

II2250 Manajemen Basis Data
Semester II 2022/2023

Praktikum V: Security & Transactions

Dipersiapkan oleh:
Asisten Laboratorium Basis Data

Pelaksanaan:
Waktu Mulai: Senin, 27 Maret 2023 – 16.00 WIB
Waktu Selesai: Senin, 27 Maret 2023 – 17.30 WIB

I. Ketentuan

Peserta kuliah akan mengerjakan praktikum mengenai **Security and Transactions** pada DBMS PostgreSQL menggunakan data hospital.sql yang tersedia di folder praktikum ini. Praktikum ini dikerjakan secara **INDIVIDU**.

Peserta dapat memanfaatkan file yang terdapat pada folder [Untuk Peserta](#) dan dokumentasi [PostgreSQL](#) dalam pengerjaan soal praktikum.

Deliverables yang harus dikumpulkan untuk Praktikum ini adalah sebagai berikut.

1. File .docx berisikan daftar query beserta screenshot eksekusi tiap query dengan format nama: **K<Kelas>_P05_<NIM>.docx**
Contoh: K01_P05_18221000.docx

Isi dari dokumen adalah sebagai berikut.

- a. Untuk setiap soal.
 - i. Perintah atau langkah yang dilakukan untuk menyelesaikan persoalan. Sediakan semua kode dalam bentuk **teks**.
 - ii. Query pengecekan dalam bentuk **teks** (jika diminta pada soal).
 - iii. Hasil perintah dari *query* tersebut

Deliverables dikumpulkan pada pranala berikut.

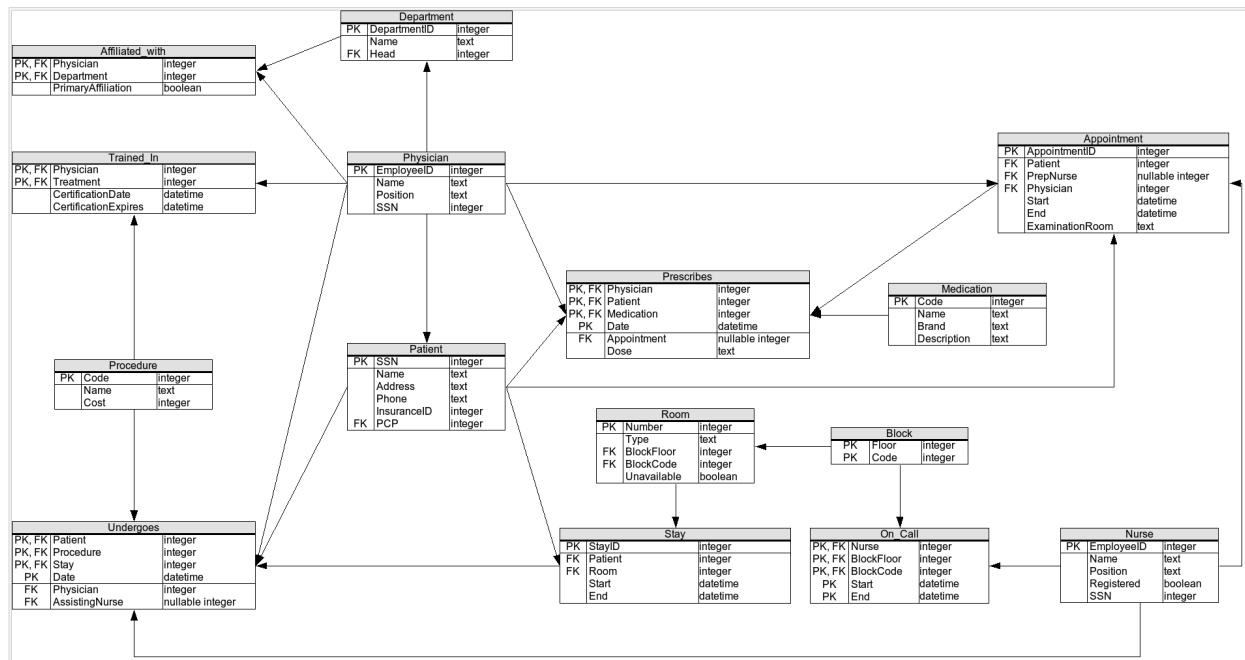
<https://s.id/uploader-prak5>

Keterlambatan pengumpulan *deliverables* akan mengakibatkan pengurangan nilai. Segala tindak kecurangan akan ditindaklanjuti dan diikuti konsekuensi serius.

