Prüfungsleistung ED im Fach Datenbankgestützte Informationssysteme (DBIS)

Kassenbon-Data Warehouse, Umsätze und Analyse mit Aggregattabellen auf Wochenbasis

Fachbereich: Management, Information, Technologie
Prof. Dr. Alfred Wulff

Aufgabe:	13
Matrikelnummer:	6013868
Name:	Tokuc
Vorname:	Kübra
Email:	kuebra.tokuc@student.jade-hs.de
Studiengang:	Wirtschaftsinformatik
Semester:	WS2019/20
Abgabetermin:	16. Januar 2020

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Aufgabenbeschreibung

In der zunehmden digitalisierten Welt entstehen in jeglichen Bereichen immer mehr Daten, die eine Verarbeitung erfordern. Dies betrifft auch den Bereich des Einzelhandels, in dem Einkaufsdaten aus Kassenbons (s. Abbildung 1) durch Big Data und BI-Technologien für die Auswertung in einen Kontext gebracht werden können. Daraus können Erkenntnisse über das Einkaufsverhalten der Kunden geschlossen werden, die als Entscheidungsgrundlage dienen können. In diesem Projekt wird dargestellt, wie eine solche Auswertung durchgeführt werden kann und welche Erkenntnisse daraus geschlossen werden können.

Abbildung 1: Kassenbon (Quelle: DBIS-Aufgaben)

Der Kassenbon enhält die Bereiche (1) Filiale, (2) Standort, (3) Bonpositionen mit Verkaufspreisen, (4) die Summe der Verkaufspreise sowie (5) Datum, Kasse, Bedingung etc. Der Fachbereich beschreibt die Abteilung, zu der die Produkte zugeordnet werden.

- Bonnummer
- Filiale
- Fachbereich/Fachabteilung
- Warengruppe
- Artikelnummer
- Produktbezeichnung
- Handelsmarke
- Trockensortiment (Ja, Nein)
- Verkaufsdatum mit Uhrzeit
- Verkaufsmenge
- Verkaufspreis
- Aktionsware (optional)
- Rabattierte Ware(optional)

2. Pflichtenheft

In diesem Kapitel werden die Kriterien für für die Erfüllung der Projektanforderungen kurz beschrieben.

2.1. Zielbestimmung

Musskriterien

1. Richten Sie die Tabellen mit den bereitgestellten SQL-Skripten in Ihrer eigenen Datenbank des MS-SQL-Servers ein.

2. Eines der gegebenen SQL-Skripte füllt die operationale Datenbank mit einem Grundbestand an Filialen und Produkten. Es existieren aber noch weitere PRODUKTE, die in der Datei „PRODUKTE1.csv“ abgelegt sind. Erstellen Sie ein Java-Programm, das die zusätzlichen Daten in die operationale Datenbank klädt. In der CSV-Datei können fehlerhafte Daten enthalten sein. Führen Sie eine geeignete Fehlerbehandlung und -protokollierung durch. Verwenden Sie das JDBC-API oder die Hibernate-Technologie (optional).

3. In dem vorgegebenen SQL-Skript wird die operationale Datenbank auch mit einem Grundbestand an Kassenbons gefüllt. Erweitern Sie den Datenbestand um min. 40 Kassenbons mit durchschnittlich 3 Bonpositionen und 7 Handelsmarken sowie 10 Fachbereiche der Filiale 14. Der Dateiname mit den zugehörigen Anweisungen soll ‚3-Insert-Student‘ lauten und im Projektordner ‚sql‘ abgelegt sein.

4. Analysieren Sie das bereitgestellte STAR-Schema für ein Data-Warehouse, welches dem Händler die Umsatzanalyse und Auswertung von Verkäufen ermöglicht. Als Faktenattribute werden Kassenbondaten berücksichtigt. dAuswertungen sollen in Bezug auf Filiale, Abteilung, Produkt und Zeiträumen möglich sein. Die Faktendaten sollen tagesgenau gespeichert werden. Es sollen zusätzlich aggregierte Faktenwerte auf Wochenbasis verfügbar sein.

5. Erstellen Sie geeignete ETL-SQL-Skripte (falls erforderlich DDL- unf DML-Anweisungen), um die Daten aus der operativen DB in das DW zu extrahieren.

6. Entwickeln Sie eine DB-Prozedur namens DW-REPORT. Zu allen oder ausgewählten Merkmalen der vorhandenen Dimensionen sollen die in einem bestimmten Zeitraum erzielten Verkäufe (z.B. Umsätze, Verkaufsmengen) berechnet und angezeigt werden. Der gewünschte Zeitabschnitt sowie Parameter der Dimensionen sollen weitgehend variabel, d.h. vom Benutzer einstellbar sein. Die Prozedur wird zur Auswertung des DW eingesetzt. Sie liefert in der Konsole eines SQL-Editors die Ergebnisse der Auswertung. Der Report soll formatiert und übersichtlich gestaltet sein. Verwenden Sie keine Reporting-Dienste, sonder erstellen Sie den SQL-Report mittels SQL-Anweisungen aus den Vorlesungsunterlagen und der Online-Dokumentation des MS SQL-Servers. Ein Beispiel für einen Prozeduraufruf mit von Ihnen vorgschlagenen Parametern ist zwingend erforderlich.

Kannkriterien

- Zusätzlich soll eine Visualisierung der Auswertungen mit der BI-Software Qlik-Sense vorgenommen werden.

2.2. Einsatz

1. Anwendungsbereiche: Data Warehousing, Auswertung, Reporting
2. Zielgruppen: Management, Controlling
3. Betriebsbedingungen: 24h Betriebsbereitschaft

2.3. Umgebung

1. Software-Client
   - Windows 10
   - JAVA JDK und JDBC
2. Software-Server
   - MS SQL-Server auf whv-fbmit3.hs-woe.de

3. Hardware
    - Client: H401/H416a/W108 Pool Rechner
    - Server: whv-fbmit3.hs-woe.de
4. Orgware: ERM

2.4. Daten

Es wird ein Grunddatenbestand bereitgestellt. Dieser ist gem. Zielbestimmungen zu erweitern. Wenn Sie für die zusätzlich erstellten Daten externe Quellen verwenden, so sind sie anzugeben. Änderungen der Datenstrukten sind zu dokumentieren.

2.5. Performance

keine spezifischen Anforderungen

2.6. Qualitätsziele

- hohe Benutzerfreundlichkeit
- hohe Änderbarkeit

2.7. Ablieferung der Ergebnisse

Folgende Bestandteile sind anzugeben bzw. einzurichten:

1. Alle DB-Schemata müssen auf der eigenen Datenbank auf dem SQL-Server whv.fbmit3.hs-woe.de eingerichtet werden. Alle Anwendungsdaten liegen in dieser DB und die Programme greifen auf die Daten zu. Multiuser-Fähigkeit sollte gewährleistet sein (durch die Angabe des Eigentümers der DB-Objekte in SQL-Statements). Die Prüfer melden sich am MS SQL-Server nicht mit ihren Login-Daten an. Deshalb müssen Sie sicherstellen, dass die Skripte durch andere DB-User aufgerufen werden können.
2. Abgabe lauffähiger, getesteter und ausführbarer SQL-Prozeduren:

- Die Skripte und Prozeduren müssen ausführbar sein. Diese sind im Projektordner **sql** abzulegen.
- Ein Beispiel für den Prozeduraufruf muss erstellt werden.
- Das Java-Programm muss lauffähig und mittels bat-Datei realisiert sein

3. Folgende Unternlagen sollen in gedruckter Form vorgelegt werden:
   - Pflichtenheft/Aufgabenstellung
   - Lösungswegbeschreibungen für (1) ETL-Prozess, (2) DW-REPORT (inkl. SQL-Anweisungen, Skripte, Prozeduren, Views, Trigger etc.) sowie zusätlich erstellte Tabellen udn Formeln.
   - extra ausgedruckt: Java-Quellcode und SQL-Skripte mit Kommentaren und Erläuterungen
   - Benutzerhandbuch für Anwendung des DW-Report
4. Folgende Unternlagen sollen in elektronischer Form (CD/DVD) vorgelegt werden:
   - Alle gedruckten Unternlagen
   - Alle Java-Programme und selbst erstellen SQL-Skripte
5. Abgabe in einem heftbaren Ordner inklusive eines fest angebrachten, leicht abnehmbaren Datenträgers.

2.8. Selbstständige Erarbeitung der Ergebnisse

Die Ergebnisse von jedem Prüfling müssen selbst erbracht werden. Wer versucht, das Ergebnis ihrer Prüfungsleistung durch Täuschung oder Benutzung nicht zugelassener Hilfsmitteln (Entwurfs- und Programmbestandteile anderer Studierender) zu beeinflussen, wird mit **nicht ausreichend** bewertet.

2.9. Organisatorisches

- Abgabe am 16.01.2020 (09:00. W108). Jeder Studierende muss zur Abgabe der Dokumentation persönlich erscheinen und dem Prüfer die Ergebnisse vorstel-

len.

3. Umsetzung

In diesem Kapitel wird der Lösungsweg für die im Pflichtheft aufgeführten Kriterien beschrieben. Es ist anzumerken, dass die Kommentare im Quellcode teils in englischer Sprache verfasst sind. Grund dafür ist die Inkompatibilität des Latex-Listing-Pakets mit UTF8 Unicode Zeichen. Im Lösungsweg werden lediglich Ausschnitte des Quellcodes gezeigt. Die vollständigen Quellcodes befinden sich im Anhang.

3.1. Einrichtung der Datenbank

Als Grundlage für das Java-Ladeprogramm, der Ausführung der ETL-Skripte und der Prozedur zum Reporting für das DW müssen zunächst die Datenbanken eingerichtet und mit Daten befüllt werden. Die Einrichtung und Befüllung der Datenbanken werden mit Hilfe der zur Verfügung gestellten Skripte nach den vorliegenden Datenmodellen erstellt:

3.2. Java-Programm

Das Java-Programm soll dazu dienen, den Datenbestand für Produkte aus der CSV-Datei **PRODUKTE.1** zu erweitern. Allerdings ist die Datei von fehlerhaften Daten befallen, welche nicht mit der Struktur der Datenbanktabelle übereinstimmen. Um diese Problematik zu lösen, wurde mit Hilfe des zur Verfügung gestellten Ladeprogramms ein eigenes Programm entwickelt. Einige Klassen wurden komplett selbst enwickelt und andere angepasst, während die Klasse **DateFormat** vollständig übernommen wurde. Zunächst wurde ein Importfenster (s. Abbildung 4) mit der Klasse **ImportWindow** erzeugt. Es dient dazu, die Parameter für die Verbindung mit dem Datenbankserver einzutragen und die zu hochladende CSV-Datei aus dem Computerverzeichnsi auszuwählen. Ob die Datenbankverbindung erfolgreich war, wird in dieser Klasse getestet. Die durchgeführten Aktionen werden in dem Protokoll dokumentiert. Man kann daraus entnehmen, welche Zeilen der Datei erfolgreich geladen wurden und welche aus welchem Grund fehlgeschlagen sind. Die Datenbank-

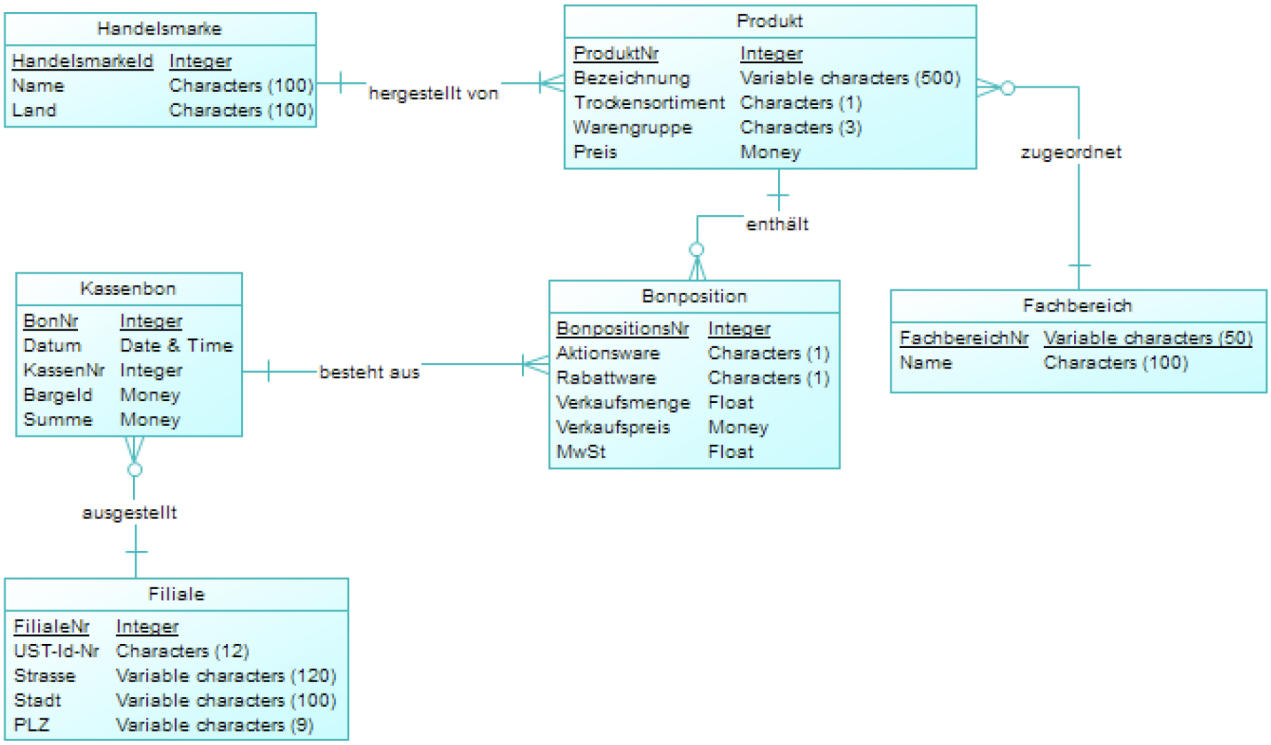

Abbildung 2: ER-Modell zur Auswertung von Kassenbons (Quelle: DBIS-Aufgaben)

verbindung wird in der Klasse **DBConnection** eingerichtet. Im Gegesatz zu der Implementierung im Beispielprogramm, wurde in diesem Beispiel ein **JDBC Data Source** Objekt verwendet, welches den gleichen Treiber verwendet wie das Beispielprogramm. Der Grund für die Entscheidung ist die zum Einen die Übersichtlichkeit und zum anderen die Tatsache, dass diese Herangehensweise mittleweile bevorzugt sei (Wulff, 2019).

```
1  public synchronized Connection getConnection() throws SQLException{
2
3  // Connection with DataSource
4
5  try {
6   if (con == null || con.isClosed()) {
7    DataSource ds = new TdsDataSource();
8    ((TdsDataSource) ds).setServerName(serverName);
```

