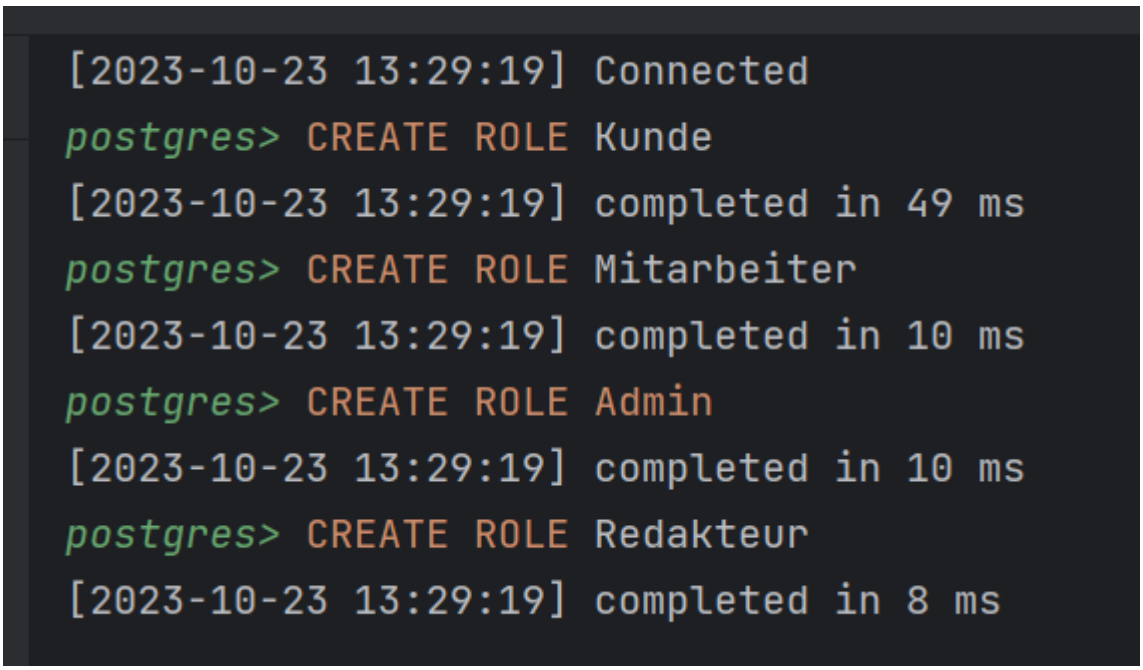
WS 01 _ Benutzerverwaltung unter Posgre SQL [GK]

1: Erstellen der Rollen

Um die Rollen zu erstellen, wird der SQL Befehl `CREATE ROLL` eingesetzt. Um die gewünschten Rollen aus der Angabe zu erstellen, werden diese Commands verwendet:

```
CREATE ROLE kunde;
CREATE ROLE mitarbeiter;
CREATE ROLE admin;
CREATE ROLE redakteur;
```

Hier ist der Output in DataGrip:



Rechte der Rollen:

```
GRANT SELECT, INSERT ON payment TO Mitarbeiter;
REVOKE UPDATE on public.payment FROM Mitarbeiter;
GRANT SELECT, INSERT, DELETE, UPDATE ON payment TO Admin;
REVOKE SELECT, INSERT,UPDATE, DELETE ON payment FROM PUBLIC;
REVOKE SELECT (replacement_cost) ON film FROM Kunde;

GRANT admin TO admin_user;
GRANT mitarbeiter to mitarbeiter_user;
GRANT kunde to kunden_user;
```

Output von DataGrip:

```
postgres.public> GRANT SELECT, INSERT ON payment TO Mitarbeiter
[2023-10-23 13:56:10] completed in 51 ms
postgres.public> GRANT SELECT, INSERT, DELETE, UPDATE ON payment TO Admin
[2023-10-23 13:56:10] completed in 13 ms
postgres.public> REVOKE SELECT, INSERT,UPDATE, DELETE ON payment FROM PUBLIC
[2023-10-23 13:56:10] completed in 14 ms
postgres.public> REVOKE SELECT (replacement_cost) ON film FROM Kunde
[2023-10-23 13:56:10] completed in 4 ms
```

Verändern der Zugriffsbedingungen

In dem `pg_hba.conf` kann man die Zugriffsmethoden bearbeiten.

Ich will ja, wie ich es aus der Angabe verstanden habe, keine unsicheren Verbindungen erlauben, sondern nur über SSL.

Das mache ich in dem config File so:

```
host  all  all  0.0.0.0/0  reject
hostssl all all 0.0.0.0/0 md5
```

View und Policy:

View:

```
CREATE VIEW customers_filtered AS
SELECT *
FROM customers
WHERE activebool = true
WITH CHECK OPTION;
GRANT SELECT ON customers_filtered TO "mitarbeiter";
```

Policy:

```
ALTER TABLE customers ENABLE ROW LEVEL SECURITY;

CREATE POLICY email_restrict_policy
  ON customers
  FOR SELECT, UPDATE, DELETE
  USING (activebool = true)
  WITH CHECK (activebool = true)
  FOR "Mitarbeiter";

ALTER TABLE customers ENABLE ROW LEVEL SECURITY;
```

Aber damit das auch funktioniert muss man RLS aktivieren:

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

Und um Sachen auszuprobieren, erstelle ich mir ein paar Benutzer:

```
-- Erstelle separate Benutzerkonten für Kunde, Mitarbeiter, Admin und Redakteur
CREATE USER kunden_user WITH PASSWORD 'kunden_password';
CREATE USER mitarbeiter_user WITH PASSWORD 'mitarbeiter_password';
CREATE USER admin_user WITH PASSWORD 'admin_password';
CREATE USER redakteur_user WITH PASSWORD 'redakteur_password';

-- Weise den Benutzern die entsprechenden Rollen zu
ALTER USER kunden_user SET ROLE Kunde;
ALTER USER mitarbeiter_user SET ROLE Mitarbeiter;
ALTER USER admin_user SET ROLE Admin;
ALTER USER redakteur_user SET ROLE Redakteur;
```

Hier zb will ich als Kunde auf replacement_cost zugreifen:

```
root@12250f2d99f6:/# psql -d postgres -U kunden_user
WARNING:  permission denied to set role "kunde"
psql (16.0 (Debian 16.0-1.pgdg120+1))
Type "help" for help.

postgres=> SELECT replacement_cost from film
postgres-> ;
ERROR:  permission denied for table film
postgres=>
```

Und hier will auf die Payments zugreifen:

```
postgres=> select * from payment;
ERROR:  permission denied for table payment
postgres=>
```

Und wenn ich jetzt als Mitarbeiter payments verändern will, dann kann ich das nicht:

```
root@12250f2d99f6:/# psql "host=0.0.0.0 port=5432 dbname=postgres user=mitarbeiter_user password=mitarbeiter_password"
WARNING:  permission denied to set role "mitarbeiter"
psql (16.0 (Debian 16.0-1.pgdg120+1))
Type "help" for help.

postgres=> DELETE from payment where payment_id=17503;
ERROR:  permission denied for table payment
postgres=> Select * from payment;
 payment_id | customer_id | staff_id | rental_id | amount |        payment_date
------------+-------------+----------+-----------+--------+----------------------------
      17503 |         341 |        2 |      1520 |   7.99 | 2007-02-15 22:25:46.996577
      17504 |         341 |        1 |      1778 |   1.99 | 2007-02-16 17:23:14.996577
```

Als Admin kann ich natürlich auf replacement cost und so sehen:

```
postgres=> select replacement_cost from film;
 replacement_cost
------------------
            14.99
```