1. Transakce

1. Vytvoření operation_room(izolace Read Committed)

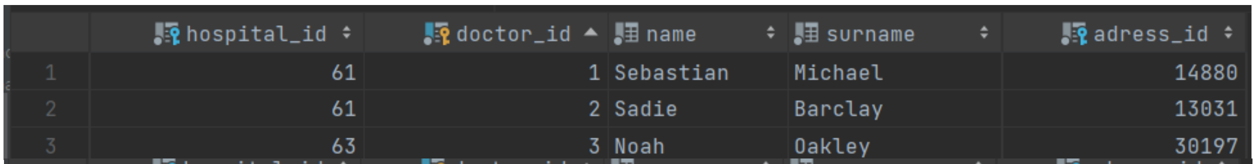
Při vytváření operation_room používáme tabulky hospital a operation_room, aby se zabránit případ, kdy jsme vytvořili hospital, ale nebyl vytvořen alespon 1 operation_room, používáme transakce

```
BEGIN;
DO $$
DECLARE hosp_id INT;
BEGIN
INSERT INTO hospital(hospital_id, name, web_site, adress_id)
VALUES (103, 'Odesa national hospital', 'onh.ua', 103)
RETURNING adress_id INTO hosp_id;
INSERT INTO operation_room
VALUES (hosp_id, 1001, 1);
END $$;
COMMIT;
```

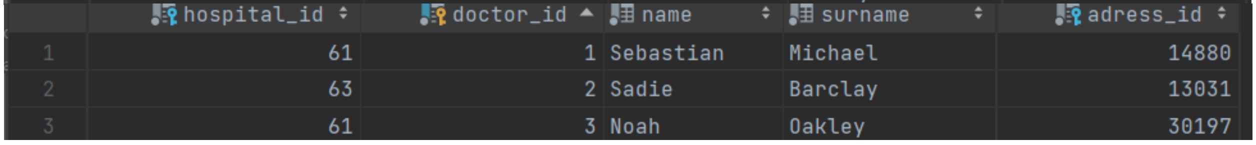
2. Výměna 2 lékařů mezi nemocnicemi.(izolace Serializible)

Podstatou úkolu je, že 2 lékaři na výměnném programu změní zaměstnání (dočasně nebo ne, to je jedno), jejich hospital_id bude změněno. Nevěděl jsem co jiného vymyslet. Doufám, že za toto řešení nebudete strhávat body:D.

```
BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
DO $$
DECLARE hospital1 INT;
DECLARE hospital2 INT;
BEGIN
SELECT doctor.hospital_id INTO hospital1 FROM doctor WHERE
doctor_id = 2;
SELECT doctor.hospital_id INTO hospital2 FROM doctor WHERE
doctor_id = 3;
IF (SELECT doctor.hospital_id FROM doctor WHERE doctor_id = 2)
!=
   (SELECT doctor.hospital_id FROM doctor WHERE doctor_id = 3)
    THEN UPDATE doctor SET hospital_id = hospital2 WHERE
doctor_id = 2;
    UPDATE doctor SET hospital_id = hospital1 WHERE doctor_id =
3;
END IF;
END $$;
COMMIT;
```



	hospital_id	doctor_id	name	surname	adress_id
1	61	1	Sebastian	Michael	14880
2	61	2	Sadie	Barclay	13031
3	63	3	Noah	Oakley	30197

