Modells

- Erzeugen eines neuen Fensters für ein logisches Modell (STRG+N).
- Einstellen der Projekteinstellungen (Darstellungen der Beziehungen, Texte)
- Erstellen der Entities (STRG+E)
 - Eintragen der Attribute, ohne Fremdschlüssel
 - Festlegen der Primärschlüssel
 - Zusätzliche Eintragungen (Check, Unique, Weak-Entities, Generator, Sequenz)
 - Eintragen der Texte (Beschreibungen etc.)
- Erstellen der Beziehungen (STRG+R)
- Aufruf der Generierung zum SQL-Skript (F9)

2.4.1 Auswahl des logischen Modell

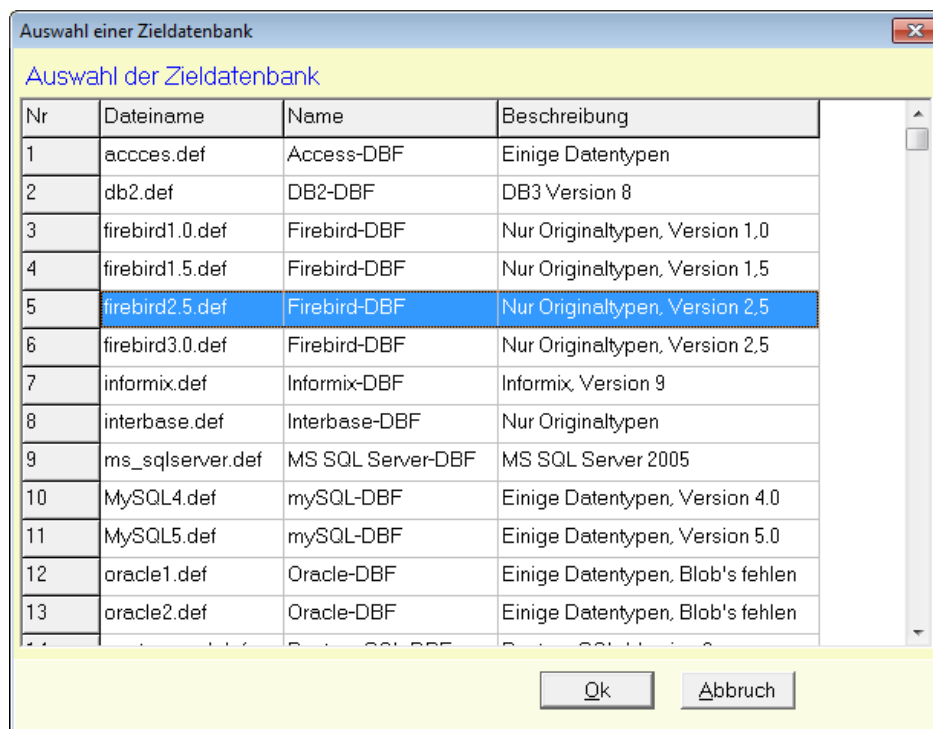


Abbildung 8 Auswahl des logischen Modells

2.4.2 Neues Entity

Einstellungen eines Entitys: MITARBEITER

Allgemein | Attribute | Indizes | Triggers | Constraints

Tabellen-Skripte | Beschreibungen | zu Erledigen | Farben | Position

Name: MITARBEITER

Kurzname: mitarbei

PK Constraint Name: PK_MITARBEITER

Weak-Entity: ☐ Entity ist ein Weak-Entity

Tabellen-Optionen Beschreibungen

Ok Abbruch

Abbildung 9 Eintragen des Namens für das Entity

Einstellung eines Attributs: Attribut1

Allgemein | Default / Unique | Checks | Linked-Attribute | Beschreibung | Zu Erledigen | ObjectID

Name: KUNDENNR

☒ Primärschlüssel ☒ Not Null ☐ Fremdschlüssel

Datentyp-Definition

Domain: keine Domain

Datentyp: INTEGER

Formel:

Datentyp: kein Datentyp

Autonumber

☐ aktiviert Start: 0 Increment: 1

Neue Domain

Ok Abbruch

Abbildung 10 Primärschlüssel Kundennr

Einstellung eines Attributes: Attribut2

Allgemein | **Default / Unique** | Checks | Linked-Attribute | Beschreibung | Zu Erledigen | Object-ID

Name: GEHALT

☐ Primärschlüssel ☐ Not Null ☐ Fremdschlüssel

Datentyp-Definition

Domain: keine Domain Neue Domain

Datentyp: NUMERIC 7 2

Formel:

Datentyp: kein Datentyp 1 0

Autonumber

☐ aktiviert Start: 0 Increment: 1

Ok Abbruch

Abbildung 11 Attribut Gehalt mit genau zwei Nachkommastellen

Einstellung eines Attributes: Attribut2

Allgemein | **Default / Unique** | Checks | Linked-Attribute | Beschreibung | Zu Erledigen | Object-ID

Defaultwert: (NUMERIC)

☒ Defaultwert

Wert: 1000

Datum, Zeit, Boolean, Currency werden als String gespeichert (Ohne Prüfung).

Unique-Bedingungen

☐ Unique

Check-C. Name:

Ok Abbruch

Abbildung 12 Attribut Gehalt mit dem Defaultwert

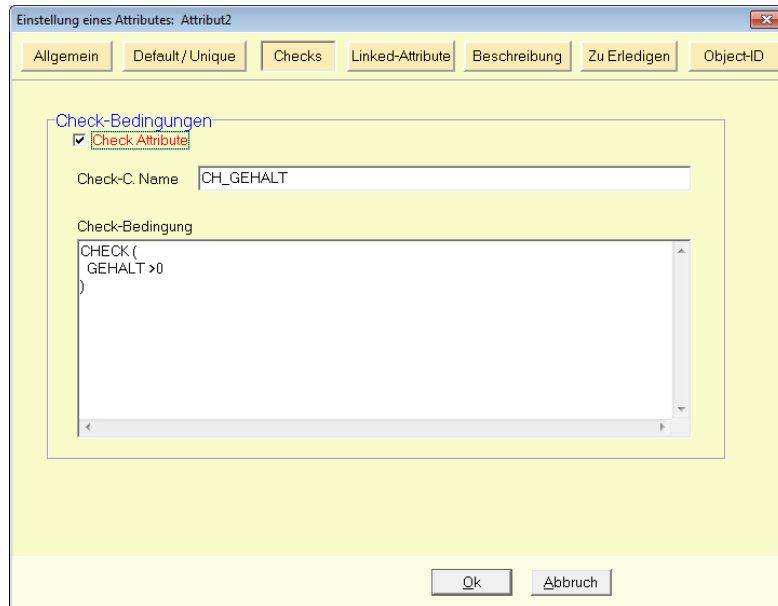


Abbildung 13 Attribut Gehalt mit einer Prüfbedingung

Es werden nur Gehälter größer Null eingetragen. Im Fehlerfall gibt es eine Fehlermeldung seitens des Datenbanksystems.

