Institut für Informatik
Ludwig-Maximilians-Universität München

3. Dezember 2009

Thomas Bernecker, Tobias Emrich

Datenbankpraktikum

WS 2009/10

Abschlussprojekt

Stellen Sie sich folgende Situation vor:

Sie gehören einem Entwicklerteam des Systemhauses *BoilingPoint AG* an und sollen als Projekt ein Auftragsabwicklungssystem für das Handelsunternehmen *Haufkof* entwickeln.

Die Firma *Haufkof* ist ein reines Handelsunternehmen und lebt vom An- und Verkauf von Produkten, die es regelmäßig von seinen Lieferanten zu günstigen Großhandelspreisen bezieht und anschließend teurer an den Endkunden weiterverkauft. Die Preisvorteile bei den Lieferanten erzielt *Haufkof* vor allem durch langfristige Abnahmeverträge, die den Lieferanten die regelmäßige Abnahme Ihrer Produkte in genau vorgegebenen Zeitintervallen garantieren. Eine flexible Nachbestellung, die den momentanen Kundenbedürfnissen angepasst wird, ist daher nicht möglich.

Zur besseren Planung und Verwaltung sollen Sie eine relationale Datenbank mit *Oracle* konzipieren und erstellen, die über Clientprogramme bedient werden soll. Das System soll sowohl die Verwaltung der Kundenbestellungen und Zulieferungen von Lieferanten als auch die Lagerbestände an Produkten verwalten. Zusätzlich sollen natürlich noch Stammdaten wie Kunden, Lager, Produkte und Lieferanten verwaltet werden.

Ein weiteres Ziel des Systems ist es, bei Anlage einer Bestellung zu entscheiden, ob der vom Kunden genannte Bestelltermin eingehalten werden kann. Zusätzlich wäre die Berechnung des nächstmöglichen Liefertermins von großem Nutzen.

Als letzter wichtiger Punkt sollen mehrere Produktanalysen implementiert werden. Hierzu soll ein Reportingsystem erstellt werden, das diese Standardauswertungen ermöglicht.

Die Daten, die in die neue Anwendung übernommen werden sollen, stammen aus einem proprietären Altsystem und wurden gemäß des alten Datenbankschemas in ein gesondertes Schema PROJEKT_2009 auf Ihrer Datenbank importiert. Um die neuen Anforderungen an das System zu erfüllen, ist allerdings eine Neukonzeption als relationales Schema notwendig.

Im Altsystem stehen folgende Relationen zur Verfügung:
KUNDEN, PRODUKTE, LIEFERANT, ZULIEFERUNG (genaue Beschreibung siehe Anhang).

Der Fertigstellungstermin Ihrer Applikation ist der 11. Februar 2010. Sollten Sie bis zu diesem Zeitpunkt nicht fertig sein, muss die *BoilingPoint AG* wegen der dann von der Firma *Haufkof* verhängten Konventionalstrafe Konkurs anmelden und Sie verlieren Ihren Job.

Anhang:

