

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG



Praktikum 1 Susulan

Integrity Constraints

Selasa, 8 Oktober 2024

Oleh:

13522053 - Erdianti Wiga Putri Andini

13522089 - Abdul Rafi Radityo H

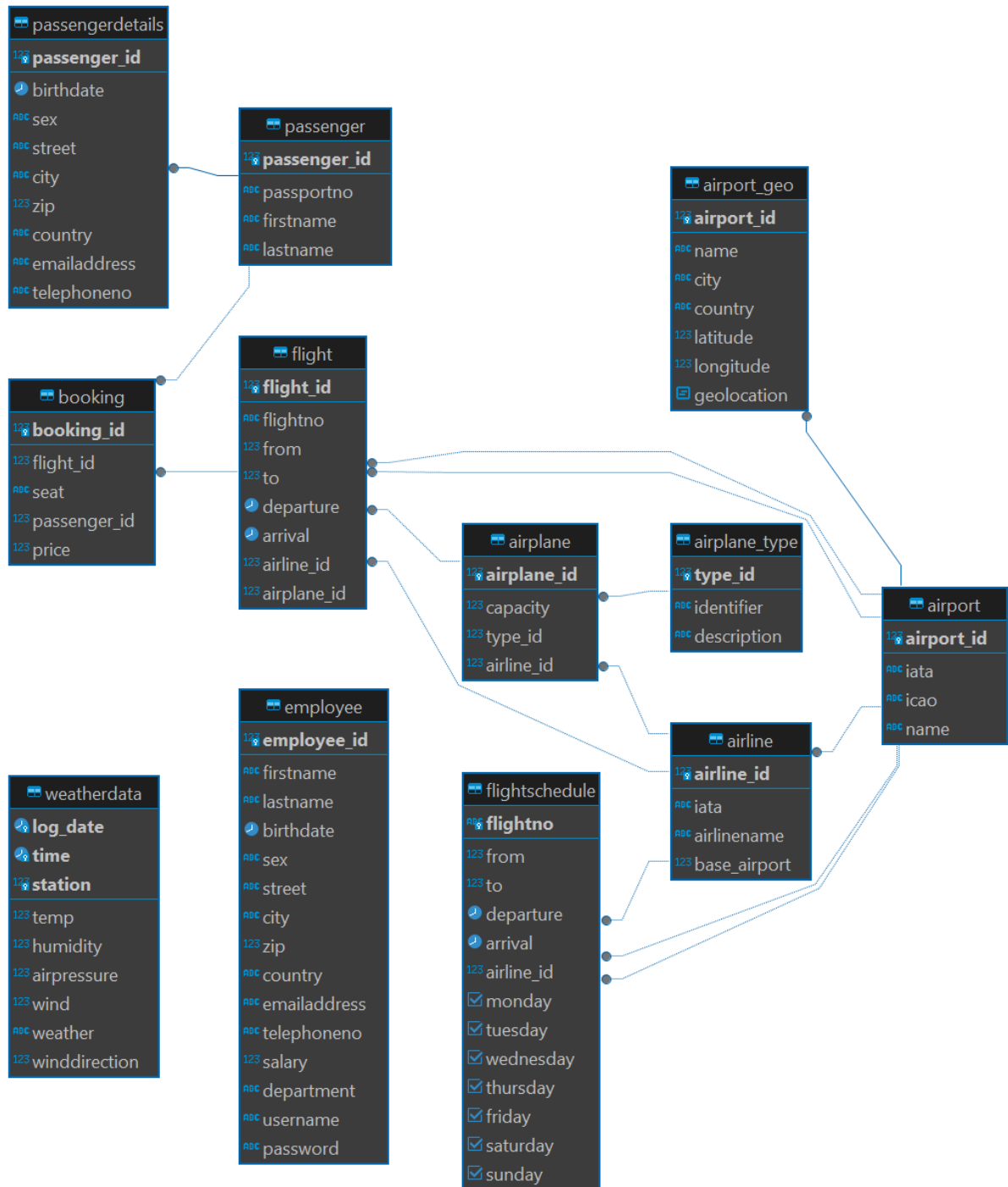
IF3140 - Sistem Basis Data

2024

Petunjuk

1. Kerjakan setiap soal praktikum ini dengan baik
2. Diperkenankan untuk menggunakan dokumentasi PostgreSQL resmi
3. Kerjakan secara berkelompok
4. **Dilarang Menyontek**

