TEIL II SQL und ORACLE

Der zweite Praktikumsteil befasst sich mit SQL unter der Datenbank ORACLE 9i. Dazu sind einige Vorbereitungen und Erläuterungen notwendig.

Das Programm JSQL

JSQL ist ein SQL-Editor, mit dem Sie auf eine beliebige relationale Datenbank zugreifen können, für die es einen JDBC-Treiber (JDBC: Java Data Base Connectivity) gibt.
Im Praktikum verwenden wir die Datenbank ORACLE 9i (9.0.1) auf einem Linux-System (SuSE 7.1)

In den Räumen 9-029 und 9-030 lässt sich JSQL über Start/Programme/Tools/JSQL/Verknüpfung mit JSQL.bat starten. Diese Batch-Datei setzt die Java-Pfade und startet JSQLClient.

Bedienung von JSQL

Um auf eine Datenbank zugreifen zu können muss in JSQL zunächst der gewünschte Treiber registriert werden:
Wählen Sie unter dem Menü-Punkt "Driver" den Treiber "oracle.jdbc.driver.OracleDriver"

Jetzt können Sie eine Verbindung zur Datenbank herstellen:
Wählen Sie unter dem Menü-Punkt "Connection" den Eintrag "Connect to database" und trage Sie folgendes ein:

Database URL: jdbc:oracle:thin:@widbsrv4.fh-reutlingen.de:1521:dbsrv4
User Name: wi***xx*** (***xx***: die Nummer erhalten Sie bei Ihrem Betreuer)
Password: wi***xx***

Nach der Anmeldung ändern Sie bitte Ihr Oracle-Kennwort:
Tragen Sie dazu im Befehlsfenster (oberes Fenster) des JSQL folgendes ein:

```
grant connect to wixx identified by <neues_passwd>;
```

!ACHTUNG! Das Passwort wird im Klartext angezeigt!

Markieren Sie den Text oder setzen Sie den Cursor in die Zeile. Wählen Sie unter dem Menü-Punkt "Commands" den Eintrag "Execute". Alternativ geht auch „CTRL – E“.
Jetzt können weitere SQL-Befehle eingegeben werden. Die Ergebnisse werden im mittleren Fenster (Ergebnisfenster) angezeigt, wenn es sich um Daten handelt. Fehler- und Statusmeldung werden im unteren Fenster (Statusfenster) ausgegeben.
Die Syntax von SQL ist im Anhang A in Kurzform wiedergegeben. Eine ausführliche Syntax der SQL-Befehle bietet die Online Hilfe von JSQL

Einige Bedienungshilfen erleichtern die Eingabe von Befehlen:

<ctrl-E> := Execute (SQL-command at cursor position or selected commands)
<ctrl-K> := Kill execution thread (cancel SQL-command)
<ctrl-A> := Select all (mark all text)
<ctrl-C> := Copy (selected text to clipboard)
<ctrl-X> := Cut (selected text to clipboard)
<ctrl-V> := Insert (text from clipboard)

Sie können ein SQL-Skript (=beliebige Textdatei mit SQL-Befehlen) im Befehlsfester öffnen (Menü-Punkt "File"-"Open") und danach ausführen oder Ihre eingegebenen Befehle als Datei abspeichern (Menü-Punkt "File"-"Save As").

Wenn Sie die Aufgaben des Praktikums lösen, vergessen Sie bitte nicht, Ihre Lösungen (SQL Befehle) zu speichern. Ihre gespeicherten Lösungen aller Aufgaben sind am Ende des Praktikums als Text-Datei abzugeben.

Aufgabe 1) (Tabelle anlegen und Daten eingeben [create table, insert into])

a) Rufen Sie den SQL-Editor „JSQL“ auf, laden den Oracle-Treiber aus dem Menü Driver und stellen eine Verbindung mit jdbc:oracle:thin:@dblabor2.fh-reutlingen.de:1521:db2 her. Melden Sie sich unter Ihrer ORACLE-Kennung „wi***xx***“ an.

b) Legen Sie eine Tabelle PFLANZEN an indem Sie folgenden SQL-Befehl eintippen :

```
create table PFLANZEN (ART_CODE  char(3) primary key,
                       PFLANZENNAME      varchar(32) not NULL,
                       SORTE             varchar(10) not NULL,
                       FARBE             varchar(8),
                       HOEHE             integer check (hoehe > 0),
                       PREIS             numeric(6,2) not NULL);
```

