DESAIN BASIS DATA DENGAN MENGGUNAKAN PL/PGSQL DAN CHECK CONSTRAINT

Stevi Ema Wijayanti [1)]

[1)] *Magister Teknik Informatika STMIK AMIKOM Yogyakarta*
Jl. Ring Road Utara Condong Catur Depok Sleman Yogyakarta
Email : steviema.wijayanti@gmail.com

Abstrak

Saat ini basis data menjadi salah satu komponen yang sangat penting bagi sistem informasi. Untuk itu desain basis data harus lebih diperhatikan. Basis data yang baik adalah basis data yang memiliki batasan berupa aturan-aturan tertentu sehingga data yang tersimpan dalam tabel berkualitas. Aturan yang diberikan dalam basis data dapat menggunakan bahasa prosedural PL/PgSQL atau disebut juga dengan PL/SQL. Dengan adanya aturan tersebut maka integritas basis data akan terjaga.

Kata kunci : *pl/pgsql, check constraint, integritas basis data*

1. Pendahuluan

1.1 Latar Belakang

Basis data merupakan komponen yang penting dalam sebuah sistem informasi modern. Sebagian besar sistem informasi dewasa ini hampir semuanya menggunakan Relational Database Management System (RDBMS), Sistem Basis Data Relational. Software RDBMS yang umum digunakan dalam sistem informasi adalah Oracle, Ms SQL Server, PostgreSQL, DB2, FirebirdSQL atau MySQL. [1]

Sebuah sistem dikatakan baik, apabila memiliki desain basis data yang baik sehingga dapat mendukung kinerja dari sistem tersebut. Hal ini menunjukkan bahwa basis data memegang peranan penting dalam sebuah sistem informasi. Basis data dituntut untuk memenuhi seluruh kebutuhan berbagai informasi dan data penggunanya.

1.2 Tinjauan Pustaka

Basis data adalah kumpulan data yang saling berelasi. Data sendiri merupakan fakta mengenai obyek, orang dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter atau simbol). [2]

Basis Data terdiri dari 2 kata, yaitu Basis dan Data. Basis kurang lebih diartikan sebagai markas/gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu obyek seperti manusia, barang, hewan, dan sebagainya. [3]

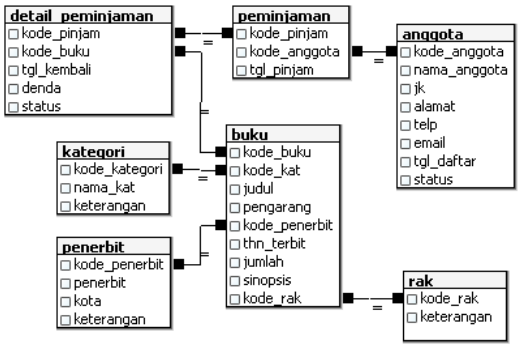
Constraint adalah fitur-fitur yang ada didalam *Database Management System* (DBMS) yang dapat melakukan validasi terhadap data-data yang diisikan pada sisi server dan bukan lagi pada sisi aplikasi *client* seperti yang selama ini digunakan oleh *programmer*. [4]

Contoh *constraint* antara lain :

1. Primary Key
   Primary Key atau Kunci Utama dalam sebuah tabel merupakan kunci yang akan membatasi pengisian record dalam sebuah tabel agar tidak duplikat (*redudant*). [4]
2. Foreign Key
   Foreign Key atau Kunci Tami merupakan kunci yang digunakan sebagai penghubung antara satu tabel dengan tabel lainnya. [4]
3. Unique
   Constraint Unique merupakan sebuah constraint yang akan membatasi pengisian record yang sama ke dalam sebuah kolom jika kolom tersebut diberikan constrainy Unique dalam sebuah tabel. [4]
4. Not Null
   Not Null merupakan constraint yang digunakan untuk menjamin pengisian record ke sebuah tabel agar nilai record tersebut harus berisi data. [4]
5. Check
   Check merupakan constraint yang memaksa integritas domain dengan membatasi pemasukan nilai yang mungkin di masukkan ke dalam sebuah kolom atau beberapa kolom. [4]

2. Hasil dan Pembahasan

Basis Data yang digunakan sebagai contoh dalam penelitian ini adalah desain database untuk perpustakaan, seperti gambar di bawah ini :



Gambar 4. 1 Relasi Tabel

Metode-metode yang digunakan untuk menjaga integritas data yang tersimpan dalam tabel antara lain :

1. Pemberian kunci primer (*primary key*) pada *field* di masing-masing tabel untuk menghindari adanya redudansi dan inkonsistensi data.
2. Penggunaan kunci tamu (*foreign key*) untuk menjaga integritas referensial bagi tabel-tabel yang saling berelasi.
3. Pemilihan tipe data yang sesuai dengan kebutuhan data yang akan disimpan serta pemberian parameter *null* dan *not null* pada setiap *field* pada tabel. Hal tersebut perlu diperhatikan mengingat ada beberapa data yang harus diisikan dan tidak harus diisi.
4. *Constraint check* difungsikan untuk membantu memvalidasi data yang akan disimpan dalam basis data selain telah menentukan tipe data dan parameter untuk masing-masing *field*.

