**Alle Aufgaben, die in SQL99 zu lösen sind, müssen in der Laborumgebung unter Oracle implementiert werden! Bei der Präsentation müssen die Lösungen somit in sqlplus vorzeigbar sein. Derartige Beispiele sind mit einem Stern [⋆] gekennzeichnet.**

**Aufgabe 1** []

Vorliegender Realitätsausschnitt ist zu modellieren:

> *Ein Student wird durch seine Matrikelnummer eindeutig identifiziert und weist weiters einen Vornamen und einen Nachnamen auf.*
>
> *Jeder Student kann in unterschiedlichen Semestern Lehrveranstaltungen besuchen, die durch eine eindeutige LV-Nummer, einen Namen und eine Stundenanzahl charakterisiert werden.*
>
> *Jede Lehrveranstaltung muss nun genau einem Institut zugeordnet sein, wobei ein Institut mehrere Lehrveranstaltugen anbieten kann. Das Institut wird durch eine Instituts-Nummer und einen Namen beschrieben.*
>
> *Für ein Institut kann ein Institutsvorstand existieren, der seinerseits eine Vorstand-Nummer, einen Vornamen, einen Nachnamen und ein Gehalt aufweist. Das Gehalt eines Institutsvorstands liegt immer zwischen EUR 4.000- und EUR 9.000,-.*

a. Erstellen Sie ein entsprechendes ER-Diagramm, das sämtliche Entitäten, Beziehungen und Attribute beinhaltet. Sachverhalte und Zusatzannahmen, die sich nicht direkt im ER-Diagramm darstellen lassen, sind explizit (z. B. natürlichsprachlich) zu formulieren.

b. Leiten Sie das konzeptuelle Modell in ein entsprechendes relationales DB-Schema über. Bitte geben Sie Relationenschemata mit Attributen, Schlüssel(-kandidaten) sowie Fremdschlüsselbedingungen an. Sollten funktionale Abhängigkeiten existieren, deren linke Seite kein Schlüsselkandidat ist, dann notieren Sie diese bitte ebenfalls.

**Aufgabe 2** [⋆]

Erstellen Sie in SQL-99 ein Datenbankschema (Tabellen, Attribute und Einschränkungen), das auf der Lösung des Beispiels 1 aufbaut. Bitte geben Sie bei den einzelnen Strukturkomponenten an, ob sie aus dem Realitätsausschnit und den zugehörigen verbalen Erklärungen direkt hergeleitet wurden, oder ob sie vielmehr das Resultat Ihrer individuellen Designentscheidungen sind. Es müssen jedenfalls sämtliche Einschränkungen, die im ER-Schema vorhanden sind berücksichtigt werden.

**Aufgabe 3** [⋆]

Erstellen Sie die folgenden Sichten (Views) auf das Datenbankschema der Aufgabe 2:

a. Die Sicht soll die Lehrveranstaltungsnummer und den Lehrveranstaltungsnamen, sowie die Matrikelnummer und den Nachnamen der zugehörigen Studenten für alle Lehrveranstaltungen beinhalten, die eine Stundenanzahl von 2 aufweisen und im SS2003 durchgeführt wurden.

b. Die Sicht soll die Institutsnummer sowie den Nachnahmen des zugehörigen Institutsvorstands aller Institute aufweisen, die zumindest eine Lehrveranstaltung anbieten bzw. angeboten haben.

**Aufgabe 4** [⋆]

Gegeben sind folgende Relationenschemata:
produkt (<u>modellnr</u>, hersteller , typ)
pc (<u>modellnr</u>, taktrate, ram, preis); pc.modellnr◊produkt
laptop (<u>modellnr</u>, taktrate, ram, hd, bildschirm, preis); laptop.modellnr◊produkt

produkt.typ IN {'pc','laptop'}

Formulieren Sie die folgenden Abfragen in Relationenalgebra:

a. Ermitteln Sie alle Hersteller, die zumindest 3 unterschiedliche PC-Modelle erzeugen.

b. Ermitteln Sie Modellnummer und Preis aller PCs des Herstellers B.

**Aufgabe 5** [⋆]

Formulieren Sie, basierend auf das Schema der Aufgabe 4, die folgende Abfrage in SQL99:

a. Bestimmen Sie für jeden Hersteller den durchschnittlichen PC-Preis pro vorhandener Taktrate.