~ Selamat mengerjakan! :)) ~

II. Skema Basis Data

Berikut ini adalah skema dari basis data Rumah Sakit Basdat dari file hospital.sql.



Affiliated_with = (Physician, Department, PrimaryAffiliation)

Appointment = (AppointmentID, Patient, PrepNurse, Physician, StartDate, EndDate, ExaminationRoom)

Block = (BlockFloor, BlockCode)

Department = (DepartmentID, Name, Head)

Medication = (Code, Name, Brand, Description)

Nurse = (EmployeeID, Name, Position, Registered, SSN)

On_call = (Nurse, BlockFloor, BlockCode, OnCallStart, OnCallEnd)

Patient = (SSN, Name, Address, Phone, InsuranceID, PCP)

Physician = (EmployeeID, Name, Position, SSN)

Prescribes = (Physician, Patient, Medication, Date, Appointment, Dose)

Procedures = (Code, Name, Cost)

Room = (RoomNumber, RoomType, BlockFloor, BlockCode, Unavailable)

Stay = (StayID, Patient, Room, StayStart, StayEnd)

Trained_in = (Physician, Treatment, CertificationDate, CertificationExpires)

Undergoes = (Patient, Procedures, Stay, DateUndergoes, Physician, AssistingNurse)

III. Soal

(**Note:** Pastikan bahwa pada komputer yang digunakan, telah terdapat sebuah database bernama hospital. Jika belum, buatlah sebuah database bernama **hospital** dan import hospital.sql ke dalam database tersebut!)

1. Buatlah sebuah database bernama **rumahsakit** yang merupakan duplikat dari database **hospital**.
2. Di rumah sakit basdat, terdapat beberapa *receptionist* yang berperan dalam mengelola *appointment* antara pasien dengan dokter dan perawat.
 - A. Buatlah sebuah *role* dengan nama **receptionist**. Role ini memiliki akses untuk melihat, menambahkan, mengubah, dan menghapus data terhadap tabel Appointment.
 - B. Buatlah sebuah *user* dengan nama **receptionist1** dengan password **recept123** lalu berikan akses role **receptionist** kepada user tersebut.
 - C. Login pada database sebagai user **receptionist1** dan coba lakukan **penambahan** data pada tabel Appointment dengan rincian sebagai berikut.

AppointmentID = 1000 Patient = 11 PrepNurse = 9 Physician = 2 StartDate = '2023-03-27 09:00:00' EndDate = '2023-04-27 18:00:00' ExaminationRoom = 'J'

Lalu coba lakukan **pengubahan** data ruangan tersebut menjadi 'K'. Setelah itu, coba untuk **menghapus** data yang baru saja Anda tambahkan.

HINT command login: `psql -U <user> <database>`

Jawaban:

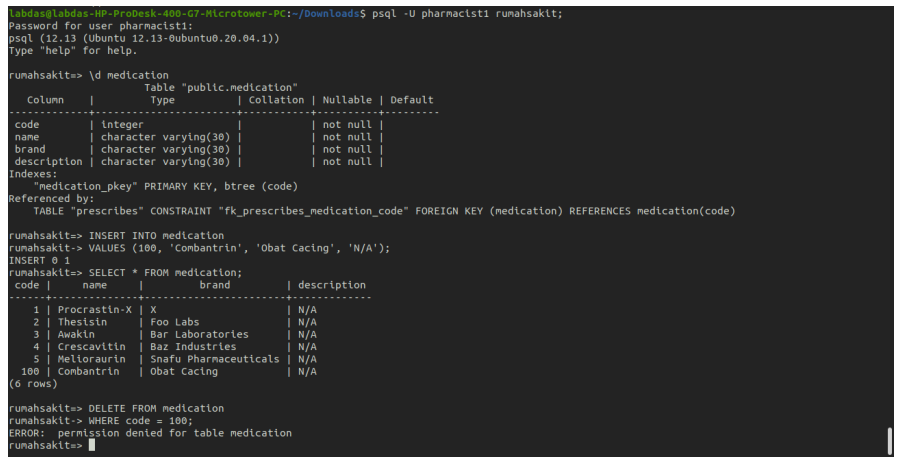
Query Pembuatan Role dan	<pre>CREATE ROLE receptionist; GRANT SELECT, INSERT, UPDATE, DELETE ON appointment TO receptionist;</pre>
---------------------------------	---

Pemberian Akses Role (A)	
Query Pembuatan User dan Pemberian Akses Role (B)	<pre>CREATE USER receptionist1 LOGIN PASSWORD 'recept123' IN ROLE receptionist;</pre>
Query pada Tabel Appointment (C)	<pre>INSERT INTO appointment VALUES(1001, 11, 9, 2, '2023-03-27 09:00:00', '2023-04-27 18:00:00', 'J'); UPDATE appointment SET examinationroom = 'K' WHERE appointmentid = 1001; DELETE FROM appointment WHERE appointmentid = 1001;</pre>
SS Hasil Query pada Tabel Appointment (C)	<pre>labdas@labdas-HP-ProDesk-400-G7-Microtower-PC:~\$ psql -U receptionist1 rumahsakit Password for user receptionist1: psql (12.13 (Ubuntu 12.13-0ubuntu0.20.04.1)) Type "help" for help. rumahsakit=> rumahsakit=> INSERT INTO appointment rumahsakit-> VALUES(1001, 11, 9, 2, '2023-03-27 09:00:00', '2023-04-27 18:00:00', 'J'); INSERT 0 1 rumahsakit=> SELECT * FROM appointment WHERE appointmentid = 1001; appointmentid patient prepurse physician startdate enddate examinationroom -----+-----+-----+-----+-----+-----+----- 1001 11 9 2 2023-03-27 09:00:00 2023-04-27 18:00:00 J (1 row) rumahsakit=> rumahsakit=> UPDATE appointment SET examinationroom = 'K' WHERE appointmentid = 1001; UPDATE 1 rumahsakit=> SELECT * FROM appointment WHERE appointmentid = 1001; appointmentid patient prepurse physician startdate enddate examinationroom -----+-----+-----+-----+-----+-----+----- 1001 11 9 2 2023-03-27 09:00:00 2023-04-27 18:00:00 K (1 row) rumahsakit=> rumahsakit=> DELETE FROM appointment WHERE appointmentid = 1001; DELETE 1 rumahsakit=> SELECT * FROM appointment WHERE appointmentid = 1001; appointmentid patient prepurse physician startdate enddate examinationroom -----+-----+-----+-----+-----+-----+----- (0 rows) rumahsakit=></pre>

- Salah satu pihak yang berperan agar pasien dapat menerima obat sesuai dengan anjuran dokter adalah seorang *pharmacist*.

- A. Buatlah sebuah *role* dengan nama **pharmacist1** yang memiliki password **password123**. Role ini memiliki akses untuk melihat, menambahkan, dan mengubah data pada terhadap tabel Medication. Selain itu, role ini juga memiliki akses untuk melihat data pada tabel Prescribes.
- B. Login pada database sebagai user **pharmacist1** dan coba untuk lakukan penambahan data pada tabel Medication. Setelah itu, coba untuk menghapus data yang baru saja Anda tambahkan.
- C. Tampilkan data pada tabel Prescribes sebagai user **pharmacist1**. Setelah itu, coba untuk lakukan penambahan *entry* data baru pada tabel tersebut.

