# WS-01 _ User-Management

**@author:** Kacper Bohaczyk
**@date** 30.11.2023

---

# Voraussetzungen

- Basiskenntnisse von postgres
- Durchführung der 1 INSY Aufgabe
- Betriebsystem Ubuntu

## 1. Rollen

Es werden alle Rollen angelegt, wobei Admin und Kunde Login-Rollen (User) sind, also Rollen mit denen man sich direkt anmelden kann (das ist äquivalent zu einem User).

```
CREATE USER admin  WITH PASSWORD 'schueler' SUPERUSER;
CREATE USER kunde WITH PASSWORD 'schueler';
```

Die weiteren Rollen sind zur Gruppierung von Usern.

```
CREATE ROLE mitarbeiter;
CREATE ROLE redakteur;
```

### Überprüfung

```
SELECT rolname FROM pg_roles;
```

## 2. Berechtigungen

Zahlungen einsehen & anlegen darf nur der Administrator und der Mitarbeiter

```
GRANT SELECT, INSERT ON payment TO admin, mitarbeiter;
```

Zahlungen ändern & löschen darf nur der Administrator

```
GRANT SELECT, INSERT, UPDATE, DELETE ON payment TO admin;
```

Der Kunde darf die Spalte replacement_cost in der Tabelle film nicht sehen (oder: darf alles sehen außer replacement_cost)

```
GRANT SELECT (film_id, title, description, release_year, language_id,
rental_duration, rental_rate, length, rating, last_update, special_features,
fulltext) ON film TO kunde;
```

## Überprüfung

### Mitarbeiter

```
SELECT * FROM payment;
```

--> Die Zahlungen werden zurückgegeben

```
UPDATE payment SET amount=4.99 WHERE payment_id=17503;
```

--> `ERROR: permission denied for table payment`

### Admin

```
UPDATE payment SET amount=4.99 WHERE payment_id=17503;
```

--> Erfolgreich

### Kunde

```
SELECT title FROM film;
```

--> Gibt die Film-Titel erfolgreich zurück

```
SELECT title, replacement_cost FROM film;
SELECT * FROM film;
```

--> `ERROR: permission denied for table film`

# 3. Rollen vs. Accounts

Die Rollen für Mitarbeiter und Redakteur wurden bereits erstellt. Nun kann ein User/Account erstellt werden und die Rechte der Rolle mittels GRANT zugewiesen werden.

```
CREATE USER mitarbeiter1 WITH PASSWORD 'schueler';
GRANT mitarbeiter TO mitarbeiter1;
```

```
CREATE USER redakteur1 WITH PASSWORD 'schueler';
GRANT redakteur TO redakteur1;
```

# 4. Berechtigungen über die Datei pg_hba.conf

Datei `/var/lib/postgresql/14/main/pg_hba.conf` bearbeiten.

```
sudo nano /var/lib/postgresql/14/main/pg_hba.conf
```

## Nur SSL

In der Datei pg_hba.conf alle `host`-Zeilen zu `hostssl` ändern, um **ausschließlich** SSL-Verbindungen zu erlauben.

Beispiel:

```
hostssl    all             all             ::1/128                 trust
```

# 5. View und Policy

---

**Fehler:**

Falls man sich nicht wieder mit der DATA-BASE connecten kann liegt das an keinen Rechten für den Port
Dieser Fehler tritt auf beim neustarten des PC

- der Server läuft aber kein Port ist aktiv

---

## Mit View

View erstellen.

```
CREATE VIEW active_customer_emails AS SELECT email FROM customer WHERE
active=1;
```

Berechtigungen für die View erteilen.

```
GRANT SELECT ON active_customer_emails TO mitarbeiter;
```

Überprüfung:

```
SELECT * FROM active_customer_emails;
```

--> 584 Ergebnisse (599 wären es mit inaktiven)

## Mit Policy

Zuerst wird der Zugriff auf die E-Mail-Spalte beschränkt und dann die Policy für active=true erstellt.

```
GRANT SELECT (email) ON customer TO mitarbeiter;
CREATE POLICY active_customers ON customer TO mitarbeiter USING (active=1);
```

Dann muss noch die Row-Level-Security für die customer-Tabelle aktiviert werden.

```
ALTER TABLE customer ENABLE ROW LEVEL SECURITY;
```

Überprüfung:

```
SELECT email FROM customer;
```

--> 584 Ergebnisse (599 wären es mit inaktiven)