*ISSN : 2302-3805*

# INTEGRITY CONSTRAINT BASIS DATA RELASIONAL DENGAN MENGGUNAKAN PL/PGSQL DAN CHECK CONSTRAINT

**Suwanto Raharjo** [1)]
[1)] *Jurusan Teknik Informatika*
*Fakultas Teknologi Industri Institut Sains & Teknologi AKPRIND Yogyakarta*
Jl. Kalisahak No. 28, Kompleks Balapan, Yogyakarta
*E-mail: wa2n@akprind.ac.id*

## Abstrak
*Desain basis data merupakan hal yang sangat penting di dalam sebuah sistem informasi. Kualitas data dan informasi yang tersaji sangat terpengaruh oleh bagaimana basis data tersebut didesain. Pemberian pembatasan terhadap data yang masuk dengan memberikan aturan-aturan tertentu dalam sebuah basis data atau tabel akan membantu menjaga integritas basis data. Selain dengan menggunakan program antarmuka seperti PHP, VB atau Delphi dapat pula digunakan PL/PgSQL. PL/PgSQL merupakan bahasa prosedural dalam PostgreSQL yang dapat digunakan bersama dengan constraint CHECK untuk memberikan pembatasan terhadap data yang akan dimasukan dalam suatu tabel.*

**Kata kunci :**
*integritas basis data, plpgsql, check constraint, postgresql*

## 1. Pendahuluan

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 Relasional. Software RDBMS yang umum digunakan dalam sistem informasi adalah Oracle, Ms SQL Server, PostgreSQL, DB2, FirebirdSQL atau MySQL [3].

Salah satu peran basis data, yaitu sebagai sumber informasi bagi Sistem Informasi Manajemen (SIM), mengimplikasikan bahwa basis data yang digunakan harus mampu memenuhi kebutuhan berbagai informasi (dan data) bagi para penggunanya, baik untuk saat ini maupun mendatang [4].

Basis data relasional dapat dilihat sebagai suatu data yang disimpan dalam suatu tabel. Tabel tersebut dirancang sedemikian rupa agar data yang tersimpan didalamnya dapat diolah untuk menjadi informasi yang berkualitas. Performance dari sistem informasi bergantung pada 3 faktor (triad factors); [5]

1. Desain basis data dan implementasinya
2. Desain aplikasi dan implementasi
3. Prosedur administastif

Dari tiga faktor diatas desain basis data merupakan faktor yang utama dalam membentuk informasi yang berkualias. Informasi yang berkualitas hanya akan bisa didapat dari data yang berkualitas. Data berkualitas dapat dibentuk dari pembuatan basis data, tabel yang berkualitas. Salah satu parameter yang dapat digunakan untuk mengukur kualitas basis data adalah bagaimana basis data tersebut dapat menyimpan data dengan valid.

## 2. Tinjauan Pustaka

Validitas data yang disimpan dalam sebuah basis data relasional dapat dijaga dengan menggunakan constraint integritas. Dalam basis data relasional secara umum terdapat 3 tipe constraint untuk menjaga integritas basis data yakni :

1. Constraint Entitas
2. Constraint Referensial
3. Constraint Domain

Perancang basis data pada saat membuat tabel akan memberikan constraint atau aturan-aturan yang mencerminkan tingkah laku dari suatu tabel atau kolom. Sebagai contoh apakah kolom suatu tabel hanya dapat menerima data yang bersifat unik, tidak menerima nilai *null* atau hanya menerima suatu karakter tertentu saja. Constraint yang umum digunakan dalam merancang tabel adalah *Primary Key* (kunci primer) dan *Foreign Key* (kunci tamu) yang digunakan untuk menjaga integritas entitas dan referensial. Constraint *domain* umumnya hanya digunakan dengan memberikan tipe data yang tepat pada suatu kolom.