(1) Anforderungsanalyse für das neue System

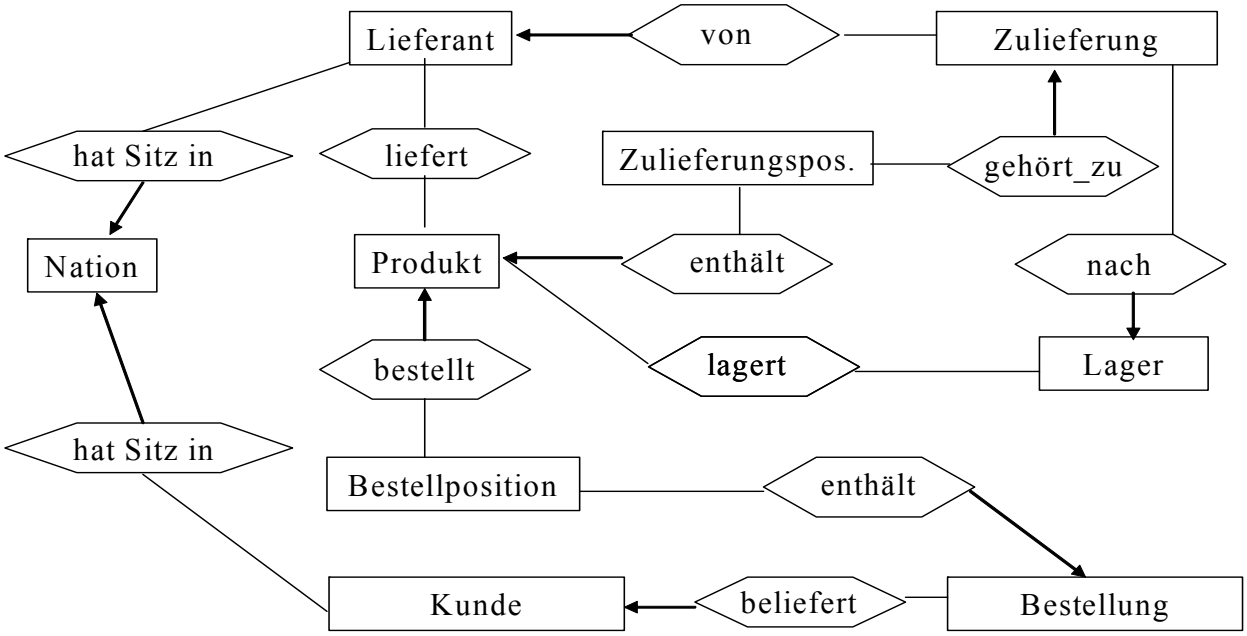
(2) Zeitplan für das Projekt

(3) Beschreibung der Altdaten

1. Anforderungsanalyse für das neue System

1.1 ER-Modell

Laut Anforderungsanalyse sollen im gewünschten System folgende Entities existieren, die in den angegebenen Beziehungen zueinander stehen:



Beschreibung der Entities (Schlüsselattribute sind unterstrichen):

Nation: NID, Name.

Lieferant: LID , Name, Adresse, Telefonnummer, Konto.

Kunde: KID, Name, Adresse, Telefonnummer, Konto, Branche.

Produkt: PID, Name, Hersteller, Marke, Typ, Konfektion, Groesse, Einzelverkaufspreis (wird in eine neue Bestellung kopiert).

Zulieferung: ZLID, Liefertext, Liefertermin, Aenderungsdatum, Anleger, Anlagedatum, Status (’OFFEN’, ’ERLEDIGT’), Erledigt_Termin (Datum der Lieferung).

Zulieferungsposition: POSNR (nur innerhalb einer Zulieferung eindeutig), Anzahl, Preis (gilt für diesen Auftrag), Positionstext.

Bestellung: BSTID, Bestelltext, Anleger, Anlagedatum, Aenderungsdatum, Status (’OFFEN’, ’BESTAETIGT’, ’ERLEDIGT’), Bestelltermin (falls Status = ’OFFEN’: Wunschtermin, anderer Status: zugesagter Termin), Erledigt_Termin (Datum der Lieferung).

Bestellposition: POSNR (nur innerhalb einer Bestellung eindeutig), Anzahl, Preis (gilt für diesen Auftrag), Positionstext.

Lager: LAGID, Adresse.

Relationships mit eigenen Attributen:

lagert: Anzahl.

liefert: Preis.

Die übrigen Relationships haben keine eigenen Attribute.

Konzipieren Sie ein relationales Datenbankschema nach obigem ER-Modell. Realisieren Sie Ihr Relationenschema in *Oracle* und erzeugen Sie die notwendigen Integritätschecks, um die Datenbank konsistent zu halten. Achten Sie dabei besonders auf Primärschlüsselbeziehungen und referentielle Integrität. Erzeugen Sie ein SQL-Skript, das den Aufbau der Datenbank vornimmt.

Achtung:

Die erweiterten Funktionalitäten, die im folgenden beschrieben werden, können das Anlegen weiterer Entitäten, Attribute und Beziehungen sinnvoll machen. Mit anderen Worten: Ihr Datenbankschema soll die genannten Objekte und Beziehungen verwalten, muss sich aber nicht auf diese beschränken.

Hinweise:

Sie sollten das Schema *eines* Team-Mitglieds als Datenbankschema für Ihr System verwenden. Dieses Mitglied sollte den anderen die benötigten Zugriffsrechte einräumen, also z.B. `select`, `insert`, `delete` und `update` für Tabellen und `select` für Sequenzen:

```
grant select, insert, delete, update on <table_name> to <member_name>;
grant select on <sequence_name> to <member_name>;
```

Weitere Informationen zum **`grant`**-Befehl gibt es in der *Oracle SQL Reference*.