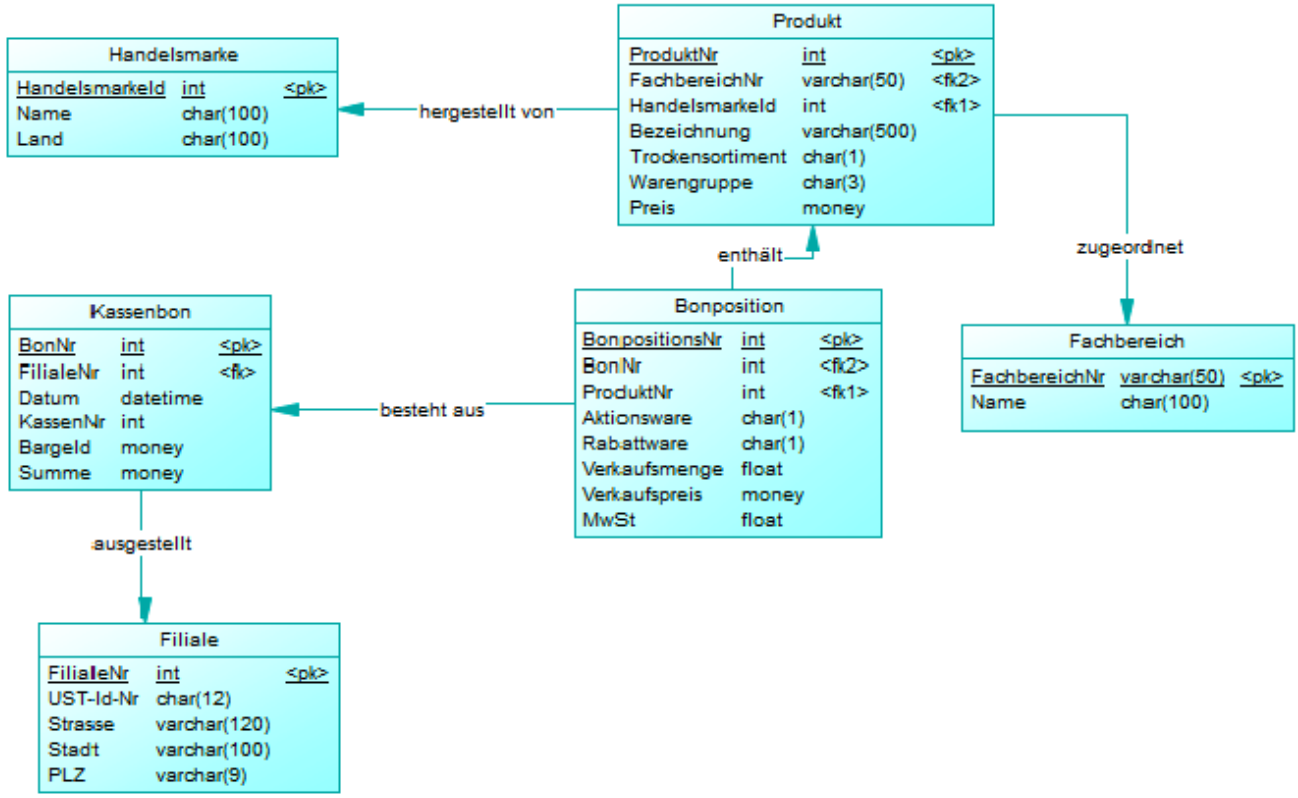

Abbildung 3: Physikalisches Datenmodell (Quelle: DBIS-Aufgaben)

```
9     ((TdsDataSource) ds).setPortNumber(Integer.parseInt(port));
10    con = ds.getConnection(username, password);
11    con.setCatalog(databaseName);
12   }
13  } catch (SQLException e) {
14   log.log(Level.WARNING, "DB Connection failed.", e);
15   JOptionPane.showMessageDialog(new JFrame(),
16     " DB Connection failed!\n" +
17       "Check your connection input",
18       "CSV Import", JOptionPane.ERROR_MESSAGE);
19  }
20  return con;
```

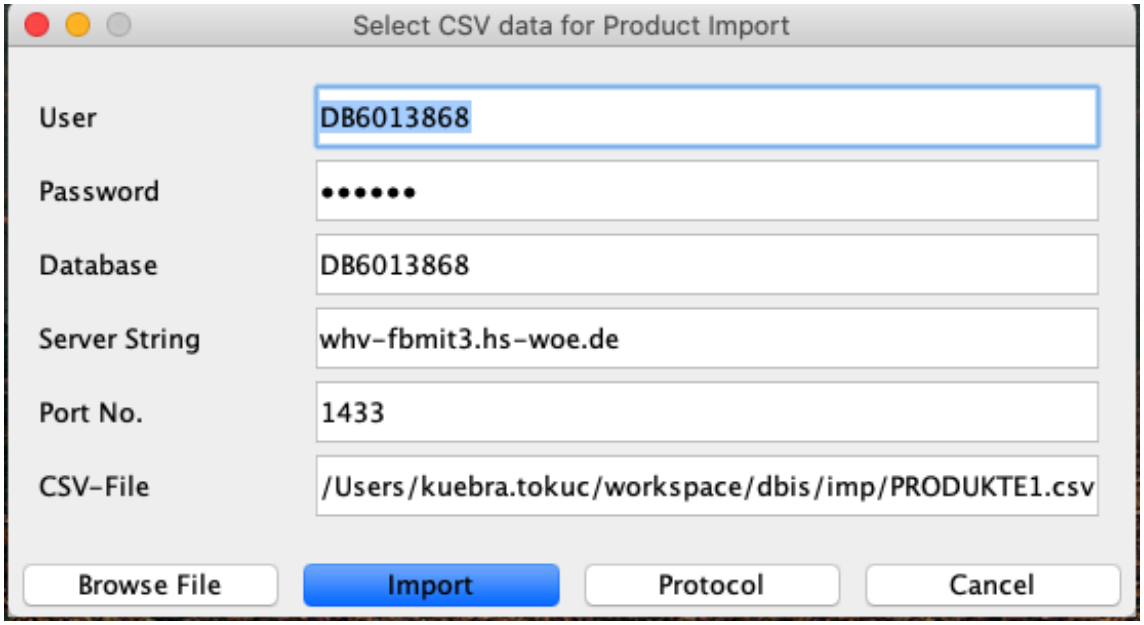

Abbildung 4: Importfenster des Java-Programms

```
21
22  }
```

Die Klasse **FileLineParser** für die Erzeugung eines Objekts für den Import wurde ebenfalls angepasst und erweitert. Zum Beispiel werden die Warengruppennummern und Fachbereichsnummern mit übergeben, damit sie nicht nachträglich angepasst werden müssen. Zudem wird eine Methode zum Erstellen eines neuen Fachbereichs in der DB für Gewürze angestoßen, damit die Zuordnung korrekt stattfindet. Die Methode **isDub** prüft, in einer Hashmap, ob Einträge in einer Zeile redundaten ist und gibt bei Redundanzen den Wert true zurück. Dies ist für die Fehlerbehandlung von großer Bedeutung.

```
1   // Ubergabe der Warengruppennummer
2
3  public static String getWarengruppe(String s_warengruppe) {
4
5   if (s_warengruppe == null)
6    return null;
7
```

```
8   String warengruppe_id = null;
9
10  if (s_warengruppe.equalsIgnoreCase("Milch"))
11   warengruppe_id = "002";
12  else if (s_warengruppe.equalsIgnoreCase("Musli & Cerealien"))
13   warengruppe_id = "003";
14  else if (s_warengruppe.equalsIgnoreCase("Saucen"))
15   warengruppe_id = "004";
16  else if (s_warengruppe.equalsIgnoreCase("Gewurze"))
17   warengruppe_id = "005";
18  else
19   warengruppe_id = "006";
20
21  return warengruppe_id;
22 }
23
24 // Ubergabe der Fachbereichsnummer
25 // wird auch Fachbereich fuer Gewurze erstellen
26
27 public static String getFachbereichsNummer(String warengruppe_id)
       throws SQLException {
28
29  if (warengruppe_id==null) {
30   return null;
31  }
32  String fb_id= null;
33  if (warengruppe_id=="002") {
34   fb_id= "1014";
35  }else if(warengruppe_id == "003") {
36   fb_id= "1023";
37  }else if(warengruppe_id == "004") {
38   fb_id= "1015";
39  }else if(warengruppe_id == "005") {
```

```
40    ImportRoutine.createNewFB("Gewurze", "1039");
41    fb_id="1039";
42   }
43
44   return fb_id;
45  }
46
47  // Methode zum Prufen von Redundanten Eintragen
48
49  public static boolean isDub(String bezeichnung) {
50
51   String[] words = bezeichnung.split("[\\s+]");
52
53   Map<String, Integer> occurrences = new HashMap<String, Integer>();
54
55   boolean isdub =false;
56   Integer oldCount=0;
57   for ( String word : words ) {
58    oldCount = occurrences.get(word);
59    if ( oldCount == null ) {
60     oldCount = 0;
61    }
62    occurrences.put(word, oldCount + 1);
63    if(oldCount>2) isdub = true;
64    System.out.println("oldcount : "+ oldCount+ "word"+ word + "isdub"
        + isdub);
65   }
66   System.out.println("IS DUB? :"+ isdub);
67   return isdub;
68  }
69
70 }
```

Die Klasse **ImportRoutine** wurde ebenfalls angepasst. Zudem wurde eine Methode zum Erzeugen eines neuen Fachbereichs hinzugefügt. In **DBImport** findet die Ausführung der Datenbanktransaktionen statt. Auch der Insert des Fachbereichs findet hier statt.

```
1   public static void createNewFB(String name, String fb_nr) throws
        SQLException {
2
3         DBImport.insertFB(name, fb_nr, con);
4
5       }
```

3.3. Data Warehouse

3.3.1. Erweiterung des Datenbestandes

Da in Datenanalysen die Menge der Daten einen wichtigen Einfluss hat, wird der Datenbestand um 9 Handelsmarken, 9 Fachbereiche, 40 Kassenbons mit je 3 Bonpositionen erweitern. Die Verkaufspreise Summen im Kassenbon sind zu diesem Zeitpunkt noch nicht auf die Beträge der Bonpositionen angepasst und sind „Dummy-Werte“

```
1
2   --- Beispiel fur einige Kassenbons ---
3   insert into KASSENBON (FILIALENR, DATUM, KASSENNR, BARGELD, SUMME)
4 VALUES
5
6 (14, CONVERT([datetime], '2019-02-10 08:22:00.000', 20), 1, 400.00,
      400.00),
7 (14, CONVERT([datetime], '2019-02-11 10:22:00.000', 20), 1, 33.00,
      33.00),
8 (14, CONVERT([datetime], '2019-02-16 08:22:00.000', 20), 1, 12.00,
      12.00),
9 (14, CONVERT([datetime], '2019-02-17 10:22:00.000', 20), 1, 123.00,
      150.00),
10 --- Beispiel fur Bonpositionen des Bons Nr. 10 ---
```

```
11 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
12 values
13 (10, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
14 (10, 2, 'N', 'N', 1, 11.9900, 11.9900 * 0.07),
15 (10, 3, 'Y', 'Y', 10, 24.900, 24.900 * 0.07);
16
17 --- Hinzufugen von Handelsmarken ---
18 insert into HANDELSMARKE (NAME, LAND)
19 values
20 ('Schar', 'Deutschland'),
21 ('Nestle', 'Schweiz'),
22 ('Kellog''s', 'USA'),
23 ('Coca Cola', 'USA'),
24 ('Goutess', 'Deutschland'),
25 ('Gutfried', 'Deutschland'),
26 ('Jever', 'Deutschland'),
27 ('Nick', 'USA'),
28 ('Basic', 'USA');
29
30 -- ** Gewurze wurden bereits aus dem Java-Programm heraus hinzugefugt
       *** --
31 insert into FACHBEREICH (FACHBEREICHNR, NAME)
32 values
33 (1040,'Snacks'),
34 (1041,'Oriental'),
35 (1042,'Asia'),
36 (1043,'Italienisch'),
37 (1044,'Drogerie'),
38 (1045,'Haushalt'),
39 (1046,'Getranke'),
40 (1047,'Desserts');
41 (1048,'Tierfutter');
```

3.3.2. Aufbereitung des Datenbestandes

Es wurden zwar Daten hinzugefügt, aber waren noch nicht den richtigen Handelsmarken zugeordet. Daher wurde gepüft, welche Handelsmarken ID der Produktbezeichnung zugefügt werden soll. Außerdem wurde die Summe der Kassenbons nachträglich aus der Summe der Verkaufspreisen der einzelnen Bonpositionen korrigiert. Damit das Bargeld höher ist als die Summe, wurden standardmäßig 50 Euro addiert.

```
1   -- *** Beispiel Zuordnung der Handelsmarken *** --
2
3  update PRODUKT
4  set HANDELSMARKEID = 40
5  where BEZEICHNUNG LIKE '%Kellog%';
6
7  update PRODUKT
8  set HANDELSMARKEID = 42
9  where BEZEICHNUNG LIKE '%Goutess%';
10
11 -- ** Update der Summen in Kassenbons ** --
12
13 UPDATE KASSENBON
14 SET SUMME = t.summe_einkauf
15 FROM KASSENBON AS kb
16 INNER JOIN
17     (
18         SELECT BONNR, SUM(VERKAUFSPREIS) summe_einkauf
19         FROM BONPOSITION
20         GROUP BY BONNR
21     ) t
22     ON t.BONNR = kb.BONNR
23
24 -- ** Bargeld an Summe anpassen, um Deckung zu gewaehrleisten ** --
25
26 update KASSENBON
```

```
27  set BARGELD = Round(SUMME, 0)
28
29 -- ** Bargeld standardmaessig erhoehen, um Differenz zu erzeugen ** --
30
31 update KASSENBON
32  set BARGELD = BARGELD + 50
33
34 -- ** runden ** --
35
36 update KASSENBON
37  set BARGELD = Round(Bargeld,-1)
```

3.3.3. Analyse des Starschemas

Nachdem der Datenbestand erweitert und korrekt aufbereitet worden ist, wird ein Star-Schema für ein Data-Warehouse zur Umsatzanalyse und Auswertung von Verkaufsdaten nach einem vorliegenden Schema (s. Abbildung 5) erstellt. Das Star-Schema besteht aus Dimensionen- und Faktentabellen, welche die Struktur und den Inhalt einer multidimensionalen Datenmenge definieren (Wulff, 2019). Die Werte in der Faktentabelle werden durch die Primary Keys der Dimensionstabelle mit ihr in Abhängigkeit gebracht (s. Abbildung 6). Allerdings ergibt die Analyse des Datenmodells Inkosistenzen zwischen den verschiedenen Tabellen:

1. Kein Fachbereich im Sternschema, stattdesen Abteilung
2. Kassenbon_Fakten AbteilungsNr FK int und DIM_Abteilung AbteilungsNr PK int
3. FilialeNr int der operativen DB und FilialeNr char(3) der DIM_Filiale
4. Warengruppe char(3) der operativen DB (ODB) und Warengruppe char(1) der DIM_Produkt
5. Bezeichnung des Produkts in ODB varchar(500) und in DIM_Produkt varchar(100)

Es stellte sich jedoch heraus, dass diese Unterschiede keinen Einfluss auf den Report hatten.

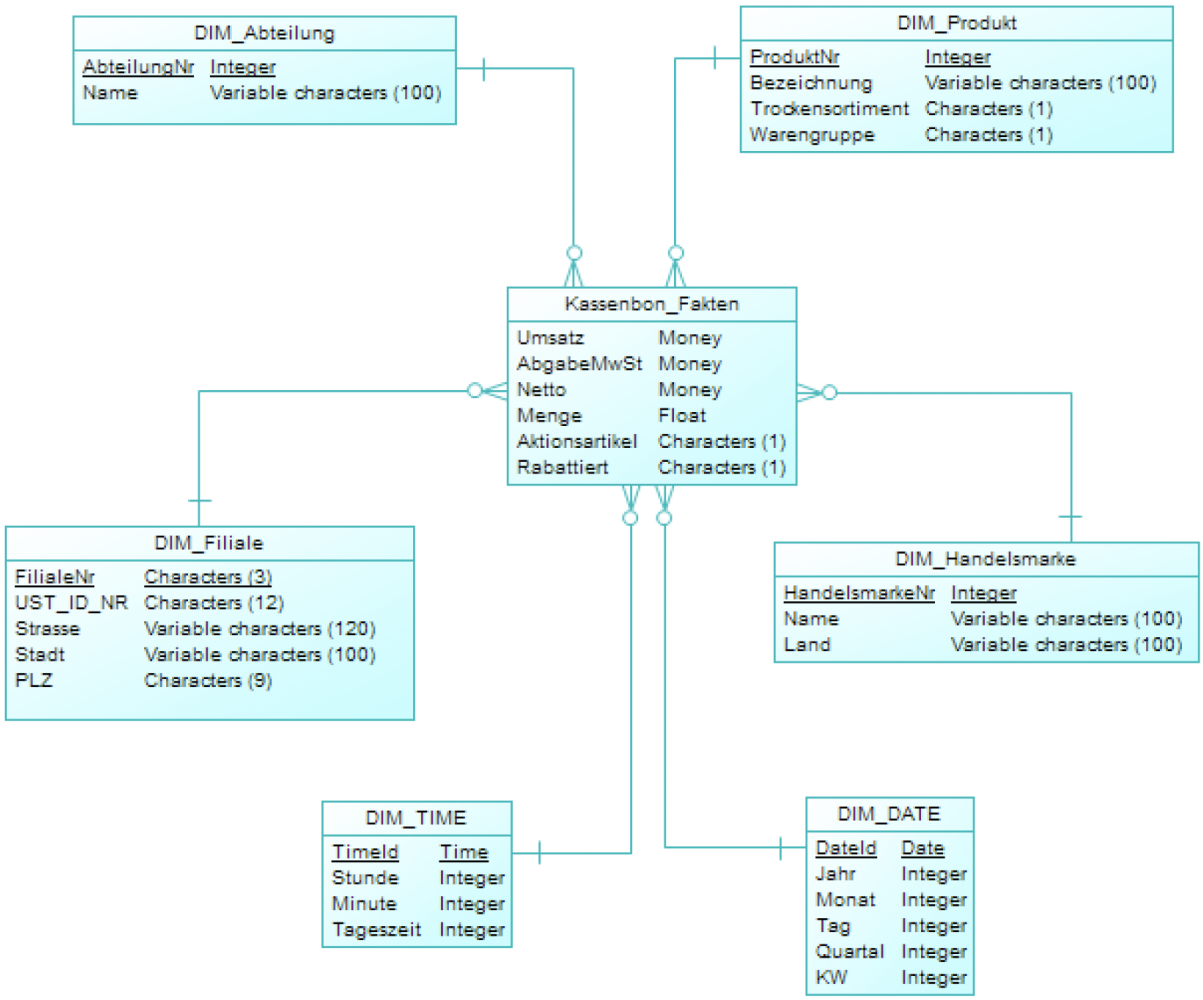

Abbildung 5: Konzeptionelles Datenmodell des Data-Warehouses (Quelle: DBIS-Aufgaben)