AirportDB



1.Reviu Integrity Constraint

Di bawah ini adalah komponen-komponen yang umum untuk digunakan dalam DBMS, yaitu:

1. *Not Null*,
2. *Enum*,
3. *Trigger*,
4. *Check*, dan
5. *Unique*

Untuk setiap komponen tersebut, pada skema *integrity constraint* apa masing-masing komponen dapat digunakan untuk menjaga integritas data? Berikan penjelasan singkat atas jawaban yang kamu berikan! Untuk setiap komponen, jawaban dapat berjumlah lebih dari satu *integrity constraint*.

Jawaban Soal
<ol style="list-style-type: none">1. <i>Not null</i> menjaga integritas relasional. Memastikan bahwa kolom tidak boleh memiliki nilai NULL.2. <i>Enum</i> menjaga integritas atribut. Memperbolehkan user untuk membuat tipe data baru untuk atribut tertentu agar memperkecil kemungkinan adanya data yang tidak sesuai.3. <i>Trigger</i> menjaga integritas database dan relasional.4. <i>Check</i> menjaga integritas atribut dan relasional. Untuk menetapkan aturan yang harus dipenuhi oleh nilai-nilai dalam suatu kolom.5. <i>Unique</i> menjaga integritas relasional. Untuk memastikan bahwa semua nilai di kolom tertentu unik, namun tetap memungkinkan NULL.

2.Karyawan Baru

Bu Didot, seorang ahli database diminta untuk membuat procedure untuk menambahkan employee baru, namun dengan syarat sebagai berikut:

- Username employee baru belum pernah digunakan sebelumnya
- Usia employee baru lebih dari 25 tahun
- Gaji employee baru lebih dari 1000

Tampilkan pula pesan kesalahan yang sesuai apabila salah satu constraint di atas dilanggar.

Fungsi procedure dibuat dengan nama insert_new_employee dengan parameter:

- new_username
- new_firstname
- new_lastname
- new_birthdate
- new_street
- new_city
- new_zip
- new_country
- new_salary

Query	<pre>CREATE PROCEDURE insert_new_employee(new_username text, new_firstname text, new_lastname text, new_birthdate date, new_street text, new_city text, new_zip int8, new_country text, new_salary float8) LANGUAGE plpgsql AS \$\$ BEGIN IF EXISTS (SELECT * FROM employee WHERE username = new_username) THEN RAISE EXCEPTION 'Username sudah digunakan'; ELSIF age(new_birthdate) <= interval '25 year' THEN RAISE EXCEPTION 'Employee terlalu muda'; ELSIF new_salary < 1000 THEN RAISE EXCEPTION 'Gaji Employee terlalu sedikit'; END IF; END; \$\$;</pre>
SS Hasil Query	

```
airport=# CREATE PROCEDURE insert_new_employee(new_username text, new_firstname
text, new_lastname text, new_birthdate date, new_street text, new_city text, new
_zip int8, new_country text, new_salary float8)
LANGUAGE plpgsql
AS $$
BEGIN
    IF EXISTS (SELECT * FROM employee WHERE username = new_username) THEN
        RAISE EXCEPTION 'Username sudah digunakan';
    ELSIF age(new_birthdate) <= interval '25 year' THEN
        RAISE EXCEPTION 'Employee terlalu muda';
    ELSIF new_salary < 1000 THEN
        RAISE EXCEPTION 'Gaji Employee terlalu sedikit';
    END IF;
END;
$$;
CREATE PROCEDURE
```

Query Uji	CALL insert_new_employee('Michael1', 'John', 'Doe', '1970-01-01', '123 Main St', 'New York', 10001, 'USA', 1500.00);
-----------	--

SS Hasil Query Uji

```
airport=# CALL insert_new_employee('Michael1', 'John', 'Doe', '1970-01-01', '123
Main St
', 'New York', 10001, 'USA', 1500.00);
ERROR:  Username sudah digunakan
CONTEXT:  PL/pgSQL function insert_new_employee(text,text,text,date,text,text,bi
gint,text,double precision) line 4 at RAISE
```