Jawaban:

Query Pembuatan Role (A)	<pre>CREATE ROLE pharmacist1 LOGIN PASSWORD 'password123'; GRANT SELECT, INSERT, UPDATE ON medication TO pharmacist1; GRANT SELECT ON prescribes TO pharmacist1;</pre>
Query pada Tabel Medication (B)	<pre>INSERT INTO medication VALUES (100, 'Combantrin', 'Obat Cacing', 'N/A'); DELETE FROM medication WHERE code = 100;</pre>
SS Query pada Tabel Medication (B)	 <pre>Labdas@labdas-HP-ProDesk-400-G7-Microtower-PC:~/Downloads\$ psql -U pharmacist1 runahsakit; Password for user pharmacist1: psql (12.13 (Ubuntu 12.13-0ubuntu0.20.04.1)) Type 'help' for help. runahsakit=> \d medication Table "public.medication" Column Type Collation Nullable Default ----- ----- ----- ----- ----- code integer not null name character varying(30) not null brand character varying(30) not null description character varying(30) not null Indexes: "medication_pkey" PRIMARY KEY, btree (code) Referenced by: TABLE "prescribes" CONSTRAINT "fk_prescribes_medication_code" FOREIGN KEY (medication) REFERENCES medication(code) runahsakit=> INSERT INTO medication runahsakit-> VALUES (100, 'Combantrin', 'Obat Cacing', 'N/A'); INSERT 0 1 runahsakit=> SELECT * FROM medication; code name brand description ----- ----- ----- ----- 1 Procrastin-X X N/A 2 Thesisin Foo Labs N/A 3 Awakin Bar Laboratories N/A 4 Crescavitin Baz Industries N/A 5 Melitoraurin Snafu Pharmaceuticals N/A 100 Combantrin Obat Cacing N/A (6 rows) runahsakit=> DELETE FROM medication runahsakit-> WHERE code = 100; ERROR: permission denied for table medication runahsakit=></pre>
Query pada Tabel Prescribes (C)	<pre>SELECT * FROM prescribes LIMIT 10; INSERT INTO prescribes VALUES (1, 1, 1, '2018-01-01', 2, 10);</pre>

**SS Query
pada Tabel
Prescribes (C)**

labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads

18	208	3	2013-02-25 00:00:00	2
16	509	3	2008-08-12 00:00:00	13
71	362	4	2013-07-06 00:00:00	15
82	100	4	2013-02-09 00:00:00	12
93	598	1	2018-04-28 00:00:00	3
33	556	5	2009-02-26 00:00:00	15
6	115	4	2016-12-10 00:00:00	7
83	196	3	2018-05-24 00:00:00	2
33	57	5	2012-01-27 00:00:00	3
97	199	4	2017-09-05 00:00:00	9
64	433	1	2009-05-03 00:00:00	12
23	349	5	2010-07-20 00:00:00	6
44	158	4	2015-06-26 00:00:00	13
71	408	5	2009-11-16 00:00:00	9
22	259	5	2015-05-20 00:00:00	7
36	472	3	2017-07-16 00:00:00	1
26	517	2	2018-09-24 00:00:00	14
41	488	1	2008-06-20 00:00:00	4
70	570	2	2011-06-26 00:00:00	13
77	523	5	2016-07-07 00:00:00	12
15	417	2	2013-12-27 00:00:00	10
92	446	4	2018-06-11 00:00:00	15
13	54	5	2015-05-23 00:00:00	9
3	347	1	2009-02-24 00:00:00	4
42	272	2	2010-04-23 00:00:00	15
6	126	2	2010-05-13 00:00:00	2
64	567	2	2013-03-24 00:00:00	13
30	327	2	2009-04-22 00:00:00	4
9	370	1	2013-04-11 00:00:00	2
45	239	5	2012-05-27 00:00:00	5
78	479	4	2018-12-06 00:00:00	8
95	230	5	2011-08-05 00:00:00	3
56	419	3	2018-05-01 00:00:00	3
63	272	5	2018-04-09 00:00:00	6
58	333	5	2018-07-24 00:00:00	10