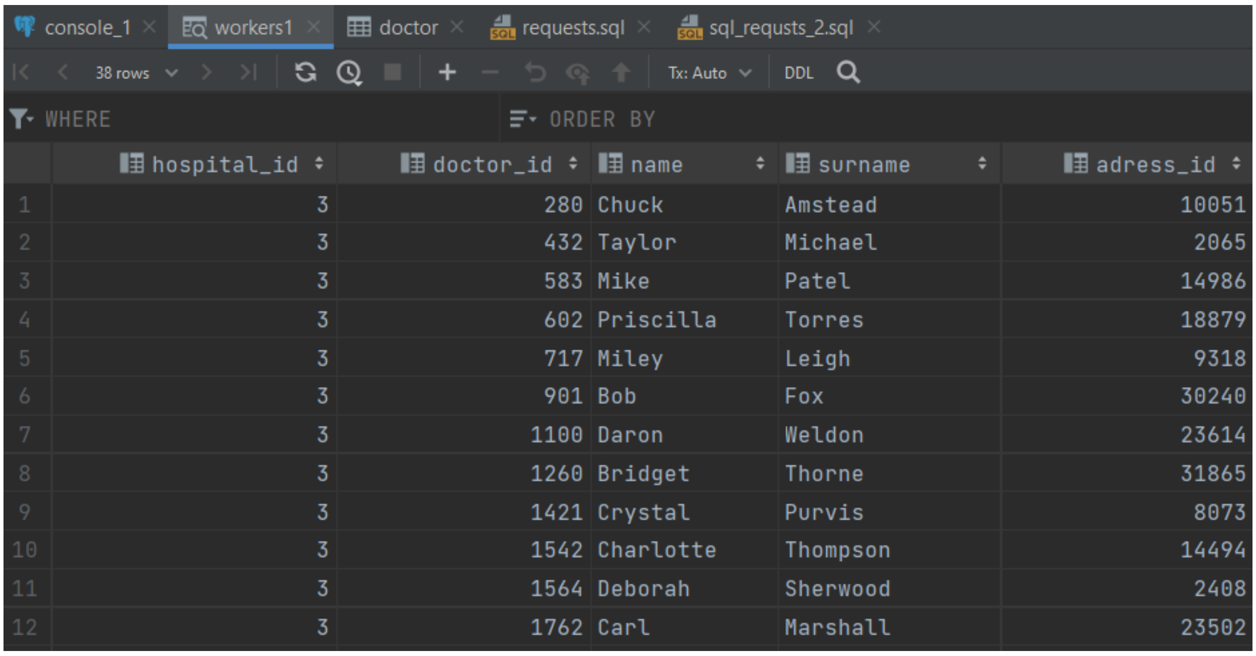


	hospital_id	doctor_id	name	surname	adress_id
1	61	1	Sebastian	Michael	14880
2	63	2	Sadie	Barclay	13031
3	61	3	Noah	Oakley	30197

2. Vytvoření a použití pohledu

1. Získání všech doktoru z hospitalú který ma ID = 3

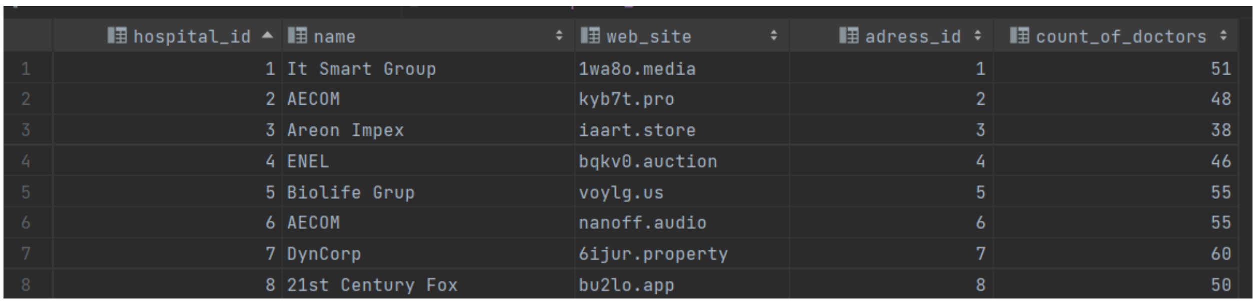
```sql
--creating
CREATE OR REPLACE VIEW workers1 AS
    SELECT * FROM doctor
WHERE hospital_id = 3;
--selecting
SELECT FROM workers1;
```



	hospital_id	doctor_id	name	surname	adress_id
1	3	280	Chuck	Amstead	10051
2	3	432	Taylor	Michael	2065
3	3	583	Mike	Patel	14986
4	3	602	Priscilla	Torres	18879
5	3	717	Miley	Leigh	9318
6	3	901	Bob	Fox	30240
7	3	1100	Daron	Weldon	23614
8	3	1260	Bridget	Thorne	31865
9	3	1421	Crystal	Purvis	8073
10	3	1542	Charlotte	Thompson	14494
11	3	1564	Deborah	Sherwood	2408
12	3	1762	Carl	Marshall	23502