Dengan mengimplementasikan metode diatas, maka data yang akan diinputkan dalam basis data lebih baik dan benar. Adanya *primary key* pada setiap tabel dapat mencegah terjadinya data ganda karena ciri dari *primary key* adalah unik yang berarti tidak boleh ada data yang sama dalam satu tabel. Begitu pula dengan *foreign key*, dengan adannya kunci tamu tersebut maka tabel-tabel yang saling berelasi melalui suatu *field* tidak dapat diisikan data sembarang, *field* tersebut hanya dapat diisikan data-data yang telah tersedia pada tabel yang dijadikan sebagai tabel master. Penentuan tipe data dan parameter pada masing-masing *field* harus diperhatikan agar data yang disimpan sesuai dengan kebutuhan. Hal tersebut diperkuat dengan diberikannya *constraint check* pada *field-field* yang membutuhkan sehingga data yang tersimpan dalam basis data sesuai.

Berdasarkan contoh yang telah tergambar di atas, perlu dibuat tabel kategori untuk menyimpan data kategori buku perpustakaan, maka *query* pembuatan tabel kategori adalah sebagai berikut :

```
CREATE TABLE kategori (
 kode_kategori CHAR(3) NOT NULL PRIMARY
 KEY
 CHECK (kode_kategori ~ 'K[0-9][0-9][0-
 9]'),
 nama_kat VARCHAR(35) NOT NULL
 CHECK (nama_kat ~ '^[A-Za-z]'),
 keterangan TEXT NULL
);
```

Tabel kategori di atas memiliki kode_kategori yang dijadikan sebagai kunci primer (primary key). *Constraint check* pada *field* kode_kategori difungsikan untuk menyamakan format kode_kategori yaitu “K01”, “K02”, “K03” dan selanjutnya. Hanya dua digit terakhir yang dapat diubah angkanya, namun karakter “K” tidak boleh diubah. Nama kategori diberikan *constraint check* agar isi dari nama kategori hanya dapat diisikan huruf mulai dari A sampai Z dan tidak dapat diisi karakter yang lainnya. Kode_kategori dan nama_kat diberikan parameter NOT NULL dan keterangan diberikan parameter NULL yang berarti data boleh diisikan, boleh tidak.

Contoh tabel yang selanjutnya adalah tabel buku, dimana tabel tersebut menyimpan semua data buku yang ada di perpustakaan.

```
CREATE TABLE buku (
 kode_buku CHAR(5) NOT NULL
 PRIMARY KEY
 CHECK (kode_buku ~ 'B[0-9][0-9] [0-
 9][0-9]'),
 kode_kat CHAR(3) NOT NULL REFERENCES
 kategori(kode_kategori),
 judul VARCHAR(100) NOT NULL
 CHECK (judul ~ '^[A-Za-z0-9]'),
 pengarang VARCHAR(100) NOT NULL CHECK
 (pengarang ~ '^[A-Za-z]'),
 kode_penerbit CHAR(4) NOT NULL
 REFERENCES penerbit(kode_penerbit),
 thn_terbit CHAR(4) NOT NULL
 CHECK (thn_terbit ~ '[1-2][0-9][0-9][0-
 9]'),
 jumlah INT NOT NULL
 CHECK (jumlah >= 0),
 sinopsis TEXT NULL,
 kode_rak CHAR(3) NOT NULL REFERENCES
 rak(kode_rak)
);
```

Seperti halnya tabel kategori, pada tabel buku memiliki satu *primary key* yaitu *field* kode_buku dan pada *field* tersebut juga diberikan *constraint check* untuk memvalidasi data yang diinputkan. Terdapat kunci tamu (*foreign key*) yaitu *field* kode_kat yang berelasi dengan tabel kategori pada *field* kode_kategori. Dengan adanya *foreign key* tersebut, maka data kode_kat yang ada tabel buku tidak dapat diisikan data yang tidak tersedia pada tabel kategori. Terdapat *constraint check* pada beberapa *field* untuk membatasi data yang akan diinputkan pada basis data, contohnya pada *field* jumlah tidak dapat diisikan data kurang dari 0.

