CP04 - expanding SQL queries

I decided to expand my database using these queries:

```
-- Trigger
CREATE OR REPLACE FUNCTION assign_accessory_to_instrument(p_accessoryISBN VARCHAR, p_instrumentModelNumber VARCHAR)
    RETURNS VOID AS $$
BEGIN

    PERFORM * FROM Accessory WHERE ISBN = p_accessoryISBN FOR UPDATE;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Accessory with ISBN % does not exist', p_accessoryISBN;
    END IF;

    PERFORM * FROM InstrumentProduct WHERE modelNumber = p_instrumentModelNumber FOR UPDATE;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Instrument with model number % does not exist', p_instrumentModelNumber;
    END IF;

    -- Update the accessory to link it to the instrument
    UPDATE Accessory SET comesWith = p_instrumentModelNumber WHERE ISBN = p_accessoryISBN;
EXCEPTION
    WHEN OTHERS THEN
        RAISE;
END;
$$ LANGUAGE plpgsql;



-- Transaction
CREATE OR REPLACE FUNCTION assign_accessory_to_instrument(p_accessoryISBN VARCHAR(50), p_instrumentModelNumber
VARCHAR(50))
    RETURNS VOID AS $$
BEGIN
    BEGIN;

    PERFORM * FROM Accessory WHERE ISBN = p_accessoryISBN FOR UPDATE;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Accessory with ISBN % does not exist', p_accessoryISBN;
    END IF;

    PERFORM * FROM InstrumentProduct WHERE modelNumber = p_instrumentModelNumber FOR UPDATE;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Instrument with model number % does not exist', p_instrumentModelNumber;
    END IF;

    UPDATE Accessory SET comesWith = p_instrumentModelNumber WHERE ISBN = p_accessoryISBN;

    COMMIT;
```

```
  EXCEPTION
    WHEN OTHERS THEN
      -- Rollback the transaction on error
      ROLLBACK;
      RAISE;
  END;
END;
$$ LANGUAGE plpgsql;

-- View
CREATE VIEW CustomerInfo AS
SELECT p.userNickname, p.userMail, p.fullName,
    p.city, p.zip, p.street, p.phoneNumber
FROM privateinfo p
  JOIN customeruser c ON p.userNickname = c.userNickname AND p.userMail = c.userMail;




-- Indexing
CREATE INDEX idx_AccessoryType ON Accessory(type);
```

Quick explanation:

Trigger:

Trigger was created to ensure that one instrument doesn't come with same accessory type. Because most of my types wouldn't make sense as duplicates that would come with a purchase of an instrument, I had to make a trigger.

Trigger could also be used in hierarchy of entities, but I have already created functions that handle that in CP03.

Transaction:

Further building on accessories, I imagine that some users will be adding accessories to the instruments. To ensure data integrity I have created a transaction function that should be used for data manipulation, when someone wants to attach an accessory to an instrument. Since I've put the transaction into a function call there is no need to specify COMMIT or ROLLBACK.

View:

View was created to have all customer data show up in one simple SELECT. User entities have the biggest number of attributes, so to make my life easier in the future I created this view.

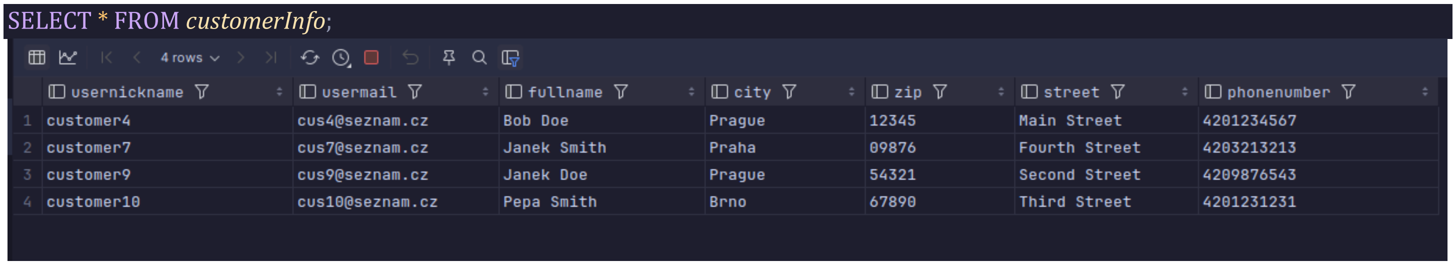
Indexing:

As the accessory entity is the biggest table yet I felt like it would be perfect one to index first.