**Aufgabe 6** [⋆]

Formulieren Sie, basierend auf das Schema der Aufgabe 4, die folgende Abfrage in SQL99:

a. Bestimmen Sie alle Hersteller, die Laptops erzeugen, die teurer als 3000 sind und gleichzeitig PCs erzeugen, die billiger als 300 sind.

**Aufgabe 7** [⋆]

Gegeben sind folgende Relationenschemata:
kunde (id, name)
kurs (id, bezeichnung, sortorder)
kunde_kurs (kundeid, kursid, sort_kunde_kurs); kunde_kurs.kundeid◊kunde, kunde_kurs.kursid◊kurs

Formulieren Sie die folgende Abfrage in SQL99:

a. Die Abfrage ist für genau einen Kunden durchzuführen, der durch die id eindeutig identifiziert wird (d.h. Angabe von *kunde.id=1* in der Where-Clause).

   Das Ergebnis der Abfrage soll alle existierenden Kurse (id, bezeichnung, sortorder) auflisten. Sollte der gegebene Kunde einen Kurs belegt haben, dann ist zusätzlich der Kundenname und sort_kunde_kurs mit auszugeben. Die Abfrage soll zuerst alle belegten Kurse sortiert nach *sort_kunde_kurs* und danach die nicht belegten Kurse sortiert nach *sortorder* beinhalten.

   BEDINGUNG: Die Abfrage ist mittels Verwendung von LEFT JOIN zu lösen.

   Beispielergebnis:

| ID | BEZEICHNUNG | SORTORDER | NAME | SORT_KUNDE_KURS |
|---|---|---|---|---|
| 1 | 1. Kurs | 1 | Maxi | 1 |
| 4 | 4. Kurs | 4 | Maxi | 2 |
| 2 | 2. Kurs | 2 | | |
| 3 | 3. Kurs | 3 | | |

**Aufgabe 8** [⋆]

a. Die Beispielangabe ist ident mit Beispiel 7. Diesmal ist die Abfrage jedoch OHNE Verwendung eines LEFT JOIN zu lösen.

**Aufgabe 9** [⋆]

**Dieses Beispiel gilt aufgrund des Schwierigkeitsgrades für 2 Beispiele (Beispiel 9 und 10)! Der Source kann von der Übungshomepage heruntergeladen werden.**

Folgende Tabellen sind gegeben:

```
create table employee (
  id number(10) constraint pk_user primary key,
  name varchar(15) constraint user_name_nn not null
);

create table trouble_ticket (
  id number(10) constraint pk_tt primary key,
  employeeid constraint fk_tt_1 references employee(id),
  logged_dt date constraint tt_logged_dt_nn not null,
  status varchar2(10) 'default' constraint tt_status_nn not null
);

create table trouble_ticket_history (
  id number(10) constraint pk_tth primary key,
  trouble_ticket_id number constraint fk_tth_1 references trouble_ticket(id),
  change_dt date constraint tth_change_dt_nn not null,
  action varchar2(20) constraint tth_action_nn not null,
  old_value varchar2(20),
  new_value varchar2(20) constraint tth_new_val_nn not null
);
```

Die Tabelle *employee* beinhaltet Mitarbeiter einer Firma.

In der Tabelle *trouble_ticket* werden auftretende Fehler eingetragen und verwaltet. Das Attribut *logged_dt* entspricht dem Erstellungsdatum des Tickets. Zu jedem Zeitpunkt ist das Ticket genau einem Mitarbeiter *employeeid* zugeordnet, wobei dieser wechseln kann. Genauso hat jedes Ticket zu einem Zeitpunkt genau einen Status (mögliche Stati sind

vereinfachend 'open' und 'closed'), wobei auch dieser wechseln kann.