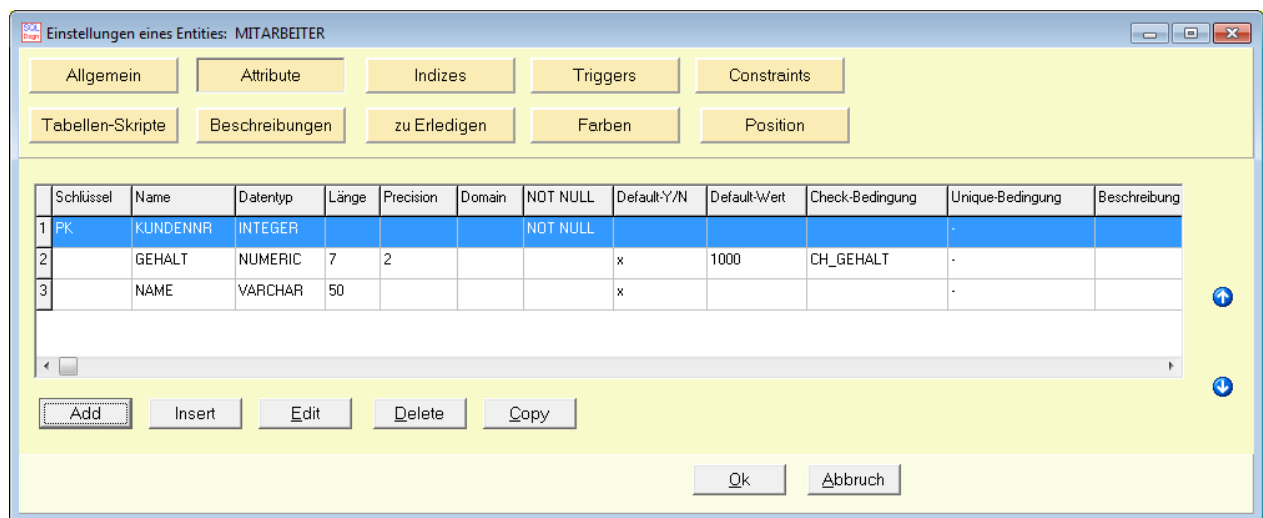


Abbildung 14 Attribute eines Kunden

2.5 Neue Beziehung

Hinweis: Man benötigt mindestens zwei Entities

Taste: Strg+R

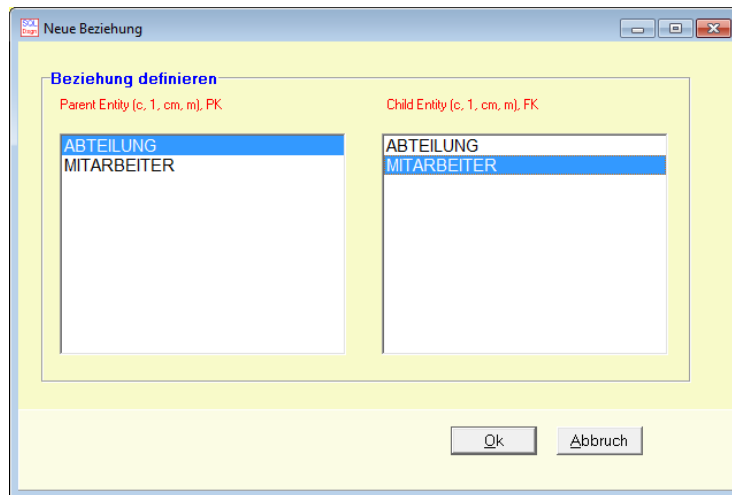


Abbildung 15 Neue Beziehung, rechts ist das abhängige Entity

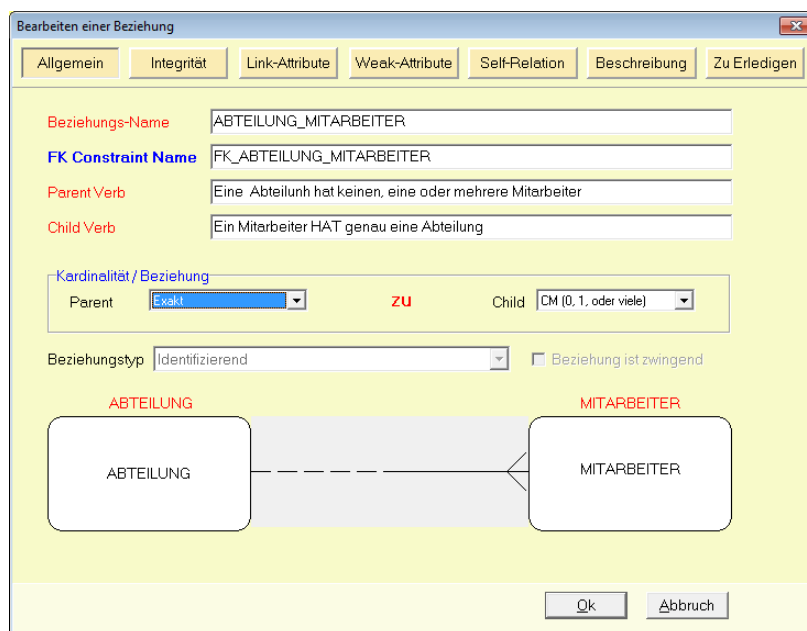


Abbildung 16 Eintragen der Texte

Nun den Schalter „link-Attribute betätigen“

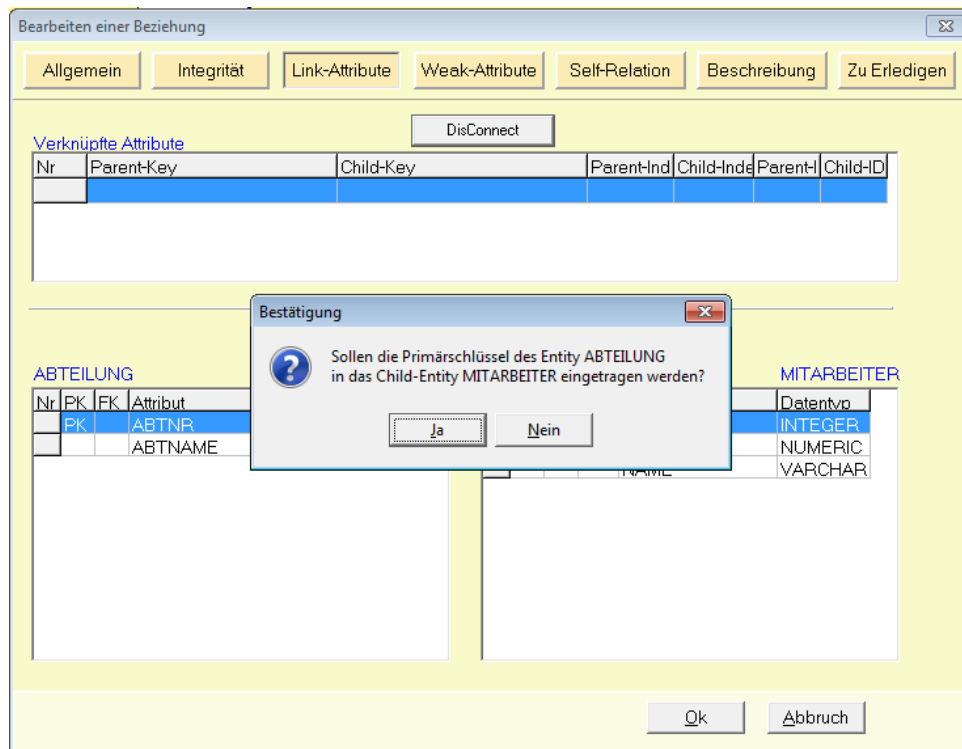


Abbildung 17 Attribut "Foreign-Key" eintragen

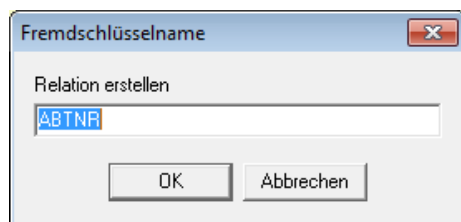


Abbildung 18 Name des "Foreign-Key" eintragen

Bearbeiten einer Beziehung

Allgemein Integrität Link-Attribute Weak-Attribute Self-Relation Beschreibung Zu Erledigen

DisConnect

Nr	Parent-Key	Child-Key	Parent-Ind	Child-Inde	Parent-I	Child-ID
1	ABTNR	ABTNR	0	3	18	42

Connect

ABTEILUNG

Nr	PK	FK	Attribut	Datentyp
1	PK		ABTNR	INTEGER
			ABTNAME	VARCHAR

MITARBEITER

Nr	PK	FK	Neu	Attribut	Datentyp
	PK			KUNDENNR	INTEGER
				GEHALT	NUMERIC
				NAME	VARCHAR
1		FK	Neu	ABTNR	INTEGER

Ok Abbruch

Abbildung 19 Beziehungsdialog mit allen Daten

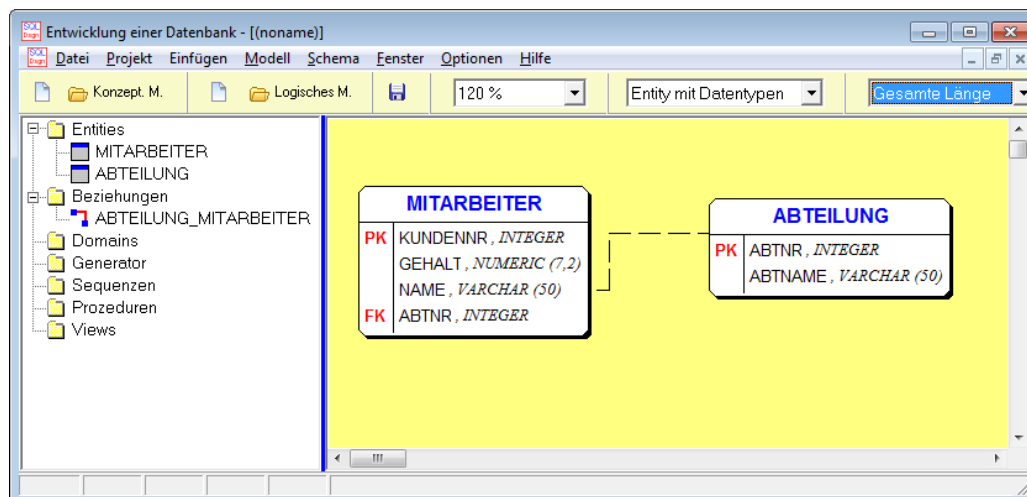


Abbildung 20 Beziehung im ERM-Abbildung

2.6 Erstellen der Skripte

Taste F9:



Abbildung 21 Dialog zum Erzeugen der SQL-Skripte

Bitte immer **ohne Kommentare** die SQL-Befehle erzeugen. Der .net Provider mag keine Kommentare, IBOConsole kann die Kommentare verarbeiten.

```
CREATE TABLE MITARBEITER (
    KUNDENNR INTEGER NOT NULL,
    GEHALT NUMERIC(7,2) DEFAULT 1000,
    NAME VARCHAR(50) DEFAULT '',
    ABTNR INTEGER NOT NULL,

    CONSTRAINT PK_MITARBEITER PRIMARY KEY (KUNDENNR)
);
```

```
CREATE TABLE ABTEILUNG (
    ABTNR INTEGER NOT NULL,
    ABTNAME VARCHAR(50) NOT NULL,
```

```
CONSTRAINT PK_ABTEILUNG PRIMARY KEY (ABTNR)
);
```

```
ALTER TABLE MITARBEITER ADD CONSTRAINT
    FK_ABTEILUNG_MITARBEITER FOREIGN KEY (ABTNR) REFERENCES
    ABTEILUNG(ABTNR);
```

```
ALTER TABLE MITARBEITER
    ADD CONSTRAINT CH_GEHALT CHECK (
        GEHALT >0
    );
```

2.7 Sequenz

2.7.1 Generator oder Sequenz definieren

Einige Datenbanken verwenden Generatoren, andere Sequenzen. Access wiederum benutzt einen Autoincrement-Datentyp. Ziel einer Sequenz ist die automatische Erzeugung eindeutiger Schlüssel. Um eine Sequenz zu erzeugen, ruft man das Menü „Einfügen“ mit dem Eintrag „Sequenz“ auf.

2.7.2 Sequenzbefehle mit SQL:

Erzeugen einer Sequenz mittels SQL:

```
CREATE SEQUENCE seq;
alter sequence seq restart with 0
```

Abfrage des aktuellen Wertes:

```
select gen_id( seq, 0 ) FROM RDB$DATABASE;
```

Anlegen einer Abteilung mit einer Sequenz:

```
INSERT INTO dept (deptno, Name, location, budget)
VALUES (GEN_ID(seq,1), 'Finanzen', 'Wernigerode', 11000 );
```

3 FBConsole

Dieses Kapitel beschreibt das Erstellen, Füllen und Abfragen einer neuen Datenbank

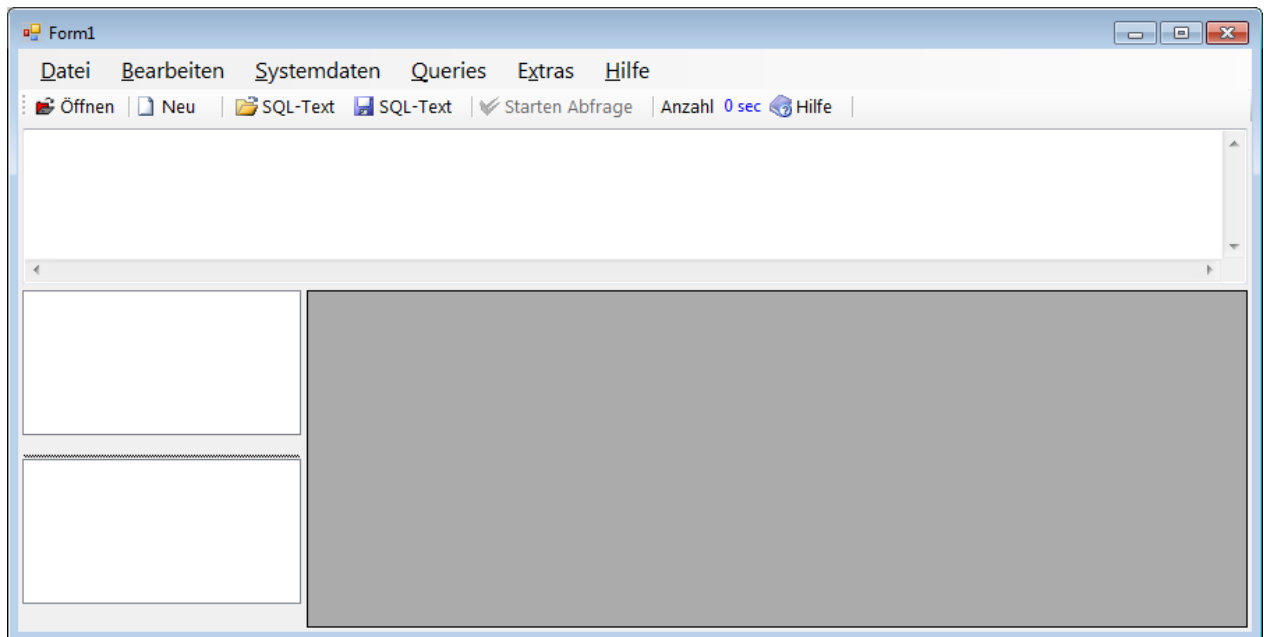


Abbildung 22 Hauptfenster der FBConsole

Schalter:

- | | |
|-------------------|---|
| • Öffnen | Öffnen einer vorhandenen Datenbank. |
| • Neu | Erstellen einer neuen Datenbank. |
| • SQL-Text | Öffnen einer vorhandenen Textdatei mit einer SQL- |
| Anweisung. | |
| • SQL-Text | Speichern der aktuellen SQL-Anweisung in eine |
| Textdatei. | |
| • Starten Abfrage | Ausführen der Anweisung im Texteditor |

Menü Datei:

- | | |
|-------------------------------|---|
| • Öffnen einer Datenbank | Öffnen einer vorhandenen Datenbank. |
| • Neue Datenbank | Erstellen einer neuen Datenbank. |
| • Öffnen einer SQL-Datei | Öffnen einer vorhandenen Textdatei mit einer SQL- |
| Anweisung. | |
| • Speichern eines SQL-Befehls | Speichern der aktuellen SQL-Anweisung in eine |
| Textdatei. | |

Menü Queries:

- | | |
|----------------------------|--|
| • Ausführen der Abfrage | Ausführen der Anweisung im Texteditor. |
| • Export der Ergebnisse | Speichern des Ergebnisses der Abfrage in die |
| Zwischenablage. | |
| • Speichern der Ergebnisse | Speichern des Ergebnisses der Abfrage in eine Textdatei. |
| • | |

3.1 Neue Datenbank erstellen

Ablauf:

- Erstellen der Skripte mittels eines Datenbank-Designers
- Erstellen einer neuen Datenbank: STRG+N

