Katalognummer: 23

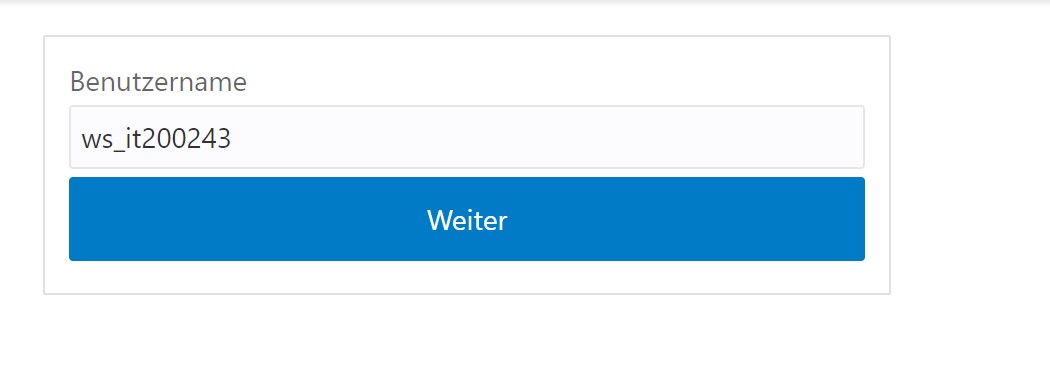
**EINFÜHRUNG**

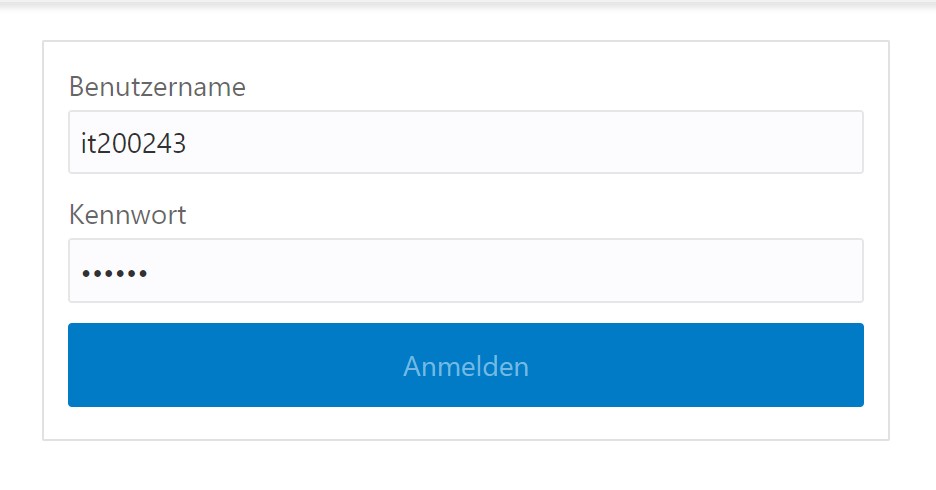
**SQL Apex Link:** [**https://apex.cloud.htl-leonding.ac.at/**](https://apex.cloud.htl-leonding.ac.at/)

**ws\_it200243**

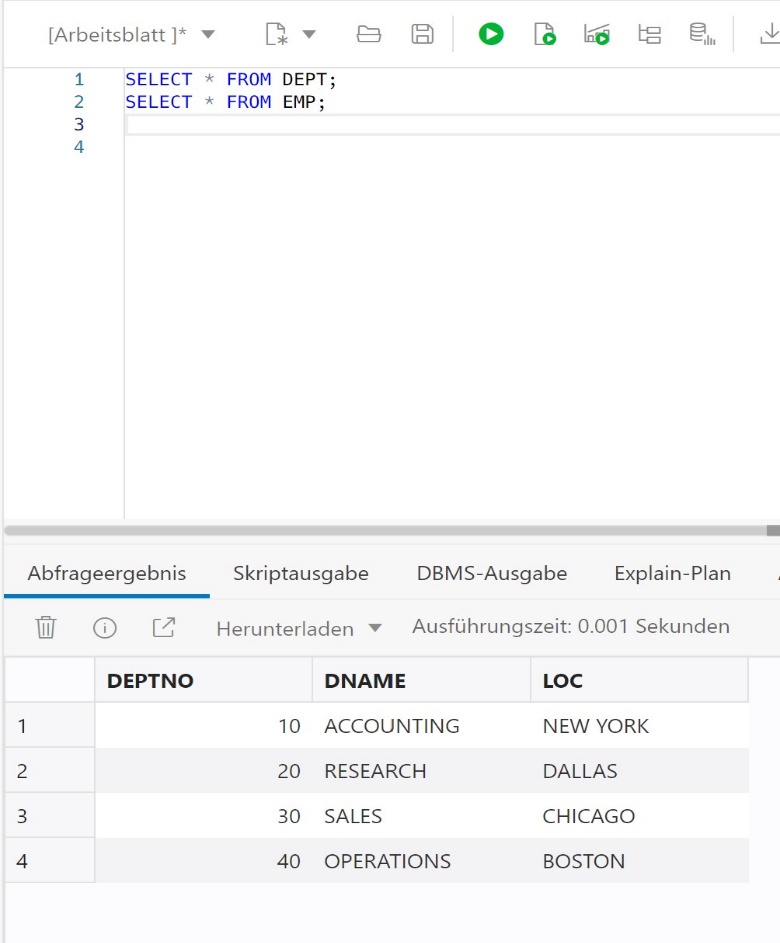
**IT200243**

**Password: oracle**

[](https://apex.cloud.htl-leonding.ac.at/ords/sql-developer)https://apex.cloud.htl-leonding.ac.at/ords/sql-developer



oracle



**Password nie verwenden: Tyga**

Oracle academy

Passwort:

Katalognummer: 23;

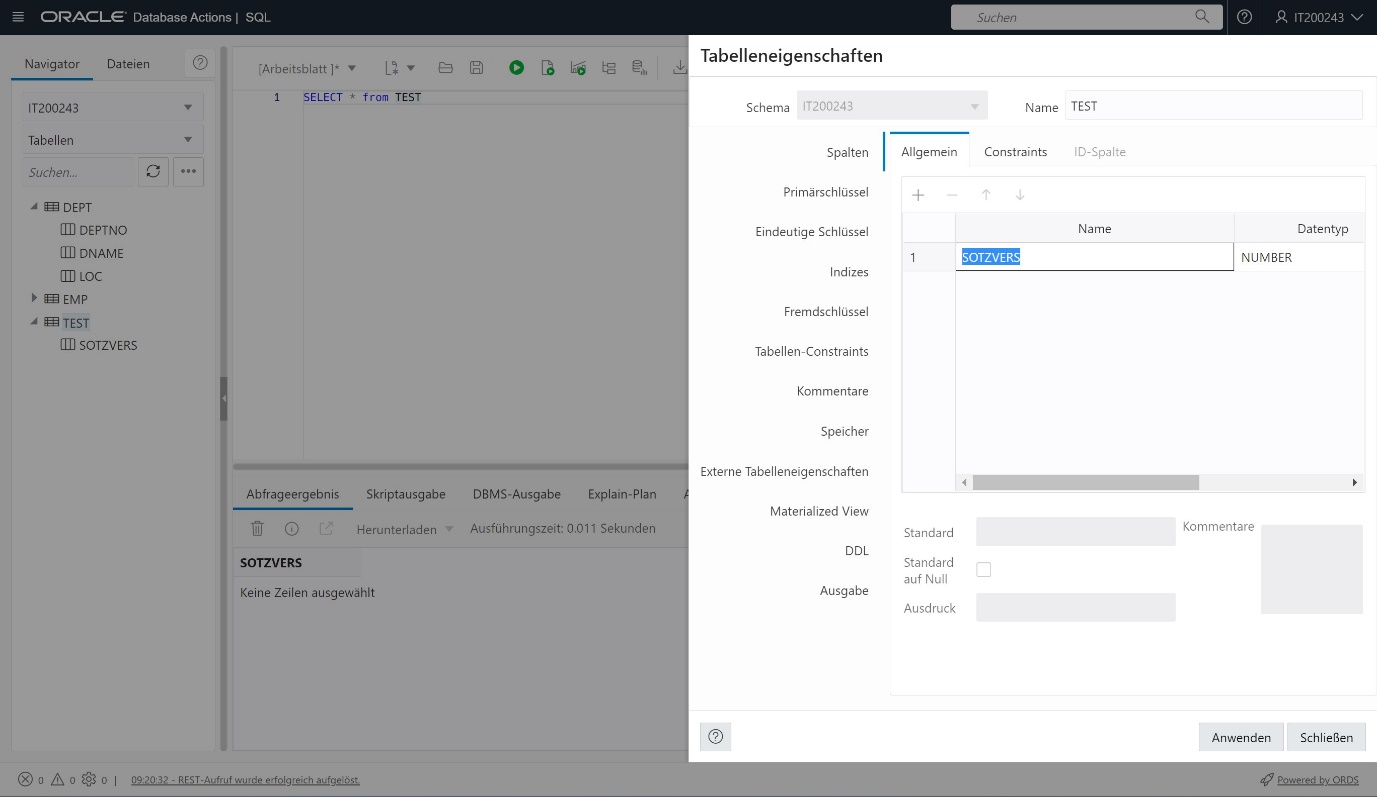
User: OA685067358;

Password: oracleacademy;

New Password: Stevan2006

SQL Syntax & Coding:

Erstellen von Tabellen:



Erzeugen neuer Tabellen 1

**Erzeugen neuer Tabellen**

Syntax:

CREATE TABLE table\_name (

    column\_name data\_type [default expression] [column integrity rule] [, column\_name ....,

....]

