

---

# **Uni DB1 Syntax Cheatsheet**

Syntax cheatsheet for the DB1 (databases) course at  
HdM Stuttgart

Felicitas Pojtinger

2022-02-01

## Inhaltsverzeichnis

<b>1</b>	<b>Data Definition Language</b>	<b>3</b>
1.1	Tabellen . . . . .	3
1.1.1	Tabelle erstellen . . . . .	3
1.1.2	Tabelle löschen . . . . .	3
1.1.3	Tabelle umbenennen . . . . .	3
1.2	Spalten . . . . .	3
1.2.1	Spalten hinzufügen . . . . .	3
1.2.2	Spalten bearbeiten . . . . .	3
1.2.3	Spalten löschen . . . . .	4
1.3	Constraints . . . . .	4
1.3.1	Constraints hinzufügen . . . . .	4
1.3.2	Constraints löschen . . . . .	4
1.4	Views . . . . .	4
1.4.1	Views erstellen . . . . .	4
1.4.2	Views löschen . . . . .	4
1.5	Indizes . . . . .	4
1.5.1	Indizes erstellen . . . . .	4
1.5.2	Indizes löschen . . . . .	5
1.6	Trigger . . . . .	5
1.6.1	Trigger erstellen . . . . .	5
1.6.2	Trigger löschen . . . . .	5
1.6.3	Exceptions handlen . . . . .	5
1.7	Functions . . . . .	6
1.7.1	Function erstellen . . . . .	6
1.7.2	Function callen . . . . .	6
1.7.3	Function löschen . . . . .	6
1.8	Procedure . . . . .	6
1.8.1	Procedure erstellen . . . . .	6
1.8.2	Procedure callen . . . . .	6
1.8.3	Procedure löschen . . . . .	7
<b>2</b>	<b>Data Manipulation Language</b>	<b>7</b>
2.1	Datentypen . . . . .	7
2.2	Zeilenoperationen . . . . .	7
2.2.1	Insert . . . . .	7
2.2.2	Update . . . . .	7

---

2.2.3	Delete . . . . .	8
2.3	Unions . . . . .	8
2.4	Joins . . . . .	8
2.4.1	Inner Join . . . . .	8
2.4.2	Left Outer Join . . . . .	9
2.4.3	Right Outer Join . . . . .	9
2.4.4	Full Outer Join . . . . .	9
2.5	Trigger . . . . .	9
2.5.1	Insert-Trigger . . . . .	9
2.5.2	Update-Trigger . . . . .	10
2.5.3	Delete-Trigger . . . . .	10
2.5.4	Instead-Of-Trigger . . . . .	11
2.6	Ort der Verdammnis . . . . .	11

“Come, let us go down and confuse their language so they will not understand each other” - Genesis 11:7, *Die Bibel*

Mehr Details unter <https://github.com/pojntfx/uni-db1-notes>. Dieses Dokument ist nur als Schnell-Übersicht gedacht.

## 1 Data Definition Language

### 1.1 Tabellen

#### 1.1.1 Tabelle erstellen

```
1 create table persons (  
2     person_id number primary key not null ,  
3     first_name varchar2(50),  
4     last_name varchar2(50) default 'Duck' not null  
5 );
```

#### 1.1.2 Tabelle löschen

```
1 drop table persons;
```

#### 1.1.3 Tabelle umbenennen

```
1 alter table persons rename to people;
```

### 1.2 Spalten

#### 1.2.1 Spalten hinzufügen

```
1 alter table persons add ( phone varchar2(20), email varchar2(100) )
```

#### 1.2.2 Spalten bearbeiten

```
1 alter table persons modify ( birthdate date null, email varchar2(255) )  
    ;
```

### 1.2.3 Spalten löschen

```
1 alter table persons drop column birthdate;
```

## 1.3 Constraints

### 1.3.1 Constraints hinzufügen

```
1 alter table purchase_orders add constraint purchase_orders_order_id_pk  
  primary key(order_id);
```

### 1.3.2 Constraints löschen

```
1 alter table purchase_orders drop constraint purchase_orders_order_id_pk  
  ;
```

## 1.4 Views

### 1.4.1 Views erstellen

```
1 create view employees_years_of_service  
2 as select  
3   employee_id, first_name || ' ' || last_name as full_name,  
4   floor(months_between(current_date, hire_date) / 12) as  
     years_of_service  
5 from employees;
```

### 1.4.2 Views löschen

```
1 drop view employees_years_of_service;
```

## 1.5 Indizes

### 1.5.1 Indizes erstellen

```
1 create index members_full_name on members(first_name, last_name);
```

### 1.5.2 Indizes löschen

```
1 drop index members_full_name;
```

## 1.6 Trigger

### 1.6.1 Trigger erstellen

```
1 create trigger customers_credit_trigger
2   before update of credit_limit
3   on customers
4 declare
5   current_day number;
6 begin
7   current_day := extract(day from sysdate);
8
9   if current_day between 28 and 31 then
10      raise_application_error(-20100, 'Locked at the end of the month
11      ');
11   end if;
12 end;
```

### 1.6.2 Trigger löschen

```
1 drop trigger customers_credit_trigger;
```

### 1.6.3 Exceptions handlen

```
1 create trigger users_ensure_trigger
2   before update
3   on users
4   for each row
5 declare
6   user_invalid exception;
7   pragma exception_init(user_invalid, -20555);
8 begin
9   raise user_invalid;
10
11 exception
12   when user_invalid then
13      raise_application_error(-20555, 'User is invalid');
14   when others then
15      dbms_output.put_line('Unexpected error: ' || sqlerrm);
```

```
16 end;
```

