Übung 11: (Abfragen auf den Tabellen von Beispiel 1)

Welches Ergebnis liefern die folgenden Abfragen für den ursprünglichen Inhalt der Tabellen (Artikel, Lieferant, liefert) aus Beispiel 1. Bevor Sie sich das Ergebnis z.B. im SQL Developer anzeigen lassen, überlegen Sie bitte zunächst, welches Ergebnis Sie ohne explizites Ausprobieren nennen würden.

Abfrage1:
```
select ANR, Bezeichnung from Artikel
where Laenge*Breite < (select Max(Laenge*Breite)/2 from Artikel)
```

Abfrage2:
```
select Name from Lieferant
where  LNR IN (select LNR from liefert
where  ANR IN (Select ANR from Artikel where Material = 'Metall'))
```

Abfrage3:
```
select   liefert.LNR, Name   from Lieferant, liefert
where  Status = 20 and ANR='S2'
```

Abfrage4:
```
select distinct Bezeichnung from Artikel
where ANR NOT IN  (Select ANR from liefert where LNR IN ( 1 , 9 ) )
```

Abfrage5:
```
select   ANR,Lieferant.LNR   from Artikel, Lieferant
where  Material='Metall' and Ort = 'Jena'  order by LNR
```

Erstellen Sie anschließend für jede der gegebenen Abfragen jeweils eine gespeicherte externe Sicht (CREATE OR REPLACE VIEW ….) mit einem entsprechenden Namen (z.B. uebung11_1 für die erste Abfrage dieser Übung).

Übung 12: (Datenbankbearbeitung mit MS-Access)

Wiederholen Sie die Übung 9 mit dem Datenbanksystem **MS-Access**. Dazu ist eine Access-Datenbank **Uebung12.mdb** gegeben, in der bereits die Tabellen **Artikel**, **Lieferant** und **liefert** angelegt und gefüllt sind.

- Ergänzen Sie die Primär und Fremdschlüssel (Beziehungen) für die Tabellen.
- Studieren Sie den Entwurf (Struktur) und den Inhalt der einzelnen bereits angelegten Tabellen.
- Erstellen Sie anschließend in der MS Access-Datenbank die Abfragen aus Übung 9 und benutzen Sie dabei die angebotene Unterstützung durch die graphische Benutzer-oberfläche (Tabellen-, Feld-Ebene), aber natürlich ohne direkte Eingabe der SQL-Anweisungen.

Übung 13: (PL/SQL-Programmierung, anonyme Blöcke, Oracle-Server)

Diese Übung ist auf dem Oracle-Server zu bearbeiten. Erstellen Sie dort in Ihrem Schema eine neue Tabelle **Lief_Liste** (CREATE TABLE .....) mit zwei Spalten:

Ort	VARCHAR(20) für Ortsangabe wie in **Lieferant.Ort**
Liste	VARCHAR(250) für eine Liste aller Lieferanten aus diesem Ort, d.h. die Namen der Lieferanten sind in alphabetischer Reihenfolge durch Komma getrennt in einer Liste hintereinander aufgeführt

Die Tabelle besitzt weder (Primär)Schlüssel noch Fremdschlüssel und es gibt auch keine sonstigen Bedingungen für die Tabellenspalten.

Diese Tabelle soll nun mit einem PL/SQL-Block entsprechend gefüllt werden, z.B. für die ursprünglichen Basisdaten mit

Ort	Liste
Apolda	Runge
Gera	Mey
Jena	Karcher, Todd
Weimar	Schmidt

Vervollständigen Sie das nachfolgend gegebene Grundgerüst für den PL/SQL-Block. Dabei sind die kommentierten Stellen (…….(…)) durch die entsprechende(n) PL/SQL-Anweisung(en) zu ersetzen:

```
DECLARE

  CURSOR Lief_Lesen IS …….. (Lieferanten-Daten bereitstellen)

BEGIN
   …….. (Löschen alle Zeilen in Lief_Liste)

   FOR …… (For-Schleife über den Cursor Lief_Lesen)
   LOOP
      …… (Ändern der entsprechende „Orts-Zeile“ in Lief_Liste durch
                                    Anhängen des Namens des Lieferanten an die Spalte Liste)

      IF SQL%NOTFOUND
      THEN
         …….. (Einfügen der ersten „Orts-Zeile“ in Lief_Liste)
      END IF;

   END LOOP;
END;
/                          wichtig, hier wird der PL/SQL-Block an den Server übergeben
```

Fassen Sie das Erstellen und das Füllen der neuen Tabelle **Lief_Liste** in einer PL/SQL-Skriptdatei **uebung13.sql** zusammen, führen Sie die PL/SQL-Skriptdatei aus und überprüfen das Ergebnis in ihrem Schema.

