1. Transakce

1. Vytvoření klienta (izolace Read Committed)

Při vytváření klienta používáme tabulky person a client, aby se zabránit případ, kdy jsme vytvořili person, ale nebyl vytvořen client, používáme transakce (pak pro tuto příležitost jsem udělal funkci, která se již provádí v samostatné transakci).

```
BEGIN;

DO $$
    DECLARE
        last_person_id INT;
    BEGIN
        INSERT INTO Person(email, first_name, last_name, address,
phone_number)
          VALUES ('VeraKrchnakova@dayrep.com', 'Věra', 'Krchňáková',
'Stožická 86 391 31 Dražice', '+420724636438')
              RETURNING person_id INTO last_person_id;

        INSERT INTO Client(pass_id, person_id)
            VALUES ('U48459877', last_person_id);
END;
$$;
```

```
COMMIT;
```

2. Převod bonusů mezi zákazníky (izolace Serializable)

Klient s id 2 převádí své bonusy klientovi s id 4 a id 4, probíhá tento process ve dvou tranzakcích (nějak to dopadlo :)). A pokud nebudeme používat transakce s úrovní izolace Serializable může se stát tak, že dostane záporný počet bonusů (určitě bych mohl prostě použít CHECK, ale nic jiného jsem si nevymyslel :( ).

```
BEGIN ISOLATION LEVEL SERIALIZABLE;
DO $$
    BEGIN
        IF (SELECT bonus_points FROM client where client_id = 2) >= 20 THEN
            UPDATE client SET bonus_points = (SELECT bonus_points FROM
client WHERE client_id = 2) + 20 WHERE client_id = 3;
            UPDATE client SET bonus_points = (SELECT bonus_points FROM
client WHERE client_id = 2) - 20 WHERE client_id = 2;
        END IF;
    END;
    $$;
COMMIT;


BEGIN ISOLATION LEVEL SERIALIZABLE;
```

```
DO $$
    BEGIN
        IF (SELECT bonus_points FROM client where client_id = 2) >= 10 THEN
            UPDATE client SET bonus_points = (SELECT bonus_points FROM
client WHERE client_id = 2) + 10 WHERE client_id = 4;
            UPDATE client SET bonus_points = (SELECT bonus_points FROM
client WHERE client_id = 2) - 10 WHERE client_id = 2;
        END IF;
    END;
    $$;
COMMIT;
```

3. Vytvoření objednávky (izolace Read Committed)

Stejný případ jako “Vytvoření klienta”.

```
BEGIN;

DO $$
    DECLARE
        last_order_id INT;
    BEGIN
```

```
        INSERT INTO ClientOrder(client_id, department_from_id,
department_to_id, price)
            VALUES (3, 1, 2, 434)
            RETURNING order_id INTO last_order_id;

        INSERT INTO Recipient(order_id, first_name, last_name,
phone_number)
            VALUES (last_order_id, 'Mikita', 'Citarovic', '+420606169911');
END;
$$;
```

2. Vytvoření a použití pohledu

1. Získání všech řidičů

Vytvoření

```
CREATE OR REPLACE VIEW drivers AS
    SELECT * FROM Employee
        WHERE position_name = 'Řidič';
```

Výběr

```
SELECT * FROM drivers;
```

	employee_id	card_number	position_name	person_id	department_id
1	5	5465356	Řidič	5	3
2	6	53453634	Řidič	6	4
3	7	5346734	Řidič	7	4
4	8	5345336	Řidič	8	5

2. Získání informací o všech zákaznících a počtu jejich objednávek

Vytvoření

```
CREATE OR REPLACE VIEW clients_stats AS
SELECT person.first_name,
       person.last_name,
       person.email,
       person.phone_number,
       client.pass_id,
       COUNT(clientorder.client_id) AS orders_count
FROM   ClientOrder
       LEFT JOIN Client
              ON clientorder.client_id = client.client_id
       LEFT JOIN Person
              ON client.client_id = person.person_id
GROUP  BY clientorder.client_id,
          person.first_name,
          person.last_name,
          person.email,
          person.phone_number,
          client.pass_id
```

```
ORDER  BY COUNT(clientorder.client_id) DESC
```

