Uni DB1 Syntax Cheatsheet

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

Felicitas Pojtinger

2022-02-01

Inhaltsverzeichnis

> "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

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

1.1.2 Tabelle löschen

```
drop table persons;
```

1.1.3 Tabelle umbenennen

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

1.2 Spalten

1.2.1 Spalten hinzufügen

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

1.2.2 Spalten bearbeiten

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

1.2.3 Spalten löschen

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

1.3 Constraints

1.3.1 Constraints hinzufügen

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

1.3.2 Constraints löschen

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

1.4 Views

1.4.1 Views erstellen

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

1.4.2 Views löschen

```
drop view employees_years_of_service;
```

1.5 Indizes

1.5.1 Indizes erstellen

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

1.5.2 Indizes löschen

```
drop index members_full_name;
```

1.6 Trigger

1.6.1 Trigger erstellen

```
create trigger customers_credit_trigger
    before update of credit_limit
    on customers
declare
    current_day number;
begin
    current_day := extract(day from sysdate);

    if current_day between 28 and 31 then
        raise_application_error(-20100, 'Locked at the end of the month
            ');
    end if;
end;
```

1.6.2 Trigger löschen

```
drop trigger customers_credit_trigger;
```

1.6.3 Exceptions handlen

```
create trigger users_ensure_trigger
    before update
    on users
    for each row
declare
    user_invalid exception;
    pragma exception_init(user_invalid, -20555);
begin
    raise user_invalid;

    exception
        when user_invalid then
            raise_application_error(-20555, 'User is invalid');
        when others then
            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

```
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

```
insert into discounts(
    discount_name,
    amount,
    start_date,
    expired_date
) values (
    'Summer Promotion',
    9.5,
    date '2017-05-01',
    date '2017-08-31'
)
```

2.2.2 Update

```sql
update products
set list_price = 420
where list_price < 69;
```

2.2.3 Delete

```sql
delete from products
where list_price > 69;
```

2.3 Unions

Gleiche Anzahl von Spalten, mehr Zeilen.

```sql
select
    first_name,
    last_name,
    email,
    'contact' as role
from contacts
union select
    first_name,
    last_name,
    email,
    'employee' as role
from employees order by role
```

2.4 Joins

Mehr Spalten & mehr Zeilen

2.4.1 Inner Join

```sql
select
    a.id as id_a,
    a.color as color_a,
    b.id as id_b,
    b.color as color_b
from palette_a a
inner join palette_b b using(color);
```

2.4.2 Left Outer Join

```
select
    a.id as id_a,
    a.color as color_a,
    b.id as id_b,
    b.color as color_b
from palette_a a
left outer join palette_b b using(color);
```

2.4.3 Right Outer Join

```
select
    a.id as id_a,
    a.color as color_a,
    b.id as id_b, b.color as color_b
from palette_a a
right outer join palette_b b using(color);
```

2.4.4 Full Outer Join

```
select
    a.id as id_a,
    a.color as color_a,
    b.id as id_b,
    b.color as color_b
from palette_a a
full outer join palette_b b using(color);
```

2.5 Trigger

2.5.1 Insert-Trigger

`:old` ist nicht vorhanden.

```
create or replace trigger customers_credit_trigger
    before insert of credit_limit
    on customers
declare
    current_day number;
begin
    current_day := extract(day from sysdate);

    if current_day between 28 and 31 then
```

```
10          raise_application_error(-20100, 'Locked at the end of the month
                ');
11      end if;
12  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
                than double the old credit!');
9       end if;
10  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

```
create or replace trigger create_customer_trigger
    instead of insert on customers_and_contacts
    for each row
declare
    current_customer_id number;
begin
    insert into customers(
        name,
        address,
        website,
        credit_limit
    ) values (
        :new.name,
        :new.address,
        :new.website,
        :new.credit_limit
    ) returning customer_id into current_customer_id;

    insert into contacts(
        first_name,
        last_name,
        email,
        phone,
        customer_id
    ) values (
        :new.first_name,
        :new.last_name,
        :new.email,
        :new.phone,
        current_customer_id
    );
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:

```
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™!