Übung 14: (PL/SQL-Programmierung, Funktionen, Oracle-Server)

Erstellen Sie eine PL/SQL-Funktion **BEST_LNR** mit:

- einem Übergabeparameter (Datentyp entsprechend Artikel.ANR)
- einem Rückgabewert (Datentyp entsprechend Artikel.LNR)

die für eine gegebene Artikelnummer entweder eine Lieferantennummer liefert, die den Artikel zum kleinsten Preis liefert oder NULL, wenn der Artikel von keinem Lieferanten geliefert wird.

Benutzen Sie dabei den anonymen PL/SQL-Block aus **init_LNR_ORA.sql** als Ausgangspunkt für die Programmierung und ergänzen ihn so, dass auch der Fall abgedeckt wird, dass es keinen Lieferanten gibt, der den Artikel liefert.

Übung 15: (PL/SQL-Programmierung, Datenbanktrigger, Oracle-Server)

Ein Datenbanktrigger ist ein aktives Datenbankobjekt, das Änderungen (Insert, Update, Delete) für eine Tabelle überwachen kann und bei jeder getriggerten Änderung ohne weitere Aktivitäten durch den Anwender auf dem Datenbankserver einen PL/SQL-Block ausführt.

Erstellen Sie die PL/SQL-Skriptdatei **uebung15.sql**, in der zunächst eine neue Tabelle **ART_PROTO** mit den vier Spalten (Datentypen siehe unten)

- **USER_NAME**
- **DATUM_ZEIT**
- **ANZ_ART**
- **ART_ANR**

angelegt und anschließend ein **Datenbanktrigger** erstellt wird**,** der bei jeder **Neuanlage** eines **Artikels** einen Eintrag (eine Zeile) in die Tabelle **ART_PROTO** schreibt.

Dabei sind die einzelnen Spalten der Tabelle ART_PROTO durch:

- **USER_NAME** VARCHAR2(16)
  Name des Oracle-Benutzers, der ein Insert auf die Tabelle Artikel ausführt. Der aktuelle Wert wird **von der Systemfunktion USER** geliefert.
- **DATUM_ZEIT** VARCHAR2(20)
  aktuelles Tagesdatum einschließlich Uhrzeit. Der aktuelle Wert wird von der **Systemfunktion SYSDATE** geliefert. Verwenden Sie für die Zuweisung die interne Funktion TO_CHAR(SYSDATE, 'DD.MM.YYYY HH24:MI.SS'), die das aktuelle Datum und die Uhrzeit formatiert ausgibt.
- **ANZ_ART** DECIMAL(3)
  Anzahl der vorhandenen Artikel einschließlich der neu angelegten Artikel
- **ART_ANR** Datentyp von Artikel.ANR
  größte Artikelnummer nach der Neuanlage

definiert.

Sie können die Systemfunktionen USER und SYSDATE auch zunächst (z.B. in SQL*PLUS) mit einfachen Select-Anweisungen ausprobieren, so liefert

SELECT DISTINCT **USER** Benutzer FROM Lieferant;

eine Abfrage mit genau einer Spalte und einer Zeile (die Spaltenüberschrift wurde in Benutzer umbenannt)

Benutzer
SS12ETXYZ

und

SELECT DISTINCT TO_CHAR(**SYSDATE**,'DD.MM.YYYY HH24:MI:SS') Datum FROM Lieferant ;

eine Abfrage mit genau einer Spalte und einer Zeile (die Spaltenüberschrift wurde in Datum umbenannt), z.B.

Datum
14.04.2012 12:43:50

Führen Sie die von Ihnen fertig gestellte PL/SQL-Skriptdatei **uebung15.sql** aus.

Wenn Sie den Trigger innerhalb von sqlplus erstellen und der Datenbanktrigger nicht fehlerfrei erstellt werden konnte, dann können Sie sich mit dem Befehl „Show Errors“ die Fehlermeldungen auflisten lassen.

Sie können aber den Datenbanktrigger auch direkt im Abfragefenster vom SQL Developer 2.1 bearbeiten und übersetzen lassen. Dann bieten sich bessere Möglichkeiten für die Beseitigung von Syntax-Fehlern.

Darüber hinaus gibt es im SQL Developer 2.1 einen entsprechenden Dialog für vorhandene Trigger (Doppelklick auf den Triggernamen). Hier kann dann der PL/SQL-Block editiert und kompiliert werden.

Nachdem der Trigger fehlerfrei angelegt wurde, fügen Sie weitere Zeilen in die Tabelle Artikel ein und kontrollieren Sie den Inhalt von ART_PROTO. Bei jedem insert wird **eine** entsprechende Zeile in ART_PROTO geschrieben. Auch wenn mit einer insert Anweisung mehrere Zeilen in der Tabelle Artikel eingefügt werden, wird in ART_PROTO trotzdem nur eine Zeile geschrieben.