Výběr

```
SELECT * FROM clients_stats
```

	first_name	last_name	email	phone_number	pass_id	orders_count
1	Otakar	Tůma	OtakarTuma@teleworm.us	+420603980434	U4003010	216
2	Petr	Král	PetrKral@teleworm.us	+420314645076	U0403070	205
3	Antonie	Kubešová	AntonieKubesova@teleworm.us	+420720292683	U4503410	200
4	Marek	Jindřich	MarekJindrich@dayrep.com	<null>	U0003010	195
5	Jiří	Doucha	JiriDoucha@dayrep.com	+420603470913	U0205010	192

3. Vytvoření a použití triggeru

1. On Client Delete

Vzhledem k tomu, že tabulka client je zděděná z tabulky person, při odebrání klienta je odstraněn a odpovídající záznam v person.

```
CREATE OR REPLACE FUNCTION client_delete_handler()
     RETURNS TRIGGER
     LANGUAGE 'plpgsql'
AS
  $$
BEGIN
  DELETE
  FROM   person
  WHERE  person_id = OLD.person_id;
```

```
  RETURN NULL;
END;
$$;

CREATE TRIGGER on_client_delete
AFTER DELETE
ON client
FOR EACH ROW
EXECUTE PROCEDURE client_delete_handler();
```

2. On Employee Delete

```
CREATE OR REPLACE FUNCTION employee_delete_handler()
     RETURNS TRIGGER
     LANGUAGE 'plpgsql'
AS
  $$
BEGIN
  DELETE
  FROM   person
  WHERE  person_id = OLD.person_id;

  RETURN NULL;
END;
$$;

CREATE TRIGGER on_employee_delete
```

```
AFTER DELETE
ON employee
FOR EACH ROW
EXECUTE PROCEDURE employee_delete_handler();
```

4. Vytvoření a použití indexu

Index pro attribute creation_date_time z tabulky ClientOrder. Často budeme hledat nějaké objednavky na základě data vytvoření objednávky, takže pro rychlejší vyhledávání je lepší vytvořit index. Budeme používat typ indexu B-tree (to je defaultní typ). Nejprve provedeme analýzu, kolik budou trvat dotazy bez indexu.

SQL Dotazy a analýza

1. Vytvoření indexu

```
CREATE INDEX clientorder_creation_date_time
  ON ClientOrder(creaiton_date_time);
```

2. Vyhledávání podle určitého data

```
EXPLAIN (analyze, costs off, timing off)
     SELECT * FROM ClientOrder
     WHERE creaiton_date_time = '2022-04-19 21:37:33.814295';
```

1. Bez indexu (3 průchody)

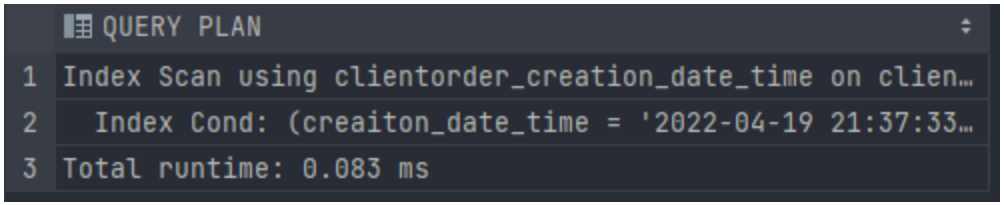
	QUERY PLAN
1	Seq Scan on clientorder (actual rows=1 loops=1)
2	Filter: (creaiton_date_time = '2022-04-19 21:37:33.814…
3	Rows Removed by Filter: 1316
4	Total runtime: 0.501 ms

