Technische Universität München
Fakultät für Informatik
Prof. R. Bayer, Ph.D.
Dipl.-Inform. Michael Bauer
Dr. Gabi Höfling

WS 2003/04,
Datenbanksysteme I
Lösungsblatt 6

1.12.2003

SQL-DDL und SQL-Anfragen

Aufgabe 1:

1.) Legen Sie die Tabelle Kategorie an. Die Kategoriebezeichnung soll ein String mit wechselnder Länge, maximal jedoch 15 Zeichen sein, das Klassifikationskriterium ein String mit wechselnder Länge (max. 100 Zeichen). Weiterhin soll die Tabelle festlegen, dass das Kategoriekriterium eingegeben werden muss und die Kategoriebezeichnung der Primärschlüssel ist. Fügen Sie dann folgende Tupel in die Tabelle ein: (´U-18´, ´Alter unter 18 Jahren´),(´G-Kunde´, ´Geschäftskunde´), (´U-18´, ´Alter unter 18 Jahren´)

CREATE TABLE Kategorie (Bezeichnung VARCHAR(15) NOT NULL PRIMARY KEY, Klassifikationskriterium VARCHAR(100) NOT NULL)

Create Table legt eine neue Tabelle an, die Spalten werden in Klammern dahintergeschrieben. Für jede Spalte werden der Spaltenname, der Typ und eventuell weitere Informationen angegeben. In unserem Fall wird festgelegt, dass die Spalten Werte enthalten müssen und der (Primär-)schlüssel für diese Tabelle entsprechend gesetzt wird.

Als Konvention wird im weiteren Lösungsblatt Großschreibung für alle Schlüsselwörter von SQL verwendet, die Identifier sind in gemischter Schreibweise.

INSERT INTO Kategorie (Bezeichnung, Klassifikationkriterium) VALUES (‘U-18´, ´Alter unter 18 Jahren´’), analog G-Kunde und noch einmal U-18.

Beim zweiten Einfügen von U-18 wird eine Fehlermeldung zurückgegeben, die besagt, dass gegen die Schlüsselbedingung verstoßen wurde. U-18 existiert ja bereits als Schlüsselwert in der Tabelle.

2.) Legen Sie die Tabelle Konto an.
Fügen sie das Tupel (Kontonr, Betrag, Gebühr): (174266,789.50,3) ein, danach das Tupel (Kontonr): (174255). Was passiert im zweiten Fall?

CREATE TABLE Konto (Kontonr INTEGER NOT NULL PRIMARY KEY, Betrag NUMERIC(10,2), Gebühr NUMERIC(10,2))

INSERT INTO Konto (Kontonr, Betrag, Gebühr) VALUES (174266,789.50,3)

INSERT INTO Konto (Kontonr) VALUES (174255)

Die zweite Zeile enthält NULL-Werte für Betrag und Gebühr.

3.) Legen Sie ähnlich wie oben die Tabelle Kunde an. Hierbei soll gelten, dass die Kategoriebezeichnung eines Kunden (15 Zeichen, variabel) nicht leer sein darf. Um den zusammengesetzten Schlüssel darzustellen, verwenden Sie das Primary Key(Spaltenname, Spaltenname, ...)-Konstrukt, das wie eine Spaltendefinition eingetragen wird.
Fügen Sie folgende Tupel ein (VN, NN, Gebdatum, Bezeichnung): (´Heinz´,´Schiller´,´05/03/1942´,´P-Kunde´) und (VN, NN, Gebdatum)(´Anette´,´Müller´, ´07/25/1953´). Was passiert im zweiten Fall?

```
CREATE TABLE Kunde (
VN VARCHAR(30) NOT NULL,
NN VARCHAR(30) NOT NULL,
Gebdat DATE NOT NULL,
Bezeichnung VARCHAR(15) NOT NULL,
PRIMARY KEY (VN,NN, Gebdat))
```

Inserts analog zur letzten Aufgabe.

Beim zweiten Insert wird eine Fehlermeldung ausgegeben, dass ein benötigter Wert fehlt, da wir Name als NOT NULL definiert haben.

Der Datentyp DATE wurde in dieser Aufgabe im amerikanischen Datumsformat gesetzt. Es funktioniert allerdings auch die gewohnte europäische Schreibweise (25.7.1953). In vielen Datenbanksystemen hängt dieses Verhalten von der installierten Sprachversion ab.