Hinweis: Mit diesem Befehl wird eine Tabelle PFLANZEN mit den Attributen ART_CODE, PFLANZENNAME etc. definiert. Der Datentyp *varchar(n)* definiert eine variable Zeichenfolge mit bis zu n Zeichen im Gegensatz zu *char(n)* bei dem mit Leerzeichen aufgefüllt wird. Der Datentyp numeric(n,m) definiert eine Zahl mit maximal n Ziffern, wobei die m Nachkommastellen mitgezählt werden; d.h. numeric(6,2) bedeutet 4 Vorkomma und 2 Nachkommastellen. Für die Attributdefinition wurden folgende Festlegungen verwendet:

primary key	= Ident-Schlüssel
not NULL	= Pflichtfeld
check (Beding.)	= Konsistenzbedingung

Eine detaillierte Beschreibung der Bezeichner (<identifier>), Konstanten (<literal>), Datentypen (<data type>) und Konsistenzbedingungen (<constraint>) finden Sie im SQL_AnhangA.

Ist der Spaltenname „HÖHE“ oder „PFLANZEN-NAME“ erlaubt?

c) Speichern Sie folgende Pflanzen mit:

```
insert into pflanzen values ('S01', 'ROSE', 'STAUDE',NULL, 150, 9.90 );
insert into pflanzen values ('B01', 'TULPE', 'BLUME',NULL, 50, 2.50 );
insert into pflanzen values ('B02', 'VEILCHEN', 'BLUME',NULL, 10, 1.95;
```

d) Überprüfen Sie, ob alle Tupel gespeichert wurden:

```
select * from PFLANZEN;
```

e) Zeigen Sie die Tabellenbeschreibung von PFLANZEN an:

```
select * from USER_TAB_COLUMNS where TABLE_NAME = 'PFLANZEN';
```

Hinweis: Die Tabelle USER_TAB_COLUMNS gehört zum Data Dictionary (Datenkatalog) der Oracle-Datenbank, das Auskunft über das aktuelle Datenbankschema gibt. In Aufgabe 8 lernen Sie weitere Befehle kennen, mit denen Sie den Datenkatalog abfragen können.

f) Erklären Sie den Unterschied zwischen d) und e).

g) Laden Sie die Tabelle PFLANZEN mit weiteren Daten :

Speichern Sie Ihre „SQL“-Datei!
Öffnen Sie die Datei **pflanzen.dat** in JSQL und führen Sie sie aus.

Aufgabe 2) (Einfache Datensuche [select ... from ... where ...])

Beispiel: Suchen Sie alle Pflanzen (ART_CODE, PFLANZENNAME, SORTE, PREIS) mit einem Preis zw. 2 und 10 €.

```
select ART_CODE, PFLANZENNAME, SORTE, PREIS from PFLANZEN
where PREIS >= 2 and PREIS <= 10;
```

a) Selektieren Sie alle Pflanzen, die billiger als 5 € sind.

b) Selektieren Sie alle Pflanzen, die billiger als 5 € sind. Geben Sie nur PFLANZENNAME und PREIS aus.

c) Suchen Sie alle Pflanzen (ART_CODE, PFLANZENNAME), deren Name mit B oder W beginnt.

d) Für welche Pflanzen (ART_CODE, PFLANZENNAME) wurde keine FARBE angegeben?

Aufgabe 3) (Abfragen mit Funktionen und Gruppierungen [select ..,fkt(..),.. from ... where; select ... from ... where ... group by ... having ...])

Beispiel: Welches ist die maximale Höhe der Bäume in der Tabelle PFLANZEN ?

```
select max(HOEHE) from PFLANZEN
where SORTE = 'BAUM';
```

a) Suchen Sie die teuerste Pflanze (ART_CODE, PFLANZENNAME) der Sorte ‚BLUME'.

b) Der Mittelwert der Preise aller Buschpflanzen (SORTE = ‚BUSCH') ist zu ermitteln.

c) Welche Blütenfarben (FARBE) gibt es in der Pflanzentabelle. Bei der Ausgabe soll sich die Farbe nicht wiederholen.