);

* Oracle ist nicht case sensitiv (Statements können auch klein geschrieben werden)
* [ ]  heißt optional
* Es gibt keine [blanks] alles wird mit [\_] gentrennt: bspw: Hugo\_boss

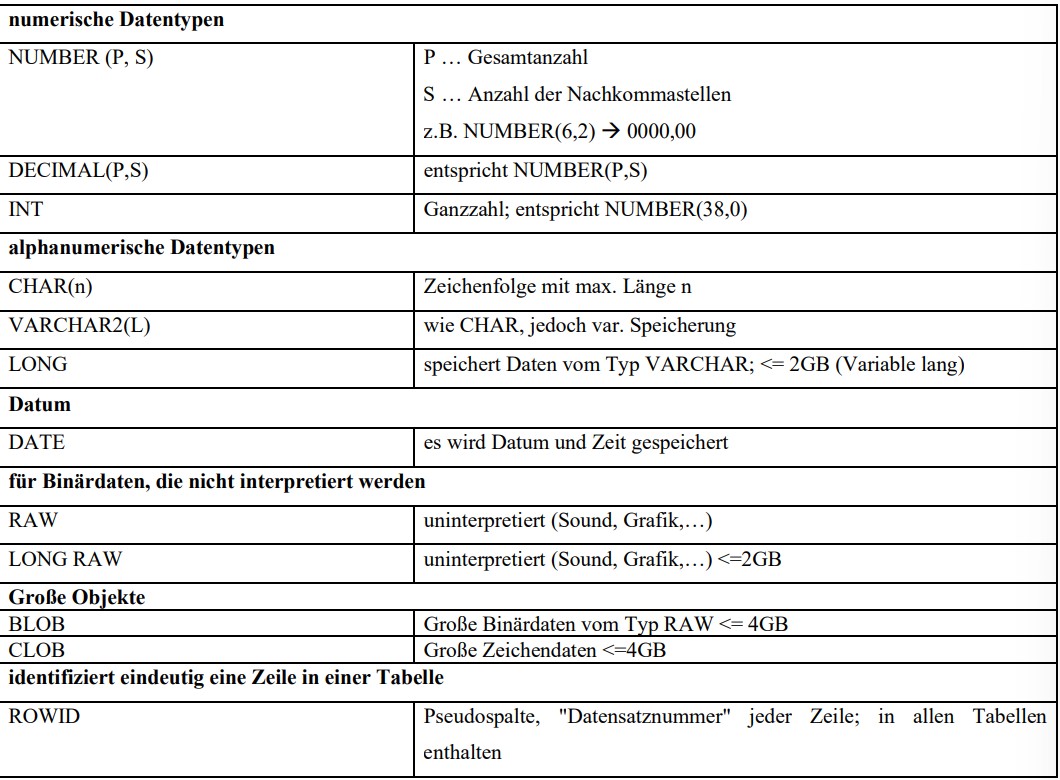
CREATE TABLE test2

(

    sozvers number(10,0)

);

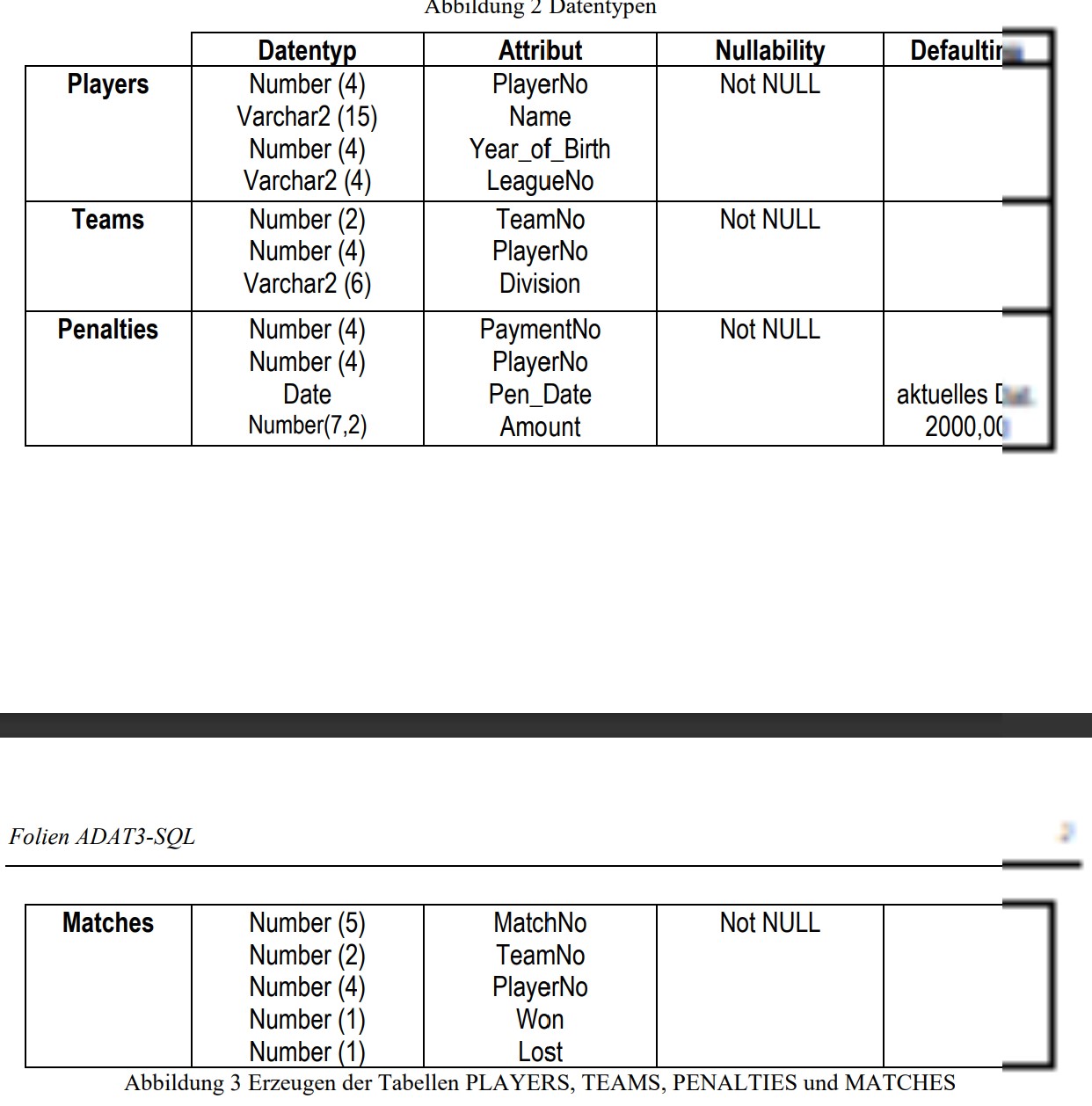
**Datentypen**



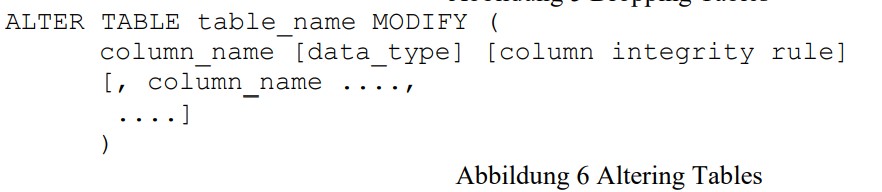
Bei Number (6 gesamt, 2 nachkomma stellen)

Datentypen am anfang

* VARCHAR2()
* STRING()
* NUMBER()

Abbildung 3)

5) Ändern von Spalten



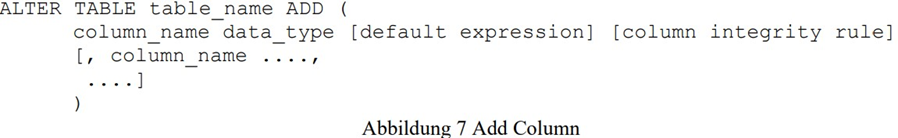
**Hinweise:**

- Modify immer nur für ein Attribut

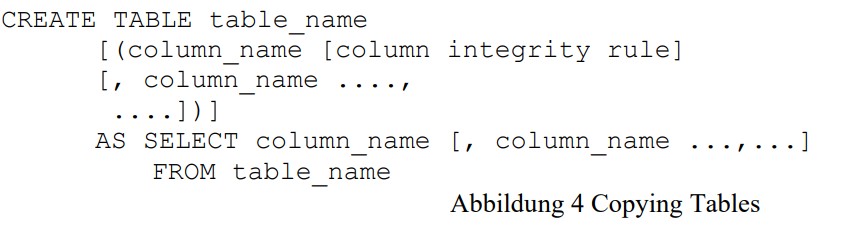
- Erhöhung der Länge ist erlaubt

- Ändern des Datentyps nur möglich, wenn NULL

**6) Hinzufügen einer Spalte**



**Kopieren von Tabellen**



Auswaehlen von Spalte

select name from demo\_tennis.players;

select \* from demo\_tennis.players;

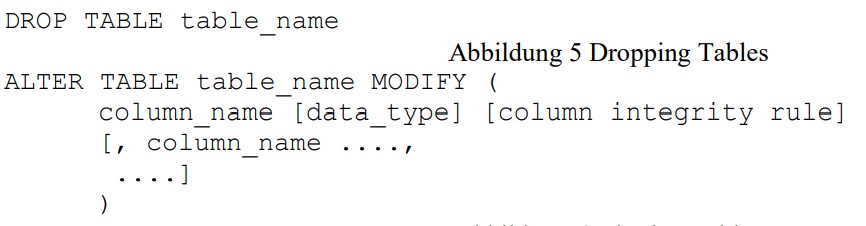
create table players\_copy(

    NAME,

    initials

) as select name, initials from demo\_tennis.players;