2. Získání informací o počtu všech doctoru ve všech nemocnicech

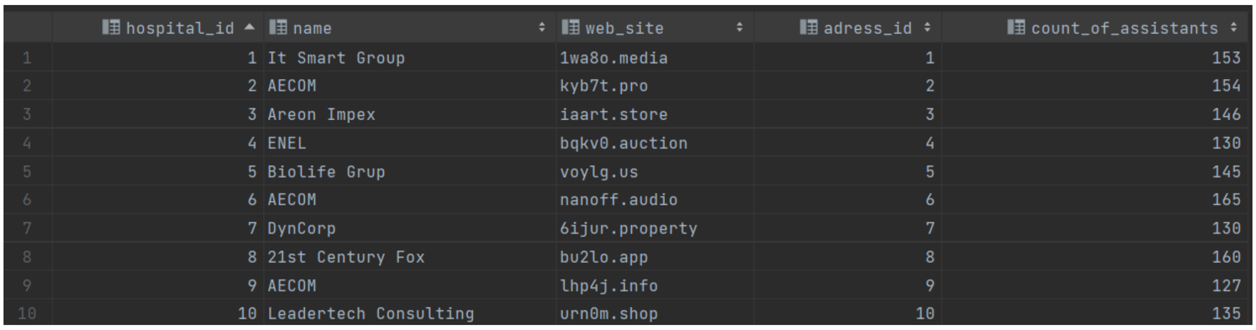
```sql
CREATE OR REPLACE VIEW count_of_doctors_per_hospital AS
SELECT hospital.hospital_id, hospital.name, hospital.web_site,
hospital.adress_id,
COUNT(hospital.hospital_id) AS count_of_doctors
FROM hospital
LEFT JOIN doctor
ON doctor.hospital_id = hospital.hospital_id
GROUP BY hospital.hospital_id, hospital.name, hospital.web_site,
hospital.adress_id;
SELECT FROM count_of_doctors_per_hospital;
```



	hospital_id	name	web_site	adress_id	count_of_doctors
1	1	It Smart Group	1wa8o.media	1	51
2	2	AECOM	kyb7t.pro	2	48
3	3	Areon Impex	iaart.store	3	38
4	4	ENEL	bqkv0.auction	4	46
5	5	Biolife Grup	voylg.us	5	55
6	6	AECOM	nanoff.audio	6	55
7	7	DynCorp	6ijur.property	7	60
8	8	21st Century Fox	bu2lo.app	8	50

3. Získání informací o počtu všech assistantech ve všech nemocnicech

```
CREATE OR REPLACE VIEW count_of_assistants_per_hospital AS
SELECT hospital.hospital_id, hospital.name, hospital.web_site,
hospital.adress_id,
COUNT(hospital.hospital_id) AS count_of_doctors
FROM hospital
LEFT JOIN doctor
ON doctor.hospital_id = hospital.hospital_id
GROUP BY hospital.hospital_id, hospital.name, hospital.web_site,
hospital.adress_id;
SELECT FROM count_of_assistants_per_hospital;
```



	hospital_id	name	web_site	adress_id	count_of_assistants
1	1	It Smart Group	1wa8o.media	1	153
2	2	AECOM	kyb7t.pro	2	154
3	3	Areon Impex	iaart.store	3	146
4	4	ENEL	bqkv0.auction	4	130
5	5	Biolife Grup	voylg.us	5	145
6	6	AECOM	nanoff.audio	6	165
7	7	DynCorp	6ijur.property	7	130
8	8	21st Century Fox	bu2lo.app	8	160
9	9	AECOM	lhp4j.info	9	127
10	10	Leadertech Consulting	urn0m.shop	10	135

3. Vytvoření a použití triggeru

1. On operation_room Delete

Vzhledem k tomu, že tabulka hospital je zděděná z tabulky operation_room, při odebrání hospital je odstraněn a odpovídající záznam v operation_room

```
CREATE OR REPLACE FUNCTION hospital_delete_handler()
RETURNS TRIGGER
LANGUAGE 'plpgsql'
AS
$$
BEGIN
DELETE
FROM hospital
WHERE hospital_id = old.hospital_id;
RETURN NULL;
END;
$$;
CREATE TRIGGER on_hospital_delete
AFTER DELETE
ON operation_room
FOR EACH ROW
EXECUTE PROCEDURE hospital_delete_handler();
```