3.3.4. ETL-Prozess

Mit dem ETL-Prozess (Extract, Transform, Load) werden die Daten aus der operativen Datenbank für die Auswertung und Analyse aufbereitet und in das Data Warehouse geladen. Weil die Fremdschlüssel und Primärschlüssel der Faktentabelle

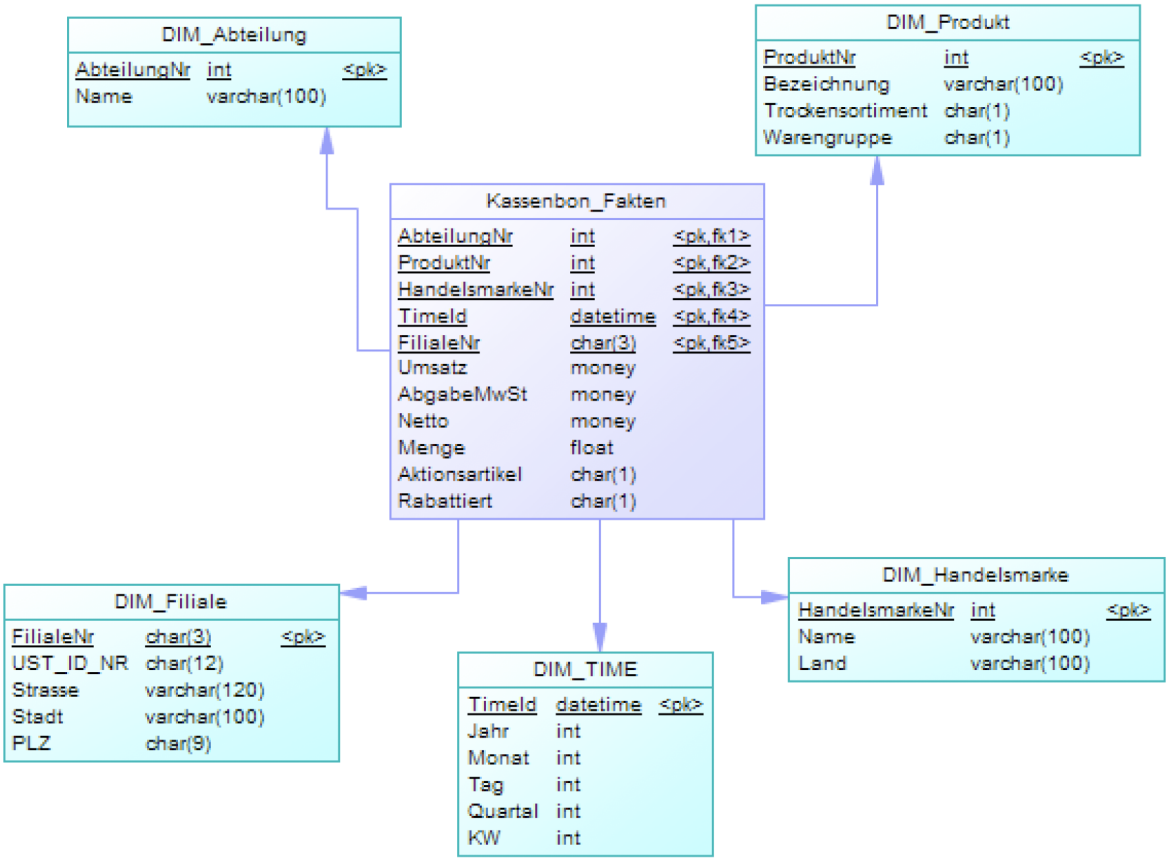

Abbildung 6: Physikaliches Datenmodell des Data-Warehouses (Quelle: DBIS-Aufgaben)

für Kassenbons aus den Primärschlüsseln der Dimensionstabellen bestehen (s. Abbildung 6), müssen zunächst Insert-Anweisungen für die Dimensionstabellen ausgeführt werden.

```
1   -- Dim-Tabellen mit Werten aus der ODB befullen --
2  --DIM_Abteilung--
3  insert into DIM_Abteilung(AbteilungNr,Name)
4  select FB.FachbereichNr,FB.Name
5  from Fachbereich as FB
6
7  --DIM_Produkt--
```

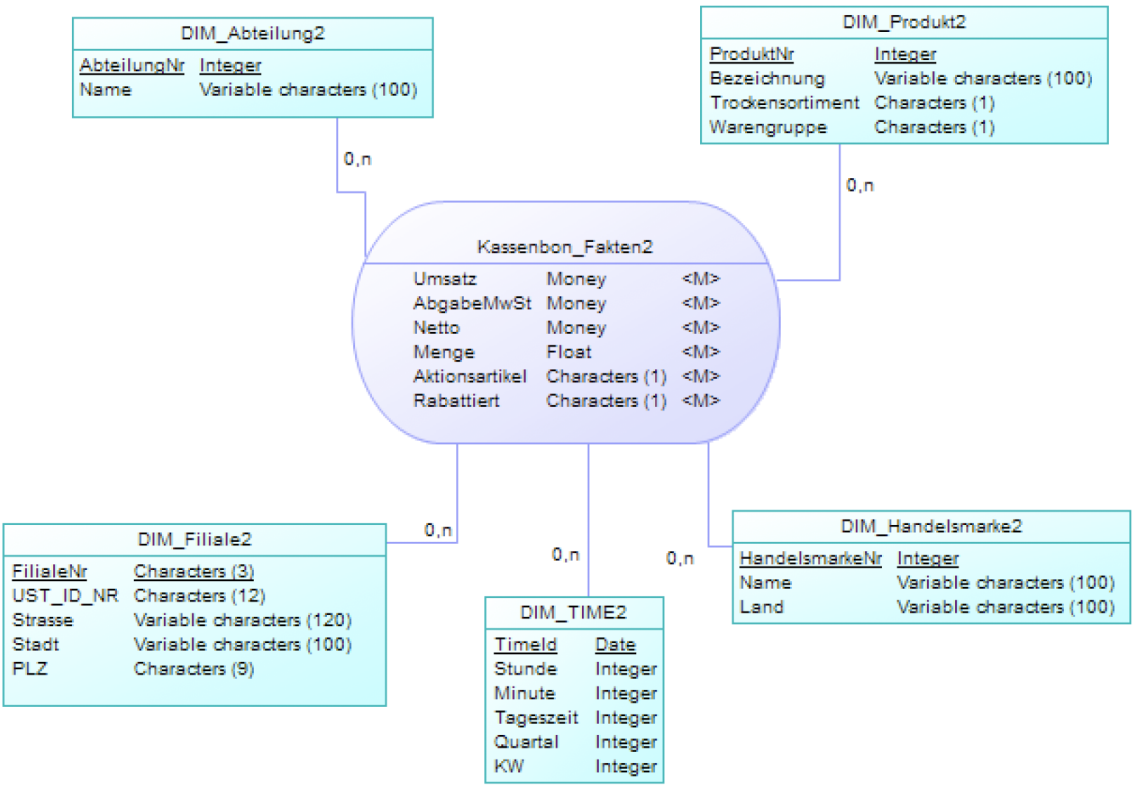

Abbildung 7: Datenmodell des DW mit Assoziierung (Quelle: DBIS-Aufgaben)

```
8  insert into
       DIM_PRODUKT(PRODUKTNR,BEZEICHNUNG,TROCKENSORTIMENT,WARENGRUPPE)
9  select P.ProduktNr, P.Bezeichnung, P.Trockensortiment,
       substring(P.Warengruppe,3,1)
10 from Produkt as P
11
12 --DIM_Filiale--
13 insert into DIM_FILIALE(FILIALENR, UST_ID_NR,STRASSE,STADT,PLZ)
14 select F.FILIALENR,F.UST_ID_NR, F.STRASSE,F.STADT, F.PLZ
15 from FILIALE as F
16
17 --DIM_Handelsmarke--
```

```
18 insert into DIM_HANDELSMARKE(HANDELSMARKENR,NAME, LAND)
19 select HM.HANDELSMARKEID, HM.NAME, HM.LAND
20 from HANDELSMARKE AS HM
```

Im Anschluss kann die Dimensionstabelle DIM_Date auf den Tag, die Woche, den Monat, das Quartal und das Jahrgenau gespeichert werden. Dafür wird das vom Anfangsdatum bis zum Endatum der Kassenbons eine Schleife durchlaufen. Die Dim_Time läuft auf ähnliche Weise, nur dass die Schleife einen ganzen Tag durchläufz und ihn in Zeitabschnitte unterteilt.

```
1   -DIM_Date Parameter fur die Berechnung der Attribute--
2 --Alle Tage von Anfang bis Ende ---
3 declare @vonDatum date,
4   @bisDatum date,
5   @Jahr integer, -- Attribut "Jahr"
6   @Monat integer, -- Attribut "Monat"
7   @Tag integer, -- Attribut "Tag"
8   @Quartal integer, -- Attribut "Quartal"
9   @KW integer -- Attribut "Kalenderwoche"
10
11  select @vonDatum=min(CONVERT(date,datum)) from KASSENBON
12  select @bisDatum=max(CONVERT(date,datum)) from KASSENBON
13
14 -- Zeitspanne: Schleife, bis Enddatum Erreicht wurde --
15
16  while @vonDatum <= @bisDatum
17   begin
18    set @Jahr  =year(@vonDatum)
19    set @Monat  =month(@vonDatum)
20    set @Tag  =day(@vonDatum)
21    set @Quartal =datepart(Quarter,@vonDatum)
22    set @KW   =datepart(WK,@vonDatum)
23
24    insert into DIM_DATE(DATEID, JAHR, MONAT, TAG, QUARTAL, KW)
```

```
25    values (@vonDatum, @jahr, @monat, @tag, @Quartal, @KW)
26
27   set @vonDatum = DATEADD(DD, 1,@vonDatum)
28  end
```

Zum Schluss findet der Insert der Kassenbonfakten durchgeführt werden. Dafür werden die Primärschlüssel der operativen DB gejoined. Mit dem Insert der Zeitdimensionen kann die Faktentabelle zum Beispiel das Verkaufsdatum auf das Produkt beziehehen oder Umsätze nach Produkt generieren. Zu beachten war außerdem, die Differenz zwischen der summierten Verkaufspreise und der Mehrwertsteuern zu erzeugen, um einen Nettobetrag zu erhalten.

```
1
2   --- Insert der Kassenbon_Fakten ---
3   INSERT INTO KASSENBON_FAKTEN
4  select fbr.FACHBEREICHNR, P.PRODUKTNR, hm.HANDELSMARKEID,
5  convert(date,DATEID), convert(time(0),TIMEID),
6     F.FILIALENR, sum(BP.Verkaufspreis), sum(BP.MWST),
7  sum(BP.Verkaufspreis) - sum(BP.MWST),
8  sum(BP.Verkaufsmenge),
9  bp.AKTIONSWARE,
10  bp.RABATTWARE
11
12  --- Gruppierung der Faktentabelle nach: -----
13  group by
14 fbr.FACHBEREICHNR, P.PRODUKTNR, hm.HANDELSMARKEID,
15 DD.DATEID, DT.TIMEID, F.FILIALENR,
16  BP.AKTIONSWARE, BP.RABATTWARE,BP.BONNR
```

3.3.5. Aggregate auf Wochenbasis

Um Aggregate auf Wochenbasis zu erhalten, wurde auf ähnliche Weise wie die Faktentabelle für Kassenbons eine Faktentabelle für Kassenbons auf Wochenbasis erstellt. Sie stellt dar, in welchen Wochen der meiste Umsatz mit welchem Produkt und welcher Menge erzielt worden ist.

	ABTEILUNGNR	PRODUKTNR	HANDELSMARKENR	JAHR	WOCHE	TIMEID	FILIALENR	Umsatz	ABGABEMWST	NETTO	MENGE	AKTIONSARTIKEL	RABATTIERT
1	1000	1	2	2019	1	08:45:00	1	6395	447.65	5947.35	500	N	N
2	1000	1	2	2019	1	09:00:00	1	6395	447.65	5947.35	500	N	N
3	1000	1	2	2019	1	09:12:00	1	6395	447.65	5947.35	500	N	N
4	1000	1	2	2019	1	08:05:00	1	1279	89.53	1189.47	100	N	N
5	1000	6	2	2019	1	08:21:00	1	124.9	8.743	116.157	10	N	Y
6	1014	10	22	2019	1	08:35:00	1	54	3.78	50.22	100	Y	Y
7	1014	11	22	2019	1	08:35:00	1	54	3.78	50.22	100	Y	Y
8	1014	12	22	2019	1	08:35:00	1	54	3.78	50.22	100	Y	Y
9	1014	13	22	2019	1	08:15:00	1	299	20.93	278.07	100	Y	N
10	1014	13	22	2019	1	08:35:00	1	104.5	7.315	97.185	50	Y	Y
11	1014	14	22	2019	1	08:15:00	1	299	20.93	278.07	100	Y	N
12	1014	15	22	2019	1	08:15:00	1	299	20.93	278.07	100	Y	N
13	1014	19	37	2019	1	08:45:00	1	665	46.55	618.45	1000	Y	Y
14	1014	19	37	2019	1	09:00:00	1	665	46.55	618.45	1000	Y	Y
15	1014	20	37	2019	1	08:45:00	1	745	52.15	692.85	1000	Y	Y
16	1014	20	37	2019	1	09:12:00	1	745	52.15	692.85	1000	Y	Y
17	1014	21	37	2019	1	08:45:00	1	745	52.15	692.85	1000	Y	Y
18	1014	21	37	2019	1	09:12:00	1	745	52.15	692.85	1000	Y	Y
19	1014	22	37	2019	1	08:15:00	1	233	16.31	216.69	1000	N	Y
20	1014	22	37	2019	1	08:21:00	1	116.5	8.155	108.345	50	N	N
21	1000	3	2	2019	6	08:22:00	14	174.3	12.201	162.099	70	Y	Y
22	1000	1	2	2019	6	08:22:00	14	89.53	6.2671	83.2629	7	N	N
23	1000	2	2	2019	6	08:22:00	14	83.93	5.8751	78.0549	7	N	N
24	1000	1	2	2019	7	08:22:00	14	89.53	6.2671	83.2629	7	N	N
25	1000	1	2	2019	7	10:22:00	14	89.53	6.2671	83.2629	7	N	N
26	1000	2	2	2019	7	10:22:00	14	16786	1175.02	15610.98	1400	Y	Y
27	1000	4	2	2019	7	10:22:00	14	146.3	10.241	136.059	70	Y	Y

Abbildung 8: Aggregate auf Wochenbasis

```
1   INSERT INTO KASSENBON_FAKTEN_WOCHEN
2  select
3  FB.FACHBEREICHNR, P.PRODUKTNR, H.HANDELSMARKEID,
4  DD.JAHR, DD.KW, convert(time(0),TIMEID),
5  F.FILIALENR, sum(BP.Verkaufspreis),
6  sum(BP.MWST), sum(BP.Verkaufspreis)-sum(BP.MWST),
7  sum(BP.Verkaufsmenge), BP.AKTIONSWARE, BP.RABATTWARE
8  from
9  FACHBEREICH as FB join Produkt as P
10   on FB.FACHBEREICHNR= P.FachbereichNr join
11 HANDELSMARKE as H
12   on P.HandelsmarkeId = H.HANDELSMARKEID
13  join Bonposition as BP on P.ProduktNr = BP.ProduktNr
```

```
14  join Kassenbon as KB on BP.BonNr = KB.BonNr
15  join FILIALE as F on KB.FilialeNr = F.FILIALENR
16   join Dim_Date as DD on DATEPART(yyyy,KB.Datum) = DD.JAHR and
17   DATEPART(wk,KB.DATUM) = DD.KW
18  join Dim_Time as DT on convert(time(0),KB.Datum) = DT.TIMEID
19  group by
20  FB.FACHBEREICHNR,
21  P.PRODUKTNR,
22  H.HANDELSMARKEID,
23  DD.JAHR,
24  DD.KW,
25     DT.TIMEID,
26  F.FILIALENR,
27  BP.AKTIONSWARE,
28  BP.RABATTWARE
```