1.2 Übernahme und Bereinigung der Altdaten

Datenmigration:

Nachdem Sie die Datenbank angelegt haben, ist diese mit Daten zu füllen. Die Daten aus dem Altsystem stehen dabei im Schema PROJEKT_2009 in vier Relationen zur Verfügung. Da das Altsystem einen hierarchischen Aufbau hatte, enthalten die folgenden vier "logischen Datenbanken" (Ausdruck des Altsystems) alle notwendigen Daten (genauere Beschreibung in Abschnitt 3).

KUNDEN: Enthält für jede Nation alle Kunden, deren Bestellungen und die dazugehörigen Bestellpositionen.

PRODUKTE: Enthält für alle Lager alle sich darin befindenden Produkte und deren Bestände.

LIEFERANT: Enthält für jede Nation alle Lieferanten mit den Produkten, die sie liefern können.

ZULIEFERUNG: Enthält die Lieferungen, die noch erwartet werden. Zu jeder Lieferung sind deren Positionen enthalten.

Importieren Sie die Daten aus diesen vier Relationen in Ihr Datenbankschema. Achten Sie darauf, dass die Transformation verlustlos und abhängigkeitserhaltend ist. Erzeugen Sie ein SQL-Skript, das die Daten in Ihr eigenes Schema überträgt. Dies ist sinnvoll, da Sie vor der Übergabe das System mit den initialen Daten füllen sollen.

Initiale Bestätigung der Bestellungen:

Der Zustand aller Belege (Zulieferungen und Bestellungen) ist unmittelbar nach der Datenübernahme OFFEN. Dies ist für die Zulieferungen korrekt, da nur noch nicht eingetroffene Zulieferungen übernommen werden. Für Bestellungen ergibt sich jedoch das Problem, dass erst noch überprüft werden muss, ob der Termin, der im Feld BESTELLDATUM angegeben ist, wirklich haltbar ist. Bisher wurde dieser Termin immer geschätzt, was zu Misskredit bei den Kunden durch fehlerhafte Einschätzungen führte. Das neue System soll daher alle offenen Bestellungen initial bestätigen, die mit den aktuellen Beständen und den bis jetzt eingegebenen Zulieferungen haltbar sind. Dabei darf das System nicht zulassen, dass eine bereits bestätigte Bestellung nicht ausgeliefert werden kann. Wir gehen außerdem davon aus, dass unsere Lieferanten immer wie vorgesehen liefern.

Der Status einer Bestellung soll also auf BESTAETIGT gesetzt werden, wenn alle ihre Positionen zum angegebenen Termin lieferbar sind. Um zu testen, ob eine Position lieferbar ist oder nicht, prüfen Sie, ob der aktuelle Bestand zuzüglich der Zulieferungen, die vor diesem Termin eintreffen, größer sind als die Bestellmenge zuzüglich aller bestätigten Positionen, die vor diesem Termin liegen.

Bei diesem Vorgehen werden Bestellungen, die einen früheren Liefertermin haben gegenüber solchen, die einen späteren Liefertermin haben, bevorzugt. Außerdem gilt, dass eine Bestellung nur dann bestätigt wird, falls alle in ihr vorkommenden Produkte termingerecht vorliegen. Bestellungen, die nicht wunschgerecht geliefert werden können, bleiben daher OFFEN und werden für die Terminierung anderer Bestellungen nicht berücksichtigt.

1.3 Beschreibung der Vorgänge auf den Daten

Die Mitarbeiter von *Haufkof* möchten über ein komfortables Anwendungsprogramm auf das Handelsinformationssystem zugreifen. Daher ist es Ihre Aufgabe, einen Datenbank-Client in Java zu implementieren, der Sachbearbeitern von *Haufkof* benutzerfreundliche Änderungsoperationen auf der Datenbank ermöglicht.

Alle Transaktionen, die von der Oberfläche aus angeboten werden, sollen nach dem ACID-Prinzip realisiert werden. Alle in einer Transaktion vorgenommenen Änderungen sollen also erst am Ende der Transaktion bestätigt oder verworfen werden. Achten Sie darauf, dass keine Inkonsistenzen, z.B. durch das doppelte Belegen von Produktbeständen entstehen können. Falls ein Sperrkonflikt auftritt, soll eine der beiden Transaktionen warten, bis die andere die gesperrten Daten wieder freigibt.