4.) Setzen Sie die restlichen Relationen analog zu 1-3 um.

```
CREATE TABLE Festgeldkonto (
Kontonr INTEGER NOT NULL PRIMARY KEY,
Faelligkeit DATE,
zinssatz NUMERIC(4,2))

CREATE TABLE Girokonto (
Kontonr INTEGER NOT NULL PRIMARY KEY,
Dispolimit NUMERIC(10,2),
Telebanking CHAR(1))

CREATE TABLE Kontobewegung (
Datum DATE NOT NULL,
VN VARCHAR(30) NOT NULL,
NN VARCHAR(30) NOT NULL,
Gebdat DATE NOT NULL,
Kontonr INTEGER NOT NULL,
Betrag NUMERIC(10,2),
Bewegungsart VARCHAR(15),
PRIMARY KEY (Datum, VN, NN, Gebdat, Kontonr))

CREATE TABLE Kontoauszug(
Erstelldat DATE NOT NULL PRIMARY KEY,
```

```
Kontonr INTEGER,
Startdat DATE,
Gesamt NUMERIC (10,2)
)

CREATE TABLE besitzt (
VN VARCHAR(30) NOT NULL,
NN VARCHAR(30) NOT NULL,
Gebdat DATE NOT NULL,
Kontonr INTEGER NOT NULL,
PRIMARY KEY (VN, NN, Gebdat, Kontonr)
)

CREATE TABLE listet (
Erstelldat DATE NOT NULL,
Kontonr INTEGER NOT NULL,
Datum DATE NOT NULL,
VN VARCHAR(30) NOT NULL,
NN VARCHAR(30) NOT NULL,
Gebdat DATE NOT NULL,
PRIMARY KEY (Erstelldat, Kontonr, Datum, VN, NN, Gebdat)
)
```

5.) Oft ist es notwendig, eine Tabelle nachträglich noch zu verändern.
Fügen Sie zur Tabelle Kunde eine neue Spalte Wohnort mit Typ VARCHAR(30) hinzu.
Um Tabellen nachträglich weitere Spalten hinzuzufügen, gibt es die Anweisung
ALTER TABLE Tabellenname ADD COLUMN Spaltendefinition
Welche Werte sind in der neuen Spalte?

```
ALTER TABLE KUNDE ADD COLUMN Wohnort VARCHAR(30)
```
Die neue Spalte enthält NULL

6.) Legen Sie einen Index zu den Bezeichnungen in der Tabelle Kunde an.

```
CREATE INDEX  bezindex ON Kunde(Bezeichnung)
```

7.) Versuchen Sie, einen Index auf Bezeichnung in der Tabelle Kategorie anzulegen.
Welchen Schluss können Sie aus der Fehlermeldung ziehen?

```
CREATE INDEX katbezindex ON Kategorie (Bezeichnung)
```
Die Fehlermeldung besagt sinngemäß, dass ein Index mit den gleichen Eigenschaften schon existiert. Viele DBMS legen automatisch einen Index auf den Primärschlüssel an, so auch DB2

8.) Löschen Sie den Index aus Aufgabe 5.) wieder.

```
DROP INDEX bezindex
```

Ähnlich können Tabellen gelöscht werden.

Aufgabe 2: SQL-Anfragen, Kontobeispiel

Formulieren Sie folgende Anfragen basierend auf dem Schema aus Aufgabe 1 in SQL.

1.) Finden Sie alle Kunden.

SELECT * FROM Kunde

SELECT * gibt alle Spalten des Ergebnisses zurück. Mit der From-Klausel werden die Quellrelationen bestimmt.

2.) Finden Sie alle Kategoriebezeichnungen.

SELECT Bezeichnung FROM Kategorie

Projektion erfolgt in SQL durch die Nennung der Spalten nach dem SELECT.

3.) Finden Sie alle Kontobewegungen des Kunden Heinz Schiller (geb. am 3.5.1942), bei denen mehr als 500 Euro bewegt wurden.

SELECT * FROM Kontobewegung WHERE VN=´Heinz´ AND NN=´Schiller´ AND Gebdat = ´05/03/1942´ AND Betrag > 500

Mehrere Prädikate können durch logisches UND und ODER verbunden werden. Der Datentyp DATE wurde in dieser Aufgabe im amerikanischen Datumsformat gesetzt. Es funktioniert allerdings auch die gewohnte europäische Schreibweise (25.7.1953). In vielen Datenbanksystemen hängt dieses Verhalten von der installierten Sprachversion ab.

