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:

	id_account	account_number	account_balance	account_type	currency_type	iban
1	1	0800123456	600.00	checking	CZK	CZ0800
2	2	0800987654	5000.00	savings	EUR	CZ0800
3	3	0100123456	2000.00	checking	USD	CZ0100
4	4	0100987654	3000.00	savings	CZK	CZ0100
5	5	0300123456	1500.00	checking	EUR	CZ0300
6	6	0300987654	2500.00	savings	USD	CZ0300
7	7	2700123456	4400.00	checking	CZK	CZ2700

After:

	id_account	account_number	account_balance	account_type	currency_type	iban
1	1	0800123456	400.00	checking	CZK	CZ0800
2	2	0800987654	5000.00	savings	EUR	CZ0800
3	3	0100123456	2000.00	checking	USD	CZ0100
4	4	0100987654	3000.00	savings	CZK	CZ0100
5	5	0300123456	1500.00	checking	EUR	CZ0300
6	6	0300987654	2500.00	savings	USD	CZ0300
7	7	2700123456	4600.00	checking	CZK	CZ2700
8	8	2700987654	6000.00	savings	EUR	CZ2700

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;

17 rows

	birth_certificate_number	person_name	card_number	card_status
1	1234567890	John Doe	1111222233334444	active
2	2345678901	Alice Smith	2222333344445555	blocked
3	4567890123	Emily Brown	4444555566667777	active
4	5678901234	Daniel Wilson	5555666677778888	blocked
5	6789012345	Sophia Martinez	6666777788889999	active
6	7890123456	Matthew Taylor	7777888899990000	active
7	9012345678	Liam Rodriguez	9999000011112222	active
8	0123456789	Emma Lopez	0000111122223333	blocked
9	3456789012	Michael Johnson	3333444455556666	expired
10	8901234567	Olivia Garcia	8888999900001111	expired
11	9876543210	Noah Perez	1234567890123456	active
12	1111111111	Emma Watson	5678901234567890	active
13	2222222222	Jack Johnson	6789012345678901	active
14	3333333333	Sophie Turner	7890123456789012	active
15	4444444444	Ryan Reynolds	8901234567890123	active
16	5555555555	Natalie Portman	9012345678901234	active
17	6666666666	Leonardo DiCaprio	0123456789012345	active

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%';

Output | Result 6-2

5 rows

QUERY PLAN
Seq Scan on bank (cost=0.00..699.00 rows=3 width=41) (actual time=0.043..11.889 rows=9 loops=1)
Filter: ((iban)::text ~~ '%CZ%'::text)
Rows Removed by Filter: 31991
Planning Time: 0.472 ms
Execution Time: 11.939 ms

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.