Datenbank-Fehlermeldungen, wie z.B. Verletzungen von Constraints ab, sollen abgefangen und sinnvoll behandelt werden.

Pflegetransaktionen:

Kunde anlegen und ändern

Diese Transaktion soll Kundendaten anlegen und ändern können. Das System soll automatisch eine eindeutige KID vorschlagen. Der Sachbearbeiter kann diese auf Wunsch ändern. NAME, ADRESSE und TELEFONNR des neuen Kunden sind textuell einzugeben. Zur Bestimmung der NID wird dem Sachbearbeiter ein Menü mit den in der Tabelle NATION gespeicherten Ländernamen angeboten. Es ist keine textuelle Eingabe möglich. KONTO des neuen Kunden wird mit 0.00 initialisiert. BRANCHE kann der Sachbearbeiter wahlweise textuell eingeben oder aus folgender Liste auswählen: ’AUTOMOBILE’, ’BUILDING’, ’FURNITURE’, ’HOUSEHOLD’ und ’MACHINERY’.

Produkttransaktionen:

Zulieferung einbuchen

Diese Transaktion soll die Produktbestände einer Zulieferung auf das in der Zulieferung angegebene Lager einbuchen und den Status auf ERLEDIGT setzen. Außerdem muss beim Einbuchen das Feld ERLEDIGT_DATUM auf das aktuelle Datum gesetzt werden. Als Eingabe bekommt die Transaktion die Nummer der Zulieferung.

Bestand umbuchen

Um Produktbestände von einem Lager auf ein anderes Lager zu transferieren, soll diese Transaktion ein Ursprungslager, ein Ziellager, ein Produkt und eine Menge erhalten. Anschließend prüft die Transaktion, ob genügend Bestand im Ursprungslager vorhanden ist. Falls der aktuelle Bestand nicht ausreicht, soll eine Fehlermeldung ("Bestand reicht nicht aus! Nur <X> Stück vorrätig!") ausgegeben werden. Falls der Bestand ausreicht, soll die entsprechende Menge auf den Ziellagerort transferiert werden. Ursprungs- und Ziellager sollen aus den in der Datenbank vorkommenden Lagern gewählt werden.

Bestelltransaktionen:

Bestellung anlegen und ändern

Beim Anlegen einer neuen Kundenbestellung soll zunächst der Kopf der Bestellung angelegt werden. Hier soll der Kunde einen Wunschliefertermin angeben können. Anschließend sollen einzelne Positionen mit den Bestellmengen für einzelne Produkte zugewiesen werden können. Nach der Angabe von Produkt und Menge soll die Transaktion automatisch den gültigen Preis aus

der Datenbank (PRODUKT.EINZELVERKAUFSPREIS) lesen und den Gesamtbetrag der Position berechnen. Außerdem sollen in der Transaktion sowohl die Bestell- als auch die Positionstexte eingegeben werden.

Nach Eingabe der erforderlichen Daten sollen zwei Optionen zum Speichern der Bestellung angeboten werden. "Bestellung speichern" soll die Bestellung mit dem Status OFFEN in der Datenbank ablegen. "Bestellung bestätigen" soll zunächst testen, ob der Liefertermin haltbar ist. Wenn ja, soll die Bestellung mit Status BESTAETIGT in der Datenbank abgelegt werden. Falls nein, soll der Bearbeiter eine Fehlermeldung ausgegeben bekommen. Danach kann der Wunschtermin korrigiert werden (falls der Kunde bereit ist, den Termin nach hinten zu verschieben) oder die Bestellung erstmal als OFFEN abgelegt werden. Natürlich kann der Kunde auch einfach abspringen. In diesem Fall muss die Bestellung verworfen werden.

Hier ist der Verkaufsleiter von *Haufkof* eigentlich nicht glücklich und ließ sich erst nach einiger Diskussion davon überzeugen, dass diese Lösung ausreichend ist. Die Angabe des nächstmöglichen Liefertermins würde dem Sachbearbeiter hier einiges an Arbeit ersparen.