Query Uji	CALL insert_new_employee('haiduniaku', 'John', 'Doe', '2023-01-01', '123 Main St', 'New York', 10001, 'USA', 1500.00);
-----------	--

SS Hasil Query Uji

```
airport=# CALL insert_new_employee('haiduniaku', 'John', 'Doe', '2023-01-01', '1
23 Main St', 'New York', 10001, 'USA', 1500.00);
ERROR:  Employee terlalu muda
CONTEXT:  PL/pgSQL function insert_new_employee(text,text,text,date,text,text,bi
gint,text,double precision) line 6 at RAISE
```

Query Uji	CALL insert_new_employee('hailagidunia', 'John', 'Doe', '1970-01-01', '123 Main St', 'New York', 10001, 'USA', 90.00);
-----------	--

SS Hasil Query Uji

```
airport=# CALL insert_new_employee('hailagidunia', 'John', 'Doe', '1970-01-01',  
'123 Main St', 'New York', 10001, 'USA', 90.00);  
ERROR: Gaji Employee terlalu sedikit  
CONTEXT: PL/pgSQL function insert_new_employee(text,text,text,date,text,text,bi  
gint,text,double precision) line 8 at RAISE
```

Query Uji

```
CALL insert_new_employee('duniabasdatsip',  
'dunia', 'basdat', '1960-01-01', '123 Main St  
, 'New York', 10001, 'USA', 1500.00);
```

```
SELECT FROM employee  
WHERE username = 'duniabasdatsip';
```

SS Hasil Query Uji

```
airport=# CALL insert_new_employee('duniabasdatsip', 'dunia', 'basdat', '1960-01  
-01', '123 Main St  
, 'New York', 10001, 'USA', 1500.00);  
  
SELECT FROM employee  
WHERE username = 'duniabasdatsip';  
CALL  
--  
(0 rows)
```

3. Harinya Yang Mana ya?

Pak Sonny seorang data manager melihat adanya kejanggalan pada database *airport*. Pada data jadwal penerbangan (*flightschedule*), dicatat hari serta jam kedatangan dan keberangkatan setiap penerbangan. Terdapat 7 kolom (*monday-sunday*) yang bertipe *boolean* untuk menyatakan hari pada data jadwal tersebut, namun pada data yang sekarang bisa saja terdapat lebih dari 1 kolom yang bernilai '1' atau *true* . Selain itu, ada jadwal yang waktu keberangkatannya setelah waktu kedatangan. Bantulah Pak Sonny untuk mencegah agar hal tersebut tidak terjadi lagi dengan cara membuat trigger/fungsi/prosedur `validate_flight` untuk memastikan data baru yang akan di-*insert* atau *update* **hanya** memiliki 1 kolom hari yang bernilai '1' atau *true* serta waktu keberangkatan sebelum waktu kedatangan. Pesan *error* dibebaskan.

Note:

- abaikan kejanggalan pada data sebelumnya, dan fokus untuk mencegah hal tersebut agar tidak terjadi lagi.
- **[HINT]** untuk memastikan hanya 1 kolom hari yang bernilai *true*, lakukan pengecekan dengan menjumlahkan seluruh nilai pada seluruh kolom hari.
- **[HINT]** cara *typecasting* di postgres -> `variable_name::type`

Query	<pre>CREATE FUNCTION check_datetime() RETURNS TRIGGER AS \$\$ BEGIN IF (CAST(NEW.monday AS int) + CAST(NEW.tuesday AS int) + CAST(NEW.wednesday AS int) + CAST(NEW.thursday AS int) + CAST(NEW.friday AS int) + CAST(NEW.saturday AS int) + CAST(NEW.sunday AS int) = 1) AND (NEW.departure < NEW.arrival) THEN RETURN NEW; ELSE RAISE EXCEPTION 'Data tidak valid'; END IF; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER validate_flight BEFORE UPDATE OR INSERT ON flightschedule FOR EACH ROW EXECUTE FUNCTION check_datetime();</pre>
SS Hasil Query	


```

airport=# CREATE FUNCTION check_datetime() RETURNS TRIGGER AS $$
BEGIN
    IF (CAST(NEW.monday AS int) + CAST(NEW.tuesday AS int) + CAST(NEW.wednes
day AS int) + CAST(NEW.thursday AS int) + CAST(NEW.friday AS int) + CAST(NEW.sat
urday AS int) + CAST(NEW.sunday AS int) = 1) AND (NEW.departure < NEW.arrival) T
HEN RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Data tidak valid';
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER validate_flight
BEFORE UPDATE OR INSERT ON flightschedule
FOR EACH ROW
EXECUTE FUNCTION check_datetime();
CREATE FUNCTION
CREATE TRIGGER