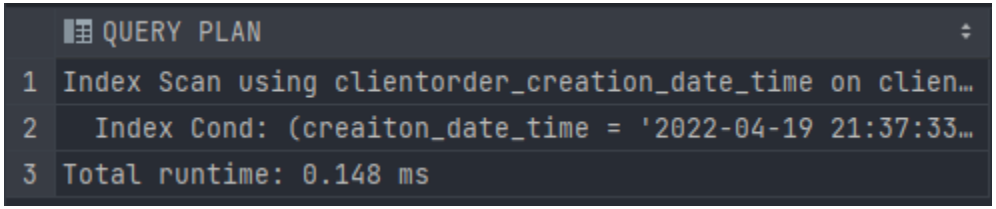
	QUERY PLAN
1	Seq Scan on clientorder (actual rows=1 loops=1)
2	Filter: (creaiton_date_time = '2022-04-19 21:37:33.814…
3	Rows Removed by Filter: 1316
4	Total runtime: 0.765 ms

	QUERY PLAN
1	Seq Scan on clientorder (actual rows=1 loops=1)
2	Filter: (creaiton_date_time = '2022-04-19 21:37:33.814…
3	Rows Removed by Filter: 1316
4	Total runtime: 0.327 ms

2. S indexem (3 průchody)

	QUERY PLAN
1	Index Scan using clientorder_creation_date_time on clien…
2	Index Cond: (creaiton_date_time = '2022-04-19 21:37:33…
3	Total runtime: 0.096 ms

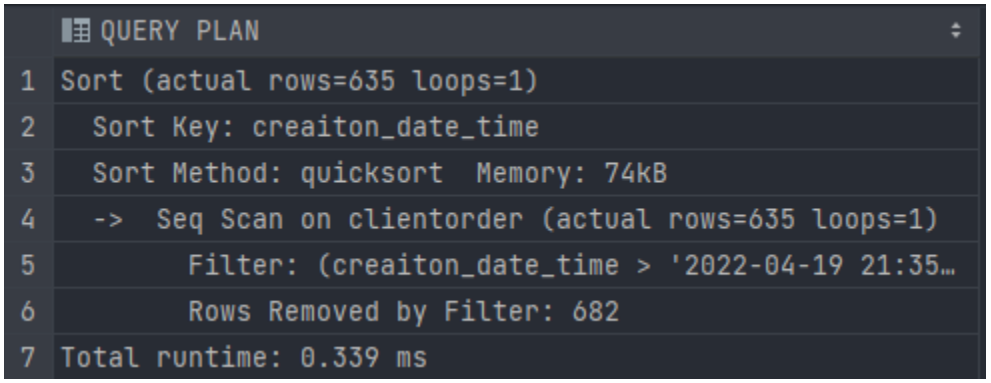




3. Vyhledávání od nějakého data

```
EXPLAIN (analyze, costs off, timing off)
      SELECT * FROM ClientOrder
      WHERE creaiton_date_time > '2022-04-19 21:35:54.625657'
      ORDER BY creaiton_date_time
desc;
```