3.4. Zugänglichkeit für die Öffentlichkeit

Mit dem folgenden Skript können auch weitere DB-Teilnehmer auf die Datenbanktabellen zugreifen, indem die Tabellen auf Public gesetzt werden.

```
1   - Datentabellen/DB freigeben --
2
3   GRANT CONNECT to public
4
5
6   GRANT SELECT ON dbo.FILIALE to public
7   GRANT SELECT ON dbo.KASSENBON to public
8   GRANT SELECT ON dbo.BONPOSITION to public
9   GRANT SELECT ON dbo.PRODUKT to public
10  GRANT SELECT ON dbo.FACHBEREICH to public
11  GRANT SELECT ON dbo.HANDELSMARKE to public
12
13  GRANT SELECT ON dbo.DIM_TIME to public
```

```
14  GRANT SELECT ON dbo.DIM_DATE to public
15  GRANT SELECT ON dbo.DIM_ABTEILUNG to public
16  GRANT SELECT ON dbo.DIM_PRODUKT to public
17  GRANT SELECT ON dbo.DIM_FILIALE to public
18  GRANT SELECT ON dbo.DIM_HANDELSMARKE to public
19  GRANT SELECT ON dbo.KASSENBON_FAKTEN to public
20  GRANT SELECT ON dbo.KASSENBON_FAKTEN_WOCHEN to public
21  GRANT SELECT ON dbo.DW_REPORT_TABLE to public
22
23  GRANT EXECUTE ON dbo.DW_REPORT to public
```

3.5. Prozedur für das Reporting

Um die Daten aus dem Data Warehouse in der Konsole eines SQL-Editors auszuwerten, kann eine Prozedur entwickelt werden. Prozeduren mit der Prozesdursprache TSQL verhalten sich ähnlich wie Methodenaufrufe in Java. Mit einem **execute**-Befehl und der Übergabe von Parametern wird die Prozedur **DW-Report** aufgerufen. Da eine Prozedur zu der Umsatzanalyse jeden Zeitraums pro Produkt ermöglichen soll, wird der Zeitraum variabel als Parameter übergeben. Da eine Anforderung aus dem Pflichtenheft die hohe Benutzerfreundlichkeit ist, muss die Ausgabe in der Konsole eine übersichtliche Darstellung ermöglichen. Für diesen Zweck wird ein Cursor eingesetzt, welcher die Select-Anweisungen aus der Hilfstabelle **DW_Report_Table** in eine einfache Tabellenstruktur positioniert. Das Ergebnis für einen Beispielaufruf wird Abbildung 9 dargestellt.

```
1  --- Beispielaufruf ----
2  declare @produktnummer int, @von date, @bis date
3  execute DW_REPORT 15,'2019-02-01','2019-08-01'
```

ProduktNr	Menge in St.	Datum	Umsatz in €
15	10	2019-02-11	2.99
15	10	2019-02-16	29.9
15	1	2019-02-17	2.99
15	1	2019-02-20	2.99
15	1	2019-02-22	2.99
15	1	2019-02-23	2.99
15	10	2019-02-25	29.9
15	10	2019-03-10	29.9
15	1	2019-04-11	12.79
15	1	2019-04-22	2.99
15	10	2019-05-22	29.9
15	10	2019-05-25	29.9
15	10	2019-06-13	29.9
15	200	2019-06-17	133
15	2	2019-06-25	24.98
		Gesamtumsatz (€) :	368.11

Abbildung 9: Reporting-Tabelle

Die Tabelle zeigt die Umsätze und die verkauften Mengen für die Produktnummer 15 in einem Zeitraum von 6 Monaten. Zunächst wird überprüft, ob die angegebenen Zeiten einen Zeitraum darstellen. Wenn die Differenz nicht negativ ist, beginnt der Insert der Daten aus der Kassenbon_Faktentabelle in die Hilfstabelle. Der Cursor KB_Cursor durchläuft jeden Tag mithilfe der Inkrementierung des Cursor-Status um 1. Wenn alle keine weiteren Verkäufe mehr für den Zeitraum und das Produkt gefunden werden, wird die Schleife beendet. Anschließend wird der Gesamtumsatz berechnet und der Cursor wird beendet.

3.6. Reporting mit QlikSense

Neben der Auswertung mit Prozeduren kann eine Datenauswertung mit der BI-Sofwtware QlikSense durchgeführt werden. Der Vorteil ist die bessere Visualisierung in anschaulichen Diagrammen.

3.6.1. Produkte nach Umsätzen

Im folgenden Kreisdiagramm wird dargestellt, welche Produkte insgesamt die umsatzstärksten sind. Es wurde nach der Dimension Produktbezeichnung und der Kennzahl sum(Umsatz) erstellt. Die Geschäftsführer können somit die Erkenntniss gewinnen, dass der Bacardi Carta Blanca z.B. länger im Sortiment behalten werden kann. Dagegen kann aus Abbildung 11 entnommen werden, welche Produkte z.B. aus dem Sortiment genommen, besser beworben oder besser platziert müssen.

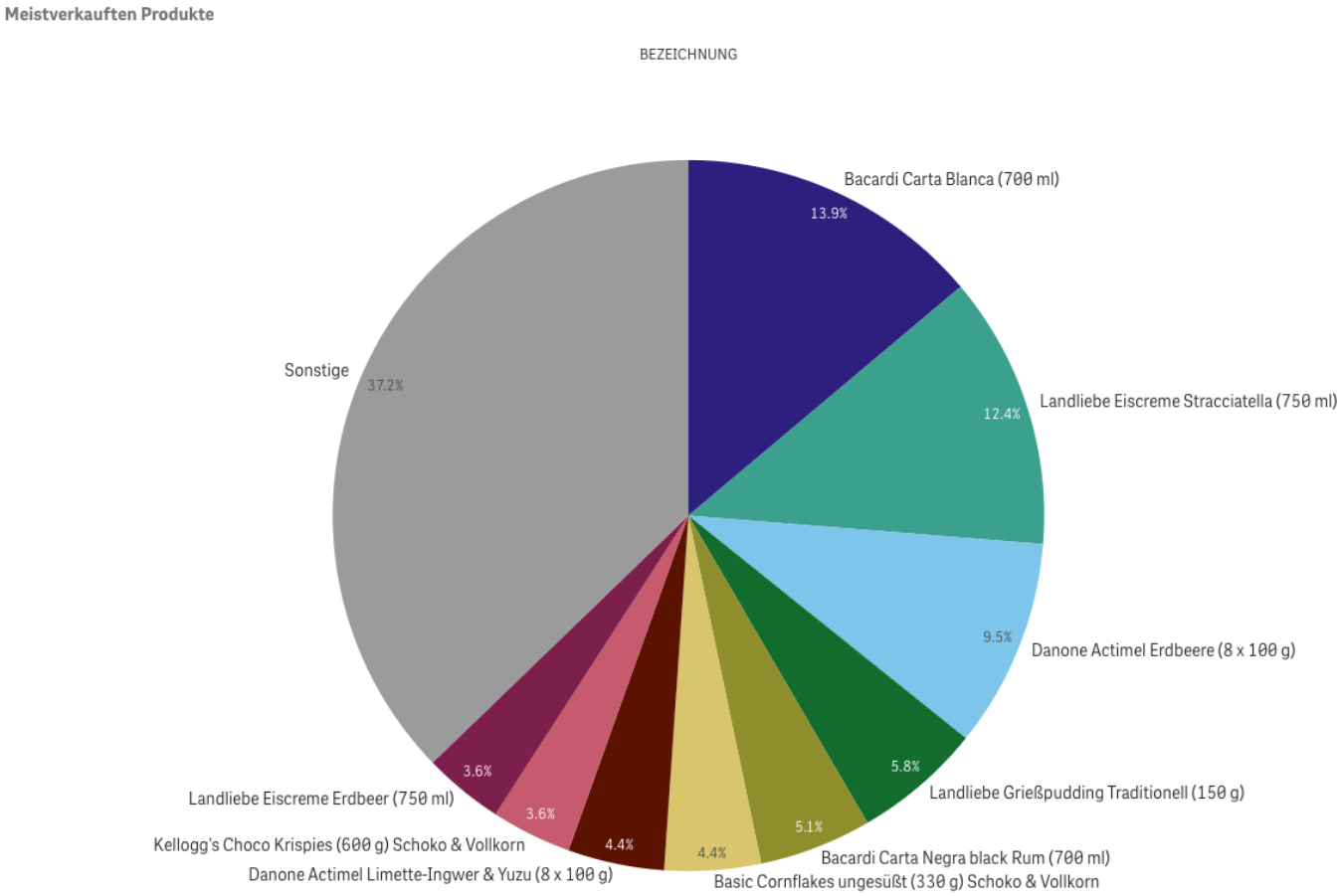

Abbildung 10: Umsatzstärksten Produkte

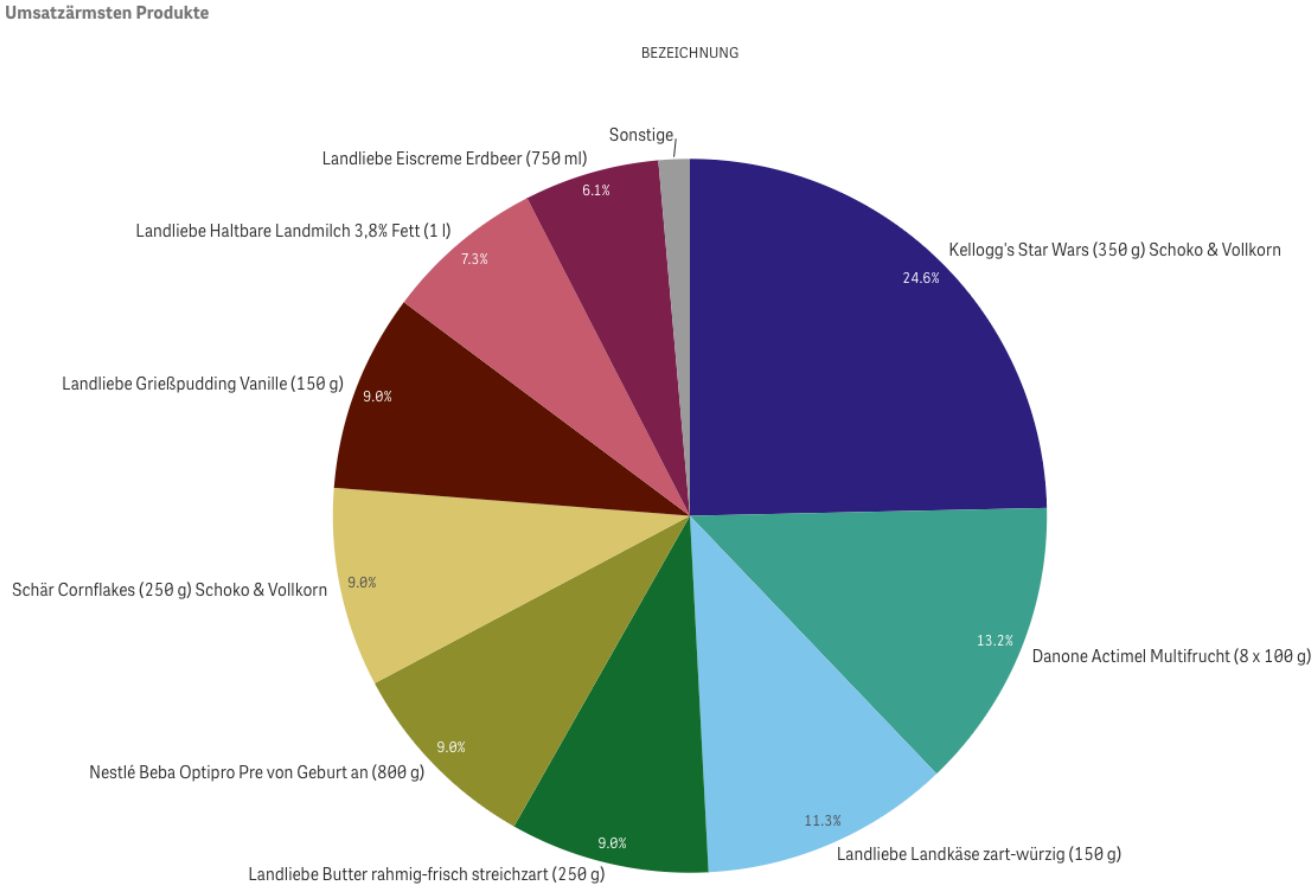

Abbildung 11: Umsatzärmsten Produkte

3.6.2. Erfolgreichsten Handelsmarken

Diese Analyse zeigt die größten Umsätze nach Handelsmarken. Wie aus der Produktanalyse ersichtlich ist, ist Bacardi eine sehr erfolgreiche Marke im Verkauf. Auch die Marken Danone und Landliebe haben großes Potenzial. Daraus kann geschlossen werden, dass die Kunden großes Interesse an den Produkten haben können, wenn z.B. bessere Werbe- oder Platzierungsmaßnahmen ergriffen werden, aber auch Produktvariationen erstellt werden.

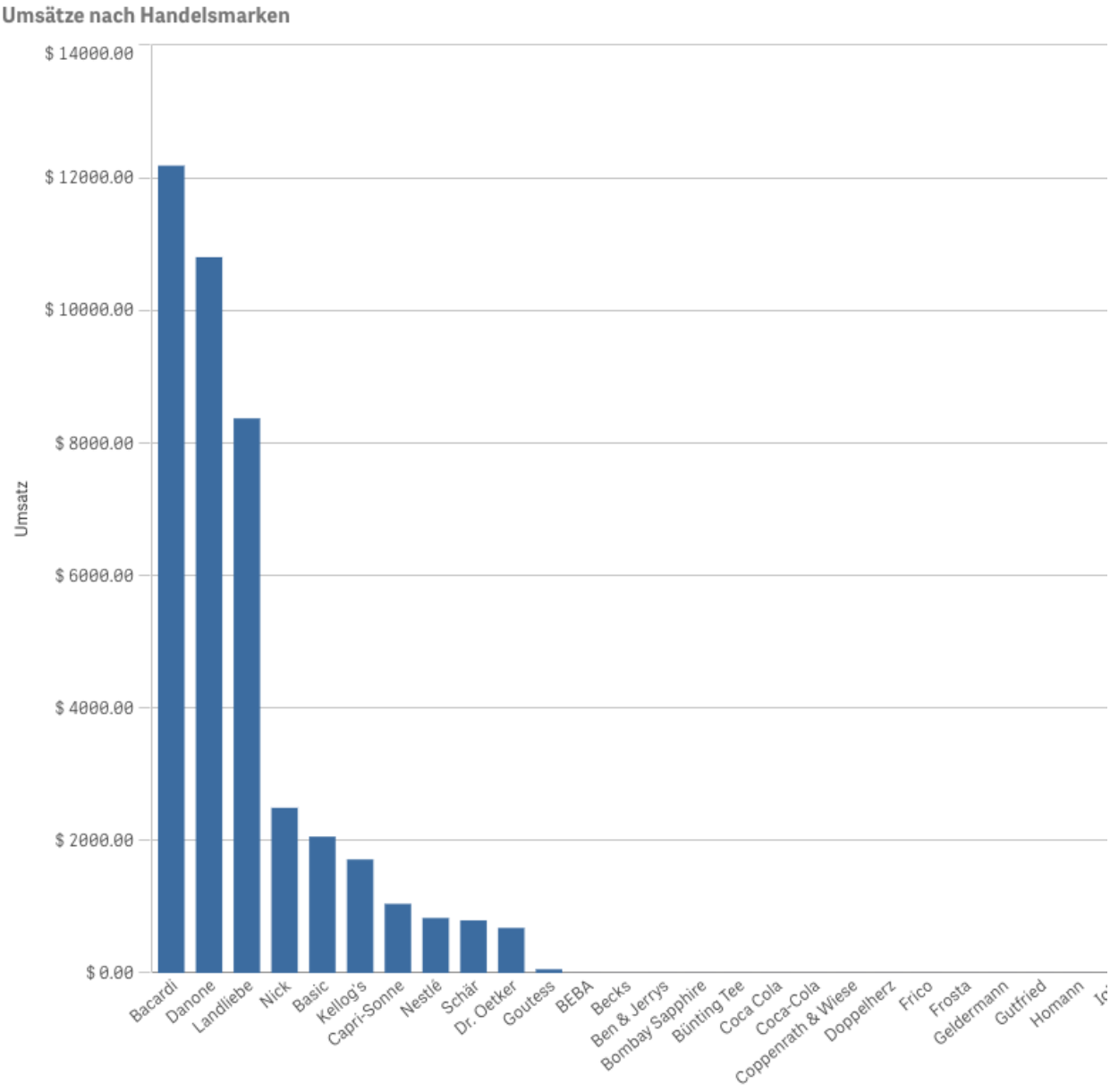

Abbildung 12: Erfolgreichsten Handelsmarken

3.6.3. Auswertung nach Kalenderwochen

Dieses Diagramm berücksichtigt die Dimension Kalenderwoche nach den durschnittlich verkauften Mengen und entsprechenden Umsätzen. Daraus kann man entneh-

men, dass die Kalenderwoche 7 die erfolgreichste war. Die Geschäftsführung könnte sich Gedanken darum machen, welche Ereignisse diesen Erfolg begründen könnten.