## 1.7 Functions

### 1.7.1 Function erstellen

```
1 create or replace function get_my_sum( a integer, b integer ) return
  integer
2 is
3     multiplier number := 2;
4 begin
5     return a + b * multiplier;
6 end;
```

### 1.7.2 Function callen

```
1 select get_my_sum(1, 2) from dual;
```

### 1.7.3 Function löschen

```
1 drop function get_my_sum;
```

## 1.8 Procedure

### 1.8.1 Procedure erstellen

```
1 create or replace procedure get_sum ( a integer, b integer )
2 is
3     multiplier number := 2;
4     result number := 0;
5 begin
6     result := a + b * multiplier;
7
8     insert into results ( result ) values ( result );
9 end;
```

### 1.8.2 Procedure callen

```
1 exec get_sum(1, 2);
```

### 1.8.3 Procedure löschen

```
1 drop procedure get_sum;
```

## 2 Data Manipulation Language

### 2.1 Datentypen

- CHAR|CHARACTER (size)
- VARCHAR2 (size)
- DATE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- INTEGER|INT
- NUMBER (precision [, scale ])
- FLOAT (precision)

### 2.2 Zeilenoperationen

#### 2.2.1 Insert

```
1 insert into discounts(  
2     discount_name,  
3     amount,  
4     start_date,  
5     expired_date  
6 ) values (  
7     'Summer Promotion',  
8     9.5,  
9     date '2017-05-01',  
10    date '2017-08-31'  
11 )
```

#### 2.2.2 Update



```
1 update products
2 set list_price = 420
3 where list_price < 69;
```

### 2.2.3 Delete

```
1 delete from products
2 where list_price > 69;
```

## 2.3 Unions

Gleiche Anzahl von Spalten, mehr Zeilen.

```
1 select
2     first_name,
3     last_name,
4     email,
5     'contact' as role
6 from contacts
7 union select
8     first_name,
9     last_name,
10    email,
11    'employee' as role
12 from employees order by role
```

## 2.4 Joins

Mehr Spalten & mehr Zeilen

### 2.4.1 Inner Join

```
1 select
2     a.id as id_a,
3     a.color as color_a,
4     b.id as id_b,
5     b.color as color_b
6 from palette_a a
7 inner join palette_b b using(color);
```

### 2.4.2 Left Outer Join

```
1 select
2     a.id as id_a,
3     a.color as color_a,
4     b.id as id_b,
5     b.color as color_b
6 from palette_a a
7 left outer join palette_b b using(color);
```

### 2.4.3 Right Outer Join

```
1 select
2     a.id as id_a,
3     a.color as color_a,
4     b.id as id_b, b.color as color_b
5 from palette_a a
6 right outer join palette_b b using(color);
```

### 2.4.4 Full Outer Join

```
1 select
2     a.id as id_a,
3     a.color as color_a,
4     b.id as id_b,
5     b.color as color_b
6 from palette_a a
7 full outer join palette_b b using(color);
```

## 2.5 Trigger

### 2.5.1 Insert-Trigger

:old ist nicht vorhanden.

```
1 create or replace trigger customers_credit_trigger
2     before insert of credit_limit
3     on customers
4 declare
5     current_day number;
6 begin
7     current_day := extract(day from sysdate);
8
9     if current_day between 28 and 31 then
```

```
10         raise_application_error(-20100, 'Locked at the end of the month
11         ');
12     end if;
13 end;
```

### 2.5.2 Update-Trigger

```
1 create or replace trigger customers_credit_limit_trigger
2   before update of credit_limit
3   on customers
4   for each row
5   when (new.credit_limit > 0)
6 begin
7   if :new.credit_limit >= 2*:old.credit_limit then
8     raise_application_error(-20101, 'The new credit cannot be more
9     than double the old credit!');
10  end if;
11 end;
```

### 2.5.3 Delete-Trigger

:new ist nicht vorhanden.

```
1 create or replace trigger customers_audit_trigger
2   after delete
3   on customers
4   for each row
5 declare
6   transaction_type varchar2(10);
7 begin
8   transaction_type := case
9     when updating then 'update'
10    when deleting then 'delete'
11  end;
12
13   insert into audits(
14     table_name,
15     transaction_name,
16     by_user,
17     transaction_date
18   ) values (
19     'customers',
20     transaction_type,
21     user,
22     sysdate
23   );
24 end;
```

### 2.5.4 Instead-Of-Trigger

```
1 create or replace trigger create_customer_trigger
2   instead of insert on customers_and_contacts
3   for each row
4 declare
5   current_customer_id number;
6 begin
7   insert into customers(
8     name,
9     address,
10    website,
11    credit_limit
12  ) values (
13    :new.name,
14    :new.address,
15    :new.website,
16    :new.credit_limit
17  ) returning customer_id into current_customer_id;
18
19   insert into contacts(
20     first_name,
21     last_name,
22     email,
23     phone,
24     customer_id
25  ) values (
26    :new.first_name,
27    :new.last_name,
28    :new.email,
29    :new.phone,
30    current_customer_id
31  );
32 end;
```

### 2.6 Ort der Verdammnis

*menhir*

Wenn einem der Syntax schon nicht kompliziert genug ist, dann darf man vor das `declare`-Statement eines Triggers auch noch folgendes sinnloses Konstrukt packen und statt `:new :neu` schreiben:

```
1 referencing new as neu old as alt
```

Danach hat man auch fünf Zeilen. Und fünf Hirnzellen weniger.

Wo wir schon dabei sind: Ist der sonst universelle Negations-Operator `!` zu einfach? Zu simpel und zu verständlich? Wie wäre es mit `<>`; macht das genau selbe, ist aber komplizierter™!