Dalam penelitian sebelumnya terlihat bahwa sebagian besar penelitian yang membahas pengembangan sistem informasi di Indonesia dengan menggunakan basis data relasional, perancangan basis datanya masih hanya menggunakan kunci primer, kunci tamu dan pemilihan tipe data bahkan beberapa perancangan sistem informasi tidak menyinggung penggunaan constraint di level basis data [3].

*ISSN : 2302-3805*

ANSI SQL-92 memuat constraint CHECK yang dapat digunakan untuk melakukan pembatasan input data dalam basis data relasional. Constraint CHECK yang terdapat dalam ANSI SQL-92 merupakan constraint paling fleksibel dan berguna [2]. Constraint CHECK dapat digunakan untuk memberikan pembatasan masukan pada basis data. Sebagai contoh adalah penggunaan constraint CHECK untuk membatasi karakter tertentu dengan menggunakan *regex* dan penggunakan operator relasional [3].

Pemberian pembatasan data yang diinputkan dapat mencegah terjadinya exploitasi antar muka program ke suatu basis data. Kesalahan dalam pembuatan program antar muka yang melakukan komunikasi ke server basis data dapat menyembabkan munculnya kerawanan keamanan terhadap data yang disimpan. Injeksi SQL melalui program antar muka dapat dilakukan jika terjadi kesalahan pemograman. Akar penyebab dari injeksi SQL adalah ketidakcukupan validasi masukan [1] [6]. Kekurangcukupan validasi masukan bukan hanya menyembabkan masuknya data yang tidak valid namun juga dapat menimbulkan implikasi keamanan data.

## 3. Metode Penelitian

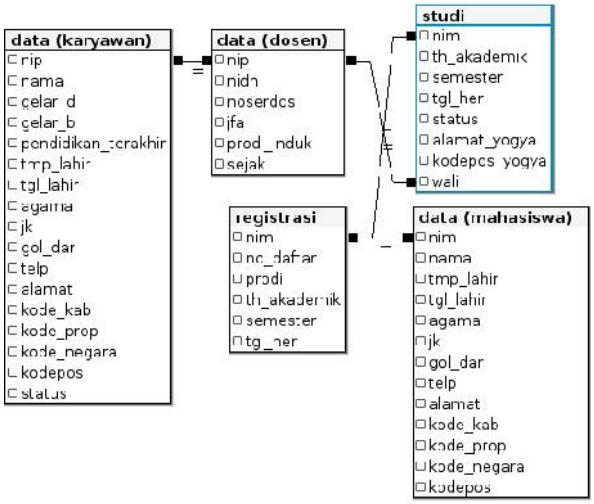
Penelitian ini akan melakukan pengembangan penggunaan constraint CHECK dengan menggunakan bahasa prosedural PL/PgSQL dalam basis data server PostgreSQL. Penelitian ini dimulai dengan melakukan pembuatan tabel yang dilengkapi constraint CHECK pada suatu kolom dimana constraint tersebut memanggil fungsi yang dibuat dengan PL/PgSQL. Fungsi yang dibuat dengan PL/PgSQL akan melakukan pengujian terhadap data yang diinputkan apakah memenuhi aturan yang ditetapkan pada tabel tersebut atau tidak.

Kemudian dilakukan pengujian dengan menginputkan dan mengubah data yang tidak sesuai dengan aturan yang ditetapkan dan dilihat apakah tabel yang dibuat dengan tambahan constraint CHECK dengan penggunaan fungsi PL/PgSQL dapat mencegah terjadinya pemasukan data yang tidak valid.

## 4. Hasil dan Pembahasan

Tabel yang digunakan sebagai contoh dalam penelitian ini adalah seperti yang tertampil pada gambar 1.

Dari gambar 1 terlihat bahwa terdapat 5 tabel yakni tabel Tabel *data (karyawan), data (dosen), registasi, studi* dan *data (mahasiswa)* yang memiliki kerelasian.