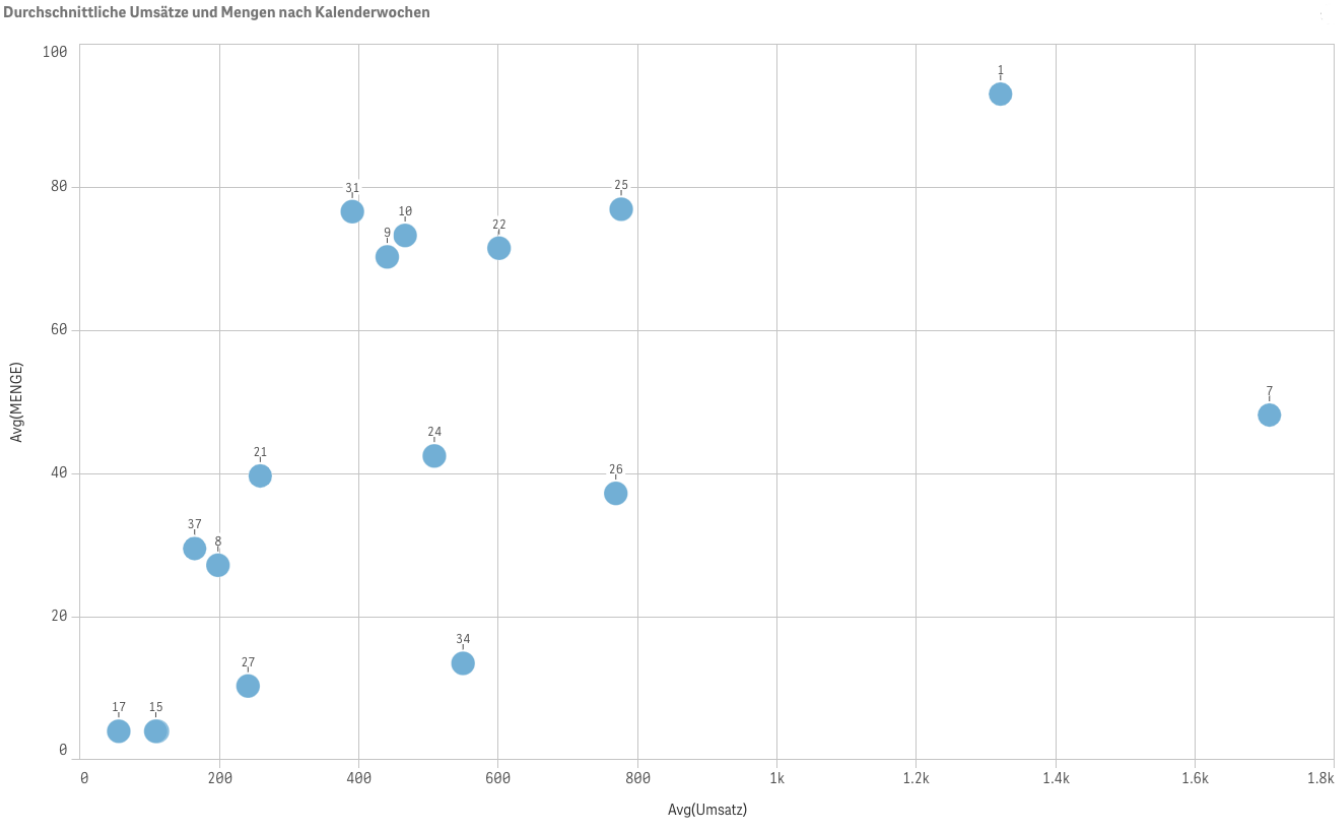

Abbildung 13: Umsätze und Mengen nach Kalenderwochen

Literatur

Wulff, A. (2019). Datenbankbasierte informationssysteme. Abgerufen 16.12.2019, von https://moodle.jade-hs.de/moodle/course/view.php?id=5151.

A. Quelltext für das Java Programm

A.1. ImportWindow

```
1
2   public class ImportWindow implements ActionListener{
3
4  // Datenbankverbindung mit JDBC Data Source
5
6  private DBConnection dbcon;
7  private Connection con;
8  private Statement statement;
9  //LoginFenster
10 public JFrame frmLoginWindow;
11 //Logger
12 private static final Logger LOG = Logger.getGlobal();
13
14 //Labels for Connection Screen
15 private JLabel userL;
16 private JLabel passwordL;
17 private JLabel databaseL;
18 private JLabel serverL;
19 private JLabel portL;
20 private JLabel fileL;
21 //Inputs in Connection Screen
22 private JTextField userF;
23 private JPasswordField passwordF;
24 private JTextField databaseF;
25 private JTextField fileF;
26 private JTextField serverF;
27 private JTextField portF;
28 //Buttons for Screens
29 private JButton btnImport;
```

```
30  private JButton btnFile;
31  private JButton btnCancel;
32  private JButton btnProtocol;
33  //Protocol
34  private String protocol_content;
35
36  //Constructor for starting Application Window
37  public ImportWindow() {
38   init();
39  }
40
41  public void init() {
42   frmLoginWindow = new JFrame("Select CSV data for Product Import");
43   frmLoginWindow.setBounds(100,100,550,300);
44   frmLoginWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
45   frmLoginWindow.setLocationRelativeTo(null);
46   frmLoginWindow.setResizable(false);
47
48   userL = new JLabel("User");
49   passwordL = new JLabel("Password");
50   databaseL = new JLabel("Database");
51   serverL = new JLabel("Server String");
52   portL = new JLabel("Port No.");
53   fileL = new JLabel("CSV-File");
54
55   //Buttons
56   btnImport = new JButton("Import");
57   btnImport.addActionListener(this);
58   btnFile = new JButton("Browse File");
59   btnFile.addActionListener(this);
60   btnCancel = new JButton("Cancel");
61   btnCancel.addActionListener(this);
62   btnProtocol = new JButton("Protocol");
```

```
63    btnProtocol.addActionListener(this);
64
65    // Input Fields
66
67    userF = new JTextField();
68    userF.setText("DB6013868");
69
70    passwordF = new JPasswordField();
71    passwordF.setText("3utbve");
72
73    databaseF = new JTextField();
74    databaseF.setText("DB6013868");
75
76    serverF = new JTextField();
77    serverF.setText("whv-fbmit3.hs-woe.de");
78
79    portF= new JTextField();
80    portF.setText("1433");
81
82    fileF = new JTextField();
83    fileF.setText(System.getProperty("user.dir")
84      + System.getProperty("file.separator") + "imp"
85      + System.getProperty("file.separator") + "PRODUKTE1.csv");
86
87    //JPanel for Labels
88
89    JPanel labelPane = new JPanel(new GridLayout(6, 1));
90    labelPane.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
91
92    labelPane.add(userL);
93    labelPane.add(passwordL);
94    labelPane.add(databaseL);
95    labelPane.add(serverL);
```

```
96   labelPane.add(portL);
97   labelPane.add(fileL);
98
99   // JPanel for Inputs
100
101  JPanel fieldPane = new JPanel(new GridLayout(6, 1));
102  fieldPane.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
103  fieldPane.add(userF);
104  fieldPane.add(passwordF);
105  fieldPane.add(databaseF);
106  fieldPane.add(serverF);
107  fieldPane.add(portF);
108  fieldPane.add(fileF);
109
110  // JPanel for Buttons
111
112  JPanel btnPane = new JPanel(new GridLayout(1,4));
113  btnPane.add(btnFile);
114  btnPane.add(btnImport);
115  btnPane.add(btnProtocol);
116  btnPane.add(btnCancel);
117
118  //Container
119  Container container = frmLoginWindow.getContentPane();
120  container.setBackground(Color.lightGray);
121
122  //Button Default
123  frmLoginWindow.getRootPane().setDefaultButton(btnImport);
124
125  //place panels
126  container.add(labelPane, BorderLayout.CENTER);
127  container.add(fieldPane, BorderLayout.LINE_END);
128  container.add(btnPane, BorderLayout.SOUTH);
```

```
129
130  frmLoginWindow.setVisible(true);
131 }
132
133 @Override
134 public void actionPerformed(ActionEvent e) {
135
136  if(e.getSource() == btnFile){
137   final JFileChooser fc = new JFileChooser();
138   int returnVal = fc.showOpenDialog(fc);
139
140   if (returnVal == JFileChooser.APPROVE_OPTION) {
141    File file = fc.getSelectedFile();
142    //This is where a real application would open the file.
143    fileF.setText(file.getAbsolutePath());
144    LOG.info("File selected with success");
145   }
146   System.out.println("Button geklickt!");
147  }
148  else if(e.getSource() == btnCancel) {
149   System.exit(0);
150   System.out.println("Canceled");
151   LOG.info("Canceled");
152  }
153  else if(e.getSource() == btnProtocol) {
154
155   JTextArea outputArea = new JTextArea(20, 40);
156   // Header and append
157   String header = getHeader();
158   String foot = getFoot();
159   protocol_content = header + protocol_content + foot;
160   outputArea.setText(protocol_content);
161   outputArea.setEditable(false);
```

```
162     JScrollPane scrollPane = new JScrollPane(outputArea);
163
164     JOptionPane.showMessageDialog(new JFrame(), scrollPane, "Protokoll",
165       JOptionPane.INFORMATION_MESSAGE);
166   }
167   // DB Import CSV
168   else if(e.getSource() == btnImport) {
169
170    LOG.info("Import clicked..");
171
172    try {
173     LOG.info("Connecting...");
174     dbcon = new DBConnection(serverF.getText(),
175       portF.getText(), databaseF.getText(),
176       userF.getText(), new String(
177         passwordF.getPassword()));
178     con = dbcon.getConnection();
179
180     // Testen der Verbindung
181
182     boolean isOk = dbcon.testConnection();
183
184     //Test if connection is established --> true!!
185     System.out.println("Connection established"+ isOk);
186     protocol_content+= "Connection established to" + "
          "+serverF.getText();
187
188
189     if (!isOk) {
190      JOptionPane.showMessageDialog(new JFrame(),
191       "Die Anmeldung konnte nicht durchgefuhrt werden!"
192         + "\n\nBitte uberprufen Sie Ihre Angaben!",
193         "Anmeldung fehlgeschlagen!", JOptionPane.ERROR_MESSAGE);
```

```
194     } else {
195
196      // Das Hauptfenster fur die Systemverwaltung wird geoffnet.
197      ImportRoutine dbisImport = new ImportRoutine(con, fileF
198        .getText());
199      try {
200       // Daten importieren
201       protocol_content = dbisImport.startImport();
202
203       if (dbisImport.isImportOk())
204        JOptionPane.showMessageDialog(new JFrame(),
205          "Datenimport wurde erfolgreich durchgefuhrt!", "Datenimport",
206          JOptionPane.INFORMATION_MESSAGE);
207       else
208        JOptionPane.showMessageDialog(new JFrame(),
209          "Datenimport wurde abgebrochen!",
210          "Datenimport fehlgeschlagen!", JOptionPane.ERROR_MESSAGE);
211      } catch (IOException e1) {
212       JOptionPane
213       .showMessageDialog(
214         new JFrame(),
215         "Die Datei konnte nicht gefunden oder nicht gelesen werden!"
216           + "\n\nBitte uberprufen Sie Pfad, Name und Rechte der
                Importdatei!",
217          "Import fehlgeschlagen!", JOptionPane.ERROR_MESSAGE);
218       e1.printStackTrace();
219      }
220      //****************!!!**************
221      dbcon.freeConnection();
222     }
223
224    } catch (SQLException e1) {
225     LOG.log(Level.SEVERE, "Fehler im Datensatz.", e1);
```

```
226     JOptionPane.showMessageDialog(new JFrame(),
227       "Keine Verbindung zur Datenbank!\n" +
228         "Bitte Uberprufen Sie Ihre Angaben!",
229         "Datenimport", JOptionPane.ERROR_MESSAGE);
230    }
231   }
232
233  }
234  private String getHeader() {
235   return "************************************\n" + "Start: "
236     + Date_Format.getDdMMyyyyHHMi(System.currentTimeMillis())
237     + " Uhr \n" + "Benutzer: " + userF.getText() + "\n" + "Database: "
238     + databaseF.getText() + "\n"
239     + "************************************\n\n";
240  }
241  private String getFoot() {
242   return "\n\n************************************\n" + "End: "
243     + Date_Format.getDdMMyyyyHHMi(System.currentTimeMillis())
244     + " Uhr \n" + "************************************\n\n";
245  }
246
247
248 }
```

A.2. DBConnection

```
1
2   public class DBConnection {
3
4  // Establish Connection via DataSource --> "mittlerweile bevorzugt"
5  // i-net MERLIA.jar is JDBC 3.0 / 4.0 driver for MS SQL Server
6  // Quelle 6. Dezember 2019:
       https://www.inetsoftware.de/products/jdbc-driver/ms-sql/features
```

```
7  // Class.forName("com.inet.tds.TdsDriver") - MS-SQL-Server
8  // This class is an implementation of a simple javax.sql.DataSource
       for the driver i-net ...
9  // ... OPTA-xs and MERLIA-xs. For application servers you should use
       the PDataSource or DTCDataSource.
10 // Quelle 6. Dezember TDSDataSource:
       https://www.inetsoftware.de/documentation/jdbc-driver/ms-sql/apispec/index.html?c
11 // public TdsDataSource () Methods:
12 // getConnection(username, password) throws java.sql.SQLException
       returns a Connection to database
13 // setUrl(java.lang.String.jdbcUrl) and getUrl()
14 // setServerName(String serverName)
15 // getInstanceName()
16 // setDatabaseName(String databaseName) etc ....
17 // setPort(String port) --> cast from Textfield not necessary, if not
       set, default ist 1433
18 // setUser and getUser as well as getPassword set Password both String
19
20
21 private final String serverName;
22 private final String port;
23 private final String databaseName;
24 private final String username;
25 private final String password;
26 private Connection con;
27 private static final Logger log = Logger.getGlobal();
28
29 // Constructor for DB-Connection-Object
30 public DBConnection(String serverName, String port, String
       databaseName, String username, String password) {
31
32  this.serverName = serverName;
33  this.port = port;
```

```
34   this.databaseName = databaseName;
35   this.username = username;
36   this.password = password;
37  }
38
39  public synchronized Connection getConnection() throws SQLException{
40
41   // Connection with DataSource
42
43   try {
44    if (con == null || con.isClosed()) {
45     DataSource ds = new TdsDataSource();
46     ((TdsDataSource) ds).setServerName(serverName);
47     ((TdsDataSource) ds).setPortNumber(Integer.parseInt(port));
48     con = ds.getConnection(username, password);
49     con.setCatalog(databaseName);
50    }
51   } catch (SQLException e) {
52    log.log(Level.WARNING, "DB Connection failed.", e);
53    JOptionPane.showMessageDialog(new JFrame(),
54      " DB Connection failed!\n" +
55       "Check your connection input",
56       "CSV Import", JOptionPane.ERROR_MESSAGE);
57   }
58   return con;
59
60  }
61  public synchronized void freeConnection() {
62   try {
63    System.out.println("Active DB-Verbindung wird geschlossen... ");
64    con.close();
65    con = null;
66   } catch (Exception e) {
```

```
67
68   }
69  }
70
71  public synchronized boolean testConnection() {
72   try {
73    // Connection worked!
74    log.info("Testing Connection in DBConnectionDataSource");
75    if (getConnection() != null)
76     return true;
77   } catch (Exception e) {
78    e.printStackTrace();
79   }
80   return false;
81
82  }
83  public String getSchemaname() {
84      return "dbo";
85    }
86
87
88 }
```