**Löschen von Tabellen**



**Löschen von Daten aus Tabellen**

delete \* from players\_copy

**7) Synonyme**

**Bsp:**

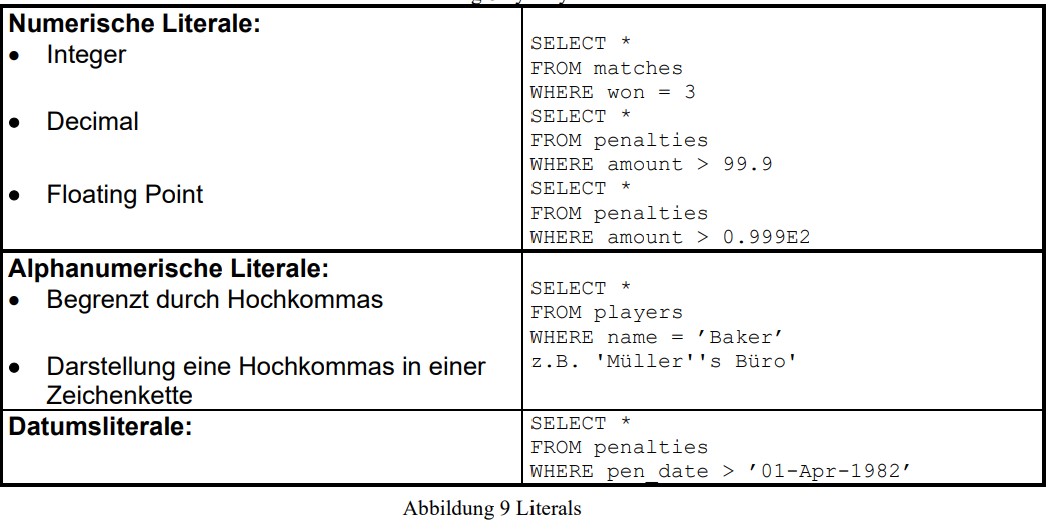
select \* from DEMO\_TENNIS.Players;

CREATE [PUBLIC] SYNONYM MEINEPLAYERS

FOR table DEMO\_TENNIS.Players;

Select \* from MEINEPLAYERS;

**Das SQL select Statement**



**Datum von Heute in sql:**

select sysdate from dual;

**Systemvariablen:**select (dual unwichtig)

* sysdate from dual;
* rownum from dual;
* User from dual

**Expressions:**

+, -, /, \*, MOD

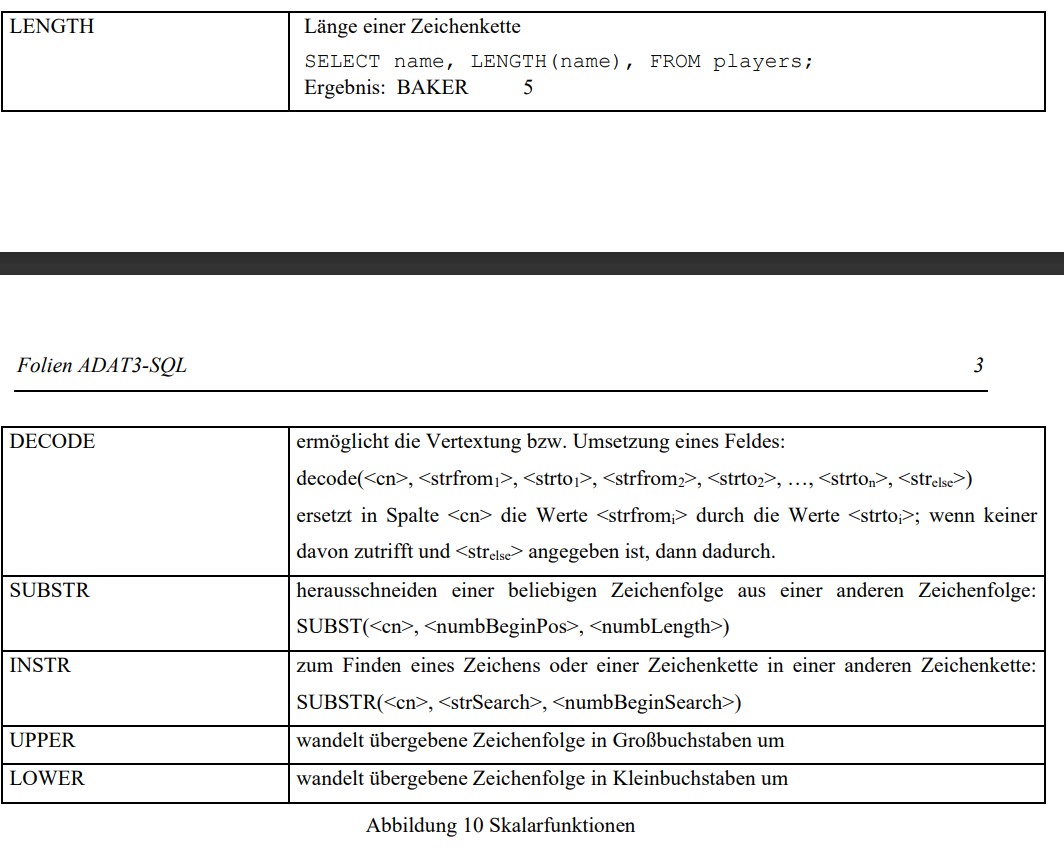
select 3\*2 from dual;

**Alphanumeric Expressions**

select ‘Hannes’ || ‘Tumfart’ from meineplayers;

**||  zusammenfügen von strings**

select name|| ‘ ’ ||playerno from MEINEPLAYERS;



**Beispiel**

select name, LENGTH(name) from meineplayers;

**Date-Expressions**

**Datumsdifferenzen**

select sysdate-pen\_date from DEMO\_Tennis.Players;

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

**Skalar Funktion Eingabe Datum:**  
to\_Date

z.b.

to\_Date(’01-01-2022’, ‘DD-MM-YYYY’);

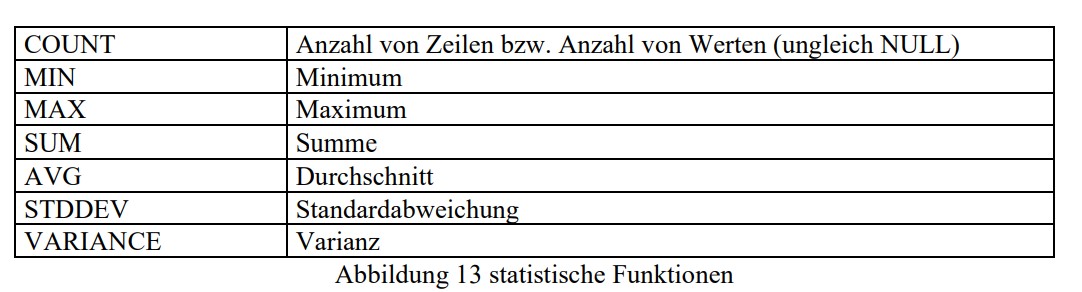
**Skalarfunktion Ausgabe Datum:**  
to\_CHAR()

z.b.

to\_Date(pen\_date, ‘DY, DD-Month-YYYY’);

Mon, 07-November-2022

**8.1 Klausel eines SELECT Statements**



Bsps)

* select count(\*) from MEINEPLAYERS;
* select sum(penalties) from MEINEPLAYERS;
  + alle penalties der tabelle meine Players;

**Spaltenüberschriften:**

* select playerno as Spielernummer from MEINEPLAYERS;
* select playerno as ‘Spielernummer der Tabelle‘ from MEINEPLAYERS;

In einem select ist nicht nur die Angabe einer column möglich sondern auch ausdrücke, konstanten, skalarfunktion

**||**

* Verbinden von strings

**Substr**

* select substr(enname, 1,1 ) || substr(job, 1, 1) as intitials, hiredate from emp;

**Entfernen identer (gleicher) Zeilen**

* select distinct playerno form penalties;

**Für count gibt es folgende Möglichkeiten:**

* count(\*)
* Doppelte werden nicht gezählt
  + count(distinct playerno)
  + gilt für alle statistische Funktionen

Beispiel: Was war die höchste Strafe:

* select max(amount) form penalties;

**8.2 WHERE Clausel**

**Bedingungen:**

* <, >, <=, <>, !=, =
* Der Alphanumerische Vergleich  ASCII Werte
* (c1,c2)=(w1,w2)

**Ergebnis**

* True
* False
* Unknown

**Verknüpfung**

* AND
* OR
* NOT

Bsp:

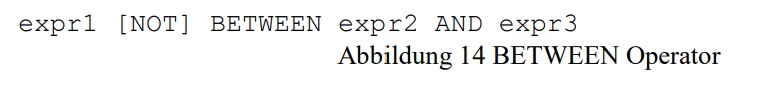
**Ausgabe der Spieler, die mit 17 dem tennis club beigetreten sind.**

* select \* from players where year\_joint – year\_of\_birth = 17;

**Ausgabe der männlichen Spieler die nach 1970 geboren wurden.**

* select \* from players where (year\_of\_birth > 1970) AND (sex = ‘M’);

**Between Operator**

* Wie Wert >= Untergrenze und <= Obergrenze

**IN Operator**

* Ist Element von [MENGE]
* Bsp:
  + select \* from tennis\_copy where year\_of\_birth in(1970, 1979, 1900);

**LIKE Operator**

Beispiel: Ausgabe der spieler deren name mit B beginnt;

Wildcards:

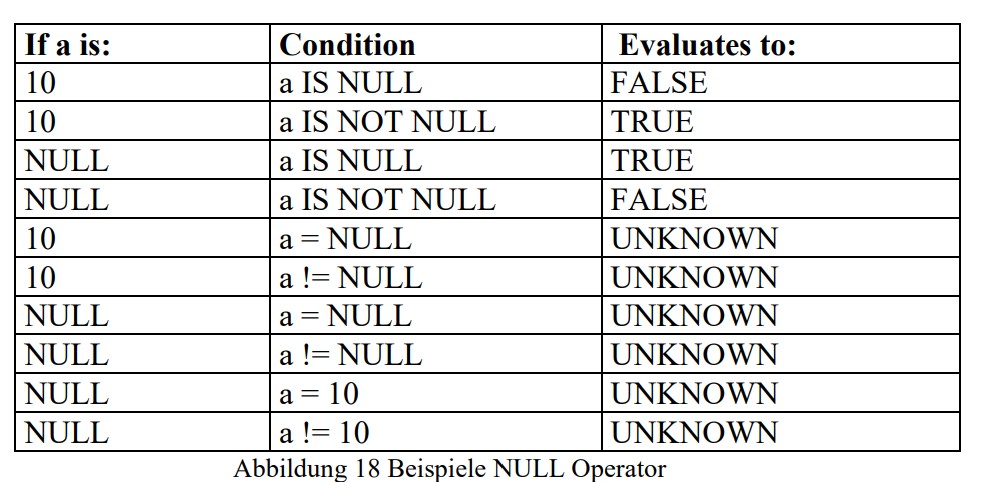
%........0,1 oder viele beliebige Zeichen

\_.........genau ein beliebiges Zeichen

Ausgabe der Spieler deren Name aus 2 Buchstaben besteht und an zweiter Stelle ein a steht.

* select \* from tennis\_copy where name like '\_a';

**NULL Operator**



* select \* from tennis\_copy where name is not NULL
* select \* from tennis\_copy where name is NULL;

**NVL**

* nvl() macht aus NULL eine  0
* select nvl(comm, 0) + 1 from emp;

****

**IN Operator mit subquery**

* Liefert subquery, keine Werte dann  False
* Sonst true

Beispiel: Ausgabe der Spieler die mindestens eine Strafe erhalten haben

**Variante 1: Ohne subquery (beim test verboten)**

•   select \* distinct playerno from penalties;

•   select \* playerno, name, from players where playerno(6,8,27,44,107);

**Variante 2: Verwenden! Mit subquery**

* select playerno, name from tennis\_copy where playerno in (select players from penalties);

**Vergleichsoperator mit subquery**

* expression comparison\_operator (subquery)

wichtig:

Das subquery darf nur eine Zeile liefern

Bsp

Ausgabe von playerno und name der Spieler, die älter als R.Parameter sind?

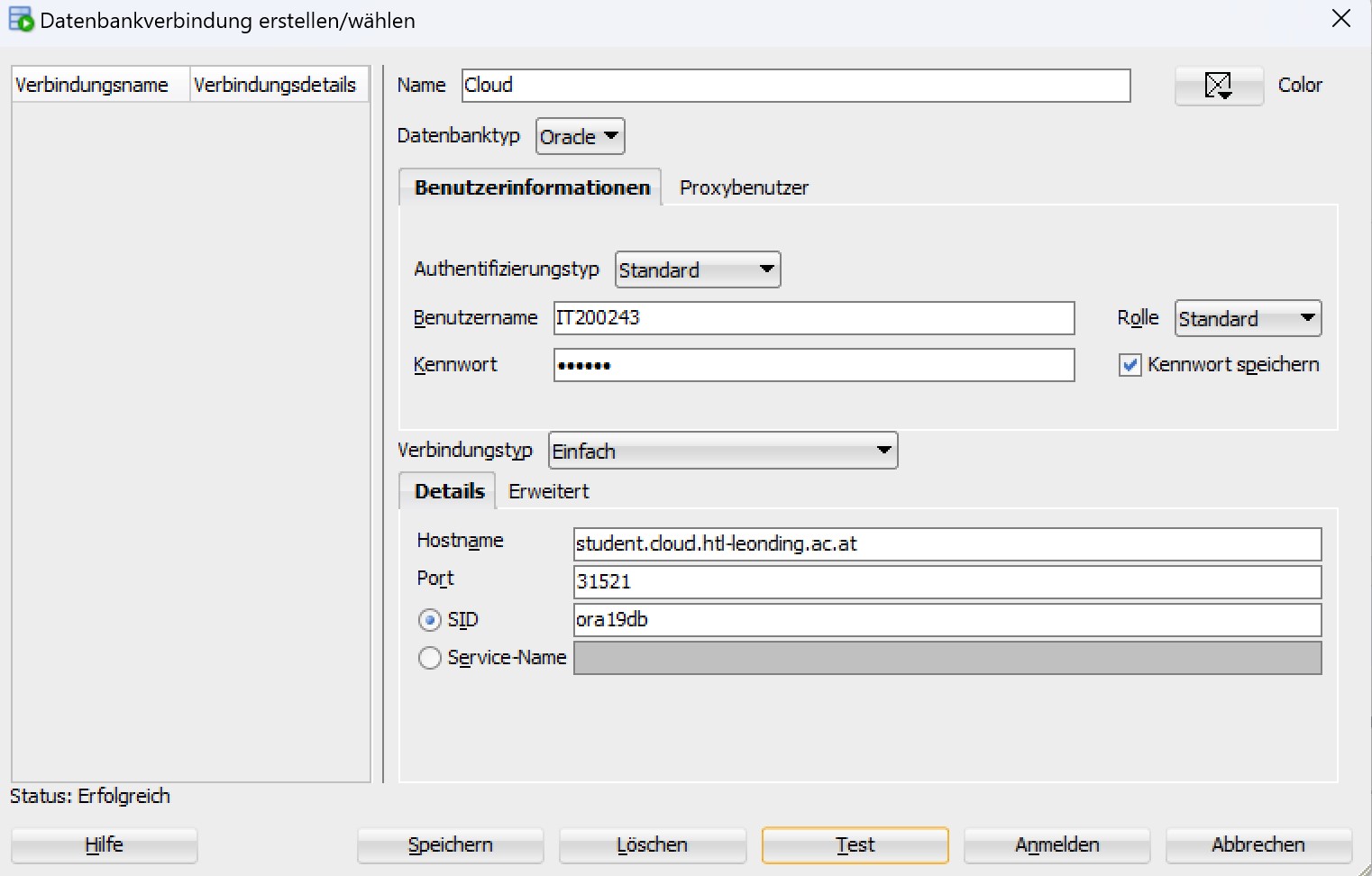
* select playerno, name from demo\_tennis.players

where year\_of\_birth <

(select year\_of\_birth from demo\_tennis.players where name

like 'Parmenter%' and initials like 'R%');

IDE Benutzen



**Die Operatoren ALL und ANY**

Syntax Diagramm

* expression comparison\_operator all subquery
* Der Ausdruck mit **[ALL]** liefert **TRUE**, wenn der Vergleich mit allen Zeichen des Subqueries erfüllt ist oder wenn das Subquery keine Zeilen liefert sonst **FALSE**
* Ausdruck mit ANY liefert **FALSE** wenn der Vergleich mit keiner Zeile des Subqueries erfüllt ist oder wenn das Subquery keine Zeilen liefert
* IN(subquery)  = ANY(subquery)
* NOT IN (subquery)  <> ALL(subquery)

Beispiel: Ausgabe von playerno, name, year\_of\_birth des ältesten Spielers?

* SELECT playerno, name, year\_of\_birth FROM PLAYERS WHERE year\_of\_birth = (SELECT MIN(year\_of\_birth) FROM PLAYERS);
* select playerno, name, year\_of\_birth from tennis\_copy

where year\_of\_birth <= ALL(select year\_of\_birth from tennis\_copy);

**Cat**

* select \* from catalog

Alle Tabelle werden angezeigt welche für die user vorhanden sind

* purge recyclebin
  + Dateien vom Recylcle bin zurückholen

**DESC emp:**

“Describe Tabelle EMP”.

* desc emp;
  + Zeigt Datenstruktur an

**8.2 Order by Clausel**

Die Ausgabe kann sortiert werden

Einschub: <https://asktom.oracle.com/>

Syntax:

**ORDER BY** [***column-Name***](https://docs.oracle.com/javadb/10.6.2.1/ref/rrefcolumnname.html#rrefcolumnname) **| ColumnPosition | Expression**

**[ ASC | DESC ]**

**[ NULLS FIRST | NULLS LAST ]**

**[ ,** [***column-Name***](https://docs.oracle.com/javadb/10.6.2.1/ref/rrefcolumnname.html#rrefcolumnname) **| ColumnPosition | Expression**

**[ ASC | DESC ]**

**[ NULLS FIRST | NULLS LAST ]**

* **DESc ist absteigend**
* **ASC ist aufsteigend (default)**

Bsp:

* select \* from demo\_tennis.players order by name;
* select \* from demo\_tennis.players order by 1 desc, 2 asc;

**8.3 Group BY and … HAVING**

* Group by verwendet man im Zusammenhang mit Statistischen Funktionen.
* Die statistische Funktion wird für jede durch GROUP BY entsprechende Gruppe ausgewertet

Bsp:

* Anzahl der Spieler in jeder Stadt?
* select count(\*),town from demo\_tennis.players GROUP by town;

Merksatz

* Hat das select statement eine Group By, so muss jede Spalte entweder eine statistische Funktion oder in der group by clausel vorkommen.