Gambar 1 Relasi Tabel yang digunakan

Beberapa metode berikut sering digunakan untuk menjaga integritas data yang dimasukan pada tabel di gambar no 1 :

1. Pemberian kunci primer, untuk mejaga integritas entitas digunakan kunci primer yakni kolom *nip* di tabel *data(karyawan)* dan *data(dosen)*, kolom *nim* di *data(mahasiswa)* dan *registrasi*, kolom *nim,th_akademik, semester* di tabel studi.
2. Penggunaan kunci tamu untuk integritas referensial, dalam tabel *data(dosen)* menggunakan kunci tamu pada kolom *nip* yang bergantung pada tabel *data(karyawan)* dengan relasi 1 ke 1, tabel *studi* memiliki *nim* sebagai kunci tamu yang bergantung pada tabel *registrasi* dimana kolom *nim* dalam tabel *registrasi* merupakan kunci tamu yang merujuk pada tabel *data (mahasiswa)*.
3. Pembatasan domain pada tipe data tertentu dan nilai tertentu dengan menggunakan contraint CHECK sebagai contoh bahwa pada kolom *semeter* di tabel *studi* dan *registrasi* hanya boleh berisi karakter 1 dan 2 saja.
4. Penggunakan constraint UNIQUE pada kolom dengan data yang harus bernilai unik seperti pada kolom *no_serdos* di tabel *data(dosen)* dan *no_daftar* di tabel *registrasi* juga digunakan untuk

*ISSN : 2302-3805*

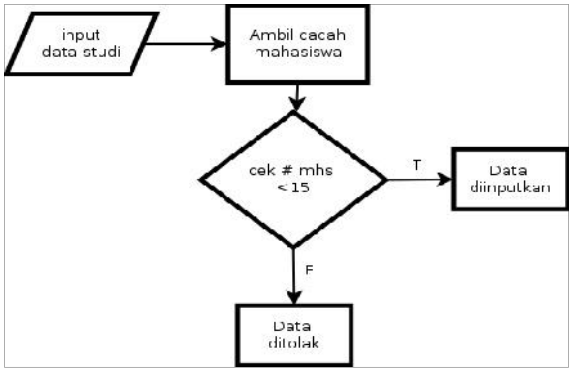
Dengan menggunakan metode-metode di atas maka data yang masuk ke dalam tabel-tabel di atas diharapkan akan valid. Setiap tabel akan berisi data yang terhindar dari duplikasi dan dapat diyakinkan untuk bisa mendapatkan tepat 1 *record* dengan adanya kunci primer. Kunci tamu menjaga bahwa tabel yang memiliki kunci tamu hanya akan dapat diisi jika di tabel yang dirujuk memiliki data yang sesuai. Tabel *data(dosen)* hanya akan dapat diisi dengan data yang ada di tabel *data(karyawan)* atau dapat dikatakan bahwa dosen merupakan bagian dari karyawan. Demikian juga dengan tabel *studi* hanya akan dapat diisi data jika terdapat data mahasiswa di tabel *registrasi*. Contoh perintah SQL untuk membuat tabel *studi* di atas adalah sebagai berikut :

```
CREATE TABLE studi (
nim CHAR(10) REFERENCES registrasi (nim),
th_akademik CHAR(9) NOT NULL,
semester    CHAR    CHECK    semester    IN
('1','2'), tgl_her date,
status CHAR DEFAULT '1' NOT NULL,
alamat_yogya  VARCHAR(150)  DEFAULT  'belum
lengkap' NOT NULL,
kodepos_yogya CHAR(5),
wali CHAR(10)  REFERENCES dosen(nip),
PRIMARY KEY
(nim,th_akademik,semester);
```

Pada saat mahasiswa baru tercatat maka data di tabel *registrasi*, *data(mahasiswa)* dan *studi* akan terisi. Kemudian apabila mahasiswa melanjutkan studi di semester berikutnya maka data mahasiswa tersebut akan terekam di tabel *studi.* Sehingga tabel *studi* digunakan untuk menyimpan data mahasiswa yang terdaftar dan diperbaharui setiap semester. Dalam tabel *studi* ini terdapat beberapa informasi yang dicatat yakni tanggal registrasi, alamat di yogyakarta dan siapa dosen wali dari mahasiswa tersebut dalam suatu tahun akademik dan semester tertentu.