A.3. FileLine

```
1   public class FileLine {
2
3  private int produktNummer;
4  private String fachbereichNr;
5  private int handelsmarkeId;
6  private String bezeichnung;
7  private String trockensortiment;
8  private String warengruppe;
```

```
9  private double preis;
10
11 // Constructor for FileLine
12
13 public FileLine(
14   String fachbereichNr,
15   int handelsmarkeId,
16   String bezeichnung,
17   String trockensortiment,
18   String warengruppe,
19   double preis
20   ) {
21  this.fachbereichNr = fachbereichNr;
22  this.handelsmarkeId = handelsmarkeId;
23  this.bezeichnung = bezeichnung;
24  this.trockensortiment = trockensortiment;
25  this.warengruppe = warengruppe;
26  this.preis = preis;
27 }
28
29 public int getProduktNummer() {
30  return produktNummer;
31 }
32
33 public String getFachbereichNr() {
34  return fachbereichNr;
35 }
36
37 public int getHandelsmarkeId() {
38  return handelsmarkeId;
39 }
40
41 public String getBezeichnung() {
```

```
42   return bezeichnung;
43  }
44
45  public String getTrockensortiment() {
46   return trockensortiment;
47  }
48
49  public String getWarengruppe() {
50   return warengruppe;
51  }
52
53  public double getPreis() {
54   return preis;
55  }
56
57
58 }
```

A.4. FileLineParser

```
1
2   public class FileLineParser {
3
4  private static final int SPALTENANZAHL = 4;
5
6  // Erzeugt aus der ubergebenen Zeile ein Object
7
8  protected static FileLine getExpFileLine(String line)
9    throws NoSuchElementException, SQLException {
10
11   String bezeichnung = getValue(line, 1).trim();
12   String trockensortiment = getValue(line, 2).trim();
13   String warengruppe = getValue(line, 3).trim();
```

```
14   String preis = getValue(line, 4).trim();
15   String  fb_id;
16
17   //Versuch: Handelsmarke und Fachbereich integrieren
18   //Hardcoded Idee fur Fachbereich: Methode getFachbereich wie
        getWarengruppe
19   //--> Fachbereichsnummer ist der Foreign Key
20
21   // Gewuerrze sind nicht als Fachbereich vorhanden
22
23   //Schwieriger: Handelsmarke -> Bezeichnung nochmal Tokenizen (1.
        Stelle oder mit Regex wegen Dr Oetker und Kellogs)
24   //Muss auch ein insert fur Handelsmarke geben, wenn nicht vorhanden
25   // Neue Handelsmarken: Weber, Kellog's, Nestle, Basic, Schaer. Nick
26
27
28
29   if(bezeichnung.length() >500) {
30    bezeichnung.substring(0, 500);
31   }
32   if(trockensortiment.length() >1) {
33    trockensortiment.substring(0, 1);
34   }
35
36
37
38   warengruppe = FileLineParser.getWarengruppe(warengruppe);
39   fb_id = FileLineParser.getFachbereichsNummer(warengruppe);
40
41   if(warengruppe.length() >3) {
42    warengruppe.substring(0,3);
43   }
44   FileLine fileLineObj = null;
```

```
45   try {
46
47    fileLineObj = new FileLine(fb_id,
          9,bezeichnung,trockensortiment,warengruppe,
          Float.parseFloat(preis));
48   } catch (Exception e) {
49    System.out
50      .println("Parsefehler: Bitte uberprufen Sie das Format in dieser
            Zeile:"
51       + line);
52    throw new NoSuchElementException();
53   }
54   return fileLineObj;
55  }
56
57  /**
58   * Liefert einen Wert, fur den entsprechenden Index
59   */
60
61  private static String getValue(String line, int keyIndex)
62    throws NoSuchElementException {
63   String value = null;
64   boolean isDub = false;
65
66   StringTokenizer lineValues = new StringTokenizer(line, ";");
67   int tokens = lineValues.countTokens();
68
69   // Uberprufung der Spaltenanzahl
70   if (tokens < SPALTENANZAHL) {
71    // Falls mehr Tokens in Zeile als notwendige Spaltenzahl, Throw
          NoSuchElementException
72    System.out.println("Falsches Format von Testdaten!"
73      + "Zu wenige Parameter gefunden: "
```

```
74      + tokens + " von " + SPALTENANZAHL + "Spalten.");
75     throw new NoSuchElementException("Falsches Format der Daten!\n");
76    } else if (tokens > SPALTENANZAHL) {
77     System.out.println("Falsches Format von Testdaten!"
78      + " Zu viele Parameter gefunden: " + tokens + " von "
79      + SPALTENANZAHL + " Spalten.");
80     throw new NoSuchElementException("Falsches Format der Daten!\n");
81    }
82
83    for (int i = 0; i < keyIndex; i++) {
84     value = lineValues.nextToken();
85    }
86
87    // Checkt, ob der Wert redundante Werte hat
88
89    isDub = isDub(value);
90    if(isDub == true) {
91     System.out.println("Falsches Format von Testdaten!"
92      + "Redundante Daten");
93     throw new NoSuchElementException("Falsches Format der Daten!\n");
94    }
95
96    return value;
97   }
98
99   // Ubergabe der Warengruppennummer
100
101  public static String getWarengruppe(String s_warengruppe) {
102
103   if (s_warengruppe == null)
104    return null;
105
106   String warengruppe_id = null;
```

```
107
108   if (s_warengruppe.equalsIgnoreCase("Milch"))
109    warengruppe_id = "002";
110   else if (s_warengruppe.equalsIgnoreCase("Musli & Cerealien"))
111    warengruppe_id = "003";
112   else if (s_warengruppe.equalsIgnoreCase("Saucen"))
113    warengruppe_id = "004";
114   else if (s_warengruppe.equalsIgnoreCase("Gewurze"))
115    warengruppe_id = "005";
116   else
117    warengruppe_id = "006";
118
119   return warengruppe_id;
120  }
121
122  // Ubergabe der Fachbereichsnummer
123  // wird auch Fachbereich fuer Gewurze erstellen
124
125  public static String getFachbereichsNummer(String warengruppe_id)
         throws SQLException {
126
127   if (warengruppe_id==null) {
128    return null;
129   }
130   String fb_id= null;
131   if (warengruppe_id=="002") {
132    fb_id= "1014";
133   }else if(warengruppe_id == "003") {
134    fb_id= "1023";
135   }else if(warengruppe_id == "004") {
136    fb_id= "1015";
137   }else if(warengruppe_id == "005") {
138    ImportRoutine.createNewFB("Gewurze", "1039");
```

```
139     fb_id="1039";
140    }
141
142    return fb_id;
143   }
144
145   // Methode zum Prufen von Redundanten Eintragen
146
147   public static boolean isDub(String bezeichnung) {
148
149    String[] words = bezeichnung.split("[\\s+]");
150
151    Map<String, Integer> occurrences = new HashMap<String, Integer>();
152
153    boolean isdub =false;
154    Integer oldCount=0;
155    for ( String word : words ) {
156     oldCount = occurrences.get(word);
157     if ( oldCount == null ) {
158      oldCount = 0;
159     }
160     occurrences.put(word, oldCount + 1);
161     if(oldCount>2) isdub = true;
162     System.out.println("oldcount : "+ oldCount+ "word"+ word + "isdub"
          + isdub);
163    }
164    System.out.println("IS DUB? :"+ isdub);
165    return isdub;
166   }
167
168  }
```

A.5. ImportRoutine

```
1
2   public class ImportRoutine {
3
4     //Schnittstelle zum Importieren von Daten
5
6       protected BufferedReader in;
7       protected final int DOPPELTER_PRIMARY_KEY_FEHLER = 2601;
8       protected final int DOPPELTER_UNIQUE_SCHLUSSEL_FEHLER = 2627;
9       protected String fileIn;
10      protected static Connection con;
11      private boolean istImportOk = true;
12
13      public ImportRoutine(Connection con, String fileIn) {
14        this.con = con;
15        this.fileIn = fileIn;
16      }
17
18      /**
19       * Liest Datei aus und speichert sie in die Datenbank.
20       */
21      public String startImport() throws IOException {
22        String protocol = "";
23        if (con == null) {
24          System.out
25              .println("Datenbankverbindung muss initialisiert
                    werden!");
26        }
27        try {
28          try {
29            // Buffer fur Zeilen
30          in = new BufferedReader(new FileReader(fileIn));
31          }
32          catch (Exception e) {e.printStackTrace();}
```

```
33
34          String line = null;
35          int lineCounter = 0;
36          int objCounter = 0;
37
38          protocol += "Datenimport wird gestartet...\n\n";
39
40          // Abbruch des Exports bei falschen Zeilen
41
42          try {
43            line = in.readLine();
44          }
45          catch (IOException e1) {
46            // TODO Auto-generated catch block
47            e1.printStackTrace();
48          }
49
50          while (line != null && istImportOk) {
51            lineCounter++;
52
53            // Konmentare ausschliessen
54            if (!line.trim().equals("") && !line.substring(0,
                1).equals("#")) {
55              try {
56                // Zeile ueberspringen
57                objCounter++;
58                FileLine fileLine = FileLineParser.getExpFileLine(line);
59                DBImport.insert(fileLine, con);
60
61              } catch (SQLException e) {
62
63                // Fehler ignorieren, wenn Zeile existiert
64
```

```
65                if (e.getErrorCode() != DOPPELTER_PRIMARY_KEY_FEHLER
66                    && e.getErrorCode() !=
                        DOPPELTER_UNIQUE_SCHLUSSEL_FEHLER) {
67                  // Der Import soll weiter laufen, wenn der Datensatz
68                  // existiert.
69                  // istImportOk = false;
70                  e.printStackTrace();
71                  protocol += "SQL-Exception in der Zeile: "
72                      + lineCounter + "\n";
73                  protocol += line + "\n\n";
74                  JOptionPane
75                      .showMessageDialog(
76                          new JFrame(),
77                          "Daten aus der Zeile "
78                              + lineCounter
79                              + " verursachten beim Ausfuhren der
                                  SQL-Anweisung Fehler."
80                              + "\nBitte uberprufen Sie die SQL-Anweisung
                                  oder zu importierenden Daten!",
81                          "Datenimport",
82                          JOptionPane.ERROR_MESSAGE);
83                } else {
84                  protocol += "Datensatz in der Zeile " + lineCounter +
                        " ist bereits vorhanden:\n";
85                  protocol += line + "\n\n";
86                }
87              } catch (ClassNotFoundException e) {
88                // istImportOk = false;
89                JOptionPane
90                    .showMessageDialog(
91                        new JFrame(),
92                        "Keine Verbindung zur Datenbank!"
93                            + "\nBitte uberprufen Sie Ihre Angaben!",
```

```
94                      "Datenimport",
95                      JOptionPane.ERROR_MESSAGE);
96                e.printStackTrace();
97                protocol += "Keine Verbindung zur Datenbank!\n\n";
98                break;
99              } catch (NoSuchElementException e) {
100               protocol += "Parse-Exception in der Zeile: "
101                   + lineCounter + "\n";
102               protocol += line + "\n\n";
103               JOptionPane
104                   .showMessageDialog(
105                       new JFrame(),
106                       "Daten aus der Zeile "
107                           + lineCounter
108                           + " konnten nicht importiert werden."
109                           + "\nBitte uberprufen Sie die zu
                              importierenden Daten!",
110                       "Datenimport",
111                       JOptionPane.ERROR_MESSAGE);
112               e.printStackTrace();
113               // break;
114             }
115           }
116           try {
117             line = in.readLine();
118           } catch (IOException e) {
119             // TODO Auto-generated catch block
120             e.printStackTrace();
121           }
122         }
123         protocol += "Anzahl der verarbeiteten Datensatze: " +
              objCounter + "\n\n";
124         protocol += "Datenimport ist abgeschlossen.\n\n";
```

```
125         } finally {
126           if (in != null)
127             try {
128               in.close();
129             } catch (IOException e) {
130               // TODO Auto-generated catch block
131               e.printStackTrace();
132             }
133         }
134         return protocol;
135       }
136 
137       // Neuer Fachbereich fur Gewurze erstellen, bevor der Import
              vollzogen wird
138 
139       public static void createNewFB(String name, String fb_nr) throws
              SQLException {
140 
141         DBImport.insertFB(name, fb_nr, con);
142 
143       }
144 
145       /**
146        * Liefert true, wenn der Import ohne Fehler durchgefuhrt wurde.
147        */
148       public boolean isImportOk() {
149         return istImportOk;
150       }
151 
152   }
```

A.6. DBImport

```
1
2   // Klasse zum Importieren der CSV-Zeilen in die Datenbank
3
4  public class DBImport{
5
6   /**
7    * Speichert Daten der ubergebenen Zeile in der Datenbank.
8    *
9    * @throws ClassNotFoundException
10   */
11
12   // Schreiben der Daten in die Datenbank
13
14   public static int insert(FileLine fileLine, Connection con)
15     throws SQLException, ClassNotFoundException, NoSuchElementException
         {
16    PreparedStatement stmnt = null;
17
18
19    int ergebnisZeilen = 0;
20
21
22    if (fileLine.getFachbereichNr() != null) {
23     stmnt = con
24       .prepareStatement("INSERT INTO "
25        + "dbo.PRODUKT"
26        +"(fachbereichNr,handelsmarkeID,bezeichnung,trockensortiment,warengruppe,preis)
27        +"VALUES (?,?,?,?,?,?)");
28
29
30    // Werte setzen:
31    stmnt.setString(1, fileLine.getFachbereichNr());
32    stmnt.setInt(2,fileLine.getHandelsmarkeId());
```

```
33    stmnt.setString(3, fileLine.getBezeichnung());
34    stmnt.setString(4, fileLine.getTrockensortiment());
35    stmnt.setString(5,fileLine.getWarengruppe());
36    stmnt.setDouble(6,fileLine.getPreis());
37
38    System.out.println(fileLine.toString());
39    // Insert Ausfuhren
40    ergebnisZeilen = stmnt.executeUpdate();
41
42    // Transaktion beenden:
43    con.commit();
44    con.setAutoCommit(true);
45   } else
46    throw new NoSuchElementException();
47
48   if (stmnt != null)
49    try {
50     stmnt.close();
51    } catch (Exception ex) {
52    }
53   stmnt = null;
54
55
56
57    return ergebnisZeilen;
58  }
59
60  // Method for creating new Fachbereich
61
62  @SuppressWarnings("null")
63  public static void insertFB (String name, String fb_nr, Connection
        con) throws SQLException{
64
```

```
65
66   int ergebnisZeilen = 0;
67
68
69   String query = "INSERT INTO dbo.FACHBEREICH"
70      + "(FACHBEREICHNR, NAME)"
71      +"VALUES (?,?)" ;
72
73   PreparedStatement statement = con.prepareStatement(query);
74
75   // Fachbereich will be created with number and name
76
77   statement.setString(1,fb_nr);
78   statement.setString(2,name);
79   ergebnisZeilen = statement.executeUpdate();
80
81   con.commit();
82   con.setAutoCommit(true);
83   if (statement != null)
84    try {
85     System.out.println("Inserting FB");
86     statement.close();
87    } catch (Exception ex) {
88    }
89   statement = null;
90
91
92  }
93 }
```