Beispiel: Wieviel Pflanzen gibt es von jeder Sorte? Ausgabe: SORTE, Anzahl Pflanzen.

```
Select SORTE, count(*) from PFLANZEN
group by SORTE;
```

d) Der Mittelwert, Minimal- und Maximalpreise aller Pflanzensorten ist zu ermitteln.
Ausgabe: SORTE, Mittelwert(Preis), Minimalwert(Preis), Maximalwert(Preis)

e) Ermitteln Sie die Sorten, deren Differenz zw. Minimal- und Maximalpreis größer als 10 € ist.
Ausgabe: SORTE

Aufgabe 4) (Abfragen mehrerer Tabellen [Verbundoperation: join])

a) Erstellen Sie folgende Tabellen MONATE, LIEFERANTEN, BESTELLUNGEN, BESTELLDATEN und ANGEBOTE. Die Felder sind nachfolgend charakterisiert :

MONATE

NR	num	2 Stellen (Identifikationsschlüssel)
MONAT	alpha	var. Länge, max. 10 Stellen, erforderlich

LIEFERANTEN

LFR_CODE	alpha	3 Stellen (Identifikationsschlüssel)
LFR_NAME	alpha	var. Länge, max. 20 Stellen, erforderlich
ADRESSE	alpha	var. Länge, max. 35 Stellen

BESTELLUNGEN

BESTELLNR	num	4 Stellen (Identifikationsschlüssel)
LFR_CODE	alpha	3 Stellen (Fremdschlüssel zu LIEFERANTEN)
B_DATUM	Datum	erforderlich
L_DATUM	Datum	L_DATUM >= B_DATUM
BETRAG	num	8 Vor-, 2 Nachkommastellen, erforderlich, BETRAG > 0

Hinweis: Prüfen Sie die Syntax für die Kommazahlen und der Konsistenzbedingungen (vgl. Anhang A).

BESTELLDATEN

BESTELLNR	num	4 Stellen (Fremdschl. zu BESTELLUNGEN)
ART_CODE_LFR	alpha	5 Stellen Identschlüssel: BESTELLNR, ART_CODE_LFR
ANZAHL	num	3 Stellen, erforderlich, ANZAHL > 0
BESTELLPREIS	num	6 Vor- , 2 Nachkommastellen, erforderlich, BESTELLPREIS > 0

ANGEBOTE

LFR_CODE	alpha	3 Stellen(Fremdschlüssel zu LIEFERANTEN)
ART_CODE_LFR	alpha	5 Stellen Identschlüssel: LFR_CODE, ART_CODE_LFR
ART_CODE	alpha	3 Stellen(Fremdschlüssel zu PFLANZEN)
LFR_ZEIT	num	2 Stellen(Fremdschlüssel zu MONATE)
ANG_PREIS	num	6 Vor- , 2 Nachkommastellen, erforderlich, ANG_PREIS > 0

Ergänzen Sie die Tabelle Pflanzen um den Blütenzeitraum:

PFLANZEN

BL_B	num	2 Stellen (Monatsnummer für Blütenbeginn)
BL_E	num	2 Stellen (Monatsnummer für Blütenende)

b) Tragen Sie die zur Verfügung gestellten Daten in Ihre eben erstellten Tabellen ein:

Speichern Sie Ihre Datei!
Öffnen Sie in JSQL nacheinander folgende Dateien und führen Sie diese aus:
monate.dat, lieferanten.dat, bestellungen.dat, bestelldaten.dat, angebote.dat, pflanzen_4b.dat

Überprüfen Sie die korrekte Übernahme der Daten im Statusfenster!

c) Zeichnen Sie ein Bachman-Diagramm für ihre Datenbank.

d) Was passiert, wenn man folgendes eintippt ?

```
select * from BESTELLUNGEN, BESTELLDATEN;
```

e) Was geschieht, wenn Sie folgene WHERE-Bedingung hinzufügen:

```
where BESTELLUNGEN.BESTELLNR = 186 AND BESTELLDATEN.BESTELLNR = 186
```

f) Wie kann man erreichen, dass für alle Bestellungen nur die zugehörigen Bestelldaten angezeigt werden?

Hinweis: der Ausdruck in der WHERE-Bedingung kann auch eine Variable anstellen eines festen Wertes enthalten.