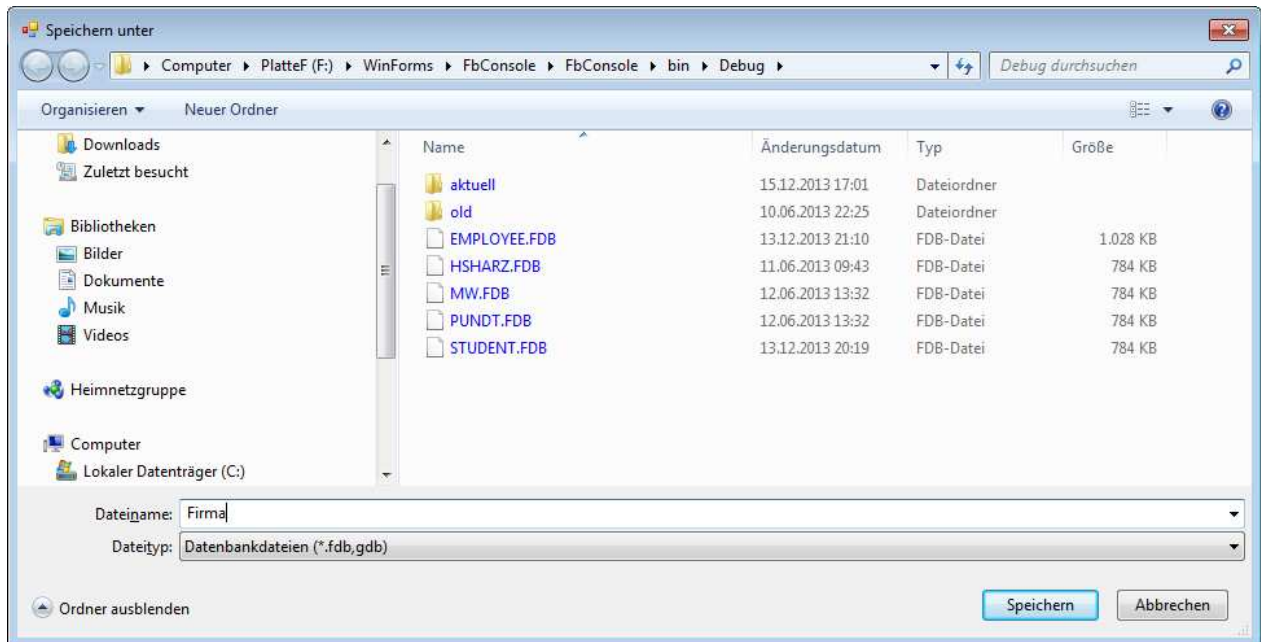
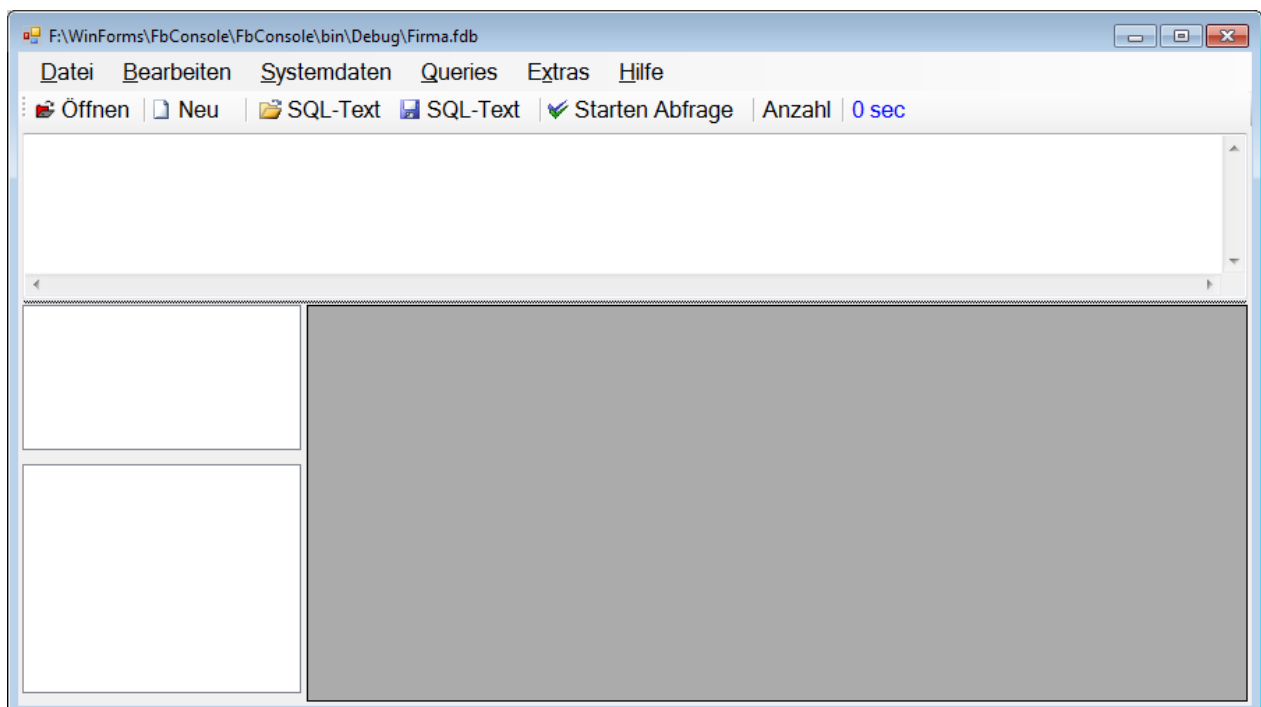


Abbildung 23 Erstellen der Datenbank "Firma.fdb"

Ergebnis:



Einfügen der Skripte in den Editor:

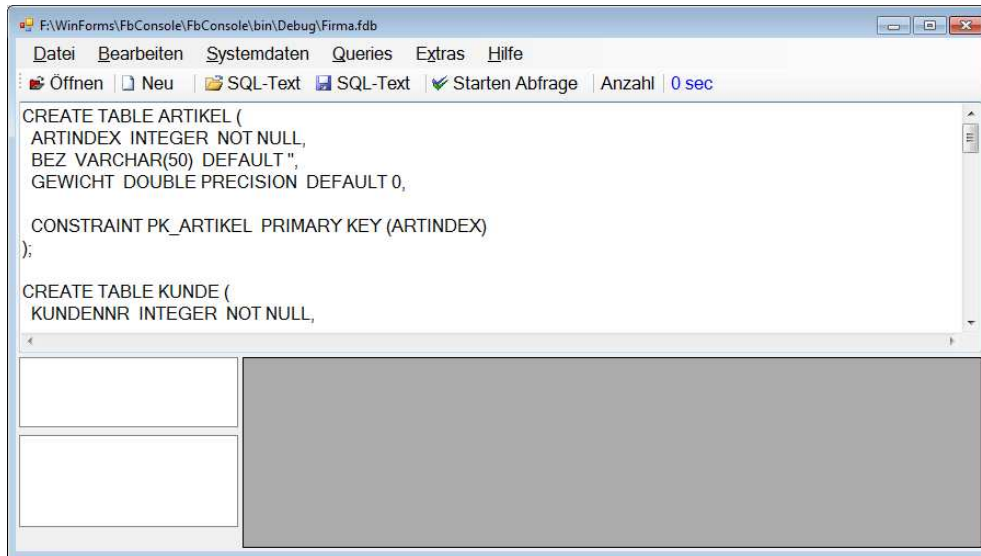


Abbildung 24 Eintragen der SQL-Skripte

Schalter „Starten Abfrage“

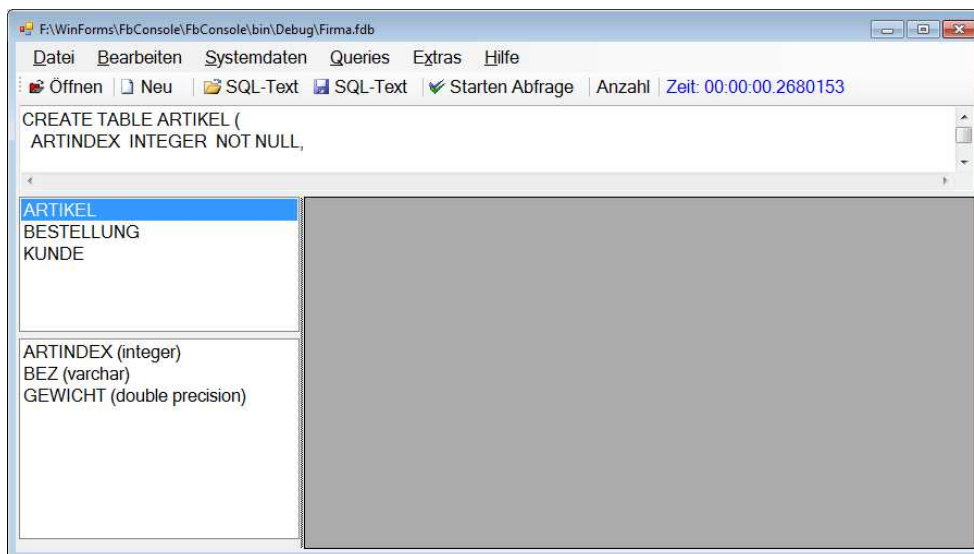


Abbildung 25 Anzeige der Datenbank-Tabellen

In der oberen Liste sind alle Entities eingetragen.

In der unteren Liste sind alle Spalten des aktuellen Entities eingetragen.