Nur noch nicht erledigte Bestellungen (STATUS <> 'ERLEDIGT') können geändert werden. Ist die Bestellung offen, können alle Felder geändert werden. Ist die Bestellung bereits bestätigt, muss vor dem Abspeichern überprüft werden, ob der aktuelle Liefertermin noch haltbar ist. Hier sollen die selben Optionen wie beim Anlegen angeboten werden. Das Datum der Änderung soll im Feld AENDERUNGSDATUM abgelegt werden.

Bestellung beliefern

Eine Bestellung wird nur beliefert, wenn der angegebene Liefertermin bereits erreicht wurde. Ist dies der Fall, soll die Transaktion die entsprechenden Waren aus den Lagern ausbuchen und die Bestellung auf den Status ERLEDIGT setzen. Außerdem soll das Feld ERLEDIGT_TERMIN mit dem aktuellen Datum gefüllt werden.

Hinweise:

Für die Generierung von eindeutigen Werten für KUNDE.KID oder BESTELLUNG.BSTID erzeugen und verwenden Sie pro Tabelle ein Schema-Objekt vom Typ SEQUENCE. Stellen Sie dabei sicher, dass diese korrekt initialisiert werden (d.h. keinen bereits in den Testdaten existierenden Schlüssel generieren).

Das Systemdatum können Sie über die SQL-Funktion SYSDATE abrufen. Zu einem gegebenen Datum *day* (Datentyp DATE) können Sie mit '*day* + *k*' einfach *k* Tage addieren.

Beim Bestätigen einer Bestellung soll die Wartezeit des Users möglichst gering gehalten werden.

Achten Sie darauf, dass beim gleichzeitigen Anlegen von Bestellungen freier Bestand nicht doppelt vergeben wird.

1.4 Beschreibung der Standardauswertungen

Produktanalyse:

Für alle Produkte eines bestimmten Typs und einer bestimmten Größe sollen folgende Daten errechnet und ausgegeben werden:

(1) Name, Hersteller, Marke, Einzelverkaufspreis

(2) den billigsten Einkaufspreis mit dessen Lieferanten (LID und Name),
den teuersten Einkaufspreis mit dessen Lieferanten (LID und Name),
den durchschnittlichen Einkaufspreis (betrachten Sie hierbei nur die aktuell angebotenen Preise und nicht die Preise, die in Zulieferungen genannt werden)

(3) den vorraussichtlichen Bestand am 31.12.2010

(4) den Umsatz im Jahr 2010 (erzielte Einkünfte durch Bestellungen des Produkts, die bestätigt oder erledigt sind und einen Bestelltermin zwischen dem 1.1. und dem 31.12. haben)

(5) die Kosten für Zulieferungen im Jahr 2010 (betrachten Sie hierfür alle Aufträge, die einen Liefertermin im Jahr 2010 haben, da der Grundsatz "Bezahlung bei Lieferung" gilt und die Lieferanten termingerecht liefern)

(6) Ertrag des Produkts ohne Bestände = Umsatz - Kosten,
Ertrag des Produkts unter Berücksichtigung der Bestände =
Umsatz - (Kosten + (Anfangsbestand - Endbestand) * (durschnittlicher Einkaufspreis))

Senken der Lieferkosten:

Geben Sie für ein angegebenes Produkt aus, wieviel Beschaffungskosten sich sparen ließen, wenn alle laufenden Zulieferungen (Status = 'OFFEN') nur vom billigsten Lieferanten bezogen werden würden. Berechnen Sie außerdem, wie das den Ertrag des Jahres 2010 verändern würde, also:
fiktiver Produktertrag - tatsächlich erwarteter Produktertrag.

Achten Sie darauf, dass das System die beiden Analysen schnell durchführen kann. Die Manager, die diese Funktionalität angefordert haben, sind ungeduldig und entscheidend für die Azeptanz des Projektes im oberen Führungskreis.