Bsp:

* select sum(houseno),count(\*), town, initials

from demo\_tennis.players GROUP by town, initials;

Anzahl der Strafen pro Jahr

* select pen\_date, count(amount) from demo\_tennis.penalties GROUP by pen\_date;

Bsp:

Ausgabe der Spieler deren Strafensumme > 10 beträgt

* select playerno, sum(amount) as Strafensumme from demo\_tennis.PENALTIES group by playerno

having sum(amount) > 100;

Having ist die Bedingung:

* **Having** geht auf das gruppierte
* **Where** geht auf die Einzelzeile

Zuerst gruppieren wenn es größer als 100 wirds abgefragt

Übung 5

**DER JOIN**

Wenn in der im FROM mind. zwei Tabellen vorkommen und die WHERE-Klausel mindestens eine Bedingung enthält, die die Spalten der Tabelle verbindet.

* Ohne WHERE Klausel hießt es
  + Kartesisches Produkt
    - select count(\*) from DEMO\_TENNIS.PLAYERS,DEMO\_TENNIS.PENALTIES;

Beim Zählen der Spieler bekommt man 8 Zeilen bei den Players und bei den Penalties 14

* 14\*8 =>112

Bsp: Ausgabe von playerno, name, amount mit den leichtestem JOIN

select *count*(\*) from DEMO\_TENNIS.PLAYERS,DEMO\_TENNIS.PENALTIES;  
  
select *count*(\*) from DEMO\_TENNIS.PLAYERS;  
  
select *count*(\*) from DEMO\_TENNIS.PENALTIES;  
  
select PLAYERNO from DEMO\_TENNIS.PLAYERS;  
  
select \* from DEMO\_TENNIS.PENALTIES;  
  
*-- Gruppieren und dann summieren*select PLAYERS.PLAYERNO, NAME, *sum*(AMOUNT) from DEMO\_TENNIS.PLAYERS, DEMO\_TENNIS.PENALTIES  
where DEMO\_TENNIS.PLAYERS.PLAYERNO = DEMO\_TENNIS.PENALTIES.PLAYERNO group by PLAYERS.PLAYERNO, NAME;

Übung 6

**Subquery (Innerselect)**

Ist ein select-statement innerhalb einer Bedingung eines select Statements.

(keine order by im subquery)

Columns der übergeordneten SELECTS dürfen im Subquery verwendet werden.

**Suchreihenfolge:**

* Suche der columns im subquery
* Wenn nicht vorhanden, suche im übergeordnetem select

**Bsp:**

Ausgabe von Spielernummer und Spielername, derjenigen Spieler die mindestens eine Strafe erhalten haben.

* SELECT playerno, name FROM players

WHERE EXISTS (SELECT \* FROM penalties WHERE playerno=players.playerno);

* select playerno, name from players where playerno in (SELECT playerno FROM penalties);
* select  pl.playerno, name from players pl, penalties pe where pl.playerno = pe.playerno group by pl.playerno,name;
* select distinct pl.playerno, name from players pl, penalties pe where pl.playerno = pe.playerno;
* select playerno, name from players where playerno = ANY (SELECT playerno FROM penalties);

Ausgabe der Spieler mit den 4 höchsten Strafen.

Standard ORACLE Port: 1521

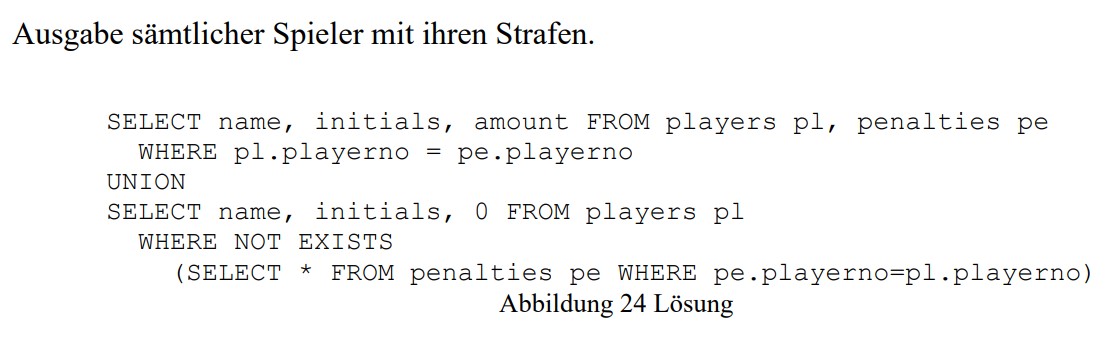
**Kombination von SELECT-Statements**

Select Statements lassen sich mit den Mengenoperationen kombinieren

* UNION (die Vereinigung)
* INTERSECT (der Durchschnitt)
* MINUS (DIFFERENZ)

Kombinieren.

**Beispiel:**



select name, initials, amount,'select 1' from demo\_tennis.players pl, demo\_tennis.penalties pe

where pl.playerno = pe.playerno

union

select name, initials, 0, 'select 2' from demo\_tennis.players pl where not exists (select \* from demo\_tennis.penalties pe where pe.playerno = pl.playerno);

Übung 7)

**Connect by Klausel**

Wird zur Abfrage hhierarchisch strukturierten Daten (z.b.Bäume) verwendet.

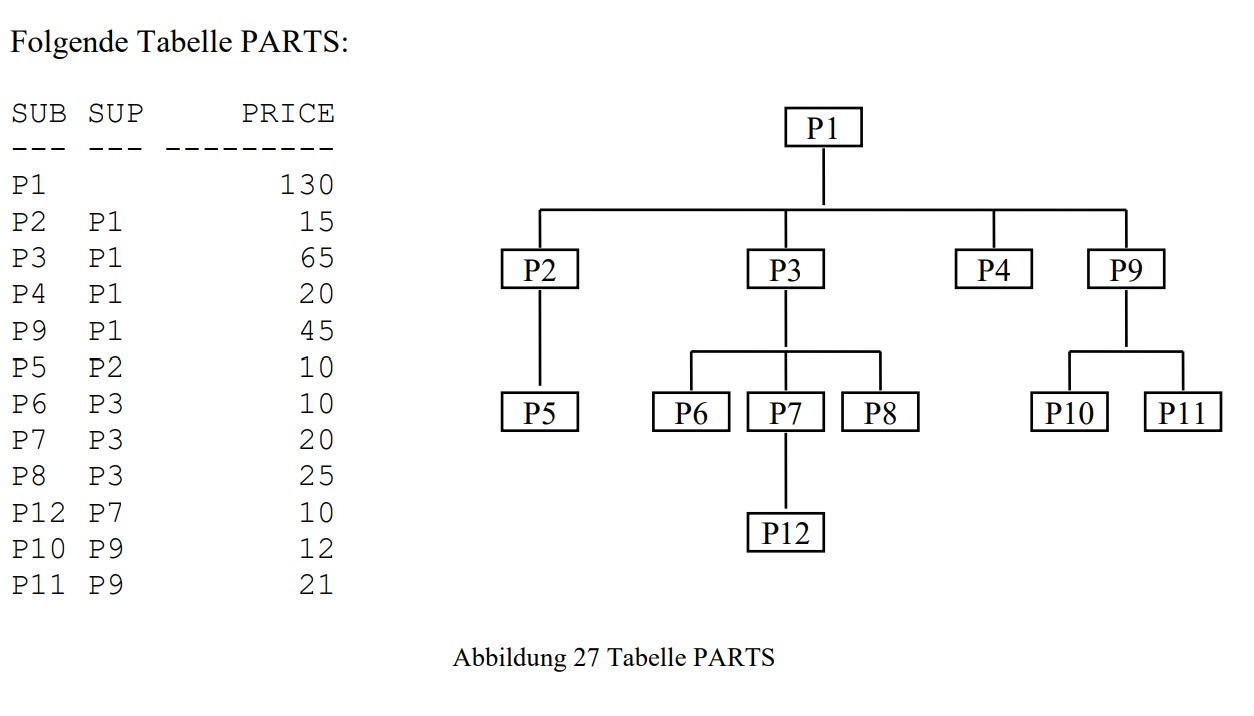
Vorsicht: Nur Oracle Standard.

**Syntax**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Bsp:**



**Beispiel:**

-- Ermittlung der Teile aus denen p3 besteht

select \* from parts

connect by prior sub = super

start with super='P3';

Unterschied zwischen start with:

* Where 🡪 werden nur die entsprechenden Datensätze entfernt und keine Kind-Datensätze

**Systemvariable Level:**

* Höhe der Hierarchie 🡪 Welche Ebene

-- Ermittlung der Teile aus denen p3 besteht

select LPAD(),Level,super,sub from parts

connect by prior sub = super

start with super='P3';

Beispiel:

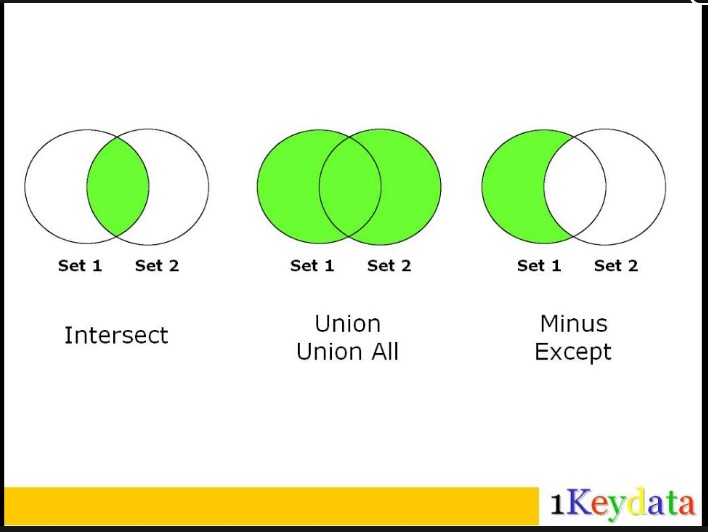
* Stelle den Baum mit Einrückungen dar.