1. Bez indexu (3 průchody)

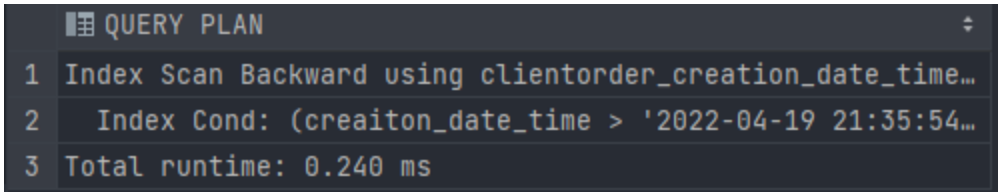


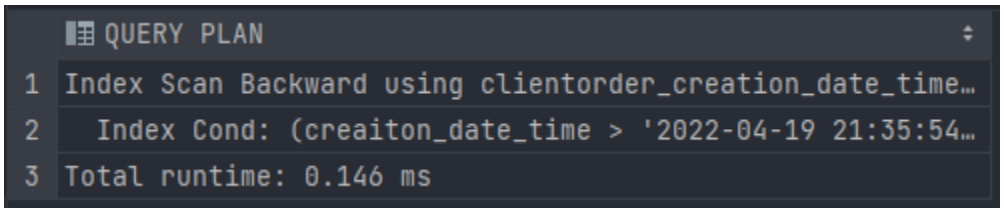
```
QUERY PLAN
Sort (actual rows=635 loops=1)
  Sort Key: creaiton_date_time
  Sort Method: quicksort  Memory: 74kB
  ->  Seq Scan on clientorder (actual rows=635 loops=1)
        Filter: (creaiton_date_time > '2022-04-19 21:35…
        Rows Removed by Filter: 682
Total runtime: 2.068 ms
```

```
QUERY PLAN
Sort (actual rows=635 loops=1)
  Sort Key: creaiton_date_time
  Sort Method: quicksort  Memory: 74kB
  ->  Seq Scan on clientorder (actual rows=635 loops=1)
        Filter: (creaiton_date_time > '2022-04-19 21:35…
        Rows Removed by Filter: 682
Total runtime: 0.617 ms
```

2. S indexem (3 průchody)

```
QUERY PLAN
Index Scan Backward using clientorder_creation_date_time…
  Index Cond: (creaiton_date_time > '2022-04-19 21:35:54…
Total runtime: 0.235 ms
```

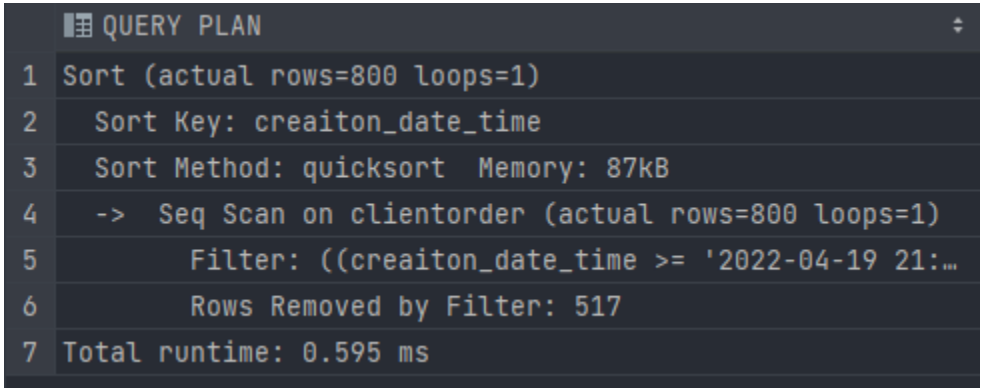




4. Hledání mezi daty

```
EXPLAIN (analyze, costs off, timing off)
      SELECT * FROM ClientOrder
      WHERE creaiton_date_time
      BETWEEN '2022-04-19 21:24:10.005322' AND '2022-05-03 19:49:24.021307'
      ORDER BY creaiton_date_time desc;
```