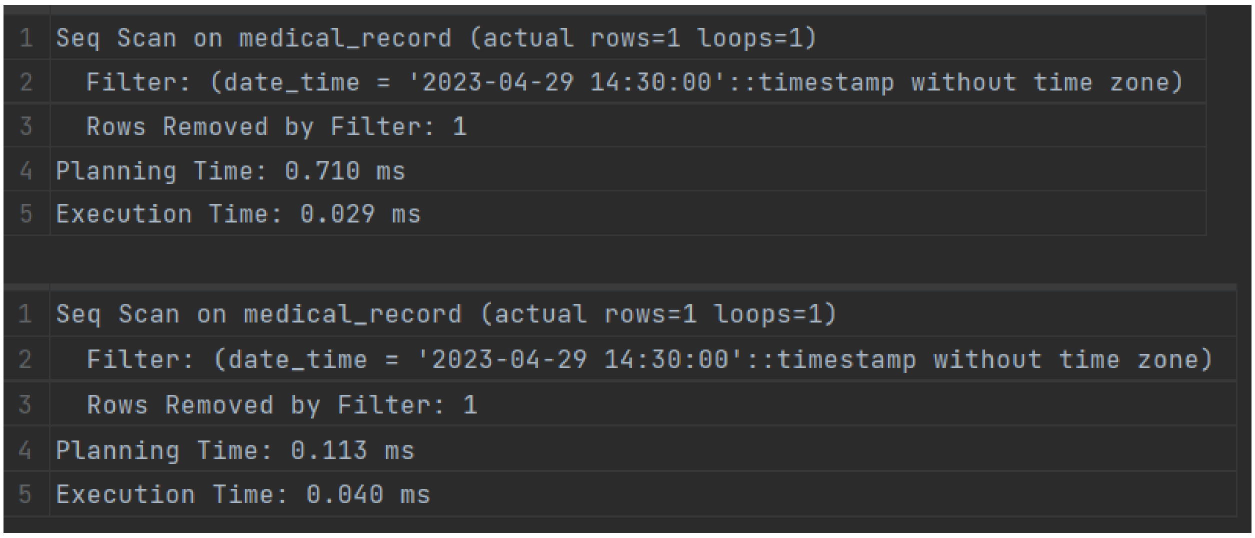
4. Vytvoření a použití indexu

Index pro attribute date_time z tabulky MedicalRecord. Často budeme hledat nějaké zapis mediční na základě data vytvoření zapisu, 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 medical_record_creation_date_time
ON medical_record(date_time);
-- 2. Vyhledávání podle určitého data
EXPLAIN (analyze, costs off, timing off)
SELECT * FROM medical_record
WHERE date_time = '2023-04-29 14:30:00';
```

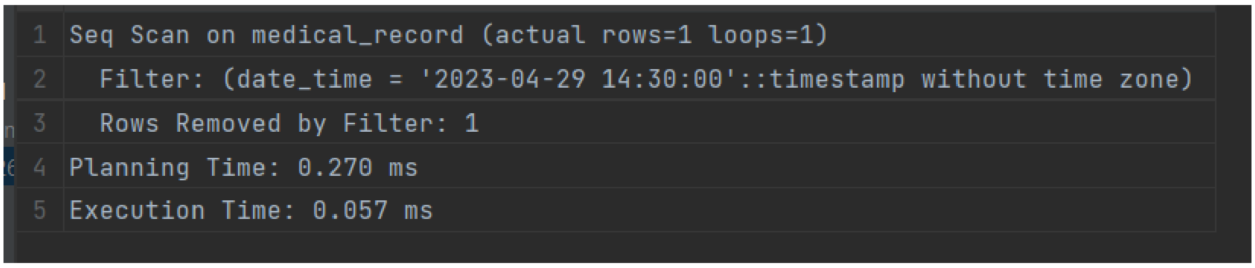
Z indexem



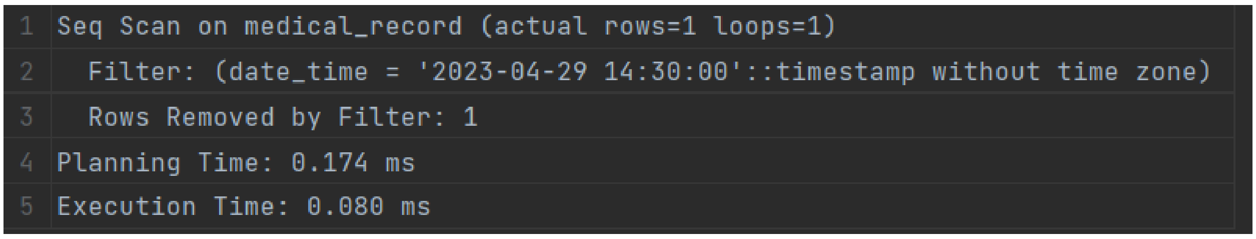
```
Seq Scan on medical_record (actual rows=1 loops=1)
  Filter: (date_time = '2023-04-29 14:30:00'::timestamp without time zone)
  Rows Removed by Filter: 1
Planning Time: 0.710 ms
Execution Time: 0.029 ms
```