Beispiel: Suchen Sie die Bestellung mit der Bestellnr = 191 und zeigen Sie alle Attribute des Bestellkopfes und der Positionen an.

```
Select * from BESTELLUNGEN b, BESTELLDATEN d
where b.BESTELLNR = d.BESTELLNR and b.BESTELLNR = 191;
```

g) Ermitteln Sie alle Bestellungen mit den zugehörigen Bestelldaten (BESTELLNR, B_DATUM, BETRAG, ART_CODE_LFR, ANZAHL, BESTELLPREIS) beim Lieferanten ‚004'.

h) Gesucht wird eine Übersicht offener Bestellungen (BESTELLNR, LFR_CODE, LFR_NAME, B_DATUM, L_DATUM, BETRAG) beim Lieferanten ‚004'.

Beispiel: Wie unterscheiden sich die Bestellpreise der bisherigen Bestellungen von den heutigen Angebotspreisen? In der Liste sind BESTELLNR, ART_CODE_LFR, ART_CODE und die positive oder negative Abweichung zum Angebotspreis anzugeben.

```
Select b.BESTELLNR, d.ART_CODE_LFR, ART_CODE, BESTELLPREIS – ANG_PREIS
     from BESTELLUNGEN b,BESTELLDATEN d, ANGEBOTE a
     where b.BESTELLNR = d.BESTELLNR and a.LFR_CODE = b.LFR_CODE
     and a.ART_CODE_LFR = d.ART_CODE_LFR;
```

i) Woher können Staudenpflanzen bezogen werden? Bitte ART_CODE, PFLANZENNAME, LFR_CODE und LFR_NAME angeben und nach ART_CODE sortieren.

j) Welche Artikelcodes der angebotenen Pflanzen gleichen irgendeinem Artikelcode eines Lieferanten? Angabe von ART_CODE, PFLANZENNAME, ART_CODE_LFR, LFR_NAME.

k) Ermitteln Sie alle Bestellungen mit zugehörigem Lieferantennamen und Adresse. Bitte geben Sie BESTELLNR, LFR_NAME, ADRESSE, B_DATUM und BETRAG sortiert nach BESTELLNR aus.

Aufgabe 5 (komplexe Abfragen)

a) Suchen Sie den Gesamtwert der Bestellungen nach Lieferanten gruppiert. Ausgabe von LFR_CODE, LFR_NAME und SUMME(BETRAG).

b) Der Mittelwert, Minimal- und Maximalpreise aller Bestellungen ist pro Lieferant zu ermitteln. Ausgabe: LFR_CODE, LFR_NAME, AVG(BETRAG), MIN(BETRAG) und MAX(BETRAG) sind auszugeben.

c) Gesucht wird eine Übersicht der Pflanzen, deren Verkaufspreise mindestens 100% über dem teuersten dazugehörigen Angebotspreis liegen. Ausgabe von ART_CODE, PFLANZENNAME, PREIS, MAX(ANG_PREIS)

Beispiel: Welche Bestellungen (BESTELLNR, BETRAG, Summe der BESTELLPREISE) haben mehr als 3% Preisnachlass erhalten ?

```
select B.BESTELLNR, BETRAG, sum(D.BESTELLPREIS*D.ANZAHL)
from BESTELLUNGEN B, BESTELLDATEN D
where B.BESTELLNR = D.BESTELLNR
and B.BETRAG <
(select 0.97 * SUM(ANZAHL * BESTELLPREIS) from BESTELLDATEN
      where BESTELLNR = B.BESTELLNR)
      group by B.BESTELLNR, B_DATUM, BETRAG;
```

d) Welche Pflanzen (ART_CODE, PFLANZENNAME) werden vom Lieferanten mit dem größten Sortiment (Anzahl Artikel) angeboten.

e) Listen Sie alle unterschiedlichen Angebote (verschied. ART_CODE) der Lieferanten auf. Die Anzeige soll die folgende Form haben :

ART_CODE PFLANZENNAME LFR_CODE LFR_NAME

Beispiel: Welche Lieferanten (LFR_CODE, LFR_NAME) kommen in den Bestellungen nicht vor?

```
select LFR_CODE, LFR_NAME from LIEFERANTEN
where not exists
( select * from bestellungen
      where LIEFERANTEN.LFR_CODE = bestellungen.lfr_code );
```

f) Erstellen Sie eine ABC-Analyse der Lieferanten bezüglich des Umsatzes. Die Grenzen sind: A = 0-50 %, B = 50-85%, C = 85-100%. Das Ergebnis sollte folgendermaßen aussehen:

Kategorie	Lief.Nr.	Lieferantenname	Umsatz	Anteil_%
A	003	Franzki H.	900	22,4 %
A	014	Dezaier L.J.	650	16,2 %
B	009	Baungarten	600	15,0 %
B	013	Spitzmann	530	13,2 %
B	020	Bluem L.Z.	421	10,5 %
C	004	Hoven G. H.	319,5	8,0 %
C	002	Bioland Gaiser	250	6,2 %
C	022	Erika GmbH	240	6,0 %
C	019	Schoene F.A.	100	2,5 %

Hinweis: Erstellen Sie zuerst einen View für den Jahresumsatz eines jeden Lieferanten (View über zwei Tabellen). Dann ermitteln Sie die laufende Summe des Umsatzes jedes Lieferantens (View oder Named Query mit laufender Summe, sortiert nach Größe des Umsatzes). Schließlich formulieren Sie eine Abfrage mit einer virtuellen Spalte Kategorie. Verwenden Sie hierzu die SQL-92 Syntax CASE...WHEN…THEN um die Werte für die Kategorie zu ermitteln.

Aufgabe 6) (DB-Schema ändern, Datenunabhängigkeit)

SQL bietet die Möglichkeit, Definitionen von Tabellen (d.h. das Datenbankschema) online zu ändern.

a) Ergänzen Sie die Tabelle Pflanzen um das Attribut Standort (STANDORT), wobei Standort nur ein Wert aus ('SCHATTEN', 'HALBSCHATTEN', 'SONNE') sein darf (check...in (...)).

b) Ergänzen Sie die vorhandenen Datensätze mit "update table". Dazu wird die Datei pflanzen_6b.dat zur Verfügung gestellt. Ermitteln Sie anschließend alle Pflanzen, die blühen (FARBE not NULL) und die in der SONNE oder im HALBSCHATTEN stehen.

c) Ändern Sie die Query von b), so dass der Monatsname für Blütenbeginn und –ende angezeigt wird.

d) Wenn die Namen der Attribute oder Tabellen geändert werden, müssen auch die Queries angepasst werden. Dies bezeichnet man als logische Datenabhängigkeit. Um logisch datenunabhängig zu werden, erstellen Sie folgenden View

```
create view PFLANZENVIEW (ART_NR, PFLANZENNAME, SORTE, FARBE, VKPREIS,
                          BLUETEN_BEGINN, BLUETEN_ENDE, STANDORT)
      as select ART_CODE, ..., PREIS, beginn.MONAT, ende.MONAT, ...
      from PFLANZEN, MONATE beginn, MONATE ende
                    /* die Tab. MONATE erhält 2 Aliasnamen)*/
      where BL_B = beginn.NR and BL_E = ende.NR
      and FARBE is not NULL;
```

Liefert die Abfrage "select * from PFLANZENVIEW" das gleiche Ergebnis wie die Query von d) ? Erklären Sie Ihre Beobachtung.

Im Falle einer Änderung an den Tabellen muss nur der View angepasst werden. Die Abfrage kann unverändert bleiben.

e) Legen Sie für den ART_CODE und den PFLANZENNAMEN einen Index an. Prüfen Sie, ob sich an der Definition für die Tabelle PFLANZEN etwas geändert hat. Löschen Sie den Index und prüfen Sie die Tabelle erneut.

Wenn die fachlichen Datenstrukturen (Tabellendefinitionen) von der internen Speicherung (z.B. Index, Cluster) unabhängig sind, spricht man physischer Datenunabhängigkeit.

Aufgabe 7) (Tabellen für andere Benutzer freigeben, Sperr- und Transaktionsmechanismus)

a) Bilden Sie Gruppen zu 3 Personen. In den folgenden Aufgaben werden die Gruppenmitglieder mit P1, P2 und P3 bezeichnet.

b) P1 soll seine Tabelle PFLANZEN für die beiden anderen Mitglieder P2 und P3 für Selektionen und Updates freigeben.
Hinweis: `grant`-Befehl verwenden.

c) Alle Gruppenmitglieder sollen den gleichen Satz (z.B. `ART_CODE = '031', 'WEIN'`) selektieren. Dann soll P1 den gleichen Satz für einen späteren Update selektieren.
Hinweis: `select .... for update of ...;`

d) P2 und P3 selektieren den gleichen Satz noch einmal. Dann Selektiert P2 den Satz für einen späteren Update. Was passiert, wenn P3 die Selektion noch einmal ohne ‚update of' wiederholt?
Warum ist P2 blockiert?

e) P1 führt den Update durch indem er/sie den Pflanzenname ändert und anschließend überprüfen P1 und P3 die Änderung.