4.) Finden Sie alle Kunden des Kontos 174266.

SELECT Kunde.VN, Kunde.NN FROM Kunde, besitzt
WHERE Kunde.VN=besitzt.VN AND Kunde.NN=besitzt.NN AND
Kunde.Gebdat=besitzt.Gebdat AND besitzt.Kontonr= 174266

Indem die beteiligten Tabellen im FROM und das verbindende Prädikat im WHERE angegeben werden, sind Joins darstellbar. Wenn der Spaltenname über alle beteiligten Tabellen eindeutig ist, muss der Tabellenname nicht angegeben werden, andernfalls ist dies notwendig.

5.) Wieviele Kunden haben mehr als 5000 Euro Guthaben.

SELECT count(*) FROM Kunde k, besitzt b, Konto kn WHERE
k.VN=b.VN AND k.NN=b.NN AND k.Gebdat=b.Gebdat
AND b.Kontonr=kn.Kontonr AND kn.Betrag > 5000

Mit Hilfe der Aggregatfunktionen kann man Maximum, Minimum, Durchschnitt, Summe oder Anzahl Werte einer Spalte bestimmt werden. Wird bei COUNT ein Stern als „Spalte“ angegeben, so wird die Anzahl Tupel gezählt. Joins können über mehrere beteiligte Tabellen formuliert werden. Zur vereinfachten Eingabe

und zur Unterstützung von Self-Joins können die Tabellen andere Namen erhalten.

6.) Finden Sie das Konto mit dem höchsten Guthaben.

SELECT max(Betrag) FROM Konto
liefert nur den höchsten Betrag.

SELECT Kontonr, Betrag FROM Konto WHERE Betrag =
(SELECT max(Betrag) FROM Konto)
liefert das gewünschte Ergebnis.

Aufgabe 3: SQL-Anfragen, Bibliotheks-DB

Gegeben ist folgende Datenbank, die das Ausleihwesen einer Bibliothek unterstützt:

- Für Leser werden die wichtigsten Daten erfasst und jedem Leser eine eindeutige Lesernummer zugeordnet.
- Bei Büchern stellt die ISBN (eine zehnstellige Zahl) einen eindeutigen Schlüssel dar. Sie wird für jede Ausgabe neu vergeben, ändert sich aber für einzelne Exemplare nicht.
- Falls ein Leser ein Buch ausleiht, wird in der Relation „Ausleihe“ ein Tupel mit der ISBN des Buches, seiner Lesernummer und dem Ausleihdatum angelegt. Wir berücksichtigen noch nicht, dass es mehrere Exemplare eines Buches geben kann, d.h. es wird nur festgehalten, dass für diesen Leser und ein solches Buch eine Ausleihe besteht.

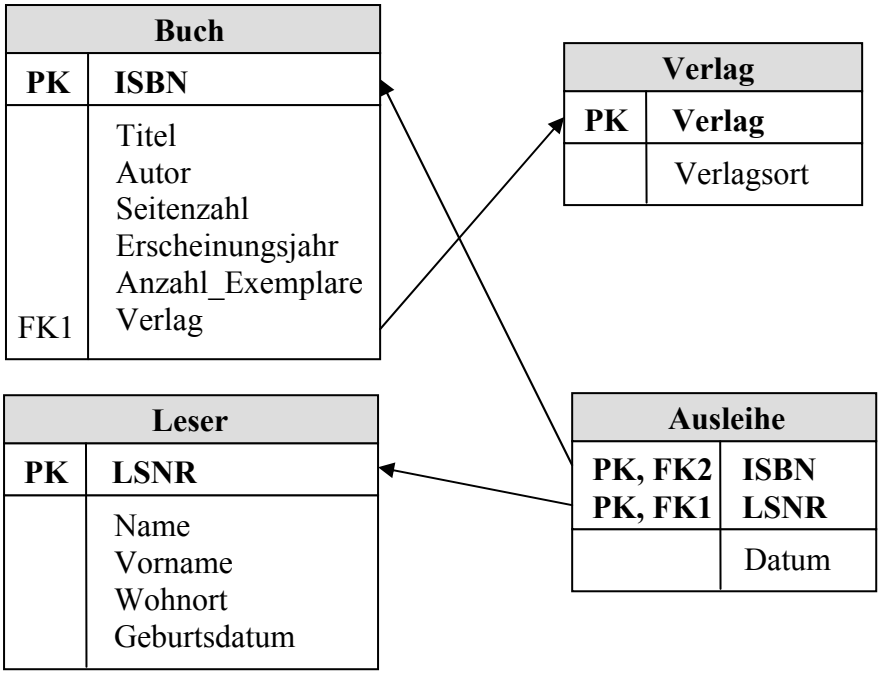
LESER (LSNR, NAME,VORNAME, WOHNORT, GEBDATUM)