```
Seq Scan on medical_record (actual rows=1 loops=1)
  Filter: (date_time = '2023-04-29 14:30:00'::timestamp without time zone)
  Rows Removed by Filter: 1
Planning Time: 0.113 ms
Execution Time: 0.040 ms
```

Bez indexu



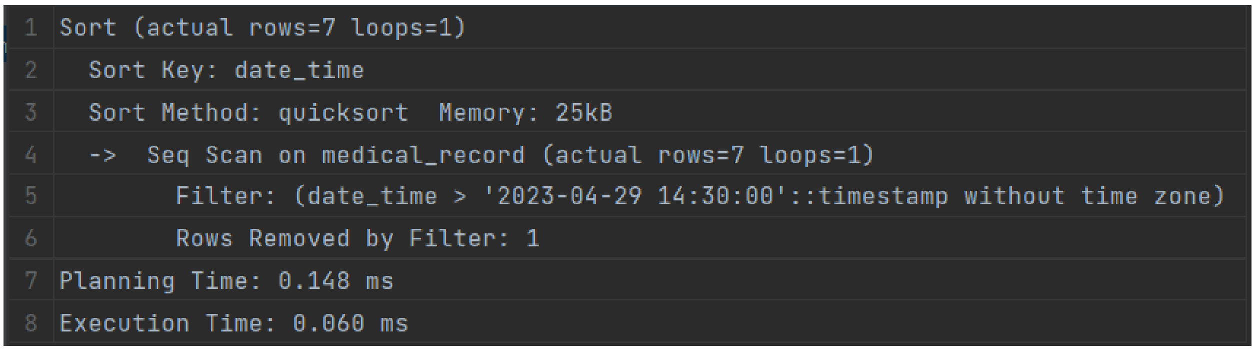
```
Seq Scan on medical_record (actual rows=1 loops=1)
  Filter: (date_time = '2023-04-29 14:30:00'::timestamp without time zone)
  Rows Removed by Filter: 1
Planning Time: 0.270 ms
Execution Time: 0.057 ms
```



```
Seq Scan on medical_record (actual rows=1 loops=1)
  Filter: (date_time = '2023-04-29 14:30:00'::timestamp without time zone)
  Rows Removed by Filter: 1
Planning Time: 0.174 ms
Execution Time: 0.080 ms
```

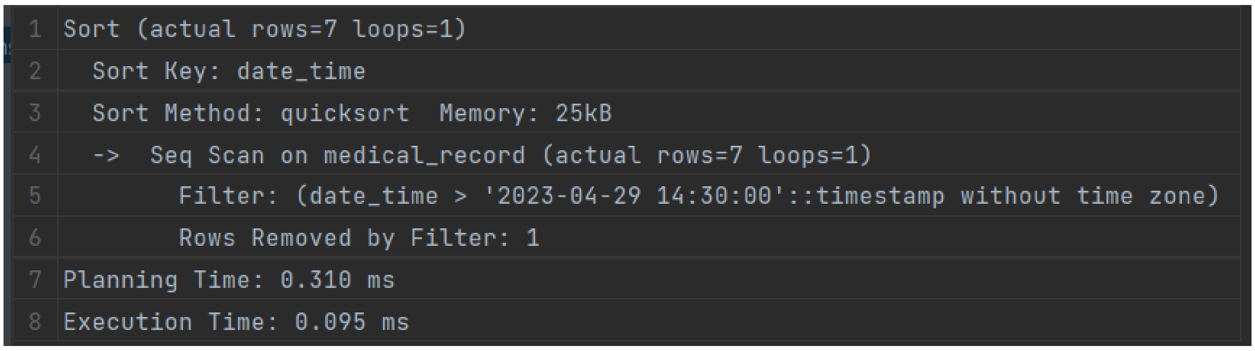
```
-- 3. Vyhledávání od nějakého data
EXPLAIN (analyze, costs off, timing off)
SELECT * FROM medical_record
WHERE date_time > '2023-04-29 14:30:00'
ORDER BY date_time;
```

Z indexem

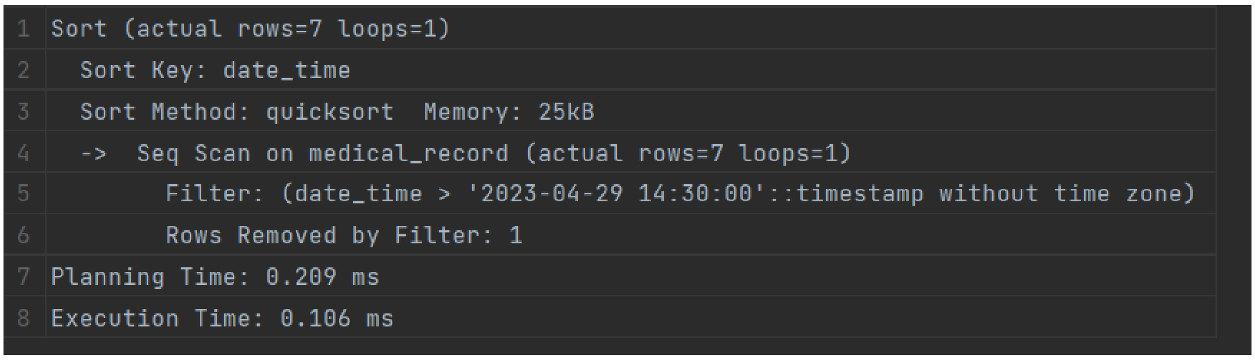