Wie erklären Sie sich die unterschiedlichen Ergebnisse? Welche Schlüsse ziehen Sie daraus bezüglich der Funktionsweise des Sperrmechanismus von ORACLE?

f) P1 bricht die Transaktion ab. Hinweis: rollback;
Was passiert, wenn nun P1 und P2 die Selektion von e) wiederholen?

g) Führen sie die Sequenz c) bis e) noch einmal durch und schließen mit commit ab.
Welche Ergebnisse erhalten Sie dann?

Aufgabe 8) (SQL-1999)

Anmerkung: Die Tabellen die Sie bisher erstellt haben werden nun nicht mehr zwingend benötigt und können gelöscht werden.

WICHTIG: Der JSQL-Editor unterstützt die in den folgenden Aufgaben verwendeten Datentypen nicht vollständig. Aus diesem Grund sollten die Aufgaben an der Console mit dem Oracle-Tool SQL-Plus bearbeitet werden:

Anmeldung an „dblabor2“ per Telnet:	login:	wi
	passwort:	dbprakt
Starten von SQLPlus:	sqlplus	
Anmelden an der Datenbank:	<eigenes login>	wiXX
	<eigenes Passwort>	

a) Bestellnummern haben eine andere Semantik als Zahlen. Um Missverständnisse zu vermeiden ändern Sie die Definition der Bestelltabellen indem Sie die Bestellnummer als benutzerdefinierten Datentyp (UDT) festlegen. Damit ist sichergestellt, dass Sie Bestellnummern nicht addieren oder mit anderen Zahlen vergleichen. (siehe create Type in der *Oracle SQL Reference)*
Hinweis: Dazu ist die Tabelle Bestellungen zu löschen und neu anzulegen. Oracle erlaubt nicht die Abänderung des Datentyps. Beim Einfügen in die neue Tabelle muss der neue Datentyp für die Bestellnummer berücksichtigt werden.
Beispiel: insert into Bestellungen values(BestellNr(13), '002', '25-FEB-92', '04-MAR-92', 250);

Eigentlich würde ein individualisierter (abgeleiteter) Datentyp (distinct datatype) genügen, der jedoch von Oracle 8i nicht unterstützt wird.

Überprüfen Sie folgende Abfragen und erläutern Sie Ihre Beobachtungen:

```
select * from Bestellungen where BestellNr = Betrag;
select * from Bestellungen where BestellNr = 15;

select * from bestellungen where bestellnr = Bestellnrtyp(15);
```

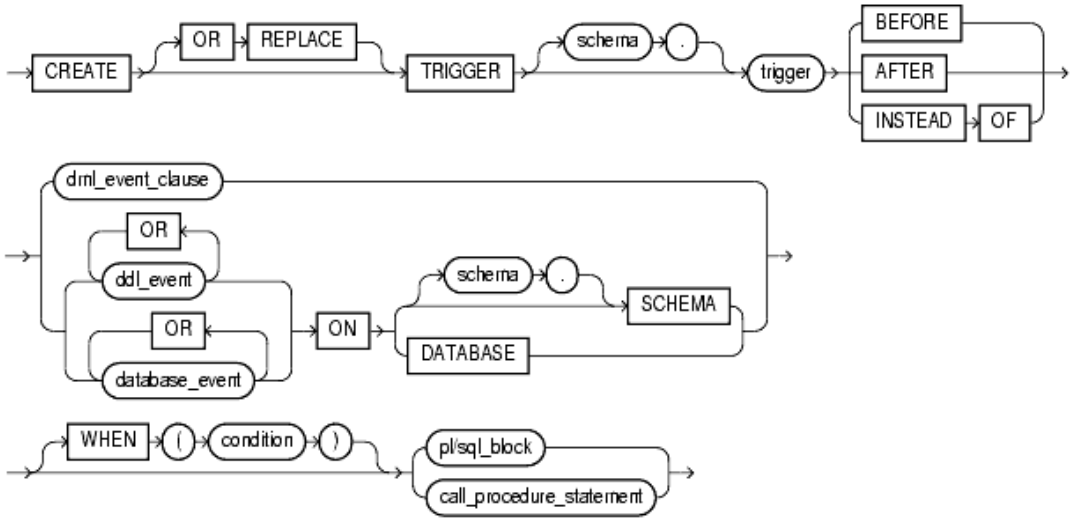
b) Eine weitere semantische Schwäche stellen die Monate für Blütenbeginn und –ende in der Tabelle Pflanzen dar. Zum einen benötigen Sie eine Tabelle mit den Namen der Monate, anderseits brauchen Sie die Monatsnummer um Vergleiche anzustellen. Machen Sie aus Monat einen benutzerdefinierten Datentyp (UDT) mit der gewünschten Semantik.
Hinweis: Wenn Sie Ihre Definition geeignet wählen, können die Pflanzendaten aus der Datei **pflanzen_8b.dat** zum Laden der neuen Tabelle verwendet werden.