3.2 Daten eintragen

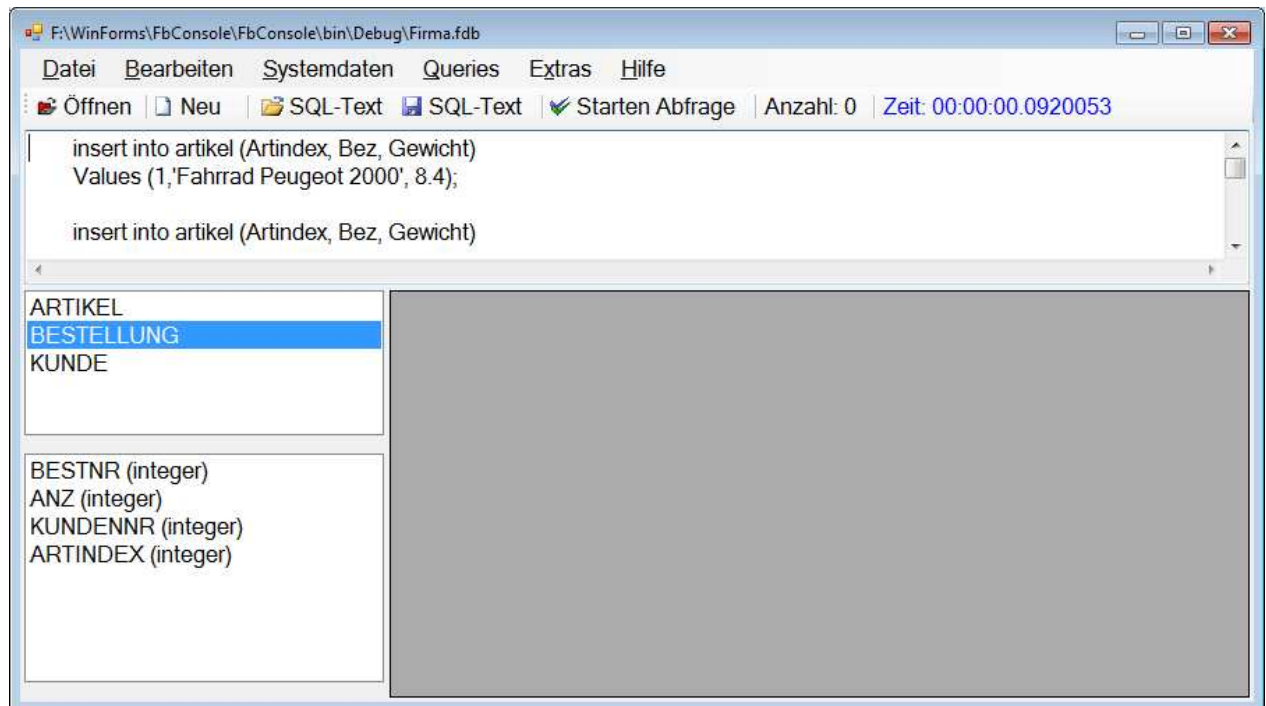


Abbildung 26 Daten mittels „Insert-Into“ eintragen

3.2.1 SQL-Befehle zum Eintragen:

```
insert into kunde (kundennr, name, vorname, strasse, plz, ort)
Values (101,'Müller', 'Hans', 'Langer Weg 6', '01256',
'Weißenfels');
```

```
insert into kunde (kundennr, name, vorname, strasse, plz, ort)
Values (102,'Schmidt', 'Andrea', 'Kastanienallee 42', '39114',
'Magdeburg');
```

```
insert into kunde (kundennr, name, vorname, strasse, plz, ort)
Values (103,'Schmidt', 'Andrea', 'Kastanienallee 42', '39114',
'Magdeburg');
```

```
insert into kunde (kundennr, name, vorname, strasse, plz, ort)
Values (104,'Meier', 'Antonia', 'Friedrichstraße 58', '38855',
'Wernigerode');
```

```
insert into artikel (Artindex, Bez, Gewicht)
Values (1,'Fahrrad Peugeot 2000', 8.4);
```

```
insert into artikel (Artindex, Bez, Gewicht)
Values (2,'Tablet Omega', 0.6);
```

```
insert into artikel (Artindex, Bez, Gewicht)
Values (3,'Desktop HP 1200', 7.5);
```



```

insert into artikel (Artindex, Bez, Gewicht)
Values (4, 'Monitor ASUS 1800', 2.6);

insert into bestellung (bestnr, anz, kundenr, artindex)
Values (201, 2, 101, 1);

insert into bestellung (bestnr, anz, kundenr, artindex)
Values (202, 3, 104, 2);

insert into bestellung (bestnr, anz, kundenr, artindex)
Values (203, 1, 102, 4);