A.7. DateFormat

```
1
```

```
2
3  public class Date_Format {
4
5   /**
6      * Liefert das Datum im Format "dd.MM.yyyy" zuruck.
7      * @param time long
8      * @return String
9      */
10    public static String getDdMMyyyyHHMi(long time) {
11      java.util.Date d = new java.util.Date(time);
12      TimeZone z = TimeZone.getDefault();
13
14      Calendar c = Calendar.getInstance(z, Locale.GERMANY);
15      SimpleDateFormat sd = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
16      sd.setCalendar(c);
17      return sd.format(d);
18    }
19
20    /**
21     * Wandelt Datum als String in einen Longwert.
22     */
23    public static long getDdMMyyyy(String sDate) throws ParseException {
24      if ( (sDate == null) || sDate.trim().equals("")) {
25        return 0;
26      }
27      SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
28      // Pruft, dass es sich tatsachlich um die richtigen Datumsangaben
           handelt.
29      dateFormat.setLenient(false);
30      GregorianCalendar date = new GregorianCalendar();
31      date.setTime(dateFormat.parse(sDate));
32      return date.getTime().getTime();
33    }
```

```
34
35    /**
36     * Wandelt Datum als String in ein GregorianCalendar-Objekttyp.
37     */
38    public static GregorianCalendar getDayAsGregorianCalendar(String
          sDate) throws ParseException {
39      if ( (sDate == null) || sDate.trim().equals("")) {
40        return null;
41      }
42      SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
43      // Pruft, dass es sich tatsachlich um die richtigen Datumsangaben
            handelt.
44      dateFormat.setLenient(false);
45      GregorianCalendar date = new GregorianCalendar();
46      date.setTime(dateFormat.parse(sDate));
47      return date;
48    }
49
50    /**
51     * Wandel Date-Objekt in einen String im Format 'YYYY-MM-dd'
52     */
53      public static String getYyyyMMdd(java.sql.Date time) {
54        TimeZone z = TimeZone.getDefault();
55
56        Calendar c = Calendar.getInstance(z, Locale.GERMANY);
57        SimpleDateFormat sd = new SimpleDateFormat("yyyy-MM-dd");
58        sd.setCalendar(c);
59        return sd.format(time);
60      }
61
62      /**
63       * Wandel Long-Objekt in einen Date-Objekt
64       */
```

```
65      public static java.sql.Date getDate(long time) {
66        return new java.sql.Date(time);
67      }
68  }
```

B. Quelltext für SQL-Skripte

B.1. 3-Insert-Student

```
1
2    -- ******************** Kassenbonerstellung
         ************************** --
3
4  insert into KASSENBON (FILIALENR, DATUM, KASSENNR, BARGELD, SUMME)
5  VALUES
6
7  (14, CONVERT([datetime], '2019-02-10 08:22:00.000', 20), 1, 400.00,
       400.00),
8  (14, CONVERT([datetime], '2019-02-11 10:22:00.000', 20), 1, 33.00,
       33.00),
9  (14, CONVERT([datetime], '2019-02-16 08:22:00.000', 20), 1, 12.00,
       12.00),
10 (14, CONVERT([datetime], '2019-02-17 10:22:00.000', 20), 1, 123.00,
       150.00),
11 (14, CONVERT([datetime], '2019-02-20 11:29:00.000', 20), 1, 134.00,
       200.00),
12 (14, CONVERT([datetime], '2019-02-22 13:22:00.000', 20), 1, 22.00,
       100.00),
13 (14, CONVERT([datetime], '2019-02-23 14:25:00.000', 20), 1, 43.00,
       50.00),
14 (14, CONVERT([datetime], '2019-02-25 08:30:00.000', 20), 1, 40.00,
       100.00),
15 (14, CONVERT([datetime], '2019-03-10 12:29:00.000', 20), 1, 100.00,
```

```
    100.00),
16  (14, CONVERT([datetime], '2019-04-22 14:20:00.000', 20), 1, 223.00,
    230.00),
17  (14, CONVERT([datetime], '2019-04-11 13:20:00.000', 20), 1, 433.00,
    450.00),
18  (14, CONVERT([datetime], '2019-05-22 15:20:00.000', 20), 1, 333.00,
    335.00),
19  (14, CONVERT([datetime], '2019-05-23 14:40:00.000', 20), 1, 555.00,
    555.00),
20  (14, CONVERT([datetime], '2019-05-24 16:22:00.000', 20), 1, 244.00,
    250.00),
21  (14, CONVERT([datetime], '2019-05-25 15:30:00.000', 20), 1, 260.00,
    260.00),
22  (14, CONVERT([datetime], '2019-05-25 16:23:00.000', 20), 1, 50.00,
    50.00),
23  (14, CONVERT([datetime], '2019-05-25 16:45:00.000', 20), 1, 78.44,
    100.00),
24  (14, CONVERT([datetime], '2019-05-25 17:44:00.000', 20), 1, 444.00,
    450.00),
25  (14, CONVERT([datetime], '2019-05-26 17:55:00.000', 20), 1, 50.00,
    50.00),
26  (14, CONVERT([datetime], '2019-05-27 12:42:00.000', 20), 1, 33.50,
    35.00),
27  (14, CONVERT([datetime], '2019-05-28 17:50:00.000', 20), 1, 447.00,
    450.00),
28  (14, CONVERT([datetime], '2019-06-02 12:44:00.000', 20), 1, 1000.00,
    1000.00),
29  (14, CONVERT([datetime], '2019-06-12 10:44:00.000', 20), 1, 23.00,
    34.00),
30  (14, CONVERT([datetime], '2019-06-12 12:44:00.000', 20), 1, 25.00,
    37.00),
31  (14, CONVERT([datetime], '2019-06-13 10:44:00.000', 20), 1, 399.00,
    400.00),
```

```
32 (14, CONVERT([datetime], '2019-06-14 12:44:00.000', 20), 1, 23.00,
       23.00),
33 (14, CONVERT([datetime], '2019-06-17 12:30:00.000', 20), 1, 12.00,
       50.00),
34 (14, CONVERT([datetime], '2019-06-20 10:33:00.000', 20), 1, 66.00,
       100.00),
35 (14, CONVERT([datetime], '2019-06-25 10:44:00.000', 20), 1, 33.00,
       100.00),
36 (14, CONVERT([datetime], '2019-06-25 10:55:00.000', 20), 1, 13.00,
       23.00),
37 (14, CONVERT([datetime], '2019-06-25 10:20:00.000', 20), 1, 124.00,
       200.00),
38 (14, CONVERT([datetime], '2019-06-28 09:55:00.000', 20), 1, 45.00,
       50.00),
39 (14, CONVERT([datetime], '2019-06-30 13:45:00.000', 20), 1, 66.00,
       70.00),
40 (14, CONVERT([datetime], '2019-07-01 16:50:00.000', 20), 1, 898.00,
       1000.00),
41 (14, CONVERT([datetime], '2019-08-02 10:44:00.000', 20), 1, 87.00,
       100.00),
42 (14, CONVERT([datetime], '2019-08-24 12:44:00.000', 20), 1, 34.00,
       35.00),
43 (14, CONVERT([datetime], '2019-08-25 15:26:00.000', 20), 1, 400.00,
       500.00),
44 (14, CONVERT([datetime], '2019-09-11 09:22:00.000', 20), 1, 298.00,
       300.00),
45 (14, CONVERT([datetime], '2019-09-11 12:33:00.000', 20), 1, 500.00,
       700.00),
46 (14, CONVERT([datetime], '2019-09-12 11:20:00.000', 20), 1, 500.00,
       100.00)
47
48 -- ******************** BONPOSITIONEN ************************** --
49
```

```
50
51 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
52 values
53 (10, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
54 (10, 2, 'N', 'N', 1, 11.9900, 11.9900 * 0.07),
55 (10, 3, 'Y', 'Y', 10, 24.900, 24.900 * 0.07);
56
57 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
58 values
59 (11, 2, 'Y', 'Y', 200, 2398.00, 2398.00 * 0.07),
60 (11, 15, 'Y', 'N', 10, 2.9900, 2.9900 * 0.07),
61 (11, 4, 'Y', 'Y', 10, 20.9, 20.9 * 0.07);
62
63 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
64 values
65 (12, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
66 (12, 21, 'Y', 'N', 1, 2.4900, 2.4900 * 0.07),
67 (12, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07);
68
69 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
70 values
71 (13, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
72 (13, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07),
73 (13, 15, 'Y', 'N', 1, 2.9900, 2.9900 * 0.07);
74
75 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
76 values
77 (14, 10, 'Y', 'Y', 20, 10.8, 10.8 * 0.07),
```

```
78 (14, 20, 'Y', 'Y', 200, 149.0, 149.0 * 0.07),
79 (14, 15, 'N', 'N', 1, 2.9900, 2.9900 * 0.07);
80
81 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
82 values
83 (15, 14, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
84 (15, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
85 (15, 15, 'N', 'N', 1, 2.9900, 2.9900 * 0.07);
86
87 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
88 values
89 (16, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
90 (16, 14, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
91 (16, 15, 'Y', 'Y', 1, 2.9900, 2.9900 * 0.07);
92
93 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
94 values
95 (17, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07),
96 (17, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
97 (17, 161, 'N', 'N', 1, 9.9000, 9.9000 * 0.07);
98
99 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
100 values
101 (18, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
102 (18, 13, 'Y', 'Y', 10, 20.9, 20.9 * 0.07),
103 (18, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07);
104
105 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
```

```
106 values
107 (19, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
108 (19, 12, 'Y', 'Y', 10, 7.900, 7.900 * 0.07),
109 (19, 15, 'Y', 'Y', 1, 2.9900, 2.9900 * 0.07);
110
111 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
112 values
113 (20, 15, 'Y', 'N', 1, 12.7900, 12.7900 * 0.07),
114 (20, 169, 'N', 'N', 1, 3.7900, 3.7900 * 0.07),
115 (20, 172, 'Y', 'Y', 10, 29.900, 29.900 * 0.07);
116
117
118 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
119 values
120 (21, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
121 (21, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
122 (21, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07);
123
124 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
125 values
126 (22, 167, 'Y', 'Y', 10, 28.900, 28.900 * 0.07),
127 (22, 169, 'Y', 'Y', 1, 3.7900, 3.7900 * 0.07),
128 (22, 172, 'Y', 'Y', 200, 2.9900, 2.9900 * 0.07);
129
130 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
131 values
132 (23, 12, 'Y', 'Y', 10, 7.90, 7.90 * 0.07),
133 (23, 158, 'Y', 'Y', 1, 3.9500, 3.9500 * 0.07),
134 (23, 160, 'Y', 'Y', 10, 45.500, 45.500 * 0.07);
```

```
135
136
137 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
138 values
139 (24, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
140 (24, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
141 (24, 16, 'Y', 'Y', 20, 59.800, 59.800 * 0.07);
142
143
144 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
145 values
146 (25, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
147 (25, 16, 'Y', 'Y', 1,2.9900, 2.9900 * 0.07),
148 (25, 19, 'Y', 'Y', 200, 133.0, 133.0 * 0.07);
149
150 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
151 values
152 (26, 12, 'Y', 'Y', 10, 7.900, 7.900 * 0.07),
153 (26, 163, 'Y', 'Y', 1, 3.9900, 3.9900 * 0.07),
154 (26, 164, 'Y', 'Y', 10, 19.900, 19.900 * 0.07);
155
156
157 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
158 values
159 (27, 169, 'Y', 'N', 10, 37.900, 37.900 * 0.07),
160 (27, 172, 'Y', 'Y', 15, 44.85, 44.85 * 0.07),
161 (27, 159, 'Y', 'Y', 10, 45.500, 45.500 * 0.07);
162
163 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
```

```
    VERKAUFSMENGE, VERKAUFSPREIS, MWST)
164 values
165 (28, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
166 (28, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07),
167 (28, 7, 'N', 'Y', 20, 27.8, 27.8 * 0.07);
168
169 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
    VERKAUFSMENGE, VERKAUFSPREIS, MWST)
170 values
171 (29, 6, 'Y', 'N', 10, 144.900, 144.900 * 0.07),
172 (29, 4, 'Y', 'Y', 10, 127.900, 127.900 * 0.07),
173 (29, 3, 'N', 'Y', 1, 2.4900, 2.4900 * 0.07);
174
175
176 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
    VERKAUFSMENGE, VERKAUFSPREIS, MWST)
177 values
178 (30, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
179 (30, 19, 'Y', 'Y', 200, 133.0, 133.0 * 0.07),
180 (30, 6, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
181
182 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
    VERKAUFSMENGE, VERKAUFSPREIS, MWST)
183 values
184 (31, 12, 'Y', 'Y', 20, 10.8, 10.8 * 0.07),
185 (31, 19, 'Y', 'Y', 200, 133.0, 133.0 * 0.07),
186 (31, 170, 'Y', 'Y', 200, 2.1900 * 200, 2.1900 *200 * 0.07);
187
188 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
    VERKAUFSMENGE, VERKAUFSPREIS, MWST)
189 values
190 (32, 169, 'Y', 'N', 43, 3.7900 *43, 3.7900 *43 * 0.07),
191 (32, 21, 'Y', 'Y', 10, 24.900, 24.900 * 0.07),
```

```
192 (32, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07);
193
194 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
195 values
196 (33, 12, 'Y', 'Y', 30, 0.7900 * 30, 0.7900 * 30 * 0.07),
197 (33, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07),
198 (33, 6, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
199
200
201 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
202 values
203 (34, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
204 (34, 19, 'Y', 'Y', 200, 133.0, 133.0 * 0.07),
205 (34, 160, 'Y', 'Y', 1, 4.5500, 4.5500 * 0.07);
206
207 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
208 values
209 (35, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
210 (35, 12, 'Y', 'Y', 10, 7.900, 7.900 * 0.07),
211 (35, 6, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
212
213 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
214 values
215 (36, 170, 'Y', 'Y', 21, 2.1900 * 21, 2.1900 * 21 * 0.07),
216 (36, 15, 'Y', 'Y', 200, 133.0, 133.0 * 0.07),
217 (36, 9, 'N', 'Y', 20, 0.7900 * 20, 0.7900 * 20 * 0.07);
218
219 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
```

```
220 values
221 (37, 172, 'Y', 'N', 20, 2.9900 * 20, 2.9900 * 20 * 0.07),
222 (37, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
223 (37, 164, 'Y', 'Y', 200, 1.9900 * 200, 1.9900 * 200 * 0.07);
224
225 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
226 values
227 (38, 18, 'Y', 'Y', 10, 21.900, 21.900 * 0.07),
228 (38, 12, 'Y', 'Y', 30, 0.7900 * 30, 0.7900 * 30 * 0.07),
229 (38, 15, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
230
231
232 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
233 values
234 (39, 165, 'Y', 'Y', 200, 1.9900 * 200, 1.9900 * 200 * 0.07),
235 (39, 168, 'Y', 'Y', 100, 100 * 2.8900, 100 * 2.8900 * 0.07),
236 (39, 6, 'N', 'Y', 3, 14.4900 * 3, 14.4900 * 3 * 0.07);
237
238 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
239 values
240 (40, 166, 'Y', 'N', 50, 14.9900 * 50, 14.9900 * 50 * 0.07),
241 (40, 165, 'N', 'Y', 8, 1.9900 * 8, 1.9900 * 8 * 0.07),
242 (40, 171, 'Y', 'Y', 15, 1.9900 * 15, 1.9900 * 15 * 0.07);
243
244 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
245 values
246 (41, 2, 'Y', 'N', 10, 11.9900 * 10, 11.9900 * 10 * 0.07),
247 (41, 3, 'N', 'N', 45, 2.4900 * 45, 2.4900 * 45 * 0.07),
248 (41, 4, 'Y', 'Y', 1, 12.7900, 12.7900 * 0.07);
```

```
249
250 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
251 values
252 (42, 163, 'N', 'N', 5, 3.9900 * 5, 3.9900 * 5 * 0.07),
253 (42, 17, 'Y', 'Y', 50, 2.1900 * 50, 2.1900 * 50 * 0.07),
254 (42, 18, 'Y', 'Y', 30, 2.1900 * 30, 2.1900 * 30 * 0.07);
255
256 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
257 values
258 (43, 169, 'Y', 'Y', 20, 3.7900* 20, 3.7900* 20 * 0.07),
259 (43, 20, 'Y', 'Y',10, 24.900, 24.900 * 0.07),
260 (43, 21, 'N', 'Y', 1, 2.4900, 2.4900 * 0.07);
261
262 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
263 values
264 (44, 170, 'Y', 'Y', 10, 21.900, 21.900 * 0.07),
265 (44, 162, 'Y', 'Y', 20, 2.9900 * 20, 2.9900 * 20 * 0.07),
266 (44, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07);
267
268 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
269 values
270 (45, 15, 'Y', 'N', 10, 29.9, 29.9 * 0.07),
271 (45, 1, 'N', 'N', 1, 12.79, 12.79 * 0.07),
272 (45, 162, 'Y', 'Y', 20, 2.9900 * 20, 2.9900 * 20 * 0.07);
273
274 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
       VERKAUFSMENGE, VERKAUFSPREIS, MWST)
275 values
276 (46, 12, 'Y', 'Y', 20, 10.8, 10.8 * 0.07),
```

```
277 (46, 162, 'Y', 'Y', 20, 2.9900 * 20, 2.9900 * 20 * 0.07),
278 (46, 170, 'Y', 'Y', 10, 21.900, 21.900 * 0.07)
279
280 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
        VERKAUFSMENGE, VERKAUFSPREIS, MWST)
281 values
282 (47, 13, 'Y', 'Y', 10, 20.9, 20.9 * 0.07),
283 (47, 21, 'Y', 'Y', 200, 149.0, 149.0 * 0.07),
284 (47, 6, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
285
286 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
        VERKAUFSMENGE, VERKAUFSPREIS, MWST)
287 values
288 (48, 170, 'Y', 'Y', 10, 21.900, 21.900 * 0.07),
289 (48, 162, 'Y', 'Y', 20, 2.9900 * 20, 2.9900 * 20 * 0.07),
290 (48, 15, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
291
292 insert into BONPOSITION (BONNR, PRODUKTNR, AKTIONSWARE, RABATTWARE,
        VERKAUFSMENGE, VERKAUFSPREIS, MWST)
293 values
294 (49, 13, 'Y', 'Y', 10, 20.9, 20.9 * 0.07),
295 (49, 170, 'Y', 'Y', 10, 21.900, 21.900 * 0.07),
296 (49, 6, 'N', 'Y', 2, 24.98, 24.98 * 0.07);
297
298 -- ******* Tabelle HANDELSMARKE ******************************** --
299
300
301 insert into HANDELSMARKE (NAME, LAND)
302 values
303 ('Schar', 'Deutschland'),
304 ('Nestle', 'Schweiz'),
305 ('Kellog''s', 'USA'),
306 ('Coca Cola', 'USA'),
```

```
('Goutess', 'Deutschland'),
('Gutfried', 'Deutschland'),
('Jever', 'Deutschland'),
('Nick', 'USA'),
('Basic', 'USA');

-- ********************************************************** --
-- ******* Tabelle FACHBEREICH ******************************** --
-- ********************************************************** --

-- ** Gewurze wurden aus dem Java-Programm heraus hinzugefugt *** --
insert into FACHBEREICH (FACHBEREICHNR, NAME)
values
(1040,'Snacks'),
(1041,'Oriental'),
(1042,'Asia'),
(1043,'Italienisch'),
(1044,'Drogerie'),
(1045,'Haushalt'),
(1046,'Getranke'),
(1047,'Desserts');
```

