

### Prüfregeln

Überlegen sie sich Regeln, welche korrekte und konsistente Daten sichern.

- Das Gehalt muss größer Null sein und kleiner 10.000
- Das Datum der letzten Veranstaltung muss größer 1990 sein

Setzen sie diese Regeln in Ihr Datenschema um.

Legen sie hier zu eine Tabelle Referent2 an!

```
CREATE TABLE Referent2 (  
    R_Nr number,  
    P_Nr number,  
    S varchar(1) not null,  
    Name varchar(30) not null,  
    Vorname varchar(30) not null,  
    Firma varchar(50),  
    constraint Referent2_pk primary key(R_Nr),  
    Gehalt number check (Gehalt BETWEEN 0 AND 10000));
```

Geben sie folgenden Datensatz ein.

```
INSERT INTO Referent2 VALUES (1,234234,'I','Münch','Falko','','20000);
```

Warum lässt sich dieser Datensatz nicht einfügen?

**Er verstößt gegen die Check-Regel.**

Sie sitzen im Studentensekretariat kurz vor dem beginn des Wintersemesters. Es kommt ein Student nach dem anderen und möchte sich bei ihnen persönlich immatrikulieren.

Welche Daten müssten sie vom Studenten eingeben?  
Wo sollte sie ihr System unterstützen und warum?

Man müsste immer erst die letzte freie Matrikelnummer per Hand suchen.

Lösung: Anlegen einer Sequenz.

```
CREATE SEQUENCE PersonSequenz  
INCREMENT BY 1  
START WITH 1000;
```

Eine Oracle Sequenz ist ein Generator, der aufeinanderfolgende Werte für künstliche Schlüssel erzeugt.

Jeder Zugriff auf eine Oracle Sequenz ergibt den nächsten verfügbaren Wert. Diesen können Sie dem künstlichen Schlüssel zuweisen. Die Werte sind immer Ganzzahlen.

Diese Sequences werden automatisch verwaltet. Die Werte der Sequence müssen nach wie vor in die Tabellen, in denen sie den Primärschlüssel bilden, eingetragen werden.  
Generieren Sie zu diesem Zweck in der Datenbank folgenden **Trigger**.

```
1 create or replace trigger  
2   Erstelle_PersonMatrikelnr  
3   before insert on Person  
4   for each row  
5   begin
```

```
6      select PersonSequenz.nextval into :new.matrikel_nr from dual;  
7      end;  
8      /
```

**Erklärung:**

- 1 Erzeuge Trigger neu, oder ersetze bestehenden Trigger gleichen Namens.
- 2 Name des Triggers, der in den User\_Triggers (Systemtabelle) eingetragen wird.
- 3 VOR einem Insert auf die Tabelle wird der Primärschlüssel ergänzt, ansonsten müsste das RDBMS auf den fehlenden Schlüssel mit einer Fehlermeldung reagieren.
- 4 FOR EACH ROW bedeutet das Zünden des Triggers für jeden neu eingefügten Datensatz. Ansonsten würde der Trigger nur auf Tabellenebene, nicht aber auf Datenebene, gelten.
- 5 Beginne PL/SQL Anweisungsblock.
- 6 Der nächste Wert der Sequence wird direkt in die Eigenschaft :new des Feldes Matrikel\_Nr geschrieben.
- 7 Ende des PL/SQL Anweisungsblocks.
- 8 Der Slash am Ende des Skriptes zwingt das RDBMS zu einem ersten Compile des Triggers, also zur eigentlichen Prüfung auf semantische und syntaktische Richtigkeit.