In der Tabelle *trouble_ticket_history* werden sämtliche Veränderungen an einem Ticket mitgespeichert. Mögliche Veränderungen sind bei diesem Beispiel 'Status Change' und 'Supporter Change' was in dem Attribut *action* dementsprechend gespeichert wird. Das Attribut *trouble_ticket_id* referenziert das ursprüngliche Trouble-Ticket. *Change_dt* beinhaltet das Datum, an dem der Statuswechsel erfolgte. Das Attribut *old_value* beinhaltet den Wert vor dem Wechsel; der neue Wert wird in *new_value* gespeichert. Bei der Neuanlage eines Tickets werden immer sofort 2 History-Einträge eingetragen: Ein History Eintrag beinhaltet den Status Change von " (NULL) auf 'open', der andere beinhaltet den Supporter Change von " (NULL) auf '*emp_id*'(=der erste Employee, der mit der Bearbeitung beauftragt ist).

Folgende Datensätze sind in die Tabellen einzufügen:

```
delete from trouble_ticket_history;
delete from trouble_ticket;
delete from employee;
insert into employee values (1,'Fritzi');
insert into employee values (2,'Maxi');
insert into trouble_ticket (id,employeeid,logged_dt,status) values (1,1,'29.09.2005','closed');
insert into trouble_ticket_history values (1,1,'29.09.2005','Status Change','','open');
insert into trouble_ticket_history values (2,1,'29.09.2005','Supporter Change','','1');
insert into trouble_ticket_history values (3,1,'11.10.2005','Status Change','open','closed');
insert into trouble_ticket (id,employeeid,logged_dt,status) values (2,2,'23.09.2005','open');
insert into trouble_ticket_history values (4,2,'23.09.2005','Status Change','','open');
insert into trouble_ticket_history values (5,2,'23.09.2005','Supporter Change','','2');
insert into trouble_ticket (id,employeeid,logged_dt,status) values (3,2,'23.09.2005','open');
insert into trouble_ticket_history values (6,3,'23.09.2005','Status Change','','open');
insert into trouble_ticket_history values (7,3,'23.09.2005','Supporter Change','','2');
insert into trouble_ticket_history values (8,3,'29.09.2005','Supporter Change','2','1');
insert into trouble_ticket_history values (9,3,'11.10.2005','Supporter Change','1','2');
insert into trouble_ticket (id,employeeid,logged_dt,status) values (4,1,'23.09.2005','open');
insert into trouble_ticket_history values (10,4,'23.09.2005','Status Change','','open');
insert into trouble_ticket_history values (11,4,'23.09.2005','Supporter Change','','1');
insert into trouble_ticket_history values (12,4,'27.09.2005','Status Change','open','closed');
insert into trouble_ticket_history values (13,4,'29.09.2005','Status Change','closed','open');
insert into trouble_ticket_history values (14,4,'03.10.2005','Supporter Change','1','2');
insert into trouble_ticket_history values (15,4,'04.10.2005','Supporter Change','2','1');
insert into trouble_ticket_history values (16,4,'08.10.2005','Supporter Change','1','2');
```

Das folgende Select-Statement gibt für jeden Employee die offenen Trouble Tickets im Wochenabstand aus:

```
SELECT e.name,
       open_trouble_tickets(e.id,to_date('28.09.2005')) as offen_am_28_09_2005,
       open_trouble_tickets(e.id,to_date('05.10.2005')) as offen_am_05_10_2005,
       open_trouble_tickets(e.id,to_date('12.10.2005')) as offen_am_12_10_2005
FROM employee e;
```

Bei den oben eingetragenen Daten ergibt die Ausführung des Statements folgendes Ergebnis:

| NAME | OFFEN_AM_28_09_2005 | OFFEN_AM_05_10_2005 | OFFEN_AM_12_10_2005 |
|---|---|---|---|
| Fritzi | 0 | 3 | 0 |
| Maxi | 2 | 1 | 3 |

a. Erstellen Sie die Funktion *open_trouble_tickets(emp_id number, effect_date date)*, die für den übergebenen Employee *emp_id* die Anzahl der offenen Trouble Tickets zum Zeitpunkt *effect_date* errechnet und als Ergebniswert retourniert.

   Hinweis: Obiges Ergebnis dient nur zu Kontrollzwecken, d.h. Ihre Funktion muss für beliebige Eingabedaten ein richtiges Ergebnis liefern.

   ACHTUNG: Functions sind in der letzten Zeile mit einem / (Slash) abzuschließen, der ganz links am Zeilenanfang stehen muss.

**Aufgabe 10 [⋆]**

Dieses Beispiel kann angekreuzt werden, wenn Beispiel 9 angekreuzt wurde.