c) Verbessern Sie die Adressangabe beim Lieferanten durch eine Strukturierung der Adresse. Die Adresse soll die Felder Strasse_Nr, PLZ und Ort enthalten. Verwenden Sie hierzu die ROW oder CREATE TYPE Syntax (Oracle 8i unterstützt nur CREATE TYPE).
Hinweis: Wenn Sie Ihre Definition geeignet wählen, können die Lieferantendaten aus der Datei **lieferanten_8c.dat** zum Laden der neuen Tabelle verwendet werden.

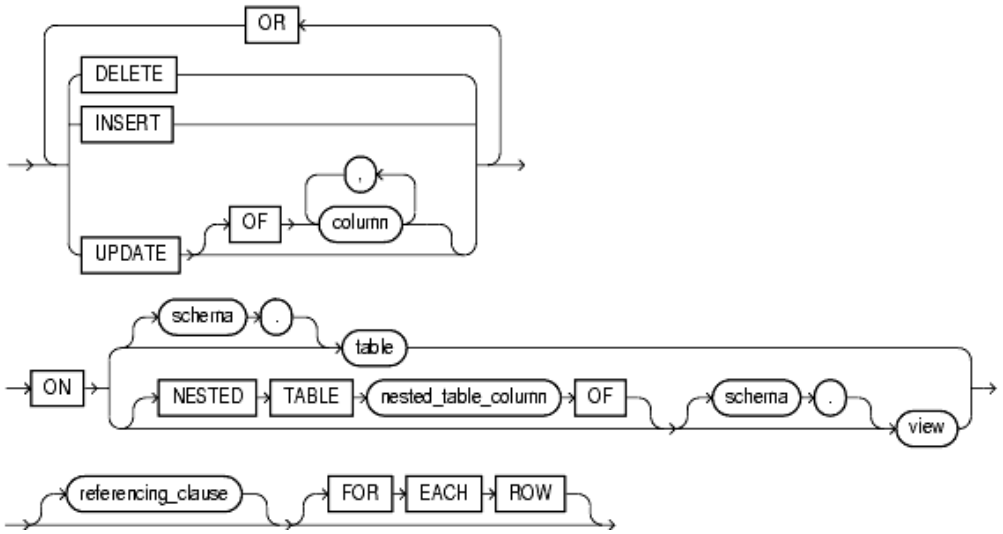
d) Stellen Sie durch die Definition eines Triggers sicher, dass für jede neuen oder geänderten Eintrag in Tabelle PFLANZEN der Verkaufspreis so korrigiert wird, dass er wenigstens 10% und mindestens 0,20 € über dem maximalen Einkaufspreis (ANG_PREIS) liegt. Erfüllt der Verkaufspreis bereits beide Kriterien, bleibt er unverändert