SELECT LPAD('-', 6\*(LEVEL-1)) ||LEVEL || super ||sub from parts

CONNECT BY PRIOR sub=super

START WITH super='P1';

**Prior:**

Bewirkt eine rekursive Verknüpfung über die Spaltennamen, wobei die Prior untergeordnet ist.

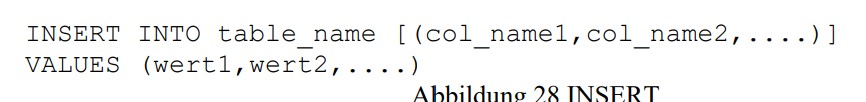
Übung 8 🡪 Ende

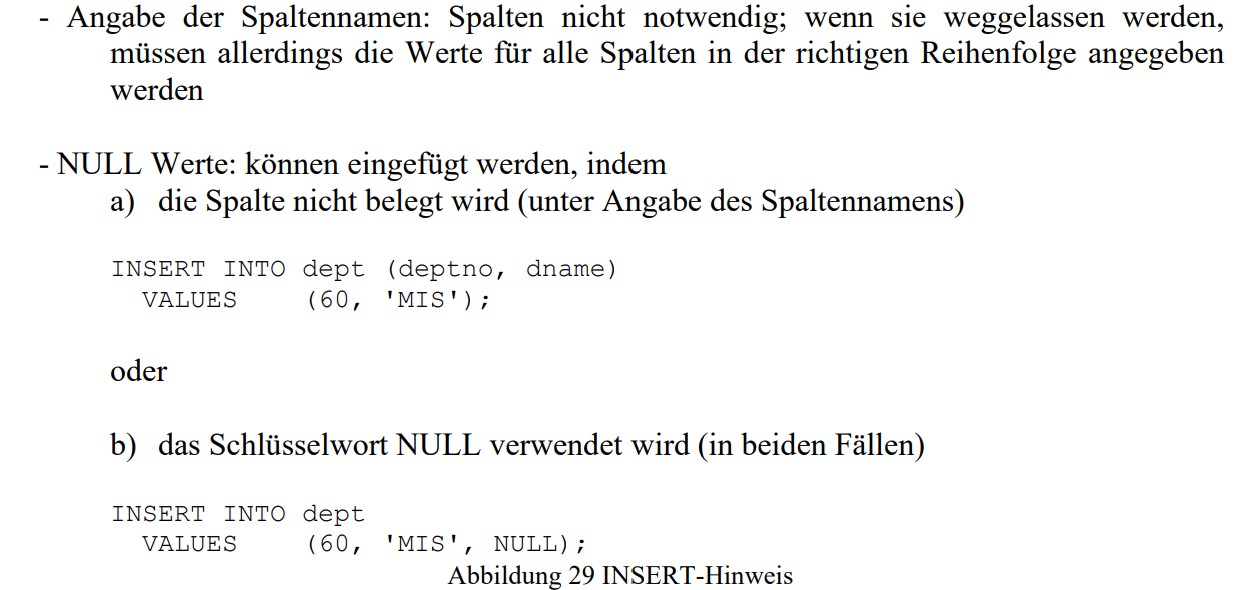
Über Teams gibt es einen Probetest

Keine Theorie 🡪 nur SELECTS

Teststoffende

**9) Daten manipulieren**

9.1 Einfügen



**Sicherung für Echte anwendungsfälle:**

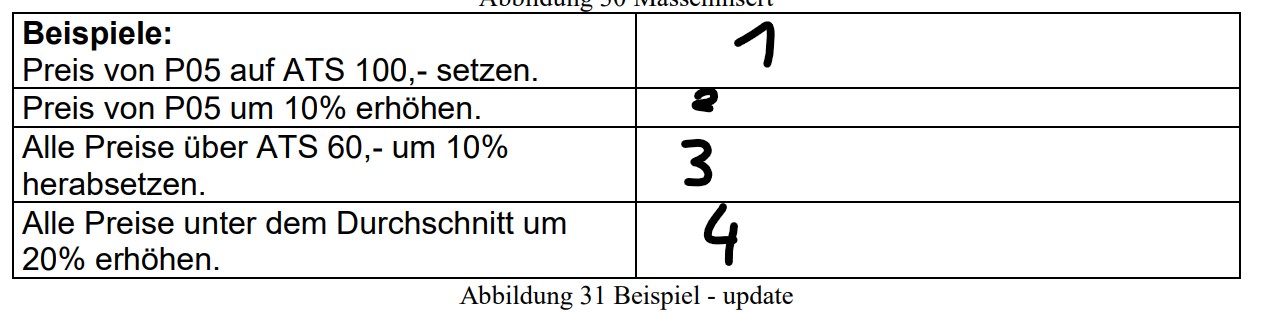
Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

**Wichtig: Die Datentypen müssen zusammenpassen**

Verwendet wirds zur Sicherung von Tabellen.

9.2 Ändern



Syntax

{

    UPDATE table-Name [[AS] correlation-Name]

        SET column-Name = Value

        [ , column-Name = Value} ]\*

        [WHERE clause] |

    UPDATE table-Name

        SET column-Name = Value

        [ , column-Name = Value ]\*

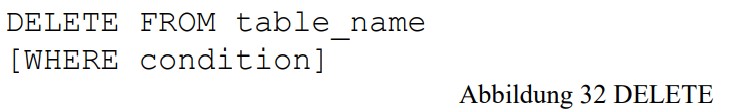
        WHERE CURRENT OF

}

**Lösung für die obige Tabelle**

* 1) update parts set price = 100 where sub = 'P5';
* 2) udate parts set price = price  (price \* 10)/100  where sub = 'P5';
* 3)update parts set price = price - (price \* 10)/100 where price > 60;
* 4) update parts set price = price + (price\*20)/100 where price < any(select avg(price) from parts);

9.3 Löschen



Unterschiede:

* Delete: Löscht Daten aus den Tabellen…………..…………………….DML
* Drop: Löscht Tabelle ……..……..……..……..……………………………….DDL
* Truncate: Löscht alle Reihen 🡪 Struktur bleibt vorhanden……DDL

(

* DML……..Data-Manipulation-Language
* DDL…......Data-Definition-Language

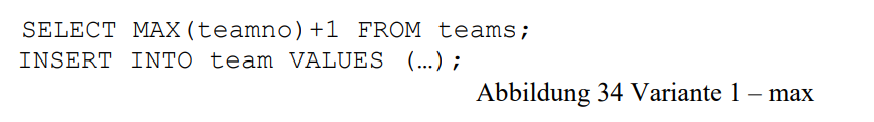
)

**11. Nummernfolgen (Sequences)**

Nummernfolgen werden für künstliche Schlüssel verwendet. (z.b. 1,2,3,4,5,6…usnw.)

Zur Vergabe von künstlichen Schlüsseln gitb es mehrere Möglichkeiten:

1. Max

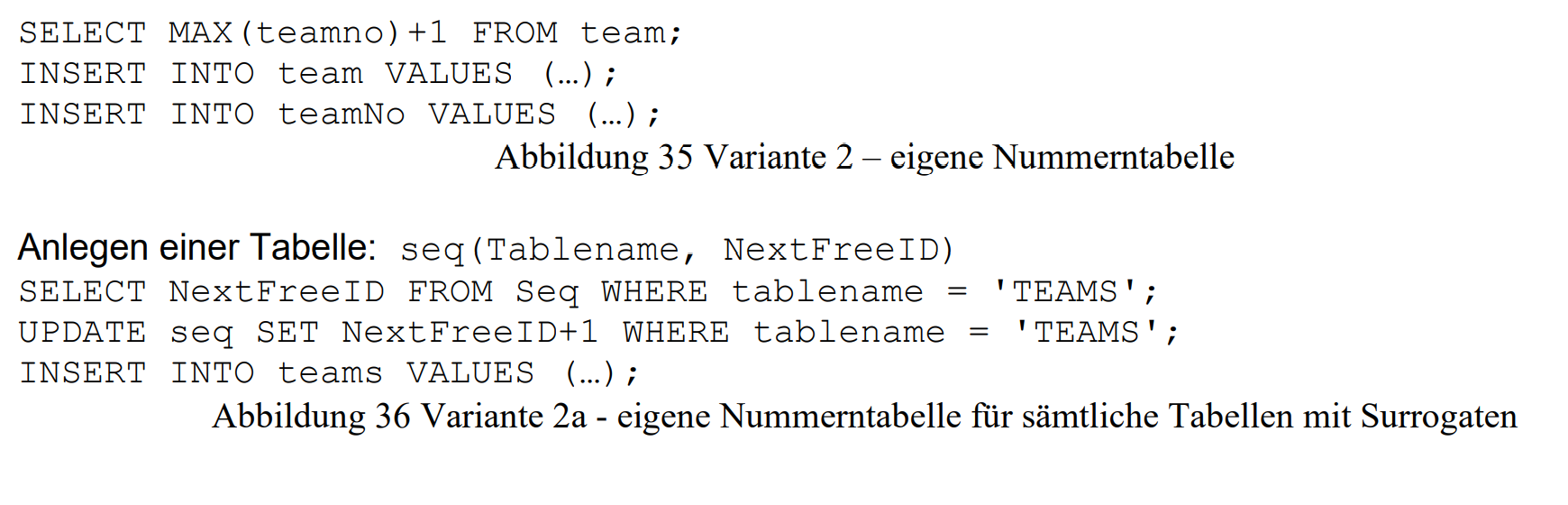


Problem: Gleichzeitig bekomm zwei Funktionen die gleich teamno

Lösung:

* Transaktion mit Lock sperren

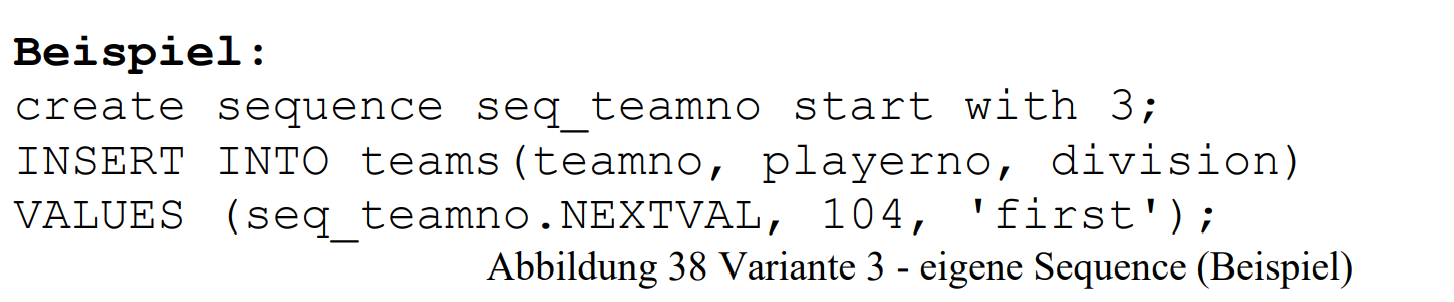
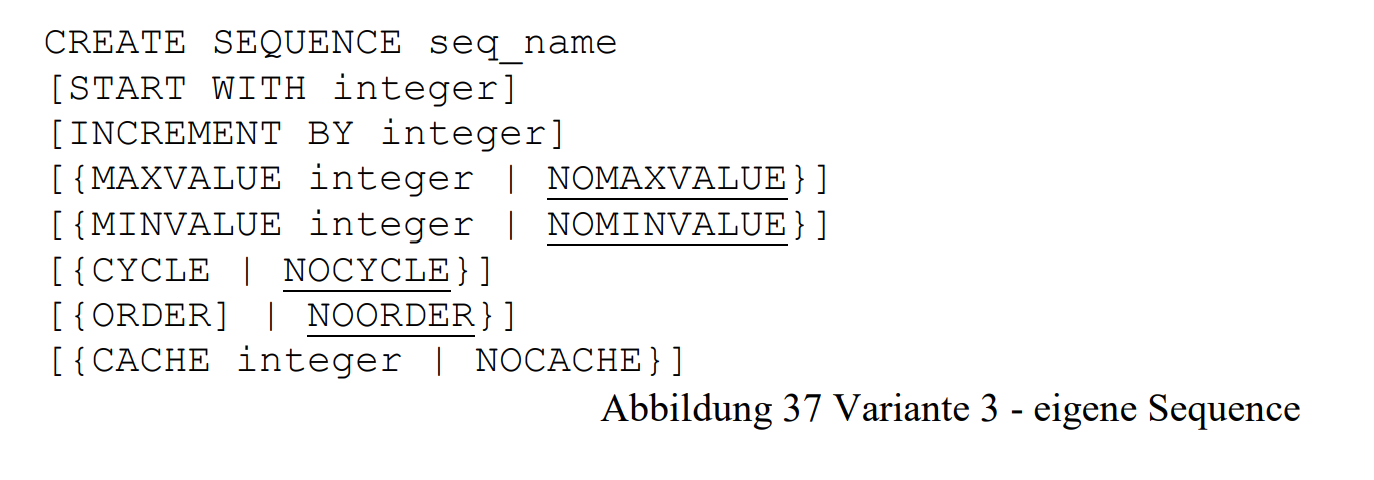
1. Eigene Nummern-Tabelle



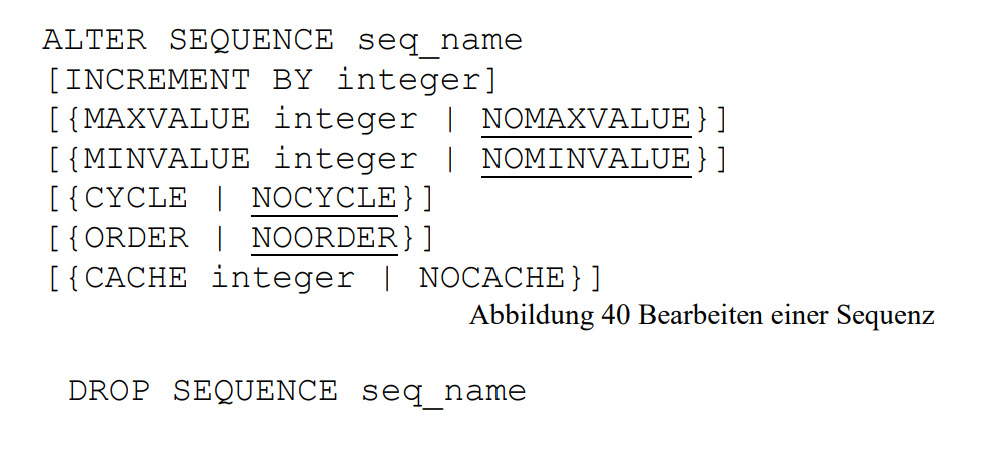
Problem: Gleichzeitig bekomm zwei Funktionen die gleich teamno

Lösung:

* Transaktion mit Lock sperren

1. Sequence

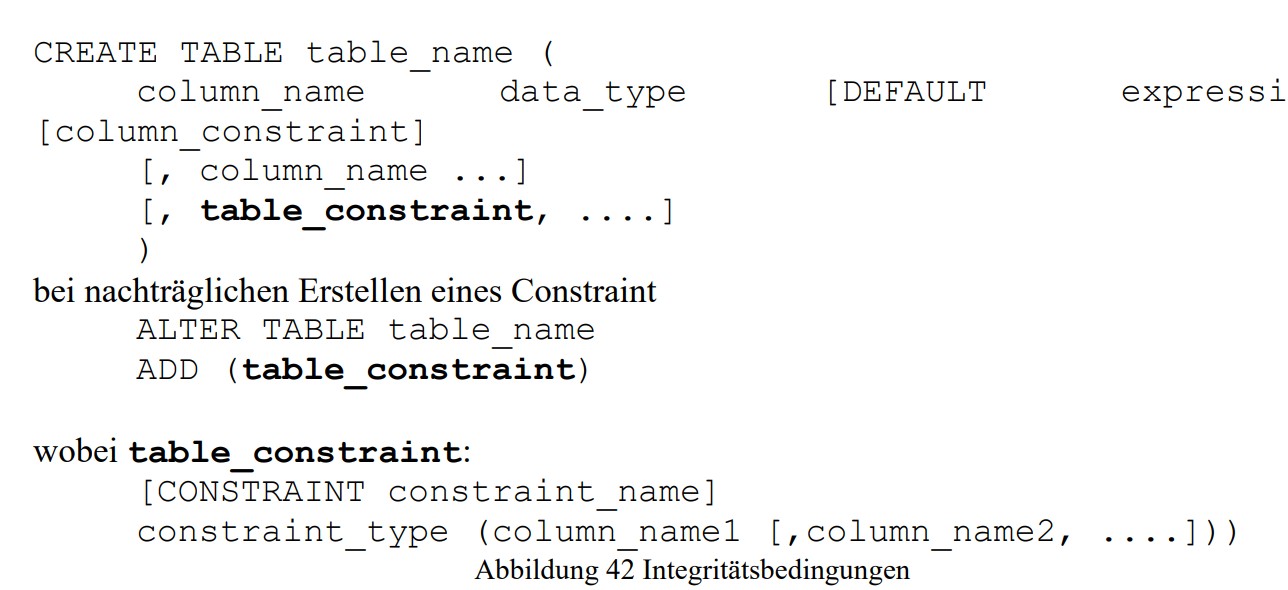
Increment by: default auch negative Werte

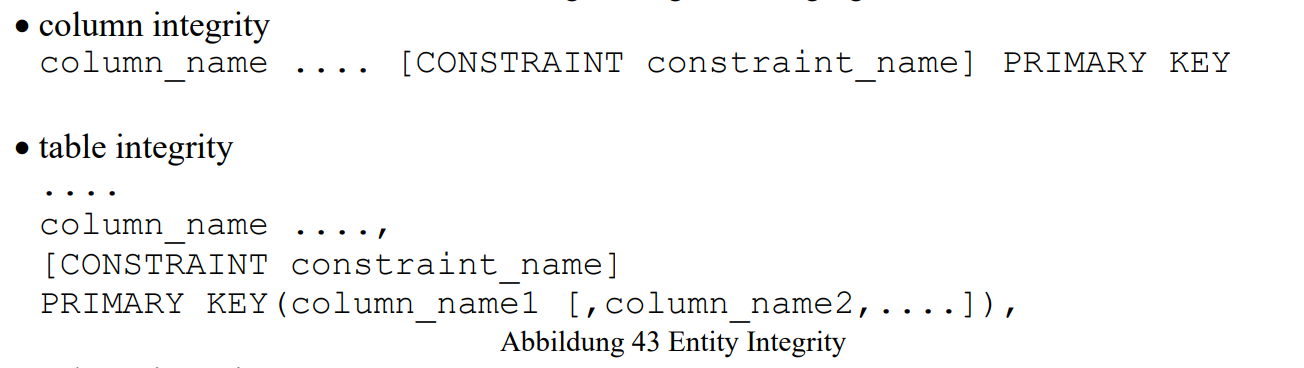


**12. Integrity Rule (Integritätsregel, Constraints)**

* PRIMARY KEY (PK)
* FOREIGNER KEY (FK)
* NOT NULL
* UNIQUE
* CHECK

Erstellen von Constraints





**12.1) Not Null**

**Bsp:**

create table test(

name varchar2(10) not null

);

create table test(

name varchar2(10),

dummy2 varchar2(10)

constraint not\_null\_dummy not null

);