(399906 rows)

rumahsakit=>

rumahsakit=> SELECT * FROM prescribes LIMIT 10;

physician	patient	medication	date	appointment	dose
26	487	2	2008-03-02 00:00:00	170	10
1	49	1	2016-04-08 00:00:00	605	4
93	181	3	2014-07-16 00:00:00	258	9
58	210	3	2012-10-16 00:00:00	542	10
20	456	2	2014-05-04 00:00:00	257	8
6	587	4	2010-07-16 00:00:00	893	7
52	93	1	2013-12-15 00:00:00	495	9
95	187	2	2010-03-16 00:00:00	807	8
70	184	4	2017-09-11 00:00:00	123	13
5	572	3	2008-10-24 00:00:00	817	2

(10 rows)

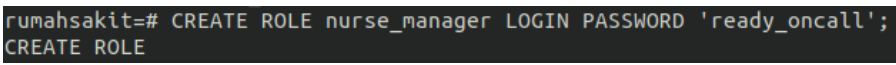
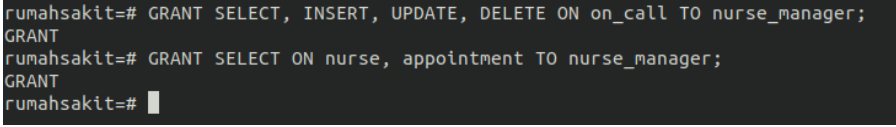
rumahsakit=>

rumahsakit=> INSERT INTO prescribes
rumahsakit-> VALUES (1, 1, 1, '2018-01-01', 2, 10);
ERROR: permission denied for table prescribes
rumahsakit=>

4. Di rumah sakit ini juga terdapat seorang **Nurse Manager** yang bertugas untuk memastikan bahwa terdapat **Nurse** yang selalu siaga setiap waktu untuk kebutuhan pasien. Ia adalah yang bertanggungjawab **mengatur keseluruhan** jadwal dan penugasan **shift on-call** Nurse. Sebelum membuat jadwal, seorang **Nurse Manager** juga perlu mengetahui **profil Nurse** tersebut dan **jadwal janji temu Nurse** dengan **Patient**.

- Berdasarkan deskripsi tersebut, akses privilege apa saja yang sesuai dengan deskripsi? Sebutkan nama tabel, tipe privilege, dan alasannya.
- Buatlah sebuah role bernama **nurse_manager** dengan password **ready_oncall**, dan berikan akses privilege kepada role tersebut berdasarkan jawaban Anda pada bagian A.

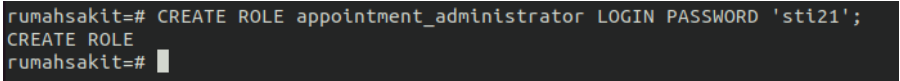
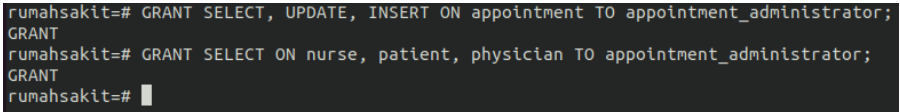
Jawaban:

<p>A.</p>	<p>Nurse Manager memiliki akses privilege:</p> <ol style="list-style-type: none"> 1. SELECT, INSERT, UPDATE, DELETE pada tabel on_call karena nurse manager memiliki hak untuk mengatur keseluruhan jadwal dan penugasan shift on-call nurse. SELECT digunakan untuk melihat jadwal yang sudah ada, INSERT digunakan untuk memasukkan jadwal baru, UPDATE digunakan jika Nurse Manager salah memasukkan data dan ingin melakukan updating, dan DELETE digunakan jika Nurse Manager ingin menghapus data masukan yang invalid. Nurse Manager tidak boleh memiliki akses TRUNCATE karena pengaturan keseluruhan jadwal bukan berarti boleh menghapus seluruh data pada tabel tersebut. 2. SELECT pada tabel nurse karena Nurse Manager perlu mengetahui profil nurse. 3. SELECT pada tabel appointment karena Nurse Manager perlu mengetahui jadwal janji temu nurse dengan patient.
<p>B. Query Pembuatan Role & Screenshot</p>	<pre>CREATE ROLE nurse_manager LOGIN PASSWORD 'ready_oncall';</pre> 
<p>Query Pemberian Akses Privilege & Screenshot</p>	<pre>GRANT SELECT, INSERT, UPDATE, DELETE ON on_call TO nurse_manager; GRANT SELECT ON nurse, appointment TO nurse_manager;</pre> 