```

3.3 Testabfragen

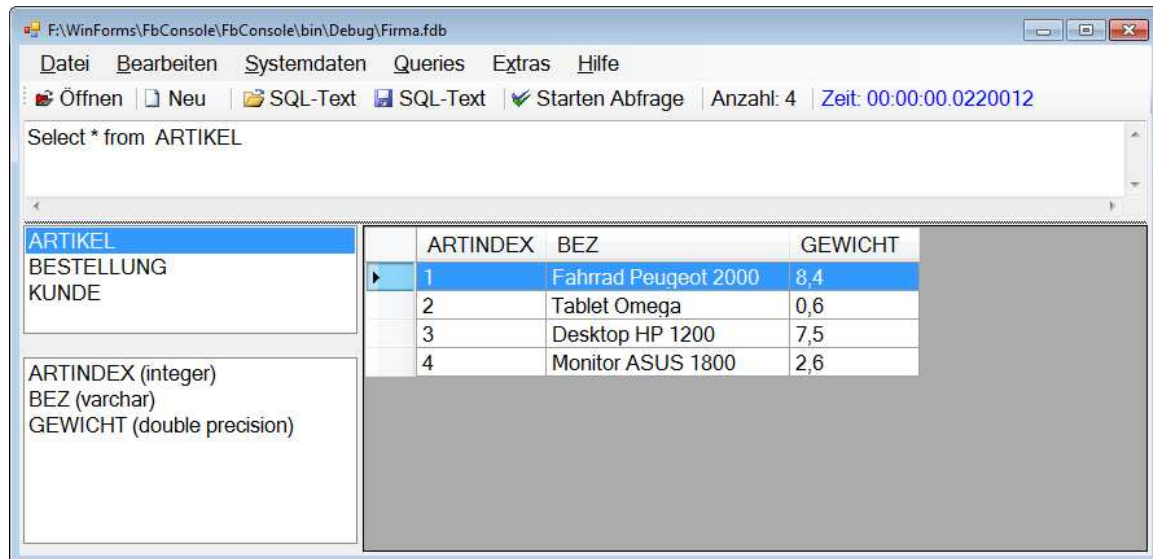


Abbildung 27 Abfrage bezüglich der Artikel

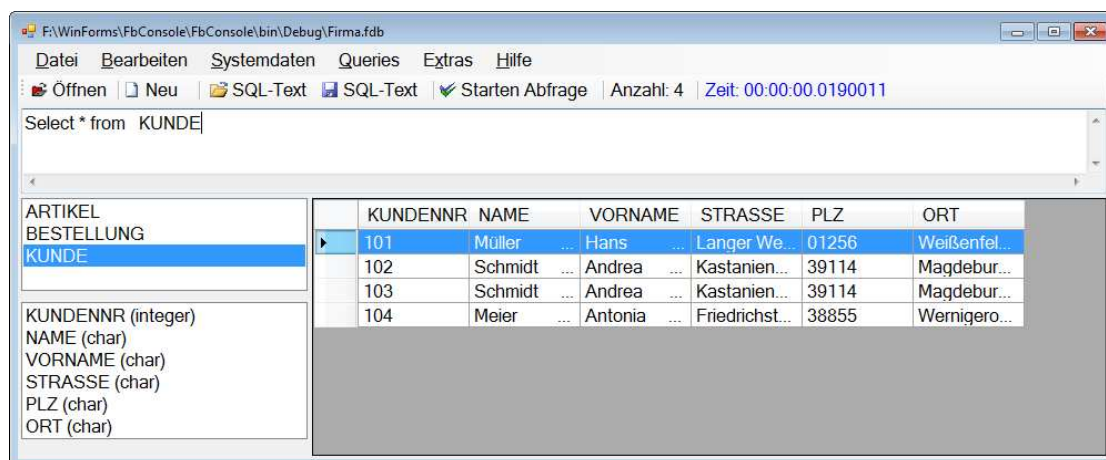


Abbildung 28 Abfrage bezüglich der Kunden

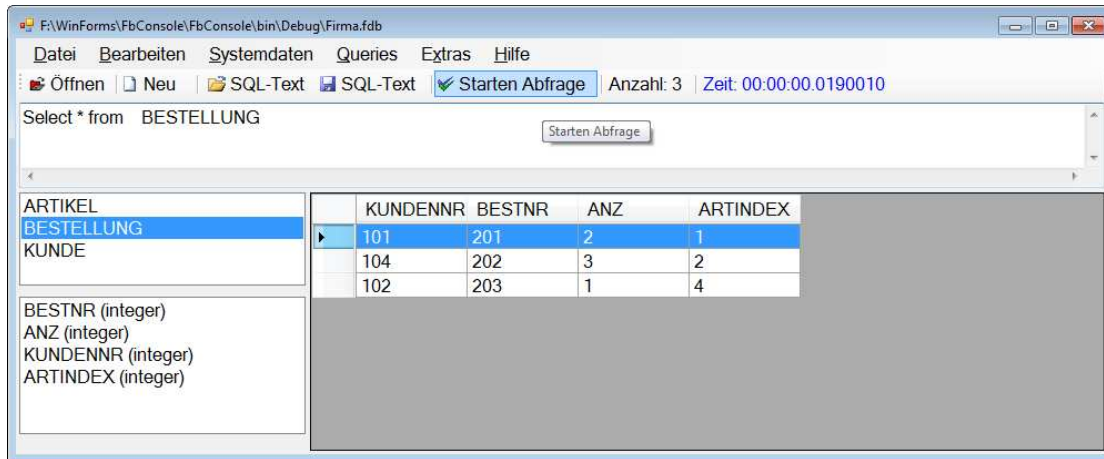


Abbildung 29 Abfrage bezüglich der Bestellungen

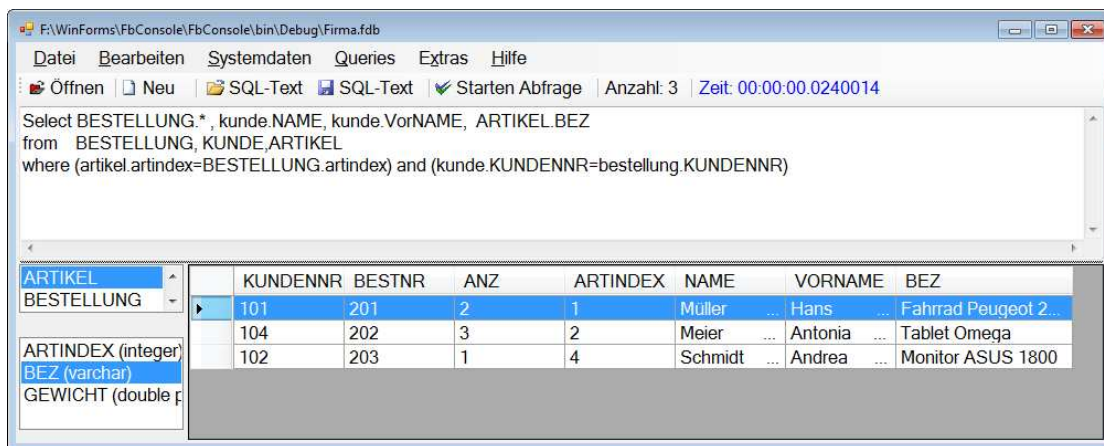


Abbildung 30 Abfrage bezüglich der Bestellungen

```

Select BESTELLUNG.* , kunde.NAME, kunde.VorNAME,  ARTIKEL.BEZ
from      BESTELLUNG, KUNDE,ARTIKEL
where (artikel.artindex=BESTELLUNG.artindex) and
      (kunde.KUNDENNR=bestellung.KUNDENNR)

```

4 SQL-Anweisungen

4.1 Aufbau der SQL-Sprache

Datenbanken werden über standardisierte Sprache angesprochen. Diese Sprache ist leicht zu erlernen und ist weniger kompliziert als eine Programmiersprache. SQL ist in folgende Teile aufgebaut:

- **Datenabfrage**
 - SELECT
- **Data Manipulation Language- DML**
 - INSERT Einfügen
 - UPDATE Ändern
 - DELETE Löschen
 - MERGE Zusammenfügen
- **Data Definition Language - DDL**
 - CREATE Erzeugen einer Tabelle
 - ALTER Ändern einer Tabelle
 - DROP Löschen einer Tabelle
 - RENAME Umbenennen einer Tabelle
 - TRUNCATE Abschneiden einer Tabelle
- **Transaktionssteuerung**
 - COMMIT Änderungen speichern
 - ROLLBACK Änderungen rücknehmen
 - SAVEPOINT Sicherheitspunkt
- **Data Control Language - DCL**
 - GRANT Rechte erlauben
 - REVOKE Rechte verbieten

4.2 SELECT-Anweisung

Anzeige aller Attribute

```
SELECT *
FROM emp;
```

Anzeige aller Mitarbeiter, deren Mitarbeiternummer größer 1000 ist

```
SELECT * FROM emp where empno>1000;
```

Anzeige aller Attribute, geordnet nach dem "full_name"

```
SELECT *
FROM emp;
ORDER BY full_name
```

Anzeige aller Attribute, absteigend geordnet nach dem "full_name"

```
SELECT *  
FROM emp;  
ORDER BY full_name DESC
```

Anzeige aller Attribute, aufsteigend geordnet nach dem "full_name"

```
SELECT *  
FROM emp;  
ORDER BY full_name ASC
```

Anzeige aller Attribute, aufsteigend geordnet nach dem "last_name" und "first_name"

```
SELECT *  
FROM emp;  
ORDER BY last_name ASC, first_name ASC
```

Problem: Müller, Hans Müller, Sabine

4.3 DISTINCT

```
SELECT distinct deptno  
FROM employee
```

```
SELECT distinct empno  
FROM employee
```

Unterschied??

4.4 BETWEEN

Die Syntax lautet dabei:

BETWEEN untereGrenze AND obereGrenze

Beispiel:

Suche alle Mitarbeiter, deren Gehalt zwischen 40.000,00 und 60.000,00 liegt.

```
SELECT full_name, salary  
FROM employee  
WHERE salary BETWEEN 40000.00 AND 60000.00
```

4.5 LIKE

- Verwenden Sie den LIKE-Operator, um eine Platzhaltersuche nach gültigen Zeichenfolgenwerten durchzuführen.
- Die Suchkriterien können entweder literale Zeichen oder Zahlen enthalten.

Suchzeichen:

- % steht für kein, ein, oder beliebige Zeichen (entspricht dem *)
- _ steht für genau ein Zeichen (entspricht dem ?)

Beispiel:

Gesucht sind alle Mitarbeiter, deren Namen mit dem Buchstaben „B“ anfängt.

```
Select Full_name, emp_no, salary
From employee
Where last_name LIKE "B%";
```

Gesucht sind alle Mitarbeiter, die irgendwo ein m haben.

```
Select Full_name, emp_no, salary
From employee
Where last_name LIKE "%m%";
```

4.6 Join

Ein Join ist ein kartesisches Produkt zweier Tabellen (A und B). Jeder Datensatz der Tabelle A wird mit jedem Datensatz der Tabelle B verknüpft. Damit ist das Ergebnis unter Umständen eine sehr Tabelle. Mit Hilfe der WHERE-Klausel kann die Datenmenge eingeschränkt werden.

Kreuzprodukt der beiden Tabellen

```
SELECT *
FROM employee, department
```

Kreuzprodukt der beiden Tabellen mit der korrekten Einschränkung

```
SELECT *
FROM employee, department
WHERE employee.dept_no = department.dept_no
```

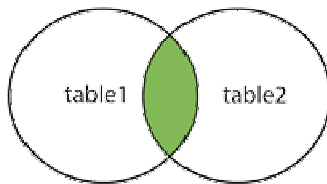
Typen der Joins:

- INNER JOIN
 - Ausgabe aller Zeilen, wenn beide Abfragen (links,rechts) wahr sind
- LEFT JOIN:
 - Ausgabe aller Zeilen, wenn die linke Abfragen wahr sind.
- RIGHT JOIN:
 - Ausgabe aller Zeilen, wenn die rechte Abfragen wahr sind.
- FULL JOIN:
 - Ausgabe aller Zeilen, wenn eine Abfragen wahr ist.

4.6.1 Inner-Join

```
SELECT *  
FROM employee  
INNER JOIN department  
ON employee.dept_no = department.dept_no;
```

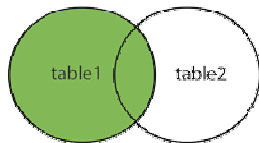
Ausgabe aller Mitarbeiter, die einer Abteilung zugeordnet sind.