1. Bez indexu (3 průchody)

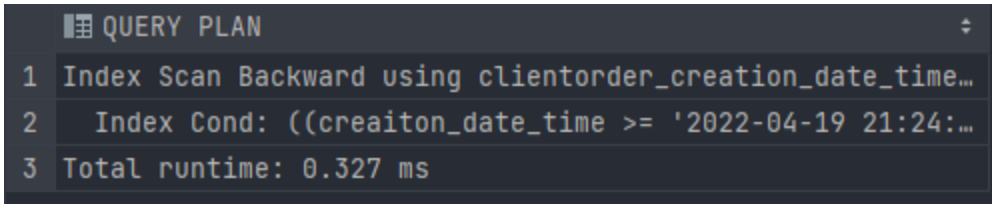


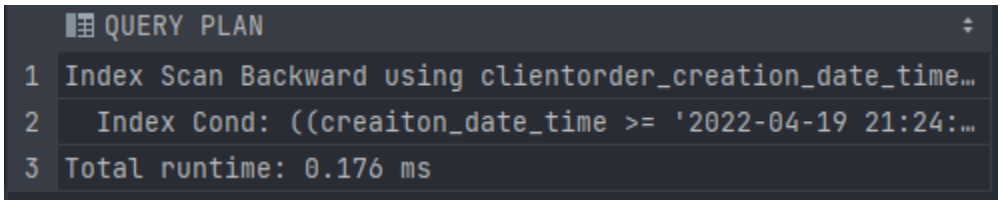
	QUERY PLAN
1	Sort (actual rows=800 loops=1)
2	Sort Key: creaiton_date_time
3	Sort Method: quicksort Memory: 87kB
4	-> Seq Scan on clientorder (actual rows=800 loops=1)
5	Filter: ((creaiton_date_time >= '2022-04-19 21:…
6	Rows Removed by Filter: 517
7	Total runtime: 0.747 ms

	QUERY PLAN
1	Sort (actual rows=800 loops=1)
2	Sort Key: creaiton_date_time
3	Sort Method: quicksort Memory: 87kB
4	-> Seq Scan on clientorder (actual rows=800 loops=1)
5	Filter: ((creaiton_date_time >= '2022-04-19 21:…
6	Rows Removed by Filter: 517
7	Total runtime: 0.841 ms

2. S indexem (3 průchody)

	QUERY PLAN
1	Index Scan Backward using clientorder_creation_date_time…
2	Index Cond: ((creaiton_date_time >= '2022-04-19 21:24:…
3	Total runtime: 0.260 ms





Z výsledků je vidět, že index urychlil dotazy. Výsledek by byl názornější, kdybychom měli více dat.

5. Vytvoření funkce

1. Add New Order

```
CREATE OR REPLACE FUNCTION add_new_order(
        P_client_id client.client_id%TYPE,
        p_department_from_id department.department_id%TYPE,
        p_department_to_id department.department_id%TYPE,
        p_price department.department_id%TYPE,
        p_recipient_first_name recipient.first_name%TYPE,
        p_recipient_last_name recipient.last_name%TYPE,
        p_recipient_phone_number recipient.phone_number%TYPE)
    RETURNS void
    LANGUAGE plpgsql
AS
$$
```

```
    DECLARE last_order_id INT;
    BEGIN
       INSERT INTO ClientOrder(client_id, department_from_id, department_to_id, price)
              VALUES (p_client_id, p_department_from_id, p_department_to_id, p_price)
                   RETURNING order_id INTO last_order_id;

    INSERT INTO Recipient(order_id, first_name, last_name, phone_number)
        VALUES (last_order_id, p_recipient_first_name, p_recipient_last_name,
p_recipient_phone_number);
    END;
$$;
```

2. Add New Client

```
CREATE OR REPLACE FUNCTION add_new_client(
        p_email person.email%TYPE,
        p_first_name person.first_name%TYPE,
        p_last_name person.last_name%TYPE,
        p_address person.address%TYPE,
        p_phone_number person.phone_number%TYPE,
        p_pass_id client.pass_id%TYPE
    )
    RETURNS void
    LANGUAGE plpgsql
AS
$$
    DECLARE last_person_id INT;
    BEGIN
        INSERT INTO Person(email, first_name, last_name, address, phone_number)
```

```
            VALUES (p_email, p_first_name, p_last_name, p_address, p_phone_number)
                RETURNING person_id INTO last_person_id;

        INSERT INTO Client(pass_id, person_id)
            VALUES (p_pass_id, last_person_id);
    END;
$$;
```