* select \* from user\_constraints;

**Not Null darf nur auf Spaltenebene(Column-Integrity) angewendet werden.**

Default name: sys\_cnnnnnn

**Constraints kann man im DataDictionary finden**

**12.2) PRIMARY KEY(PK)**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

create table test5(

user\_id number(20) constraint pk\_user\_id primary key

);

**oder**

create table test5(

user\_id number(20),

name varchar2(10),

constraint pk\_user\_id primary key(user\_id)

);

Table Integrity immer möglich jedoch notwendig, falls pk aus mehreren spalten besteht (Zusammengesetzter Primary Key)

**12.3) UNIQUE (Alternate key integrity)**

COLUMN INTEGRITY

column-name…[constraint column-name] unique

TABLE INTEGRITY

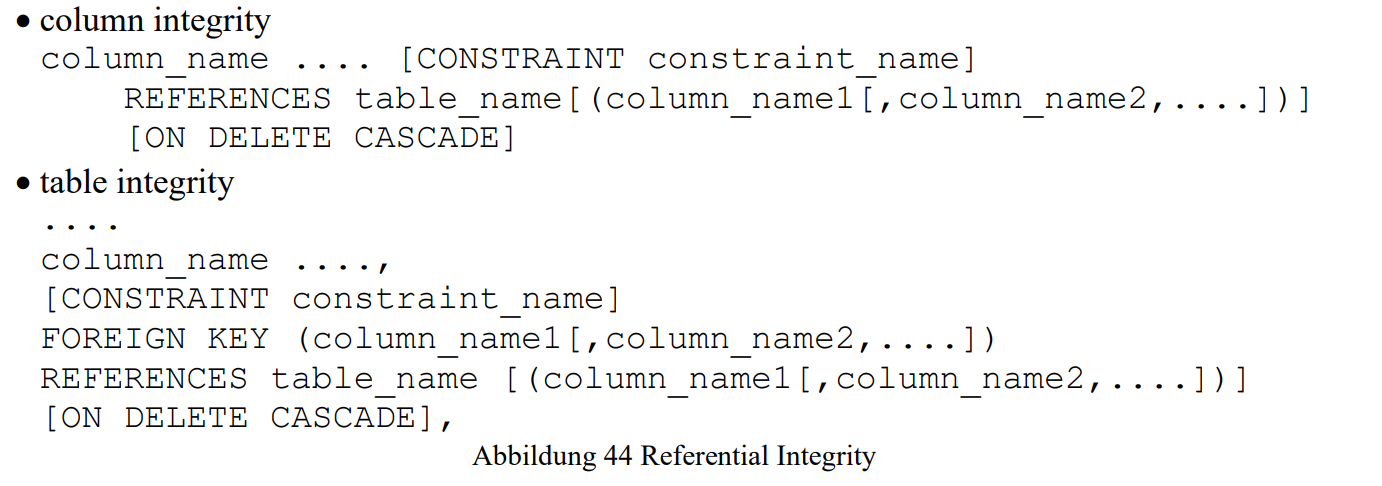
…..,constraint column\_name unique(column-name); oder

constraint column-name unique(column\_name,[])

Unterschied zu einem PK:

* Null ist erlaubt

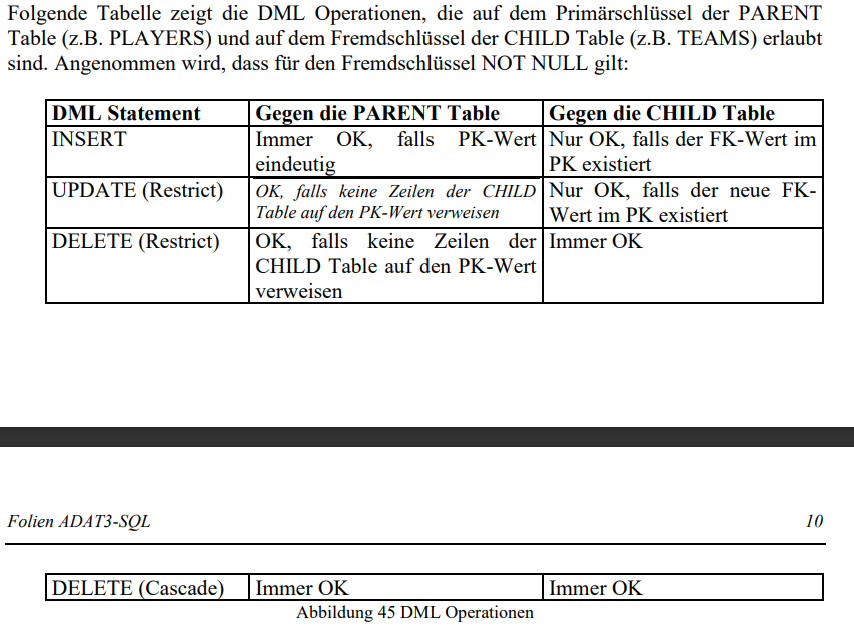
**12.4) FORGEINER KEY (Referential Integrity)**



Wird ON DELETE CASCADE angegeben, so werden die Sätze mit dem entsprechendem Fremdschlüssel (FK) automatisch mit gelöscht.

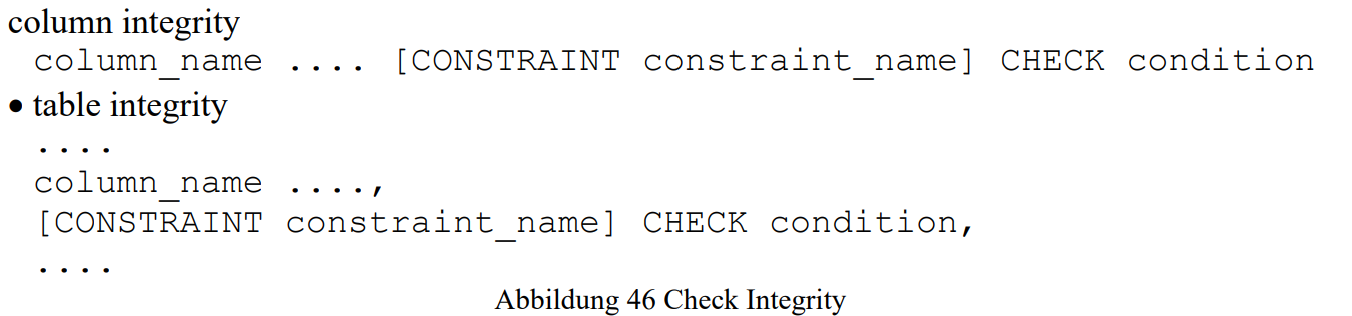
Beispiel: Players(playerno pk,….)

Teams(teamno, playerno, ….)



**Spezialfälle im Foreigner Key**:

* Primary Key und Fremdschlüssel in der gleichen Tabelle (self-referntial integrity)
  + Bspw: Hierarchische Struktur
  + Problem beim Einfügen der sätze (insert)
    - Zuerst den obersten Chef anlegen und dann die FK-Integrity nachträglich mit dem befehl alter hinzufügen
* Gegenseitige Primär/Fremdschlüsselbeziehungen (cross referntial integrity)
  + Bsp:
    - Abteilung(pk\_abtr, leiternr fk)
    - Angestellter(Ang\_nr\_pk, abntr\_fk)
  + Problem
    - Beim Erzeugen der Tabellen
      * alter table am Ende der Tabelle

**12.5) CHECK**

* Bspw: Check vergleicht das keiner über 120 jahre alt oder so…
* Bsp:
  + SEX……M,F

create table players(

    playerno number(10),

    name VARCHAR2(20),

    initials varchar2(10),

    year\_of\_birth date,

    sex char(1 BYTE) check (sex in ('M', 'W'))

);

* Bsp 2):
  + Year\_of\_birth, year\_joined

create table players5(

    playerno number(10),

    name VARCHAR2(20),

    initials varchar2(10),

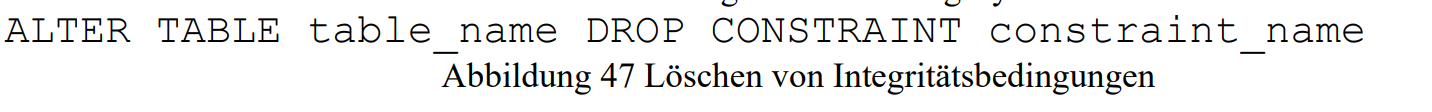
    year\_of\_birth number(38),

    year\_joined number(38),

    check (year\_of\_birth <= year\_of\_birth)

);

**12.6) CONSTRAINTS LÖSCHEN**



select \* from IT200243.PLAYERS1;  
  
create table IT200243.PKT(  
 pk number(3) constraint pk\_pk primary key,  
 c1 varchar2(20) check ( *substr*(c1,0) in('A','F') ),  
 c2 varchar2(4) check (c2 between 0.0 and 199.9)  
);  
  
create table IT200243.FKT(  
 pk1 number(2),  
 pk2 number(1),  
 c1 date,  
 c2 char(1 BYTE) check(c2 in('J','N')),  
 fk number(3) constraint fk\_fk references it200243.pkt(pk),  
 constraint pk\_pk1pk2 primary key(pk1,pk2)  
);

----------------------Teststoff ENDE----------------------