Fügen sie nun folgenden Datensatz ein:

```
INSERT INTO Person (Name,Vorname,S_ID,Semesteranzahl) VALUES ('Franz','Susanne',2,3);
```

Fragen sie nun die Daten von Franz, Susanne ab!

```
SELECT *  
FROM Person  
WHERE name = 'Franz'  
AND vorname = 'Susanne';
```

```
SELECT AVG(Gehalt), S  
FROM REFERENT  
GROUP BY S;
```

**Fragen sie alle Veranstaltungen ab, welche zwischen dem 11.02.2002 und dem 04.04.2002 statt fanden.**

```
SQL> r  
1 select *  
2 from Veranstaltungsbesuche  
3* where datum between '11.02.2002' and '04.04.2002'
```

Listen sie den Namen des Studiengangs und die Anzahl der Studenten des Studienganges auf!

```
SQL> r  
1 select Studiengang, count(s.S_ID)  
2 from person p, studiengang s  
3 where p.s_id = s.s_id  
4* group by s.S_ID, Studiengang
```

**Listen sie den Namen des Studiengangs und die Anzahl der Studenten des Studienganges auf! Was wurde in der folgenden Abfrage nicht beachtet?**

```
SQL> r  
1 select Studiengang, count(s.S_ID)  
2 from person p, studiengang s  
3* group by s.S_ID, studiengang  
  
STUDIENGANG          COUNT(S.S_ID)  
-----
```

BWL	10	
Wirtschaftsinformatik		10
Mathematik	10	
Physik	10	
Chemie	10	

Die Join-Bedingung wurde nicht aufgeführt.  
 Somit wird jeder Datensatz der Tabelle Person mit der Tabelle Studiengang verknüpft

where p.s\_id = s.s\_id

**Folgende Abfrage ist ohne Join-Bedingung entstanden.  
 Markieren sie die Datensätze, in welchen die s\_id gleich sind!  
 Führen sie danach die Abfrage mit der Join-Bedingung aus!**

Zeile 3 abgeschnitten.

```

1 select Studiengang, p.s_id "Personen S_id", s.s_id "Studiengang S_i
2*   from person p, studiengang s
3   ;

```

STUDIENANGANG	Personen S_id	Studiengang S_id
BWL	2	1
BWL	2	1
BWL	2	1
BWL	2	1
BWL	5	1
BWL	2	1
BWL	3	1
BWL	1	1
BWL	2	1
BWL	2	1
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	5	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	3	2
Wirtschaftsinformatik	1	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Mathematik	2	3

STUDIENANGANG	Personen S_id	Studiengang S_id
Mathematik	2	3
Mathematik	2	3
Mathematik	2	3
Mathematik	5	3
Mathematik	2	3
Mathematik	3	3
Mathematik	1	3
Mathematik	2	3
Mathematik	2	3
Physik	2	4
Physik	2	4
Physik	2	4
Physik	2	4
Physik	5	4
Physik	2	4

Physik	3	4
Physik	1	4
Physik	2	4
Physik	2	4
Chemie	2	5
Chemie	2	5

STUDIENGANG	Personen S_id	Studiengang S_id
Chemie	2	5
Chemie	2	5
Chemie	5	5
Chemie	2	5
Chemie	3	5
Chemie	1	5
Chemie	2	5
Chemie	2	5

50 Zeilen ausgewählt.

```
SQL> r
1 select Studiengang, p.s_id "Personen S_id", s.s_id "Studiengang S_id"
2   from person p, studiengang s
3*  where p.s_id = s.s_id
```

STUDIENGANG	Personen S_id	Studiengang S_id
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2
Chemie	5	5
Wirtschaftsinformatik	2	2
Mathematik	3	3
BWL	1	1
Wirtschaftsinformatik	2	2
Wirtschaftsinformatik	2	2

10 Zeilen ausgewählt.

**Feststellung, dass mit der Join-bedingung die markierten Datensätze aufgelistet werden!**

**Interpretieren sie die Fehlermeldungen!**

**Welche Aufgaben sollten die folgenden Statements erfüllen!**

**Korrigieren sie die Statements!**

```
SQL> CREATE TABLE Veranstaltungsbesuche (
2   Matrikel_nr number
3   constraint VeranstaltungsbesucheFK1 references Personen (Matrikel_nr),
4   V_Nr number
5   constraint VeranstaltungsbesucheFK2 references Veranstaltung (V_Nr),
6   R_Nr number
7   constraint VeranstaltungsbesucheFK3 references Referent (R_Nr),
8   Datum date);
CREATE TABLE Veranstaltungsbesuche (
*
```