Dengan adanya *check constraint* pada tabel diatas, misal ingin menginputkan data pada tabel kategori.

```
INSERT INTO kategori VALUES
('G01','Basis Data','Kumpulan buku
mengenai basis data');
```

Jika *query* tersebut dijalankan maka akan muncul pesan kesalahan dikarenakan kode_kategori sudah diberikan aturan bahwa huruf depannya harus "K". Begitu pula dengan *field* lain yang telah diberikan *check constraint* tidak dapat diisikan data secara acak, namun harus mengikuti aturan yang telah ditentukan.

```
ERROR:  new row for relation "kategori"
violates check constraint
"kategori_kode_kategori_check"
DETAIL:  Failing row contains (G01,
Basis Data, Kumpulan buku mengenai basis
data).
```

Jika memiliki tabel yang berelasi dengan tabel lain, maka tabel yang mengandung *foreign key* tidak dapat diisikan data yang tidak tersedia pada tabel master. Misalnya dalam tabel kategori terdapat data sebagai berikut :

	kode character(3)	kategori character varying(35)
1	K01	Basis Data
2	K02	Pemrograman
3	K03	Komputer Grafis

Kemudian ingin mengisi data kode_kategori pada tabel buku dengan kode_kategori yang tidak tersedia pada tabel kategori, seperti query di bawah ini :

```
INSERT INTO buku VALUES ('B0007', 'K05'
,'Mahir Komputer Grafis dalam 24 Jam',
'Rahardian', 'P001', '2006', 10, '' ,
'R03');
```

Maka akan muncul pesan kesalahan seperti contoh di bawah ini :

```
ERROR:  insert or update on table "buku"
violates foreign key constraint
"buku_kode_kat_fkey"
DETAIL:  Key (kode_kat)=(K05) is not
present in table "kategori".
```

Dari beberapa contoh yang telah tersedia, dapat diambil kesimpulan bahwa *constraint* sangat diperlukan dalam mendesain basis data. Hal tersebut dilakukan untuk menjaga integritas data yang tersimpan di dalam basis data.

Contoh lain yang dapat diambil dari basis data perpustakaan, misalnya perpustakaan ABC membuat aturan bahwa setiap anggota hanya dapat meminjam 3 buku dalam setiap peminjaman. Jika anggota tersebut telah meminjam 3 buku dan belum mengembalikan buku tersebut, maka tidak dapat melakukan peminjaman buku yang lain. Berdasarkan kasus tersebut dapat dibuatkan sebuah fungsi untuk mengecek apakah anggota tersebut diperbolehkan meminjam buku atau tidak.

```
CREATE FUNCTION cekPeminjaman(
kodeAnggota CHAR(9))
RETURNS BOOLEAN AS $BODY$
DECLARE
hasil record;
hitung INT := 0;

BEGIN
FOR hasil IN SELECT
kode_anggota,kode_buku
FROM peminjaman JOIN detail_peminjaman
USING (kode_pinjam)
WHERE kode_anggota = kodeAnggota AND
status = 'N'

LOOP
   hitung = hitung + 1;
END LOOP;

IF hitung = 3 THEN
   RETURN 'F';
ELSE
   RETURN 'T';
END IF;
END;

$BODY$ LANGUAGE 'plpgsql';
```

Kemudian tambahkan fungsi *constraint check* pada tabel peminjaman, sehingga sebelum data peminjaman dapat disimpan dalam basis data akan ada pengecekan terlebih dahulu.

```
CREATE TABLE peminjaman (
 kode_pinjam INT NOT NULL
 PRIMARY KEY,
 kode_anggota CHAR(9) NOT NULL
 REFERENCES anggota(kode_anggota),
 tgl_pinjam DATE NOT NULL,
 CONSTRAINT cek CHECK (cekPeminjaman
 (kode_anggota))
);
```

Misalnya : kode_anggota "AS1310001" dengan nama "Stevi Ema W." ingin meminjam buku di perpustakaan ABC. Namun sebelumnya telah melakukan peminjaman buku dan belum dikembalikan. Dengan menggunakan fungsi cek yang telah dibuat. Jika dilihat pada tabel detail_peminjaman, anggota tersebut telah meminjam sebanyak 3 buku dan belum ada yang dikembalikan.

	kode_anggota character(9)	kode_buku character(5)	judul character varying(100)
1	AS1310001	B0001	Konsep Basis Data
2	AS1310001	B0002	Sistem Basis Data
3	AS1310001	B0003	Pemrograman Java

Apabila anggota tersebut ingin meminjam buku perpustakaan.

```
INSERT INTO peminjaman VALUES
(2,'AS1310001','2013-11-1');
```