4.6.2 Left-Join

```
SELECT *  
FROM employee  
LEFT JOIN department  
ON employee.dept_no = department.dept_no;
```

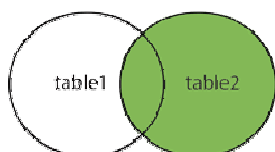
Ausgabe aller Mitarbeiter, die einer Abteilung zugeordnet sind und gleichzeitig die Mitarbeiter, die keine Abteilung haben. Wird im Allgemeinen durch eine Beziehung verhindert.



4.6.3 Right-Join

```
SELECT *  
FROM employee  
RIGHT JOIN department  
ON employee.dept_no = department.dept_no;
```

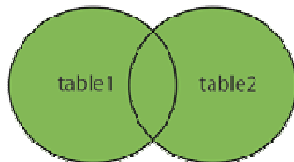
Ausgabe aller Mitarbeiter, die einer Abteilung zugeordnet sind und gleichzeitig die Abteilungen, die keine Mitarbeiter haben.



4.6.4 Outer-Join

```
SELECT *
FROM employee
FULL OUTER JOIN department
ON employee.dept_no = department.dept_no;
```

Ausgabe aller Mitarbeiter, die einer Abteilung zugeordnet sind und gleichzeitig die Mitarbeiter, die keine Abteilung haben und die Abteilungen, die keine Mitarbeiter haben.



4.7 SubSelect, SubQuery

Eine SubSelect-Anweisung verwendet mindestens zwei Abfragen, die ineinander verschachtelt sind. Es ist eine ähnliche Technik wie die Join-Anweisung.

Ausgabe der Namen und der Abteilungsnummer:

```
SELECT emp_no, first_name, last_name, employee.dept_no
FROM employee
```

Um den Abteilungsnamen mit auszugeben kann man folgende SQL-Anweisung eingeben:

```
SELECT emp_no, first_name, last_name,
       employee.dept_no, department
FROM employee, department
WHERE employee.dept_no = department.dept_no
```

Das gleiche Resultat erhält man mit folgender Anweisung:

```
SELECT emp_no, first_name, last_name, employee.dept_no,
       (
         SELECT department
         FROM department
         WHERE employee.dept_no = department.dept_no
       ) Abteilung
FROM employee
```

Im Prinzip ist es auch eine Join-Anweisung, nur ist diese einfacher zu verstehen. Welche schneller ist, hängt von der Implementierung ab.

Sinnvoller sind die SubSelect-Anweisungen mit Aggregat-Anweisungen:

```
SELECT emp_no, first_name, last_name,
(
    select sum(total_value)
    from sales
    where sales.sales_rep=employee.emp_no
) Anz_Artikel
from employee
order by Anz_Artikel asc
```

EMP_NO	FIRST_NAM	LAST_NAME	ANZ_ARTIKEL
141	Pierre	Osborne	1980,72
118	Takashi	Yamamoto	24190,4
61	Luke	Leung	37475,69
121	Roberto	Ferrari	122693
11	K. J.	Weston	139450,5
134	Jacques	Glon	462600,49
127	Michael	Yanowski	502192,23
72	Claudia	Sutherland	960008
12	Terri	Lee	
113	Mary	Page	
107	Kevin	Cook	
14	Stewart	Hall	
138	T.J.	Green	

Abbildung 31 Subselect-Beispiel

Hier gibt es also auch Mitarbeiter, die keine Verkäufe haben (Abteilungen Finanzen etc.). Das gleiche Ergebnis erhält man auch mit einem Left-Join.

Um nur die „Verkäufer“ zu selektieren kann man diese Befehle benutzen:

```
SELECT emp_no, first_name, last_name,
(
    select sum(total_value)
    from sales
    where sales.sales_rep=employee.emp_no
) Anz_Artikel
from employee
where (
    select sum(total_value)
    from sales
    where sales.sales_rep=employee.emp_no
) >0
order by Anz_Artikel asc
```

Diese Version ist langsamer, warum?

4.8 Gruppenfunktionen

Was sind Gruppenfunktionen?

Gruppenfunktionen werden auf Gruppen von Zeilen angewendet und geben ein Ergebnis pro Gruppe zurück.

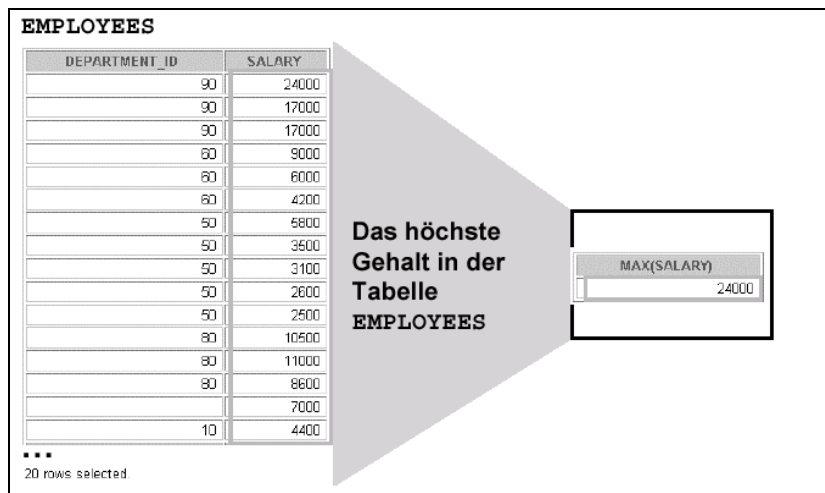


Abbildung 32 Gruppenfunktionen mit der Tabelle Employee

Die Gruppenfunktionen arbeiten mit Gruppen von Zeilen und geben ein Ergebnis pro Gruppe zurück. Bei diesen Gruppen kann es sich um ganze die Tabelle oder einzelne Gruppen der Tabelle handeln.

Funktion	Beschreibung
AVG([DISTINCT ALL] n)	Berechnet den Durchschnittswert von n-Zahlen. Eventuelle NULL-Werte werden ignoriert.
COUNT({ * [DISTINCT ALL] Ausdruck)	Anzahl der Zeilen, für die Ausdruck auf einen anderen Wert als einen NULL-Wert ausgewertet wird. Alle mit * ausgewählten Zeilen zählen, einschließlich mehrfach vorhandener Zeilen und Zeilen mit NULL-Werten.
MAX([DISTINCT ALL] n)	Größter Wert von n Werten. NULL-Werte werden ignoriert.
MIN([DISTINCT ALL] n)	Kleinsten Wert von n Werten. NULL-Werte werden ignoriert.
STDDEV [DISTINCT ALL] n)	Standardabweichung von n Werten. NULL-Werte werden ignoriert.
SUM([DISTINCT ALL] n)	Summe von n Werten. NULL-Werte werden ignoriert.
VARIANCE([DISTINCT ALL] n)	Varianz von n Werten. NULL-Werte werden ignoriert.

4.8.1 Syntax der GROUP BY-Klausel

```
SELECT      column, group_function(column)
FROM        table
[ WHERE Condition ]
[ GROUP BY group_by_expression ]
[ ORDER BY column ]
```

4.8.2 Richtlinien für die Verwendung von Gruppenfunktionen

- DISTINCT bewirkt, dass die Funktion keine doppelten Werte berücksichtigt.
- ALL bewirkt, dass alle Werte, einschließlich der doppelten Werte, berücksichtigt werden. Der Defaultwert ist ALL und muss daher nicht angegeben werden.
- Die Datentypen für Funktionen mit einem expr-Argument können CHAR-VARCHAR2, NUMERIC oder DATE sein.
- Alle Gruppenfunktionen ignorieren NULL-Werte. Um NULL-Werte durch einen Wert zu ersetzen, verwendet man die Funktionen NVL, NVL2, oder COALESCE.
- Wenn man eine GROUP BY-Klausel verwendet, sortiert der Datenbank-Server die Ergebnismenge implizit in aufsteigender Reihenfolge. Um diese Default-Sortierung außer Kraft zu setzen, verwenden Sie DESC in einer ORDER BY-Klausel.
- Wenn Sie eine Gruppenfunktion in einer SELECT-Klausel angeben, können Sie nicht gleichzeitig einzelne Ergebnisse auswählen, es sei denn, die einzelne Spalte wird in der GROUP BY-Klausel angegeben. Man erhält eine Fehlermeldung, wenn die Spaltenliste nicht in der GROUP BY-Klausel angegeben wurde.
- Mit einer WHERE-Klausel kann man Zeilen ausschließen, bevor die übrigen Zeilen in Gruppen aufgeteilt werden.
- Man muss die Spalten in der GROUP BY-Klausel angeben.
- Man kann dabei keine Spalten-Aliasnamen verwenden.
- Standardmäßig werden die Zeilen in aufsteigender Reihenfolge nach der in der GROUP BY-Liste angegebenen Spalten sortiert. Diese Sortierung kann man mit der ORDER BY-Klausel verändern.
- Bei Verwendung der GROUP BY-Klausel, muss man sicher stellen, dass alle Spalten in der SELECT-Liste, die nicht in Gruppenfunktionen (z. B. AVG) enthalten sind, in der GROUP BY-Klausel angegeben werden.

