Lars Koenigsmann - 7209439
Mika Bredehoeft - 7209429
Tobias Barthold - 7209370

Aufgabe 2 b)

Statische Integritätsbedingungen

Passwort Check

```
passwort VARCHAR(64) NOT NULL CHECK(length(passwort) > 5)
```

Überprüft ob das Passwort länger als 5 Zeichen ist, da ein kürzeres Passwort sonst zu unsicher ist. Darf zusätzlich nicht NULL sein da der Benutzer ein Passwort eingeben muss um sich einloggen zu können mit seinem Benutzernamen. Bsp:

```
INSERT INTO ACCOUNT(ID,USERNAME,PASSWORT,MAIL,STREAMKEY,MODERATORID)
VALUES(1,'username','kurz',...)
```

Klappt nicht, da das Passwort zu kurz ist.

```
Fehlerbericht -
ORA-02290: CHECK-Constraint (C##FBPOOL101.SYS_C001129024) verletzt
```

VOD Status Check

```
status VARCHAR(7) DEFAULT 'public' NOT NULL CHECK (status in ('public',
'private'))
```

Der Status einens VOD muss public oder private sein, damit das Backend weiß ob es das VOD global erreichbar macht oder nur per direktlink.

```
INSERT INTO vod(id, stream_id, clicks, status, location) values(4, 4, 69, 'priv',
'D:\TKKG_Streaming\VODs\6\6_ragingbull_stream.mp4');
```

Klappt nicht, da der Status nicht 'private' bzw. 'public' gesetzt ist. Der Insert schlägt fehl!

```
Fehlerbericht -
ORA-02290: CHECK-Constraint (C##FBPOOL101.SYS_C001129045) verletzt
```

Subscriber

```
tier CHARACTER(1) DEFAULT 1 NOT NULL CHECK (tier BETWEEN '1' AND '3')
```

Der Tier eines Subscriber muss zwischen 1 und 3 sein, da ein status drüber oder drunter für das Backend nicht auszuwerten wäre. Das Tier beschreibt wie viel der Subscriber für sein 'Sub' bezahlt. (Siehe Tier Subscribtion bei Twich)

```
INSERT INTO subscriber(partner_account_id, subscriber_account_id, tier,
subscriber_start,subscriber_end, month_count)
values(1,9,8,'10.09.2021','10.12.2021', 3);
```

Klappt nicht, da Tier 8 undefiniert ist. Dies wäre für das Backend nicht mehr auszuwerten, deswegen checkt oracle dass die Bedingung immer erfüllt sein muss.

```
Fehlerbericht -
ORA-02290: CHECK-Constraint (C##FBPOOL101.SYS_C001129053) verletzt
```

Rekursive Beziehungen

Accounts

Accounts können die rekursive Beziehung moderatorid haben. Wenn dieser Wert gesetzt ist, kann der Account der mit diesem Wert verbunden ist den anderen Account verwalten.

	ID	CREATION	USERNAME	PASSWORT	MAIL	STREAMKEY	MODERATORID
1	3	27.11.21	batmanbegins	161fe540610768de76dba672leaab2a40a2b68ab394297afbb31f4e29e91add2	batman@live.com	LIVE_NOKEY	1
2	4	27.11.21	rye	277d0488076f8ffdfd38a2e6a7a496c395bf0d7641a6681d27c7e39cb931b7e5	rye@gmx.com	live_4_VHS0YSVBETLAYLGvZ8DVGLB9jS0Z6a	1
3	5	27.11.21	dasboot	a27cd870b15cc0747b32f1b4fba1186bb37a10d37646158f8261574417e18600	dasboot@boot.com	live_5_okWgC1FW2gH5AFIM1eW1yAhKqg01tS	1

Alle Drei Accounts haben den Moderator mit der AccountID 1 zugeordnet. Das Bedeuet der Account mit der ID 1 kann Account 3,4 und 5 verwalten.
So ergibt die folgende SQL Query, das Ergebnis wer welchen Account verwaltet:

```
SELECT
    a.username as Verwalter,
    b.username as WirdVerwaltet
FROM
    accounts a,
    accounts b
WHERE
    a.id = b.moderatorid;
```

Folgenden Output:

	VERWALTER	WIRDVERWALTET
1	Ammil	batmanbegins
2	Ammil	rye
3	Ammil	dasboot
4	rye	wasdwafs

Des Weiteren kann man mit der SQL Query:

```
SELECT
    a.username as Verwaltet,
    COUNT(*) as VerwaltetCount
FROM
    accounts a,
    accounts b
WHERE
    a.id = b.moderatorID
GROUP BY
    a.username;
```

Herausfinden wie viele Accounts von einem User verwaltet werden.

Bsp Output:

	VERWALTET	VERWALTETCOUNT
1	rye	1
2	Ammil	3