```

Query Uji
Kasus
Berhasil

```

INSERT INTO flightschedule (flightno, "from",
"to", departure, arrival, airline_id, monday,
tuesday, wednesday, thursday, friday, saturday,
sunday)
VALUES ('AF1080', 9084, 2658, '10:15:00',
'20:46:00', 1, true, false, false, false, false,
false, false);

```

SS Hasil Query Uji Kasus Berhasil

```

airport=# INSERT INTO flightschedule (flightno, "from", "to", departure, arrival
, airline_id, monday, tuesday, wednesday, thursday, friday, saturday, sunday)
VALUES ('AF1080', 9084, 2658, '10:15:00', '20:46:00', 1, true, false, false, fal
se, false, false, false);
INSERT 0 1

```

Query Uji
Kasus
Gagal

```

// INSERT QUERY
INSERT INTO flightschedule (flightno, "from",
"to", departure, arrival, airline_id, monday,
tuesday, wednesday, thursday, friday, saturday,
sunday)
VALUES ('AF1079', 9084, 2658, '10:15:00',
'20:46:00', 1, true, true, false, false, false,
false, false);

// UPDATE QUERY
UPDATE flightschedule
SET departure = '15:00:00', arrival = '13:00:00'
WHERE flightno = 'AF1080';

```

SS Hasil Query Uji Kasus Gagal

```
airport=# INSERT INTO flightschedule (flightno, "from", "to", departure, arrival
, airline_id, monday, tuesday, wednesday, thursday, friday, saturday, sunday)
VALUES ('AF1079', 9084, 2658, '10:15:00', '20:46:00', 1, true, true, false, fals
e, false, false, false);

UPDATE flightschedule
SET departure = '15:00:00', arrival = '13:00:00'
WHERE flightno = 'AF1080';
ERROR: Data tidak valid
CONTEXT: PL/pgSQL function check_datetime() line 5 at RAISE
ERROR: Data tidak valid
CONTEXT: PL/pgSQL function check_datetime() line 5 at RAISE
```

Dari kasus tersebut, level *integrity constraint* mana sajakah yang dijaga? (bisa lebih dari 1)

JAWAB

Level integrity constraint yang dijaga adalah relational karena berhubungan dengan validitas data dalam satu relasi.

4. Viva La France

Marie Antoinette suka sekali dengan *French culture*. *Mariah Carey once said, viva la FRRRRRRANCEEE, we are in Paris*. Oleh karena itu, bantulah Marie Antoinette untuk membuat fungsi dengan nama `oui_oui_baguettes()` yang menerima sebuah booking id dan mengecek apakah booking id tersebut memiliki destinasi tujuan yaitu bandara yang berlokasi di Perancis (ditandai dengan `country = 'FRANCE'`). Jika ya, tampilkan tulisan “*viva la FRRRRRRANCEEE <nama lengkap pelanggan>*”, jika tidak, tampilkan “Visit France Later”

HINT: Beri petik 2 (") untuk reserved keyword SQL

Query	<pre>CREATE FUNCTION oui_oui_baguettes(booking_idRawr INT) RETURNS TEXT AS \$\$ DECLARE country text; DECLARE pfirstname text; DECLARE plastname text; BEGIN SELECT ag.country INTO country FROM booking b JOIN flight f USING(flight_id) JOIN airport_geo ag ON ag.airport_id = f.to WHERE b.booking_id = booking_idRawr; SELECT p.firstname INTO pfirstname FROM booking b JOIN passenger p USING(passenger_id) WHERE b.booking_id = booking_idRawr; SELECT p.lastname INTO plastname FROM booking b JOIN passenger p USING(passenger_id) WHERE b.booking_id = booking_idRawr; IF (country = 'FRANCE') THEN RETURN 'viva la FRRRRRRANCEEE' ' ' pfirstname ' ' plastname; ELSE RETURN 'Visit France Later'; END IF; END;</pre>
-------	--