B.2. 4-Update-Tables

```
  select * from PRODUKT;
select * from HANDELSMARKE;

-- *** Zuordnung der Handelsmarken *** --

update PRODUKT
set HANDELSMARKEID = 22
where BEZEICHNUNG LIKE '%Landliebe%';

```

```
10 update PRODUKT
11 set HANDELSMARKEID = 40
12 where BEZEICHNUNG LIKE '%Kellog%';
13
14 update PRODUKT
15 set HANDELSMARKEID = 42
16 where BEZEICHNUNG LIKE '%Goutess%';
17
18 update PRODUKT
19 set HANDELSMARKEID = 46
20 where BEZEICHNUNG LIKE '%Basic%';
21
22 update PRODUKT
23 set HANDELSMARKEID = 39
24 where BEZEICHNUNG LIKE '%Nestle%';
25
26 update PRODUKT
27 set HANDELSMARKEID = 38
28 where BEZEICHNUNG LIKE '%Schaer%';
29
30 update PRODUKT
31 set HANDELSMARKEID = 45
32 where BEZEICHNUNG LIKE '%Nick%';
33
34 update PRODUKT
35 set HANDELSMARKEID = 13
36 where BEZEICHNUNG LIKE '%Oetker%';
37
38 -- ** Update der Summen in Kassenbons ** --
39
40 UPDATE KASSENBON
41 SET SUMME = t.summe_einkauf
42 FROM KASSENBON AS kb
```

```
43 INNER JOIN
44     (
45         SELECT BONNR, SUM(VERKAUFSPREIS) summe_einkauf
46         FROM BONPOSITION
47         GROUP BY BONNR
48     ) t
49     ON t.BONNR = kb.BONNR
50
51 -- ** Bargeld an Summe anpassen, um Deckung zu gewaehrleisten ** --
52
53 update KASSENBON
54  set BARGELD = Round(SUMME, 0)
55
56 -- ** Bargeld standardmaessig erhoehen, um Differenz zu erzeugen ** --
57
58 update KASSENBON
59  set BARGELD = BARGELD + 50
60
61 -- ** runden ** --
62
63 update KASSENBON
64  set BARGELD = Round(Bargeld,-1)
```

B.3. 5-ETL-DW

```
1 use DB6013868
2 -- Dim-Tabellen mit Werten aus der ODB befullen --
3 --DIM_Abteilung--
4 insert into DIM_Abteilung(AbteilungNr,Name)
5 select FB.FachbereichNr,FB.Name
6 from Fachbereich as FB
7
8 --DIM_Produkt--
```

```
9  insert into
       DIM_PRODUKT(PRODUKTNR,BEZEICHNUNG,TROCKENSORTIMENT,WARENGRUPPE)
10 select P.ProduktNr, P.Bezeichnung, P.Trockensortiment,
       substring(P.Warengruppe,3,1)
11 from Produkt as P
12
13 --DIM_Filiale--
14 insert into DIM_FILIALE(FILIALENR, UST_ID_NR,STRASSE,STADT,PLZ)
15 select F.FILIALENR,F.UST_ID_NR, F.STRASSE,F.STADT, F.PLZ
16 from FILIALE as F
17
18 --DIM_Handelsmarke--
19 insert into DIM_HANDELSMARKE(HANDELSMARKENR,NAME, LAND)
20 select HM.HANDELSMARKEID, HM.NAME, HM.LAND
21 from HANDELSMARKE AS HM
22
23 --DIM_Date Parameter fur die Berechnung der Attribute--
24 --Alle Tage von Anfang bis Ende ---
25 declare @vonDatum date,
26   @bisDatum date,
27   @Jahr integer, -- Attribut "Jahr"
28   @Monat integer, -- Attribut "Monat"
29   @Tag integer, -- Attribut "Tag"
30   @Quartal integer, -- Attribut "Quartal"
31   @KW integer -- Attribut "Kalenderwoche"
32
33  select @vonDatum=min(CONVERT(date,datum)) from KASSENBON
34  select @bisDatum=max(CONVERT(date,datum)) from KASSENBON
35
36 -- Zeitspanne: Schleife, bis Enddatum Erreicht wurde --
37
38  while @vonDatum <= @bisDatum
39   begin
```

```
40    set @Jahr  =year(@vonDatum)
41    set @Monat  =month(@vonDatum)
42    set @Tag  =day(@vonDatum)
43    set @Quartal =datepart(Quarter,@vonDatum)
44    set @KW   =datepart(WK,@vonDatum)
45
46    insert into DIM_DATE(DATEID, JAHR, MONAT, TAG, QUARTAL, KW)
47    values (@vonDatum, @jahr, @monat, @tag, @Quartal, @KW)
48
49   set @vonDatum = DATEADD(DD, 1,@vonDatum)
50  end
51
52 -- DIM_Time aud den Tag bezogen --
53 declare
54  @vonZeit time,
55  @bisZeit time,
56  @stunde int,
57  @minute int,
58  @tageszeit int,
59  @flag int
60     --Tag --
61  set @vonZeit = '00:00'
62  set @bisZeit = '23:59'
63  set @flag = 0 --unterbricht die Schleife --
64
65 while @flag = 0
66  begin
67     -- Tageszeiten bestimmen (1,2,3,4)
68     -- Schleife, bis Tagesende
69   set @stunde = DATEPART(HOUR, @vonZeit)
70   set @minute = DATEPART(MINUTE, @vonZeit)
71
72   if @stunde > 5 and @stunde < 12
```

```
73     set @tageszeit = 1
74    else if @stunde > 11 and @stunde < 18
75     set @tageszeit = 2
76    else if @stunde > 17 and @stunde <= 24
77     set @tageszeit = 3
78    else
79     set @tageszeit = 4
80
81    insert into DIM_TIME(TIMEID,STUNDE,MINUTE,TAGESZEIT)
82     values (@vonZeit, @stunde, @minute, @tageszeit)
83
84     if @vonZeit = @bisZeit
85      set @flag = 1
86
87     set @vonZeit = DATEADD(MINUTE, 1, @vonZeit)
88   end
89
90  -- Kassenbon_Fakten mit Zeiten verbinden --
91
92   INSERT INTO KASSENBON_FAKTEN
93   select fbr.FACHBEREICHNR, P.PRODUKTNR, hm.HANDELSMARKEID,
94   convert(date,DATEID), convert(time(0),TIMEID),
95      F.FILIALENR, sum(BP.Verkaufspreis), sum(BP.MWST),
96   sum(BP.Verkaufspreis) - sum(BP.MWST),
97   sum(BP.Verkaufsmenge),
98   bp.AKTIONSWARE,
99   bp.RABATTWARE
100
101  from
102  FACHBEREICH as fbr
103  join Produkt as P
104   on fbr.FACHBEREICHNR= P.FachbereichNr
105  join
```

```
106   HANDELSMARKE as hm
107   on P.HandelsmarkeId = hm.HANDELSMARKEID
108  join
109   Bonposition as bp
110   on P.ProduktNr = bp.ProduktNr
111  join
112   Kassenbon as KB
113   on bp.BonNr = KB.BonNr
114  join
115   FILIALE as F
116   on KB.FilialeNr = F.FILIALENR
117  join
118   Dim_Date as DD
119   on convert(date,KB.Datum) = DD.DateID
120  join
121   Dim_Time as DT
122   on convert(time(0),KB.Datum) = DT.TimeID
123  group by
124  fbr.FACHBEREICHNR, P.PRODUKTNR, hm.HANDELSMARKEID,
125  DD.DATEID, DT.TIMEID, F.FILIALENR,
126  BP.AKTIONSWARE, BP.RABATTWARE,BP.BONNR
```

B.4. 6-Fakten-Woche

```
1    create table KASSENBON_FAKTEN_WOCHEN (
2   ABTEILUNGNR int not null references DIM_ABTEILUNG (ABTEILUNGNR),
3   PRODUKTNR int not null references DIM_PRODUKT (PRODUKTNR),
4   HANDELSMARKENR int not null references DIM_HANDELSMARKE
        (HANDELSMARKENR),
5   JAHR int not null,
6   WOCHE int not null,
7      TIMEID time not null,
8   FILIALENR char(3) not null references DIM_FILIALE (FILIALENR),
```

```
 9  Umsatz float not null,
10  ABGABEMWST float not null,
11  NETTO float not null,
12  MENGE int not null,
13  AKTIONSARTIKEL char(1) not null,
14  RABATTIERT char(1) not null,
15  constraint PK_FAKTEN_WOCHEN primary key
        (ABTEILUNGNR,PRODUKTNR,HANDELSMARKENR, JAHR, TIMEID,
        WOCHE,FILIALENR))
16
17
18
19  INSERT INTO KASSENBON_FAKTEN_WOCHEN
20  select
21  FB.FACHBEREICHNR,
22  P.PRODUKTNR,
23  H.HANDELSMARKEID,
24  DD.JAHR,
25  DD.KW,
26     convert(time(0),TIMEID),
27  F.FILIALENR,
28  sum(BP.Verkaufspreis),
29  sum(BP.MWST),
30  sum(BP.Verkaufspreis)-sum(BP.MWST),
31  sum(BP.Verkaufsmenge),
32  BP.AKTIONSWARE,
33  BP.RABATTWARE
34  from
35  FACHBEREICH as FB
36  join Produkt as P
37   on FB.FACHBEREICHNR= P.FachbereichNr
38  join
39   HANDELSMARKE as H
```

```
40   on P.HandelsmarkeId = H.HANDELSMARKEID
41  join
42   Bonposition as BP
43   on P.ProduktNr = BP.ProduktNr
44  join
45   Kassenbon as KB
46   on BP.BonNr = KB.BonNr
47  join
48   FILIALE as F
49   on KB.FilialeNr = F.FILIALENR
50  join
51   Dim_Date as DD
52   on DATEPART(yyyy,KB.Datum) = DD.JAHR and
53    DATEPART(wk,KB.DATUM) = DD.KW
54  join
55   Dim_Time as DT
56   on convert(time(0),KB.Datum) = DT.TIMEID
57  group by
58  FB.FACHBEREICHNR,
59  P.PRODUKTNR,
60  H.HANDELSMARKEID,
61  DD.JAHR,
62  DD.KW,
63     DT.TIMEID,
64  F.FILIALENR,
65  BP.AKTIONSWARE,
66  BP.RABATTWARE
```

B.5. 7-DW-Report

```
1   --- *** PROZEDUR mit Cursor fur das Datum ** ---
2
3 --- *** Parameter zum Aufruf der Prozedur: Produktnummer und
```

```
    Zeitspanne ** --
4
5  -- *** drop procedure DW_REPORT
6
7  CREATE PROCEDURE DW_REPORT (@produkt int ,@vonDatum date, @bisDatum
       date) AS
8  BEGIN
9   BEGIN TRANSACTION
10    if @vonDatum > @bisDatum
11    print 'ERROR. Eingabe uberprufen'
12   else
13 insert into DW_REPORT_TABLE
14 select PRODUKTNR, DATEID, sum(UMSATZ) as umsatz, sum(MENGE) as menge
       from KASSENBON_FAKTEN
15  WHERE
16  DATEID BETWEEN @vonDatum AND @bisDatum
17  and Produktnr =@produkt
18  group by DATEID, PRODUKTNR
19
20  declare @ProduktNr char(2), @von date, @bis date, @umsatz char(15),
       @menge char(4), @gesums char(10);
21  set @von = @vonDatum
22  set @bis = @bisDatum
23
24  declare KB_Cursor CURSOR FOR
25  SELECT Datum as Datum FROM DW_REPORT_TABLE
26  OPEN KB_Cursor
27  FETCH NEXT FROM KB_Cursor into @von
28  print 'ProduktNr Menge in St. Datum Umsatz in EU'
29  while @@FETCH_STATUS = 0
30  begin
31   select @umsatz = cast(umsatz as char) from DW_Report_TABLE where
       Datum = @von
```

```
32   select @menge= cast(menge as char) from DW_Report_TABLE where Datum
        = @von
33   select @ProduktNr = cast(ProduktNr as char) from DW_Report_TABLE
        where Datum = @von
34   print @ProduktNr +' | '+@menge+' | '+cast(@von as char)+'|
        '+@umsatz+'|'
35   print
        '-------------------------------------------------------------------------------
36   FETCH NEXT FROM KB_Cursor into @von
37  end
38  select @gesums = cast(sum(umsatz) as char) from DW_REPORT_TABLE
39  print' | Gesamtumsatz (EU) : '+@gesums+' |'
40  close KB_Cursor
41  deallocate KB_Cursor
42  delete from DW_REPORT_TABLE
43  commit transaction
44 END
45
46
47 -- ** fur den Aufruf der Prozedur Beispielparameter auswahlen ** --
48
49
50 declare @produktnummer int, @von date, @bis date
51 execute DW_REPORT 15,'2019-02-01','2019-08-01'
```

B.6. 8-Hilfstabelle

```
1
2
3 -- Hilfstabelle fur den Report --
4
5 create table DW_REPORT_TABLE(
6 ProduktNr int,
```

```
7 Datum datetime,
8 Umsatz float,
9 Menge int)
```

B.7. 9-Grant

```
1   -- Datentabellen/DB freigeben --
2
3   GRANT CONNECT to public
4
5
6   GRANT SELECT ON dbo.FILIALE to public
7   GRANT SELECT ON dbo.KASSENBON to public
8   GRANT SELECT ON dbo.BONPOSITION to public
9   GRANT SELECT ON dbo.PRODUKT to public
10  GRANT SELECT ON dbo.FACHBEREICH to public
11  GRANT SELECT ON dbo.HANDELSMARKE to public
12
13  GRANT SELECT ON dbo.DIM_TIME to public
14  GRANT SELECT ON dbo.DIM_DATE to public
15  GRANT SELECT ON dbo.DIM_ABTEILUNG to public
16  GRANT SELECT ON dbo.DIM_PRODUKT to public
17  GRANT SELECT ON dbo.DIM_FILIALE to public
18  GRANT SELECT ON dbo.DIM_HANDELSMARKE to public
19  GRANT SELECT ON dbo.KASSENBON_FAKTEN to public
20  GRANT SELECT ON dbo.KASSENBON_FAKTEN_WOCHEN to public
21  GRANT SELECT ON dbo.DW_REPORT_TABLE to public
22
23  GRANT EXECUTE ON dbo.DW_REPORT to public
```

Abschließende Erklärung

Ich versichere hiermit, dass ich meine Masterarbeit selbständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

<ORT>, den 13. Januar 2020 <AUTOR>