FEHLER in Zeile 1:  
ORA-00955: Es gibt bereits ein Objekt mit diesem Namen

**Anderen Tabellennamen angeben**

```
SQL> r
 1 CREATE TABLE Veranstaltungsbesuch (
 2 Matrikel_nr number
 3 constraint VeranstaltungsbesucheFK1 references Person (Matrikel_nr),
 4 V_Nr number
 5 constraint VeranstaltungsbesucheFK2 references Veranstaltung (V_Nr),
 6 R_Nr number
 7 constraint VeranstaltungsbesucheFK3 references Referent (R_Nr),
 8* Datum date)
constraint VeranstaltungsbesucheFK2 references Veranstaltung (V_Nr),
*
```

FEHLER in Zeile 5:  
ORA-00942: Tabelle oder View nicht vorhanden

**Es muss immer erst die referenzierte Tabelle angelegt werden**

```
SQL> INSERT INTO Studiengang VALUES (5, 'Chemie');
INSERT INTO Studiengang VALUES (5, 'Chemie')
*
```

FEHLER in Zeile 1:  
ORA-00001: Verstoß gegen Eindeutigkeit, Regel (USER1.STUDIENGANG\_PK)

**Der Primärschlüssel wurde schon einmal vergeben**

```
SQL> DELETE FROM Studiengang
 2 WHERE S_ID = 2;
DELETE FROM Studiengang
*
```

FEHLER in Zeile 1:  
ORA-02292: Verstoß gegen Integritätsregel (USER1.PFLICHTFAEACHERFK1).  
Untergeordneter Datensatz  
gefunden.

**In einer anderen Tabelle existiert noch ein Datensatz, welcher als Fremdschlüssel das Studienfach 2 besitzt**

```
SQL> SELECT COUNT(*) Anzahl, V_nr
 2 FROM Veranstaltungsbesuche;
SELECT COUNT(*) Anzahl, V_nr
*
```

FEHLER in Zeile 1:  
ORA-00937: keine Gruppenfunktion für Einzelgruppe

**Eine Gruppenfunktion (count) , welche Informationen über die gesamte Tabelle liefert und eine Einzelfunktion, welche sich nur auf einen Datensatz bezieht, dürfen nicht zusammen in einer Abfrage stehen.**

```
SQL> SELECT V_nr, Bezeichnung
 2 FROM Veranstaltungen v, Veranstaltungsbesuche vb, Person p
 3 WHERE v.V_NR = vb.V_NR
 4 AND vb.Matrikel_nr = p. Matrikel_nr
 5 AND Name='Schmidt'
 6 AND Vorname = 'Peter';
SELECT V_nr, Bezeichnung
*
```

FEHLER in Zeile 1:  
ORA-00918: Spalte nicht eindeutig definiert

**V-nr kommt in zwei Tabellen vor**

V\_nr → v.N\_nr

```
SQL> 1 select Studiengang, count(s.S_ID)
SQL> 2 from person p, studiengang s
SQL> 3 where p.s_id = s.s_id
SQL> 4* group by s.S_ID
SQL> r
1 select Studiengang, count(s.S_ID)
2 from person p, studiengang s
3 where p.s_id = s.s_id
4* group by s.S_ID
select Studiengang, count(s.S_ID)
*
```

FEHLER in Zeile 1:  
ORA-00979: kein GROUP BY-Ausdruck

Jeder abgefragte Wert muss auch in der group by – Klausel vorkommen.

```
SQL> r
1 select Studiengang, count(s.S_ID)
2 from person p, studiengang s
3* where p.s_id = s.s_id
select Studiengang, count(s.S_ID)
*
```

FEHLER in Zeile 1:  
ORA-00937: keine Gruppenfunktion für Einzelgruppe

Count ist eine Gruppenfunktion!

Es muss eine Gruppe angegeben werden, auf welche sich das count bezieht, in diesem Fall die Spalte s.s\_id

```
SQL> INSERT INTO Referent2 VALUES (1,234234,'I','Münch','Falko','','20000);
INSERT INTO Referent2 VALUES (1,234234,'I','Münch','Falko','','20000)
*
```

FEHLER in Zeile 1:  
ORA-02290: Verstoß gegen CHECK-Regel (USER1.SYS\_C007260)

Es wurde eine check-Regel erstellt.  
Der eingetragene Wert entspricht nicht dieser Regel!