	\$\$ LANGUAGE plpgsql;
SS Hasil Query	
<pre> airport=# CREATE FUNCTION oui_oui_baguettes(booking_idRawr INT) RETURNS TEXT AS \$\$ DECLARE country text; DECLARE pfirstname text; DECLARE plastname text; BEGIN SELECT ag.country INTO country FROM booking b JOIN flight f USING(flight_id) JOIN airport_geo ag ON ag.airport_id = f.to WHERE b.booking_id = booking_idRawr; SELECT p.firstname INTO pfirstname FROM booking b JOIN passenger p USING(passenger_id) WHERE b.booking_id = booking_idRawr; SELECT p.lastname INTO plastname FROM booking b JOIN passenger p USING(passenger_id) WHERE b.booking_id = booking_idRawr; IF (country = 'FRANCE') THEN RETURN 'viva la FRRRRRANCEEE' ' ' pfirstname ' ' plastname; ELSE RETURN 'Visit France Later'; END IF; END; \$\$ LANGUAGE plpgsql; CREATE FUNCTION </pre>	
Query Uji	SELECT oui_oui_baguettes(15327);
SS Hasil Query Uji	
<pre> airport=# SELECT oui_oui_baguettes(15327); oui_oui_baguettes ----- viva la FRRRRRANCEEE David Selby (1 row) </pre>	
Query Uji	SELECT oui_oui_baguettes(1);
SS Hasil Query Uji	

```
airport=# SELECT oui_oui_baguettes(1);
oui_oui_baguettes
-----
Visit France Later
(1 row)
```

Selanjutnya, buatlah fungsi bernama merci_beaucoup() yang menerima sebuah booking id dan mengecek apakah booking id tersebut memiliki destinasi asal yaitu bandara yang berlokasi di Perancis (ditandai dengan country = 'FRANCE'). Jika ya, tampilkan tulisan “*au revoir <nama lengkap pelanggan>*”, jika tidak, tampilkan “no France no cool”

Query	CREATE FUNCTION merci_beaucoup(booking_id INT) RETURNS TEXT AS \$\$
SS Hasil Query	
<pre>airport=# CREATE FUNCTION merci_beaucoup(booking_idRawr INT) RETURNS TEXT AS \$\$ DECLARE country text; DECLARE pfirstname text; DECLARE plastname text; BEGIN SELECT ag.country INTO country FROM booking b JOIN flight f USING(flight_id) JOIN airport_geo ag ON ag.airport_id = f.from WHERE b.booking_id = booking_idRawr; SELECT p.firstname INTO pfirstname FROM booking b JOIN passenger p USING(passenger_id) WHERE b.booking_id = booking_idRawr; SELECT p.lastname INTO plastname FROM booking b JOIN passenger p USING(passenger_id) WHERE b.booking_id = booking_idRawr; IF (country = 'FRANCE') THEN RETURN 'au revoir' ' ' pfirstname ' ' plastname; ELSE RETURN 'no France no cool'; END IF; END; \$\$ LANGUAGE plpgsql; CREATE FUNCTION</pre>	
Query Uji	SELECT merci_beaucoup(14532);

SS Hasil Query Uji	
<pre>airport=# SELECT merci_beaucoup(14532); merci_beaucoup ----- au revoir Jason Gamble (1 row)</pre>	
Query Uji	<pre>SELECT merci_beaucoup(1);</pre>
SS Hasil Query Uji	
<pre>airport=# SELECT merci_beaucoup(1); merci_beaucoup ----- no France no cool (1 row)</pre>	

5. Kita Bikin *Airport* Gas!

Untuk mewujudkan OUTDONESIA EMAS 3140, direncanakan akan dibangun 1000 bandara untuk membuat akses ke *micronation* Outdonesia, menjadi lebih mudah. Untuk itu ada beberapa hal yang harus dilakukan untuk mempermudah pencatatan 1000 bandara baru tersebut yaitu,

1. Basis data *airport* harus memastikan bahwa untuk setiap penambahan data pada tabel *airport*, maka tabel *airport_geo*, harus ada tuple yang mengacu pada *airport* baru, meskipun nilai-nilainya dibiarkan kosong atau pada angka default tertentu seperti 0
2. IATA dan ICAO harus unik untuk setiap *airport*. Untuk duplikasi data IATA dan ICAO, keluarkan pesan error

Ingatlah bahwa data pada tabel tidak hanya dapat berubah ketika terjadi insersi tetapi juga saat terjadi proses pembaharuan.