```
Sort (actual rows=7 loops=1)
  Sort Key: date_time
  Sort Method: quicksort  Memory: 25kB
  ->  Seq Scan on medical_record (actual rows=7 loops=1)
        Filter: (date_time > '2023-04-29 14:30:00'::timestamp without time zone)
        Rows Removed by Filter: 1
Planning Time: 0.148 ms
Execution Time: 0.060 ms
```

bez indexu



```
Sort (actual rows=7 loops=1)
  Sort Key: date_time
  Sort Method: quicksort  Memory: 25kB
  ->  Seq Scan on medical_record (actual rows=7 loops=1)
        Filter: (date_time > '2023-04-29 14:30:00'::timestamp without time zone)
        Rows Removed by Filter: 1
Planning Time: 0.310 ms
Execution Time: 0.095 ms
```



```
Sort (actual rows=7 loops=1)
  Sort Key: date_time
  Sort Method: quicksort  Memory: 25kB
  ->  Seq Scan on medical_record (actual rows=7 loops=1)
        Filter: (date_time > '2023-04-29 14:30:00'::timestamp without time zone)
        Rows Removed by Filter: 1
Planning Time: 0.209 ms
Execution Time: 0.106 ms
```

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

5. Vytvoření funkce

1. Add new hospital&rooms

```
CREATE OR REPLACE FUNCTION add_new_hospital_rooms(
P_hospital_id hospital.hospital_id %TYPE,
P_name hospital.name %TYPE,
P_web_site hospital.web_site %TYPE,
P_adress_id hospital.adress_id %TYPE,
P_room_id operation_room.room_id %TYPE,
P_room_number operation_room.room_number %TYPE
)
RETURNS void
LANGUAGE plpgsql
AS
$$
DECLARE hospitalid INT;
BEGIN
INSERT INTO hospital(hospital_id, name, web_site, adress_id)
```

```
VALUES (P_hospital_id, P_name, P_web_site, P_adress_id)
RETURNING hospital_id INTO hospitalid;
INSERT INTO operation_room(hospital_id, room_id, room_number)
VALUES (hospitalid, P_room_id, P_room_number);
END;
$$;
```

2. Add new address with hospital

```
CREATE OR REPLACE FUNCTION add_new_hospital_rooms(
P_hospital_id hospital.hospital_id %TYPE,
P_name hospital.name %TYPE,
P_web_site hospital.web_site %TYPE,
P_adress_id venue.adress_id %TYPE,
P_city venue.city %TYPE,
P_street venue.street %TYPE,
P_postcode venue.postcode %TYPE
)
RETURNS void
LANGUAGE plpgsql
AS
$$
DECLARE adressid INT;
BEGIN
INSERT INTO venue(adress_id, city, street, postcode)
VALUES (P_adress_id,P_city, P_street, P_postcode)
RETURNING adress_id INTO adressid;
INSERT INTO hospital(hospital_id, name, web_site, adress_id)
VALUES (P_hospital_id, P_name, P_web_site, adressid);
END;
$$;
```