Kolom *wali* merupakan kolom dengan kunci tamu yang merujuk pada tabel *data(dosen).* Kolom *wali* tersebut menunjukan siapa dosen wali dari seorang mahasiswa. Jika pada tabel *studi* ini diberikan aturan bahwa dalam satu semester pada tahun akademik tertentu 1 dosen maksimal hanya dapat menjadi dosen wali bagi 15 mahasiswa maka tabel yang dibuat dengan perintah SQL di atas belum dapat melakukan pembatasan banyaknya mahasiswa. Sehingga jika dilakukan pemasukan data dengan menggunakan perintah INSERT pada tabel *studi* maka data tersebut akan bisa dimasukan tanpa melihat dose walinya dalam 1 tahun akademik dan semester tertentu telah menjadi wali lebih dari 15 mahasiswa.

Untuk melakukan pembatasan maka dapat dilakukan pada program yang menjadi antarmuka ke basis data tersebut. Pembatasan dilakukan dapat dilakukan pada PHP, Visual Basic, Delphi dan lain sebagainya. Pada *form* input yang bertindak sebagai antar muka maka sebelum data tersebut dimasukan ke tabel *studi* terlebih dahulu dilakukan pengecekan apakah dosen wali dengan kode dosen yang bersesuaian dalam tahun akademik dan semester tertentu telah memiliki jumlah mahasiswa berjumlah 15 atau tidak. Jika cacah mahasiswa dengan perwalian dengan kode dosen tersebut telah berjumlah 15 atau lebih maka data yang akan dimasukan ditolak, seperti yang tertampil pada flowchart di gambar 2.




Gambar 2. Flowchart filter input data

Namun jika pembatasan dilakukan melalui program antarmuka maka metode tersebut memiliki kelemahan yakni, jika dilakukan pemasukan data melalui terminal (*console*) atau program bantu administrasi basis data server seperti *PHPMyadmin* untuk MySQL dan *PgAdmin* untuk PostgreSQL, maka data tersebut akan dapat dimasukan tanpa melalui program filter. Kesalahan pemograman pada program antarmuka yang dibuat dengan bahasa pemograman tertentu juga membuka peluang dilakukan SQL *Injection* sehingga data bisa dimanipulasi atau diinputkan dengan melewati pembatasan yang dilakukan oleh program.

Constraint CHECK yang sudah ditetapkan sebagai standar SQL dalam ANSI SQL-92 dapat digunakan untuk melakukan pembatasan untuk kasus di atas. Dengan menggunakan constaint CHECK yang dikombinasikan dengan bahasa prosedural yang ada dalam basis data server memungkinkan untuk dilakukan pencegahan masukan data sesuai dengan skenario di atas. Dalam PostgreSQL terdapat bahasa prosedural yang dikenal dengan nama PL/PgSQL yang secara *default* langsung dapat digunakan pada basis data yang dibuat. PL/PgSQL merupakan bahasa prosedural yang dapat digunakan untuk menjalankan suatu perintah, perhitungan, percabangan, perulangan seperti pada bahasa pemograman lainnya.

Langkah pertama yang dilakukan untuk melalukan filterisasi masukan seperti skenario di atas adalah dengan membuat fungsi dengan PL/PgSQL sesuai dengan ketentuan yang telah ditetapkan, kode sumber untuk membuat fungsi tersebut adalah seperti berikut,

```
CREATE OR REPLACE FUNCTION
```

*ISSN : 2302-3805*

```
CheckWali(vwali CHAR(10), ta CHAR(9), smt
CHAR)
RETURNS boolean AS $$
DECLARE
  jum INTEGER DEFAULT 0;
BEGIN
SELECT COUNT(wali) INTO jum FROM studi
WHERE th_akademik=ta and wali=vwali and
semester=smt;
    IF jum >= 15 THEN
    RETURN 'F';
  ELSE
    RETURN 'T';
  END IF;
END;
$$LANGUAGE plpgsql STRICT IMMUTABLE;
```