BUCH (ISBN, TITEL, AUTOR, SEITENZAHL, VERLAG, ERSCHEINUNGSJAHR,

ANZAHL_EXEMPLARE)

VERLAG (VERLAG, VERLAGSORT)

AUSLEIHE (ISBN, LSNR, DATUM)



Die Grafik stellt die Relationen noch einmal anschaulicher dar. Primärschlüssel und Fremdschlüssel sind als solche markiert, die Pfeile geben die Fremdschlüsselbeziehungen an

Formulieren Sie folgende Anfragen in SQL:

1. Welche Bücher (AUTOR, TITEL) hat der Leser Lemmi Schmöker ausgeliehen?

```
SELECT  B.AUTOR, B.TITEL
FROM    LESER L, AUSLEIHE A, BUCH B
WHERE   L.NAME = 'Schmöker' AND
        L.VORNAME = 'Lemmi' AND
        L.LSNR = A.LSNR AND
        A.ISBN = B.ISBN
```

Vorgehen:

- Lesernummer von Lemmi Schmöker finden (aus Leser)
- Mit der Lesernr die Buchnummern der ausgeliehenen Bücher (Join mit Ausleihe)
- Buchtitel und Autor bestimmen (durch Join mit Buch)

2. Welche Bücher (AUTOR, TITEL) sind in mehreren verschiedenen Ausgaben vorhanden?

```
SELECT DISTINCT B1.TITEL, B1.AUTOR
FROM    BUCH B1, BUCH B2
WHERE   B1.TITEL = B2.TITEL AND
        B1.AUTOR = B2.AUTOR AND
        B1.ISBN <> B2.ISBN
```

Vorgehen:
Bücher mit verschiedenen Auflagen haben denselben Titel und Autor, aber unterschiedliche ISBNs. Diese Bücher erhält man am einfachsten durch einen Selfjoin auf Buch.

3. Welche Leser (NAME, VORNAME) haben Bücher ausgeliehen, die an ihrem Wohnort verlegt wurden?

```
SELECT  L.NAME, L.VORNAME
FROM    LESER L, AUSLEIHE A, BUCH B, VERLAG V
WHERE   L.LSNR = A.LSNR AND
        A.ISBN = B.ISBN AND
        B.VERLAG = V.VERLAG AND
        L.WOHNORT = V.VERLAGSORT
```

Vorgehen:

- Buchnummern von ausgeliehenen Büchern durch Join zwischen Leser und Ausleihe
- Bücherdaten durch Join mit Buch
- Verlagsdaten durch Join mit Verlag
- (Ringschluss) Selektion auf Tupel mit Wohnort = Verlagsort

4. Welcher Leser (NAME, VORNAME) hat mindestens ein Buch ausgeliehen, das auch Leser Lemmi Schmöker ausgeliehen hat (Lemmi Schmöker soll nicht ausgegeben werden)?

```
SELECT  L1.NAME, L1.VORNAME
FROM    LESER L1, AUSLEIHE A1, AUSLEIHE A2, LESER L2
WHERE   L2.VORNAME='Lemmi' AND
        L2.NAME = 'Schmöker' AND
        A2.LSNR = L2.LSNR AND
        A1.ISBN = A2.ISBN AND
        L1.LSNR = A1.LSNR AND
        L1.LSNR <> L2.LSNR
```

Vorgehen: •

- Lesernummer von Lemmi Schmöker (L2)
- Bücher von Lemmi Schmöker durch Join mit A2

- Lesernummer alles Personen, die die gleichen Bücher wie Lemmi
- ausgeliehen haben, durch Join mit A1
- Daten der Leser durch Join mit L1
- Lemmi aus dem Ergebnis entfernen durch ungleiche LSNR
- Projektion auf die Daten der Leser aus L1