4.8.3 SUM-Funktion

Die SUM-Funktion berechnet die Summe eines oder mehrerer Attribute.

- Gesucht ist die Summe aller Gehälter

```
SELECT sum(salary)
FROM employee
```

Ergebnis: 115522468

Das nächste Beispiel zeigt die Benutzung mehrerer Attribute:

- Gesucht sind die Summen aller Rechnungen, Rabatte und deren Differenz.

```
SELECT  SUM(total_value), SUM (total_value*DISCOUNT)
        SUM (total_value- total_value*DISCOUNT),
FROM sales
```

Die Summe berechnet die Summe aller Rechnungen. In der zweiten Summe wird der Gesamtrabatt berechnet, der in der dritten von der ersten Summe abgezogen wird.

SUM	SUM_1	SUM_2
2250591,03	182994,9654	2067596,065

4.8.4 AVG-Funktion

Die AVG-Funktion berechnet den Durchschnitt von Werten.

- Gesucht ist der durchschnittliche Verkaufsbetrag aller Mitarbeiter

```
SELECT AVG(total_value)
FROM sales
```

liefert den durchschnittlichen Verkaufsbetrag: 68199,73 €

- Gesucht ist der durchschnittliche Verkaufsbetrag aller Mitarbeiter **pro Abteilung**

```
SELECT dept_no, AVG(salary)
FROM employee
GROUP BY dept_no
```

Gesucht ist der durchschnittliche Verkaufsbetrag aller Mitarbeiter **einer Abteilung**

```
SELECT  AVG(salary)
FROM employee
WHERE dept_no=623
```

liefert das durchschnittliche Gehalt in der Abteilung 623: 57551,65

4.8.5 MIN-Funktion

Die MIN-Funktion sucht das Minimum einer Spalte. Hier können natürlich auch Bedingungen angegeben werden.

- Gesucht ist der minimale offene Rechnungsbetrag

```
SELECT  MIN(total_value)
FROM sales
WHERE order_status="open"
```

4.8.6 MAX-Funktion

Die MAX-Funktion sucht das Maximum einer Spalte. Hier können natürlich auch Bedingungen angegeben werden.

- Gesucht ist der maximale offene Rechnungsbetrag

```
SELECT MAX(total_value)
FROM sales
WHERE order_status="open"
```

liefert den Wert 450000,49 (siehe Tabelle oben)

4.8.7 COUNT-Funktion

Die COUNT-Funktion zählt die Tupel in einer Tabelle. Hier können natürlich auch Bedingungen angegeben werden, die die Anzahl einschränken.

Hinweis:

- COUNT(deptno) liefert die Anzahl aller Abteilung, die nicht NULL sind
- COUNT(*) liefert die Anzahl aller Abteilung, die nicht NULL sind
- Bestimme die Anzahl aller Mitarbeiter

```
SELECT COUNT(emp_no)
FROM employee
```

liefert den Wert 42

- Bestimme die Anzahl aller Mitarbeiter in den Abteilungen mit der Abteilungsnummer größer 600

```
SELECT COUNT(emp_no)
FROM employee
WHERE dept_no > 600
```

liefert den Wert 21

- Bestimme die Anzahl aller Manager

```
SELECT COUNT(emp_no)
FROM employee
WHERE job_code = "Mngr"
```

liefert den Wert 4

oder auch

```
SELECT COUNT(emp_no)
FROM employee
WHERE lower(job_code) = "mngr"
```

Oder auch:

```
SELECT count(*)
FROM employee
WHERE lower(job_code) = "mngr"
```

Hier wird der Datenwert in Kleinbuchstaben umgewandelt.

4.8.8 Group by having

- Mit der where-Anweisung können Daten VOR dem Gruppierung gefiltert werden.
- Mit der having-Anweisung können Gruppenergebnisse nach der Gruppierung entfernt werden.

- Gesucht ist der durchschnittliche Verkaufsbetrag aller Mitarbeiter **pro Abteilung**

```
SELECT dept_no, AVG(salary)
FROM employee
GROUP BY dept_no
```

- Gesucht ist der durchschnittliche Verkaufsbetrag aller Mitarbeiter pro Abteilung, die **mindestens eine Million** Umsatz haben.

```
SELECT dept_no, AVG(salary)
FROM employee
GROUP BY dept_no
Having AVG(salary)>1000000
```

4.9 Codierungsfehler

Die folgende SELECT-Anweisung enthält vier Fehler. Können Sie alle vier finden?

```
SELECT emp_no, last_name
salary x 12    ANNUAL SALARAY
from employee
```

4.10 Die INSERT INTO Anweisung

```
INSERT INTO    table [(column [, column ...])]  
VALUES        (value [, value ...]);
```

Beispiele:

```
INSERT INTO KUNDE( empno, name )  
VALUES (123, 'Müller' );
```

```
INSERT INTO KUNDE( name, empno )  
VALUES ( 'Schmidt', 125 );
```

4.11 Die UPDATE Anweisung

Mit Hilfe der UPDATE-Anweisung können vorhandene Zeilen bearbeiten werden.

```
UPDATE    table  
SET       column = value [, column = value ...]  
[WHERE    condition(s)];
```

Abbildung 33.: Die UPDATE Anweisung

Beispiel:

```
UPDATE KUNDE  
SET SALDO=61615, KREDIT=81674  
WHERE ( KUNDENNR = 1 );
```

4.12 Die DELETE FROM Anweisung

Mit Hilfe der DELETE-Anweisung können vorhandene Zeilen gelöscht werden.

```
DELETE [FROM] table  
[WHERE    condition(s)];
```

Beispiele:

Löschen aller Mitarbeiter der Abteilung 110;

```
DELETE FROM EMPLOYEE;  
WHERE deptno=110
```

Löschen der gesamten Tabelle:

```
DELETE FROM EMPLOYEE;
```

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

5 DDL-Befehle

5.1 Die CREATE TABLE Anweisung

```
CREATE [GLOBAL TEMPORARY] TABLE [schema.] table
    (column datatype [DEFAULT expr], ...);
```

Beispiel für Tabelle EMP:

```
create table emp
( empno      numeric(4),
  ename      varchar2(10),
  job        varchar2(9),
  mgr        numeric(4),
  hiredate   date      default sysdate,
  sal        numeric(7,2),
  comm       numeric(7,2),
  deptno     numeric(2),
  constraint emp_pk primary key (empno),
  constraint emp_fk_emp foreign key (mgr) references emp(empno),
  constraint emp_fk_dept foreign key (deptno) references dept(deptno));
```

5.2 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

5.2.1 Syntax von Computed by

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

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

  CONSTRAINT PK_ANR PRIMARY KEY (ANR)
```

6 Literatur und Links

<http://www.w3schools.com/sql/>

- [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

);

7 Indexverzeichnis

A

Aufbau der SQL-Sprache	27
------------------------------	----

B

Beispiel1	
Generator	20
Sequenz	20
Between	28
Beziehung	15

C

Codierungsfehler	37
------------------------	----

D

Datenbank-Designer	6
DELETE	38
DISTINCT	28

E

Eigenschaften in Kurzform	6
---------------------------------	---

F

FBConsole	
Daten eintragen	24
Insert into	24
Neue Datenbank	22
Testabfragen	25
Funktionen	
avg	35
count	36
max	36
min	35
sum	34

G

Gruppenfunktionen	33
-------------------------	----

I

Inhalt der ZIP-Datei	5
INSERT INTO	38

J

Join	29
------------	----

K

Konzeptionelles Modell	9
------------------------------	---

L

LIKE	28
Logisches Modell	12

M

Menü Datei.....	7
Menü Einfügen.....	8
Menü Modell.....	8
Menü Projekt.....	8
Menü Schema.....	8
Menüfunktionen	7

N

Neue Datenbank	22
Neues Entity im konzeptionellen Modell	9, 13

R

Relation	15
----------------	----

S

SELECT-Anweisung.....	27
Sequenz	20
Spezielle Funktionen	
Computed By	39
SQL	27
SubQuery	31
SubSelect.....	31

U

UPDATE.....	38
-------------	----