Fungsi di atas akan melakukan pengecekan apakah jumlah *record* pada tabel *studi* untuk wali, tahun ajaran dan semester tertentu berjumlah >=15. Jika lebih dari 15 maka fungsi tersebut menghasilkan nilai *False* dan jika tidak maka nilai *True* yang akan dihasilkan.

```
CREATE TABLE studi (
  nim CHAR(10) NOT NULL REFERENCES
  mahasiswa.data(nim),
  th_akademik CHAR(9) NOT NULL,
  semester CHAR(1) NOT NULL,
  tgl_her date,
  status CHAR(1),
  alamat_yogya VARCHAR(150) NOT NULL
  DEFAULT 'belum lengkap',
  kodepos_yogya CHAR(5),
  wali CHAR(10) DEFAULT NULL,
  PRIMARY KEY (nim , th_akademik ,
semester ),
 CONSTRAINT Cek_Dosen_Wali CHECK
 (CheckWali(wali,th_akademik,semester))
);
```

Pada perintah SQL di atas terlihat terdapat penambahan constraint CHECK yang memanggil fungsi *CheckWali* denga parameter *wali*,*th_akademik* dan *semester*. Constraint ini akan melakukan pengecekan terlebih dahulu terhadap data yang akan dimasukan ke dalam tabel *studi* jika data sesuai dengan kriteria fungsi atau bernilai *True* maka data tersebut akan dapat dimasukan dalam tabel. Sebaliknya jika data yang akan dimasukan bernilai *False* maka data tersebut akan ditolak. Karena pengecekan data terjadi pada tabel maka walaupun input data dilakukan melalui terminal ataupun program bantu basis data maka data akan tertolak jika tidak memenuhi syarat. Sebagai contoh data pada tabel 1 diambil kolom *nim*,*th_akademik*,*semester* dan *wali* dengan perintah SQL :

```
SELECT nim,th_akademik,semester,wali from
studi WHERE wali='030775579' AND
th_akademik='2012/2013' AND semester='1';
```

Tabel 1
Isi Data Tabel Studi

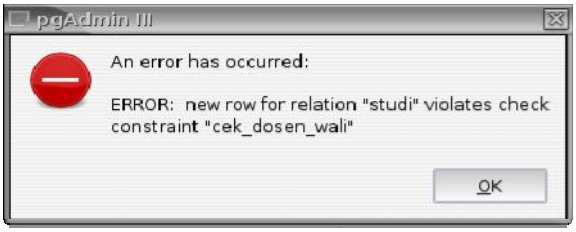
| nim | th_akademik | semester | wali |
|---|---|---|---|
| 121056026 | 2012/2013 | 1 | 030775579 |
| 121051027 | 2012/2013 | 1 | 030775579 |
| 121051028 | 2012/2013 | 1 | 030775579 |
| 121051025 | 2012/2013 | 1 | 030775579 |
| 121051001 | 2012/2013 | 1 | 030775579 |
| 121051002 | 2012/2013 | 1 | 030775579 |
| 121051003 | 2012/2013 | 1 | 030775579 |
| 121051004 | 2012/2013 | 1 | 030775579 |
| 121051005 | 2012/2013 | 1 | 030775579 |
| 121051006 | 2012/2013 | 1 | 030775579 |
| 121051007 | 2012/2013 | 1 | 030775579 |
| 121051008 | 2012/2013 | 1 | 030775579 |
| 121051009 | 2012/2013 | 1 | 030775579 |
| 121051010 | 2012/2013 | 1 | 030775579 |
| 121051011 | 2012/2013 | 1 | 030775579 |

Pada tabel 1 terlihat bahwa dosen wali dengan kode dosen 030775579 pada semester 1 tahun akademik 2012/2013 telah memiliki mahasiswa perwalian sebanyak 15 mahasiswa. Maka jika terdapat input data dengan dosen yang sama maka data tersebut akan ditolak, seperti perintah INSERT berikut,

```
INSERT INTO studi VALUES (
'121051012','2012/2013','1','2012-06-13',
'1', 'belum lengkap','00000','030775579');

ERROR:  new row for relation "studi"
violates check constraint "cek_dosen_wali"
```

Terlihat dalam perintah SQL di atas bahwa perintah INSERT akan ditolak dan menampilkan pesan kesalahan bahwa baris baru yang dimasukan melanggar constraint *cek_dosen_wali*. Hal yang sama juga terjadi jika coba dimasukan datanya melalui program bantu PgAdmin seperti terlihat pada gambar 3.