Hinweise:

- Testen und optimieren Sie Ihre SQL-Anfragen zunächst über *SQL*PLUS* bzw. im *SQL Developer*, bevor Sie diese in Ihr Anwendungsprogramm integrieren.
- **WICHTIG:** Für die Abnahme des Projektes ist der Original-Datenbestand maßgeblich.
- Wichtige Tuning-Daten zu SQL-Anfragen erhalten Sie mit folgenden *SQL*PLUS*-Befehlen:

  1. **`set timing on|off`**
     Gibt nach jeder SQL-Anfrage die benötigte Realzeit in Millisekunden aus.
  2. **`set autotrace on|off`**
     Gibt nach jeder SQL-Anfrage den vom Optimierer generierten Ausführungsplan (Execution Plan) und Informationen über die Anfragebearbeitung (Statistics) aus. Die Anzahl der logischen Blockzugriffe ergibt sich aus db block gets + consistent gets, die Anzahl der physischen Blockzugriffe (Disk-I/O) steht unter physical reads. Die Autotrace-Option setzt voraus, dass Sie in Ihrem Schema die Tabelle `plan_table` (siehe Skript `$DBPRAKT_HOME/Beispiele/SQL/utlxplan.sql`) angelegt haben.
  3. **`explain plan`**
     Manchmal ist es sinnvoll, nur den Ausführungsplan einer SQL-Anfrage zu bestimmen, ohne diese selbst bearbeiten zu lassen (z.B. zur Optimierung einer laufzeitintensiven Anfrage). Dies erreichen Sie mit `explain plan for (select ...)`. Den Ausführungsplan erhalten Sie hier aber nicht automatisch (wie unter Hinweis 2.), sondern Sie müssen diesen eigens aus dem `plan_table` auslesen. Dazu verwenden Sie am besten den View `plan_table_v` (siehe Skript `$DBPRAKT_HOME/Beispiele/SQL/plan_table_v.sql`):

     ```
     select * from plan_table_v;
     ```

     Beachten Sie, dass vor jedem `explain plan` alle Tupel aus der Tabelle `plan_table` von Hand gelöscht werden müssen:

     ```
     delete from plan_table;
     ```

2. Zeitplan für das Projekt

Hier sind ein paar Eckdaten für den Ablauf des Projekts aufgelistet:

- Ausgabe des Projekts ist der 3. Dezember 2009, die Altdaten werden voraussichtlich ab dem 4. Dezember zur Verfügung stehen.
- Für die Bearbeitung des Projekts wird genügend Zeit zur Verfügung stehen, es wird insbesondere keine Zwischenabnahme geben.
- Vor Weihnachten werden wir eventuell nützliche Zwischenresultate der ersten Schritte (wie z. B. Tupelzahlen) veröffentlichen, damit im weiteren Ablauf mit den richtigen Werten gearbeitet wird.
- Während der Bearbeitungszeit wird (bis auf die Vortragstage) kein Plenum stattfinden. Wir stehen euch aber donnerstags in der regulären Zeit von 14-17 Uhr für Fragen in einer Sprechstunde zur Verfügung.
- **Zwei Wochen** vor der Abgabe sollten die Applikationen lauffähig sein: hier beginnt die **Beta-Test-Phase**. In dieser Phase wird die Applikation jeder Gruppe von einer anderen Gruppe auf Lauffähigkeit, Konsistenz und Ergebniskorrektheit getestet. Für diese Tests ist **eine Woche** Zeit. Wir werden dazu Testbögen auf unsere Webseite stellen, welche die wichtigsten Eigenschaften beinhalten, die zu testen sind bzw. welche die Systeme erfüllen sollen. Nach Ablauf der Beta-Tests werden die Applikationen zusammen mit den ausgefüllten Testbögen direkt an die Gruppen zurückgegeben. **Die Testbögen sollen auch an uns geschickt werden.** Danach bleibt wiederum **eine Woche** Zeit, Fehler zu bereinigen.
- Die Termine für die Beta-Test-Phase sind:
  - Donnerstag, 28. Januar 2010 für die Übergabe der zu testenden Anwendungen,
  - Donnerstag, 4. Februar 2010 für die Rückgabe mit den ausgefüllten Testbögen.
- Welche Gruppe die Applikation welcher anderen Gruppe zum Testen bekommt, wird von uns kurz vorher bekanntgegeben!
- Wichtig: Natürlich erlauben wir keine fremden Zugriffe auf die Daten der Praktikumsteilnehmer. Daher wird vor dem Austausch der Anwendungen pro Gruppe je ein Account generiert, in welchen die aktuellen Relationen der Gruppe kopiert werden sollen. Der Zugriff der Applikation auf die Datenbank soll dann über diesen "öffentlichen" Account erfolgen.
- Die Abnahme der Applikationen findet schließlich am Donnerstag, den 11. Februar 2010 statt. Den genauen Zeitplan hierfür werden wir noch ausgeben.