5. Pihak Rumah Sakit akan menunjuk seorang **appointment_administrator** untuk mengolah data appointment yang ada. Untuk itu seorang **appointment_administrator** harus bisa melihat data appointment yang sudah ada, memperbaikinya, dan juga menambahkan appointment yang akan datang. **Appointment_administrator** juga harus memastikan semua relasi yang berhubungan dengan data appointment harus memiliki data yang valid dengan cara melihat data pada relasi tersebut.

- A. Berdasarkan deskripsi tersebut, akses privilege apa saja yang sesuai dengan deskripsi? Sebutkan nama tabel, tipe privilege, dan alasannya.
- B. Buatlah sebuah role bernama **appointment_administrator** dengan password sti21, dan berikan akses privilege kepada role tersebut berdasarkan jawaban Anda pada bagian A.

Jawaban:

A	<p>Akses privilege yang dimiliki appointment_administrator:</p> <ol style="list-style-type: none"> 1. SELECT, UPDATE, INSERT pada tabel appointment karena seorang appointment_administrator harus bisa melihat data appointment yang sudah ada, memperbaikinya, dan juga menambahkan appointment yang akan datang. 2. SELECT pada tabel nurse, patient, physician karena seorang appointment_administrator juga harus memastikan semua relasi yang berhubungan dengan data appointment harus memiliki data yang valid dengan cara melihat data pada relasi tersebut. Relasi yang berhubungan dengan appointment dapat ditentukan dengan melihat tabel apa saja yang diacu oleh foreign key pada tabel appointment.
B. Query Pembuatan Role & Screenshot	<pre>CREATE ROLE appointment_administrator LOGIN PASSWORD 'sti21';</pre> 
Query Pemberian Akses Privilege & Screenshot	<pre>GRANT SELECT, UPDATE, INSERT ON appointment TO appointment_administrator; GRANT SELECT ON nurse, patient, physician TO appointment_administrator;</pre> 

6. Rumah Sakit Basdat memiliki sistem booking ruangan. Pada suatu saat terdapat 2 user (user A dan user B) yang membooking ruangan nomor 27 secara bersamaan.

- a. Tentukan transaction isolation level yang cocok untuk menangani konsistensi data untuk kasus di atas, sertakan penjelasan isolation level yang dipilih.
- b. Lakukan simulasi SELECT dan UPDATE status unavailable data room nomor 27 menjadi true secara bersamaan.
Gunakan 2 terminal untuk melakukan simulasi. (anggap masing-masing terminal sebagai 1 user yang berbeda.)
- c. Dari simulasi yang telah dilakukan. Jelaskan apa yang terjadi dalam simulasi tersebut.