Hinweis: Erstellen Sie mit CREATE TRIGGER, der vor einem Update oder Insert aktiviert wird. Er muss für alle betroffen Zeilen (FOR EACH ROW) definiert werden. Die Triggeraktion ist in PL/SQL zu formulieren (Syntax: [DECLARE <var_decl_list>] BEGIN <pl/sql_statements> END;). In <pl/sql_statements> muss der maximale Einkaufspreis ap := MAX(ANG_PREIS) aus der Tabelle Angebote für die jeweiligen Pflanzen ermittelt werden. Ist dieser Preis kleiner als np := max(ap*1.1, ap +0.2), dann ist der Verkaufspreis (PREIS in Tabelle Pflanzen) auf np anzuheben. Die neuen Werte der Tabelle Pflanzen können mit dem Präfix **:new.**<attr> angesprochen werden. D.h. es muss ein PL/SQL-Statement geben wie: :new.PREIS := np;

Auszug aus der *Oracle 8i SQL Reference:*
trigger_def ::=



.
dml_event_clause ::=



e) Erstellen Sie eine Tabelle Mitarbeiter mit den Attributen, Ma_Nr PRIMARY KEY, Name VARCHAR(20), Position VARCHAR(20), Vorgesetzter (Fremdschlüssel auf Ma_Nr). Laden Sie die Tabelle mit den Daten der Datei **mitarbeiter.dat**.

f) Erstellen Sie eine hierarchische Query, welche alle Mitarbeiter gemäß ihrer Vorgesetzenhierarchie ausgibt. Hinweis: Oracle definiert eine nicht SQL-1999 konforme Syntax. Verwenden Sie den ausgeteilten Anhang A (SQL-Anleitung) bzw. Schauen Sie in der online-Dokumentation des Oracle SQL Reference Manuals (dblabor2.fh-reutlingen.de) nach.

Aufgabe 9) (Data Dictionary)

Wie jede Datenbank besitzt ORACLE auch ein Data Dictionary. Bei einer relationalen Datenbank ist es üblich, daß die Metadaten (Daten über die Datenbank, z.B. die Definition der Tabellen und ihre Beziehungen untereinander) ebenfalls mit SQL-Befehlen abgefragt werden. Durch die Abfrage

```
SQL> select table_name, comments from dictionary;
```

werden alle Systemtabellen des Dictionaries angezeigt. Einige Tabellen sind nachfolgend wiedergegeben :

TABLE_NAME	COMMENTS
ALL_CATALOG	All tables, views, synonyms, sequences accesible to the user
ALL_OBJECTS	Objects accessible to the user
ALL_TABLES	Description of tables accessible to the user
ALL_TAB_COLUMNS	Synonym for ACCESSIBLE_COLUMNS
DICT	Synonym for DICTIONARY
DICTIONARY	Description of data dictionary tables and views
DICT_COLUMNS	Description of columns in data dictionary tables and views
USER_CATALOG	Tables, Views, Synonyms, Sequences accessible to the user
USER_INDEXES	Description of the user‘s own indexes
USER_TABLES	Description of the user‘s own tables
USER_TAB_COLUMNS	Columns of user‘s tables, views and clusters
USER_TAB_PRIVS_MADE	Grants on objects for which the user is the owner, grantor or grantee

Möchte man zum Beispiel wissen, welche Tabellen man bereits angelegt hat, so kann dies mit der Abfrage:

```
SQL> select * from user_tables;
```

ermittelt werden. Analog dazu erhält man die Spaltennamen durch

```
SQL> select * from user_tab_columns
        where table_name = <table_name>;
```

Die Änderung des Data Dictionary erfolgt automatisch durch das Datenbankverwaltungsprogramm (DBMS: DataBase Management System), z.B. wenn eine neue Tabelle angelegt wird.

a) Zeigen Sie alle Tabellen an, die Ihnen gehören.
Hinweis: Verwenden Sie USER_TABLES

b) Ermitteln Sie die Felddefinitionen der Tabelle „PFLANZEN“.
Hinweis: Benutzen Sie USER_TAB_COLUMNS.

c) Welche Attribute besitzt die Tabelle „ALL_TAB_COLUMNS“.

d) Zeigen Sie alle Einträge (OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, DATA_PRECISION) der Tabelle „ALL_TAB_COLUMNS“ an. Begrenzen Sie die Ausgabe auf 20 Einträge mit Hilfe der rownum <= 20 (siehe Anhang A, A12).
Was bedeuten die Angaben der Abfrage.

e) Finden Sie heraus, wer Ihre Tabellen manipulieren darf.
Hinweis: USER_TAB_PRIVS_MADE