Gambar 3. Pesan Kesalahan di PgAdmin

Demikian juga akan terjadi penolakan oleh tabel jika terdapat data dalam tabel yang diubah (*update*) jika melanggar aturan yang sudah ditetapkan. Misalnya dalam tabel studi terdapat kode dosen 960372515 dan nilai tersebut akan diubah menjadi 030775579 dimana kode dosen tersebut diketahui telah memili jumlah perwalian sebanyak 15. Perintah SQL UPDATE untuk mengubah data seperti berikut,

```
UPDATE studi SET wali ='030775579' WHERE
```

*ISSN : 2302-3805*

```
nim='121051013' AND  semester='1' AND
th_akademik='2012/2013'                AND
wali='960372515';

ERROR:    new row for relation "studi"
violates check constraint "cek_dosen_wali"
```

Terlihat bahwa constraint CHECK yang memanggil fungsi PL/PgSQL dapat melindungi tabel dari ketidakvalidan data baik melalui perintah INSERT maupun UPDATE.

## 5. Kesimpulan dan Saran

Constraint CHECK yang digabungkan dengan pemograman PL/PgSQL dapat membuat tabel lebih terlindungi dari kemungkinan kesalahan dengan membuat aturan yang diimplementasikan dalam suatu fungsi. Penggunaan fungsi yang dituliskan dalam definisi tabel menjadikan tabel tersebut mampu menjaga input data maupun perubahan data sesuai dengan aturan yang dibuat. Pemberian pembatasan tersebut dapat membanntu menjaga integritas dari basis data tersebut.

Karena terdapat proses pengecekan data sebelum diinputkan dalam tabel dapat dipastikan bahwa terdapat proses yang lebih lama dibandingkan tanpa melalui filter. Sehingga metode filterisasi pada tabel dengan constraint CHECK dan PL/PgSQL ini akan tidak tepat digunakan untuk tabel yang cukup banyak menggunakan perintah INSERT dan UPDATE. Namun demikian apakah pertambahan waktu yang diambil oleh proses pengecekan ini signifikan atau tidak perlu penelitian lebih lanjut.

## Daftar Pustaka

[1] HALFOND, W., VIEGAS, J., AND ORSO, A. 2006. *A classification of sql-injection attacks and countermeasures*. In Proceedings of the IEEE International Symposium on Secure Software Engineering. IEEE, 65–81.

[2] MELTON, J. AND SIMON, A. 1993. *Understanding the New SQL: A Complete Guide*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers.

[3] RAHARJO, S. 2012. *Cosntranit 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).

[4] RAHARJO, S., SUTANTA , E., AND UTAMI , E. 2007. *Analisis aspek-aspek kualitas schema database (studi kasus pada database akademik ista yogyakarta)*. Seminar Nasional Teknologi 2007 (SNT 2007).

[5] ROB, P. AND CORONEL , C. 2009. *Database Systems: Design, Implementation, and Management*. Course Technology.

[6] SCHOLTE , T., BALZAROTTI , D., AND KIRDA , E. 2012. *Quo vadis? a study of the evolution of input validation vulnerabilities in web applications*. Financial Cryptography and Data Security, 284–298.

## Biodata Penulis

**Suwanto Raharjo**, Menyelesaikan Magister Komputer UGM Yogyakarta di tahun 2002. Saat ini sebagai Staf Pengajar di Jurusan Teknik Informatika IST AKPRIND Yogyakarta

*ISSN : 2302-3805*