Karena proses ini menghasilkan beberapa kolom dengan isi default kosong pada tabel *airport_geo*, buatlah procedure untuk mengupdate kolom-kolom yang kosong berdasarkan kode IATA atau ICAO masukan.

Lakukan modifikasi/penambahan fitur pada basis data untuk mewujudkan hal tersebut!

HINT:

1. Untuk memeriksa apakah ada yang tuple yang dikembalikan dari query, gunakan fungsi EXISTS
2. Gunakan `RAISE EXCEPTION '<PESAN>'` untuk mengembalikan pesan error
3. Tipe data point dapat dibuat dengan menggunakan `POINT(a,b)`; a, b: decimal/integer/float

Query	<pre>CREATE FUNCTION check_kode() RETURNS TRIGGER AS \$\$ BEGIN IF (NEW.iata IN (SELECT iata FROM airport) OR NEW.icao IN (SELECT icao FROM airport)) THEN RAISE EXCEPTION 'IATA or ICAO already exist'; ELSE RETURN NEW; END IF; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER terbang BEFORE INSERT OR UPDATE ON airport FOR EACH ROW</pre>
-------	---

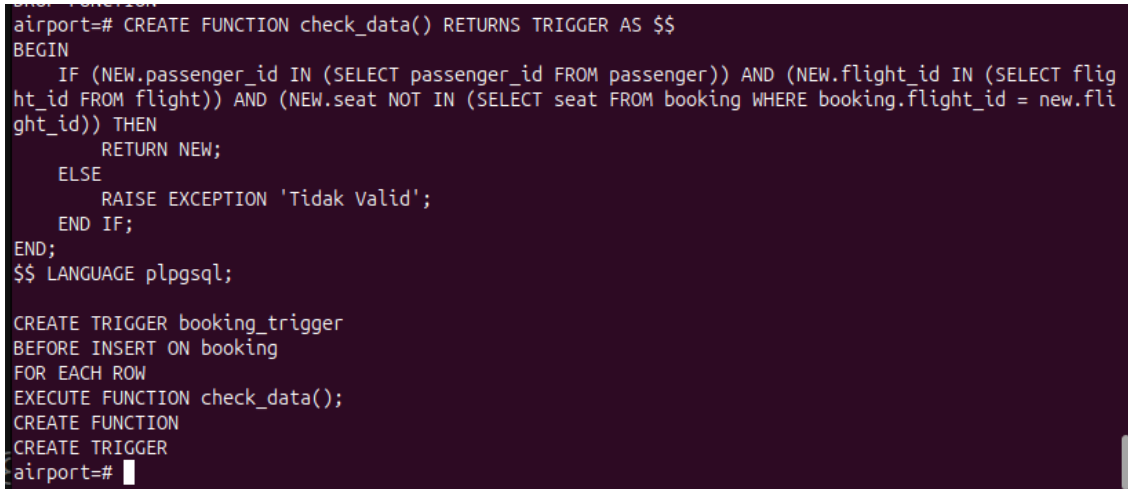
	EXECUTE FUNCTION check_kode();
Level Constraint	Constraint bekerja pada level database constraint karena menghubungkan lebih dari 1 relasi
SS Hasil Query	
<pre> airport=# CREATE FUNCTION check_kode() RETURNS TRIGGER AS \$\$ BEGIN IF (NEW.iata IN (SELECT iata FROM airport) OR NEW.icao IN (SELECT icao FROM airport)) THEN RAISE EXCEPTION 'IATA or ICAO already exist'; ELSE RETURN NEW; END IF; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER terbang BEFORE INSERT OR UPDATE ON airport FOR EACH ROW EXECUTE FUNCTION check_kode(); CREATE FUNCTION CREATE TRIGGER </pre>	
Query Uji Inseri (Lakukan 2 kali untuk uji gagal dan berhasil)	INSERT INTO airport(name, iata, icao) VALUES ('OkeGas', 'OG', 'OKG');
SS Hasil Query Uji	
<pre> airport=# INSERT INTO airport(name, iata, icao) VALUES ('OkeGas', 'OG', 'OKG'); INSERT 0 1 airport=# INSERT INTO airport(name, iata, icao) VALUES ('OkeGas', 'OG', 'OKG'); ERROR: IATA or ICAO already exist CONTEXT: PL/pgSQL function check_kode() line 4 at RAISE </pre>	
Query Uji Update Airport	UPDATE airport SET iata = 'OG', icao='OKG' WHERE name='OkeGas';
SS Hasil Query Uji	
<pre> CONTEXT: PL/pgSQL function check_kode() line 4 at RAISE airport=# UPDATE airport SET iata = 'OG', icao='OKG' WHERE name='OkeGas'; ERROR: IATA or ICAO already exist CONTEXT: PL/pgSQL function check_kode() line 4 at RAISE </pre>	
Query Uji Procedure AirportGeo	CALL update_airport_geo('OG', 'Bankarta', 'Outdonesia', 0, 0, POINT(1.0, 2.0));