Showcase:

View of customer info:

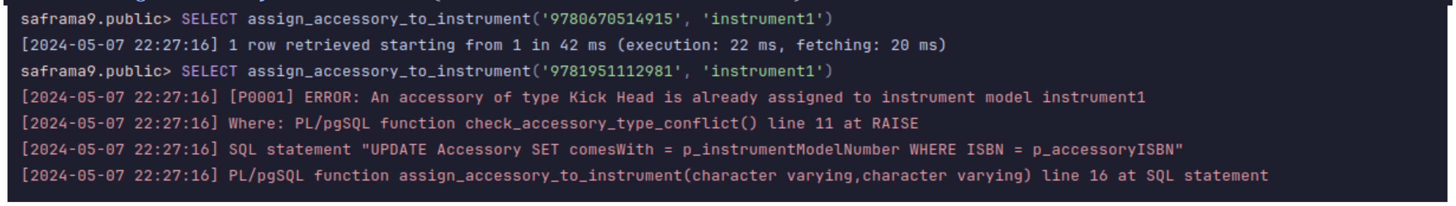
```
SELECT * FROM customerInfo;
```



usernickname	usermail	fullname	city	zip	street	phonenumber
customer4	cus4@seznam.cz	Bob Doe	Prague	12345	Main Street	4201234567
customer7	cus7@seznam.cz	Janek Smith	Praha	09876	Fourth Street	4203213213
customer9	cus9@seznam.cz	Janek Doe	Prague	54321	Second Street	4209876543
customer10	cus10@seznam.cz	Pepa Smith	Brno	67890	Third Street	4201231231

Trigger blocking update (via transaction function)

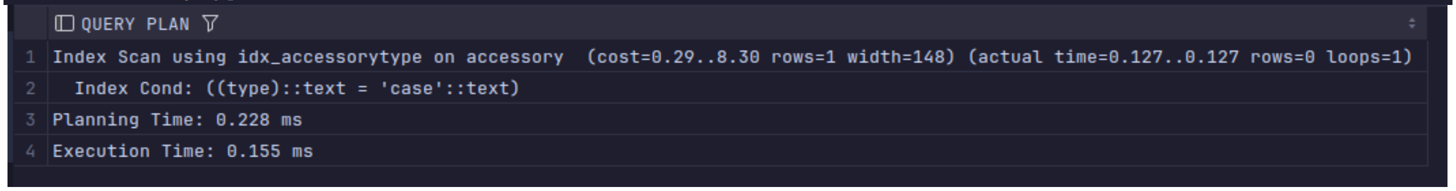
```
SELECT assign_accessory_to_instrument('9780670514915', 'instrument1');
SELECT assign_accessory_to_instrument('9781951112981', 'instrument1');
```



```
saframa9.public> SELECT assign_accessory_to_instrument('9780670514915', 'instrument1')
[2024-05-07 22:27:16] 1 row retrieved starting from 1 in 42 ms (execution: 22 ms, fetching: 20 ms)
saframa9.public> SELECT assign_accessory_to_instrument('9781951112981', 'instrument1')
[2024-05-07 22:27:16] [P0001] ERROR: An accessory of type Kick Head is already assigned to instrument model instrument1
[2024-05-07 22:27:16] Where: PL/pgSQL function check_accessory_type_conflict() line 11 at RAISE
[2024-05-07 22:27:16] SQL statement "UPDATE Accessory SET comesWith = p_instrumentModelNumber WHERE ISBN = p_accessoryISBN"
[2024-05-07 22:27:16] PL/pgSQL function assign_accessory_to_instrument(character varying,character varying) line 16 at SQL statement
```

SELECT using indexing:

```
EXPLAIN ANALYSE
SELECT * FROM accessory
WHERE accessory.type = 'case';
```



```
QUERY PLAN
Index Scan using idx_accessorytype on accessory  (cost=0.29..8.30 rows=1 width=148) (actual time=0.127..0.127 rows=0 loops=1)
  Index Cond: ((type)::text = 'case'::text)
Planning Time: 0.228 ms
Execution Time: 0.155 ms
```