Maka akan muncul pesan kesalahan karena buku yang dipinjam berjumlah tiga buah buku.

```
ERROR:  new row for relation
"peminjaman" violates check constraint
"cek"
DETAIL:  Failing row contains (2,
AS1310001, 2013-11-01);
```

Selain itu ditambahkan sebuah trigger untuk mengubah jumlah buku pada tabel buku ketika buku tersebut dipinjam oleh anggota perpustakaan. Sebelum membuat trigger, di buat fungsi terlebih dahulu yang difungsikan untuk mengubah jumlah buku di tabel buku.

```
CREATE FUNCTION update_jumlah() RETURNS
TRIGGER
AS $BODY$

BEGIN
UPDATE buku SET jumlah = jumlah-1 WHERE
kode_buku = NEW.kode_buku;
RETURN NEW;
END

$BODY$ LANGUAGE 'plpgsql';
```

Kemudian dibuat trigger, trigger yang dibuat akan dieksekusi setelah data peminjaman buku disimpan dalam tabel detail_peminjaman. Berikut ini adalah query sql pembuatan trigger.

```
CREATE TRIGGER ubah_jumlah AFTER INSERT
ON detail_peminjaman
for EACH ROW EXECUTE PROCEDURE
update_jumlah();
```

Setelah dibuat trigger, maka ketika menginputkan data peminjaman buku, secara otomatis akan mengubah stok pada tabel buku. Seperti contoh di bawah ini :

Sebelum menjalankan fungsi trigger, jumlah dari buku yang tersedia adalah :

	kode_buku character(5)	judul character varying(100)	jumlah integer
1	B0001	Konsep Basis Data	10
2	B0002	Sistem Basis Data	10
3	B0004	Pemrograman Visual	10

Buku dengan kode ‘B0002’ yang berjudul Sistem Basis Data dipinjam oleh seorang anggota. Data peminjaman buku disimpan pada tabel peminjaman dan detail_peminjaman. Berikut ini adalah query untuk insert data ke tabel peminjaman dan detail_peminjaman.

```
INSERT INTO peminjaman VALUES
(3,'AS1310002','2013-12-3');

INSERT INTO detail_peminjaman values
(3,'B0002','2013-11-7',0,'N');
```

Setelah data peminjaman di tambahkan pada tabel detail_peminjaman secara otomatis jumlah buku pada tabel buku akan berkurang.

	kode_buku character(5)	judul character varying(100)	jumlah integer
1	B0001	Konsep Basis Data	10
2	B0002	Sistem Basis Data	9
3	B0003	Pemrograman Java	9
4	B0004	Pemrograman Visual	10
5	B0005	Pembuatan Karakter	10
6	B0006	Mahir Komputer Graf	10

Dengan menggunakan trigger, tidak perlu menjalankan dua query untuk melakukan insert data dan ubah jumlah buku. Cukup menjalankan query untuk insert data ke dalam tabel detail peminjaman maka secara otomatis jumlah buku akan berubah.

3. Kesimpulan

Constraint check digabungkan dengan PL/PgSQL dapat membuat tabel lebih terlindungi dari kemungkinan terjadinya kesalahan. Penggunaan Constraint check dan PL/PgSQL yang diletakkan pada sebuah tabel dapat membuat tabel tersebut mampu menjaga data yang dimasukkan ke dalam basis data dan sekaligus dapat membantu perubahan data sesuai dengan aturan yang telah ditentukan. Pemberian aturan itulah yang akan sangat membantu untuk menjaga integritas dari basis data tersebut.

Daftar Pustaka

[1] Raharjo, S. 2012. *Constraint Basis Data sebagai Fondasi yang Kuat Dalam Pengembangan Sistem Informasi*, Seminar Nasional Aplikasi Sains & Teknologi (SNAST) Periode III ISSN: 1979-911X Yogyakarta, 3 November 2012 (SNAST 2012)
[2] Kusrini, *Strategi Perancangan dan Pengelolaan Basis Data*, Yogyakarta: Andi Offset, 2007.
[3] Fathansyah, *Basis Data*, Bandung: Informatika, 2012.
[4] Arief, M. R., *Pemrograman Basis Data Menggunakan Transact-SQL dengan Microsoft SQL Server 2000*, Yogyakarta: Andi Offset, 2006.

Biodata Penulis

Stevi Ema Wijayanti, memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Sistem Informasi STMIK AMIKOM Yogyakarta pada tahun 2013. Saat ini sedang menempuh Program Pasca Sarjana Program Studi Magister Teknik Informatika STMIK AMIKOM Yogyakarta dan sebagai tim Research & Development (R&D) PT Mataram Surya Visi (MSV) sekaligus Dosen Pengajar di STMIK AMIKOM Yogyakarta.