Transaction:

BEGIN TRANSACTION;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE account

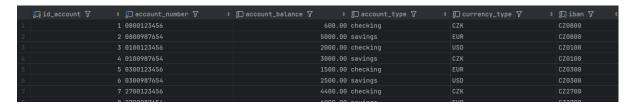
SET account_balance = account_balance - 200 where account_number = '0800123456';

UPDATE account

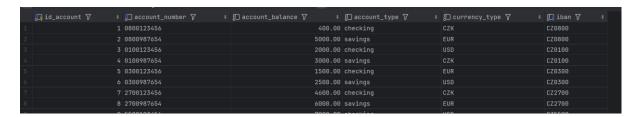
SET account_balance = account_balance + 200 where account_number = '2700123456';

COMMIT TRANSACTION;

Before:



After:



View:

CREATE VIEW person_card AS

SELECT

```
p.birth_certificate_number AS birth_certificate_number,p.full_name AS person_name,pc.card_number,pc.card_status
```

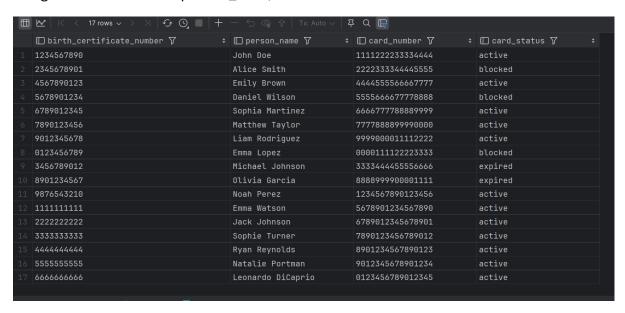
FROM

Person p

JOIN

Payment_Card pc ON p.id_person = pc.account_id;

Usage: SELECT * FROM person_card;



Trigger:

CREATE TRIGGER account_balance_trigger

BEFORE INSERT OR UPDATE ON account

FOR EACH ROW EXECUTE PROCEDURE check_balance_trigger();

CREATE OR REPLACE FUNCTION check_balance_trigger() RETURNS TRIGGER AS \$\$
BEGIN

IF NEW.account_balance IS NULL OR NEW.account_balance < 0 THEN

RAISE EXCEPTION 'Account balance cannot be null or negative';

END IF;

RETURN NEW;

END;

\$\$ LANGUAGE plpgsql;

Usage:

INSERT INTO account (iban, account_number, account_balance, account_type, currency_type)

VALUES ('some_iban', '1234567890', -100.00, 'checking', 'USD');

[P0001] ERROR: Account balance cannot be null or negative Where: PL/pgSQL function check_balance_trigger() line 4 at RAISE

Index:

CREATE INDEX IF NOT EXISTS idx_bank_iban ON Bank (iban);

EXPLAIN ANALYZE SELECT * FROM Bank where iban LIKE '%CZ%';

Creating an index on the "iban" column in the "Bank" table will help accelerate queries that utilize the LIKE operator with the '%CZ%' pattern. The index allows the database to quickly locate rows containing the substring "CZ", thereby improving application performance and reducing wait times for users.