SS Hasil Query Uji

6. Validasi data

Pak Cello sebagai pencipta *database airport* ini merasa *database* yang telah dibuat masih memiliki beberapa kekurangan. Oleh karena itu, dia menyuruh anda sebagai junior DBA untuk membuat suatu hal yaitu:

1. Buat trigger untuk booking passenger. Pastikan untuk flightid, seat, dan passenger_id valid untuk setiap insersi booking. Definisi valid disini adalah pastikan data passenger_id dan flight_id terdapat di dalam database serta seat masih tersedia.

Query	<pre>CREATE FUNCTION check_data() RETURNS TRIGGER AS \$\$ BEGIN IF (NEW.passenger_id IN (SELECT passenger_id FROM passenger)) AND (NEW.flight_id IN (SELECT flight_id FROM flight)) AND (NEW.seat NOT IN (SELECT seat FROM booking WHERE booking.flight_id = new.flight_id)) THEN RETURN NEW; ELSE RAISE EXCEPTION 'Tidak Valid'; END IF; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER booking_trigger BEFORE INSERT ON booking FOR EACH ROW EXECUTE FUNCTION check_data();</pre>
SS Hasil Query	
	
Query Uji	<pre>– Test invalid passenger_id INSERT INTO booking (booking_id, flight_id, seat, passenger_id, price) VALUES (3000001, 999, '12A', 1, 2);</pre>
SS Hasil Query Uji	

<pre>airport=# INSERT INTO booking (booking_id, flight_id, seat, passenger_id, price) VALUES (30000001, 999, '12A', 1, 2); ERROR: Tidak Valid CONTEXT: PL/pgSQL function check_data() line 6 at RAISE</pre>	
Query Uji	<p>– Test invalid flight_id</p> <pre>INSERT INTO booking (booking_id, flight_id, seat, passenger_id, price) VALUES (3000000, 999999, '12A', 5, 5);</pre>
SS Hasil Query Uji	
<pre>airport=# INSERT INTO booking (booking_id, flight_id, seat, passenger_id, price) VALUES (3000000, 999999, '12A', 5, 5); ERROR: Tidak Valid CONTEXT: PL/pgSQL function check_data() line 6 at RAISE</pre>	
Query Uji	<p>– Test invalid seat</p> <pre>INSERT INTO booking (booking_id, flight_id, seat, passenger_id, price) VALUES (3000000, 3863, '25F', 5, 5);</pre>
SS Hasil Query Uji	
<pre>airport=# INSERT INTO booking (booking_id, flight_id, seat, passenger_id, price) VALUES (3000000, 3863, '25F', 5, 5); ERROR: Tidak Valid CONTEXT: PL/pgSQL function check_data() line 6 at RAISE</pre>	

Dari kasus-kasus diatas, level *integrity constraint* mana sajakah yang dijaga? (bisa lebih dari 1)

Database constraints karena melibatkan lebih dari 1 relasi pada database

Pembagian Kerja

NIM	Pembagian Kerja
13522053	No 1, 2, 3, 4, 5, 6
13522089	No 1, 2, 3, 4, 5, 6