3. Beschreibung der Altdaten

Die Daten aus dem Altsystem sind bereits in der Datenbank unter dem Schema PROJEKT_2009 und haben folgende Formate und Bedeutungen:

KUNDEN:

NID	NUMBER(10)	ID des Landes
NATIONNAME	CHAR (25)	Name des Landes des Kunden
KID	NUMBER (10)	ID des Kunden
KUNDENNAME	VARCHAR2 (25)	Name des Kunden
ADRESSE	VARCHAR2 (40)	Adresse des Kunden
TEL	CHAR (15)	Telefonnummer der Kunden
KONTO	NUMBER (12,2)	Kontostand des Kunden
BRANCHE	CHAR (10)	Branche des Kunden
BSTID	NUMBER (10)	Bestellnummer
BESTELLTERMIN	DATE	Wunsch/Bestelltermin der Bestellung
BESTELLTEXT	VARCHAR2 (256)	Bestellkopftext der Bestellung
ANLAGEDATUM	DATE	Anlagedatum der Bestellung
AENDERUNGSDATUM	DATE	Änderungsdatum der Bestellung
ANLEGER	VARCHAR2 (12)	Bearbeiter der Bestellung angelegt hat
ERLEDIGT_TERMIN	DATE	Tag an dem Bestellung geliefert wurde
STATUS	VARCHAR2 (20)	Status der Bestellung
POSNR	NUMBER (10)	PositionsNr. in Bestellung
PID	NUMBER(10)	bestellte ProduktID der Position
ZAHL	NUMBER (10)	Mengenangabe der Position
PREIS	NUMBER (10,2)	Einzelverkaufspreis pro Stück
POSTXT	VARCHAR2 (120)	Positionstext

PRODUKTE:

LAGID	NUMBER	ID des Lagers
ADRESSE	VARCHAR2(256)	Adresse des Lagers
PID	NUMBER(10)	ID des gelagerten Produkts
NAME	VARCHAR2 (55)	Name des Produkts
HERST	CHAR (25)	Hersteller des Produkts
MARKE	CHAR (10)	Marke des Produkts
TYP	VARCHAR2 (25)	Typ des Produkts
GROESSE	NUMBER	Größe des Produkts
KONFEKT	CHAR (10)	Konfektionsgröße des Produkts
EVPREIS	NUMBER	Listenpreis pro Stück des Produkts
ANZAHL	NUMBER	Bestand auf diesem Lager

LIEFERANT:

LID	NUMBER (10)	ID des Lieferanten
NAME	CHAR (25)	Name des Lieferanten
ADRESSE	VARCHAR2 (40)	Adresse des Lieferanten
NID	NUMBER (10)	ID des Landes des Lieferanten
TEL	CHAR (15)	Telefonummer des Lieferanten
KONTO	NUMBER (12,2)	Kontostand beim Lieferanten
PID	NUMBER (10)	Produkt, das Lieferant verkauft
PREIS	NUMBER (10,2)	Einzelbezugspreis beim Lieferanten

ZULIEFERUNG:

ZLID	NUMBER (10)	ID der Zulieferung
LID	NUMBER (10)	ID des Lieferanten
LAGID	NUMBER (10)	ID des Ziellagers
LIEFERTERMIN	DATE	Termin, an dem Ware eintrifft
LIEFERTEXT	VARCHAR2 (256)	Liefertext der Zulieferung
ANLAGEDATUM	DATE	Datum, an dem Zulieferung angelegt wurde
AENDERUNGSDATUM	DATE	Datum, an dem Zulieferung geändert wurde
ANLEGER	VARCHAR2 (12)	Bearbeiter der Zulieferung angelegt hat
STATUS	VARCHAR2 (20)	Status der Zulieferung
ERLEDIGTDATUM	DATE	Datum, an dem Zulieferung erledigt wurde
POSNR	NUMBER (10)	Nummer der Position in Zulieferung
PID	NUMBER(10)	ID des Produkts, das geliefert wird
POSTXT	VARCHAR2 (120)	Positionstext
PREIS	NUMBER (10,2)	Einzeleinkaufspreis
ZAHL	NUMBER (10)	Anzahl des Produkts

Viel Erfolg beim Implementieren!

Abnahme ist voraussichtlich am 11.02.2010.