Isolation Level yang dipilih	Isolation level yang sesuai adalah SERIALIZABLE ISOLATION LEVEL.
Penjelasan Isolation Level yang dipilih	<p>Alasan memilih SERIALIZABLE adalah untuk menjaga konsistensi data saat terdapat 2 user yang melakukan update pada data yang sama pada waktu yang bersamaan.</p> <p>Jika dibandingkan dengan isolation level lainnya:</p> <ol style="list-style-type: none"> 1. READ COMMITTED: ketika salah satu user melakukan transaksi, user lain akan mengalami deadlock jika melakukan transaksi pada data yang sama. Namun, ketika user tersebut melakukan commit, user yang lain juga dapat melakukan update pada data tersebut sehingga updating data dilakukan berdasarkan commit terakhir yang artinya data tidak konsisten (tidak sesuai dengan sistem yang diinginkan). 2. REPEATABLE READ: proses deadlock sama seperti pada READ COMMITTED, namun ketika salah satu user melakukan commit, user yang lain tidak dapat melakukan update pada data yang sama (akan muncul error), namun ketika user melakukan SELECT, data belum terupdate. 3. SERIALIZABLE: proses hampir sama dengan REPEATABLE READ. Yang membedakan hanyalah pada proses SELECT, data sudah terupdate jika user lain telah melakukan commit. <p>Dari perbandingan tersebut, disimpulkan SERIALIZABLE ISOLATION LEVEL adalah yang terbaik.</p>

SS Simulasi

Simulasi SELECT pada terminal 1:

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
runahsakit=# \pst pager 0
Pager usage is off.
runahsakit=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
runahsakit=# SELECT * FROM room;
roomnumber | roomtype | blockfloor | blockcode | unavailable
-----
1 | Single | 1 | 1 | f
2 | Single | 1 | 1 | f
3 | Single | 1 | 1 | f
4 | Single | 1 | 2 | f
5 | Single | 1 | 2 | t
6 | Single | 1 | 2 | f
7 | Single | 1 | 3 | f
8 | Single | 1 | 3 | f
9 | Single | 1 | 3 | f
10 | Single | 2 | 1 | t
11 | Single | 2 | 1 | f
12 | Single | 2 | 1 | f
13 | Single | 2 | 2 | f
14 | Single | 2 | 2 | f
15 | Single | 2 | 2 | t
16 | Single | 2 | 3 | f
17 | Single | 2 | 3 | f
18 | Single | 2 | 3 | f
19 | Single | 3 | 1 | f
20 | Single | 3 | 1 | t
21 | Single | 3 | 1 | f
22 | Single | 3 | 2 | f
23 | Single | 3 | 2 | f
24 | Single | 3 | 2 | f
25 | Single | 3 | 3 | t
26 | Single | 3 | 3 | f
28 | Single | 4 | 1 | f
29 | Single | 4 | 1 | t
30 | Single | 4 | 1 | f
31 | Single | 4 | 2 | f
32 | Single | 4 | 2 | f
```

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~/Downloads
2 | Single | 1 | 1 | f
3 | Single | 1 | 1 | f
4 | Single | 1 | 2 | f
5 | Single | 1 | 2 | t
6 | Single | 1 | 2 | f
7 | Single | 1 | 3 | f
8 | Single | 1 | 3 | f
9 | Single | 1 | 3 | f
10 | Single | 2 | 1 | t
11 | Single | 2 | 1 | f
12 | Single | 2 | 1 | f
13 | Single | 2 | 2 | f
14 | Single | 2 | 2 | f
15 | Single | 2 | 2 | t
16 | Single | 2 | 3 | f
17 | Single | 2 | 3 | f
18 | Single | 2 | 3 | f
19 | Single | 3 | 1 | f
20 | Single | 3 | 1 | t
21 | Single | 3 | 1 | f
22 | Single | 3 | 2 | f
23 | Single | 3 | 2 | f
24 | Single | 3 | 2 | f
25 | Single | 3 | 3 | t
26 | Single | 3 | 3 | f
28 | Single | 4 | 1 | f
29 | Single | 4 | 1 | t
30 | Single | 4 | 1 | f
31 | Single | 4 | 2 | f
32 | Single | 4 | 2 | f
33 | Single | 4 | 2 | f
34 | Single | 4 | 3 | t
35 | Single | 4 | 3 | f
36 | Single | 4 | 3 | f
(36 rows)
runahsakit=#
```

Simulasi SELECT pada terminal 2:

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~
runahsakit=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
runahsakit=# SELECT * FROM room;
runahsakit=#
```

roomnumber	roomtype	blockfloor	blockcode	unavailable
1	Single	1	1	f
2	Single	1	1	f
3	Single	1	1	f
4	Single	1	2	f
5	Single	1	2	t
6	Single	1	2	f
7	Single	1	3	f
8	Single	1	3	f
9	Single	1	3	f
10	Single	2	1	t
11	Single	2	1	f
12	Single	2	1	f
13	Single	2	2	f
14	Single	2	2	f
15	Single	2	2	t
16	Single	2	3	f
17	Single	2	3	f
18	Single	2	3	f
19	Single	3	1	f
20	Single	3	1	t
21	Single	3	1	f
22	Single	3	2	f
23	Single	3	2	f
24	Single	3	2	f
25	Single	3	3	t
26	Single	3	3	f
28	Single	4	1	f
29	Single	4	1	t
30	Single	4	1	f
31	Single	4	2	f
32	Single	4	2	f
33	Single	4	2	f
34	Single	4	3	t
35	Single	4	3	f

```
labdas@labdas-HP-ProDesk-400-G7-Microtower-PC: ~
runahsakit=#
```

2	Single	1	1	f
3	Single	1	1	f
4	Single	1	2	f
5	Single	1	2	t
6	Single	1	2	f
7	Single	1	3	f
8	Single	1	3	f
9	Single	1	3	f
10	Single	2	1	t
11	Single	2	1	f
12	Single	2	1	f
13	Single	2	2	f
14	Single	2	2	f
15	Single	2	2	t
16	Single	2	3	f
17	Single	2	3	f
18	Single	2	3	f
19	Single	3	1	f
20	Single	3	1	t
21	Single	3	1	f
22	Single	3	2	f
23	Single	3	2	f
24	Single	3	2	f
25	Single	3	3	t
26	Single	3	3	f
28	Single	4	1	f
29	Single	4	1	t
30	Single	4	1	f
31	Single	4	2	f
32	Single	4	2	f
33	Single	4	2	f
34	Single	4	3	t
35	Single	4	3	f
36	Single	4	3	f
27	Single	3	3	f

(36 rows)

```
runahsakit=#
```

Simulasi UPDATE pada terminal 1:

```
runahsakit=# UPDATE room SET unavailable = true WHERE roomnumber = 27;
UPDATE 1
runahsakit=#
```

Simulasi UPDATE pada terminal 2:

```
runahsakit=# UPDATE room SET unavailable = true WHERE roomnumber = 27;
runahsakit=#
```

Simulasi COMMIT pada terminal 1:

```
runahsakit=# UPDATE room SET unavailable = true WHERE roomnumber = 27;
UPDATE 1
runahsakit=# COMMIT;
COMMIT
runahsakit=#
```

	<p>Yang terjadi pada terminal 2:</p> <pre>rumahsakit=# UPDATE room SET unavailable = true WHERE roomnumber = 27; ERROR: could not serialize access due to concurrent update rumahsakit=#</pre>
Penjelasan Simulasi	<p>Pada simulasi tersebut, terlihat bahwa user A (terminal 1) dan user B (terminal 2) memiliki hasil SELECT yang sama. Ketika user A melakukan update, user B akan mengalami deadlock jika juga melakukan update pada data yang sama. Namun, ketika user A melakukan commit, akan muncul error pada terminal user B. Hal ini menunjukkan bahwa data yang terupdate adalah data dari user yang terlebih dahulu melakukan update dan commit.</p>

Referensi

[PostgreSQL: Documentation: 15: CREATE ROLE](#)

[PostgreSQL: Documentation: 15: CREATE USER](#)

[PostgreSQL: Documentation: 15: GRANT](#)

[PostgreSQL: Documentation: 15: INSERT](#)

[PostgreSQL: Documentation: 15: UPDATE](#)

[PostgreSQL: Documentation: 15: DELETE](#)

[PostgreSQL: Documentation: 15: ISOLATION LEVEL](#)

Feedback Praktikum

bit.ly/feedback-praktikum