

Implementasi Constraint CHECK dalam Aspek Kebenaran pada Basis Data di Aplikasi LaundryPOS

Implementation of CHECK Constraint in Correctness Aspects of LaundryPOS's Database Application

**Firstyani Imannisa Rahma¹, Tinuk Agustin²,
Ronaldus Morgan James³, Ema Utami⁴**

^{1,2,3,4}Magister Teknik Informatika, Universitas Amikom Yogyakarta

E-mail: *¹firstyani.rahma@students.amikom.ac.id, ²gusti.north@gmail.com,

³ronaldus.1188@students.amikom.ac.id, ⁴emma@nrar.net

Abstrak

Komponen penting yang dibutuhkan dalam sistem informasi atau perangkat lunak adalah basis data. Basis data membantu perangkat lunak dalam mengolah data yang datang dari input yang masuk ke dalam sistem. Untuk menjaga integritas dan keamanan data, programmer wajib memberikan fitur validasi data pada input. Validasi data dapat dilakukan dengan membuat batasan di tingkat aplikasi maupun di tingkat basis data. Sangat penting melakukan validasi data tingkat basis data tidak hanya pada tingkat pemrograman saja. LaundryPOS adalah aplikasi kasir berbasis mobile yang diperuntukkan untuk usaha laundry. Penelitian ini akan melakukan analisis keuntungan dari CHECK constraint di database pada aplikasi LaundryPOS dalam aspek kebenaran data. Pengujian dilakukan dengan menggunakan query dan kendala. Hasil dari pengujian ini membuktikan bahwa constraint CHECK mampu menjaga aspek kebenaran pada basis data aplikasi LaundryPOS dengan menyaring data input yang tidak sesuai dengan format yang ditentukan.

Kata Kunci—CHECK constraint, integritas data, validasi data, aspek kebenaran data, MySQL

Abstract

An important components in the information system or software is database. The database helps the software process data that comes from the input that enters the system. To maintain data integrity and security, programmers must provide data validation features on the input. Data validation can be done by creating constraints at the application level or at the database level. It is very important to do database level data validation not only at the programming level. LaundryPOS is a mobile-based cashier application intended for laundry businesses. This study will analyze the benefits of CHECK constraints in the database on the LaundryPOS in terms of data correctness. Tests carried out using the query and constraints. The results of this test demonstrate that CHECK constraint is able to maintain the Correctness Aspects of the LaundryPOS database by filtering input data that does not match the specified format.

Keywords—CHECK constraints, data integrity, data validation, aspek kebenaran data, MySQL

1. PENDAHULUAN

Basis data adalah kumpulan dari data yang tersimpan dan saling terkait satu dengan lainnya untuk melayani kebutuhan banyak pengguna [1]. Penyimpanan data dengan basis data dianggap lebih baik daripada penyimpanan data dengan *file* karena penyimpanan yang lebih besar, integrasi data yang membuat akses data semakin mudah dan redundansi data yang lebih rendah.

Salah satu peran penting basis data dalam sebuah sistem informasi atau aplikasi perangkat lunak adalah sebagai sumber informasi yang mampu memenuhi kebutuhan pengguna [2].

Sejak data relasional mendominasi *software* bisnis, perkembangan perancangan basis data mengalami kemajuan yang sangat pesat [3]. Banyak peneliti yang menjelaskan tentang proses perbaikan basis data untuk menjaga kualitas data [4] dan fokus pada kendala integritas [5]. Sayangnya, masih sedikit peneliti yang membahas lebih dalam tentang penggunaan *constraint* untuk mengendalikan data yang masuk ke dalam sistem [6][7].

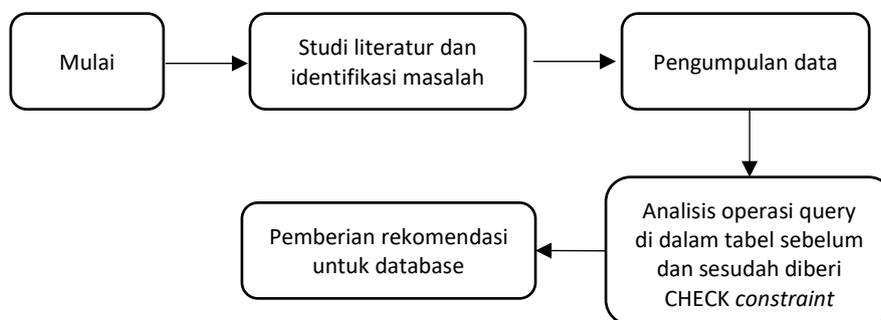
Constraint dapat digunakan pada semua RDBMS seperti pada MySQL, MYSQL SERVER, ORACLE, PostGreSQL. Penerapan *constraint* dalam tabel pada basis data dapat menjamin konsistensi dan integritas data karena data yang dapat masuk ke dalam tabel hanya data yang sesuai dengan batasan yang telah ditentukan [8]. Salah satu *constraint* yang dipakai dalam basis data adalah *constraint* CHECK.

Penelitian ini bertujuan untuk mengetahui pentingnya sebuah CHECK *constraint* dalam sebuah susunan basis data dan implementasinya khususnya pada aspek kebenaran data dengan menggunakan *database* dari aplikasi Laundry POS sebagai studi kasus permasalahan. Pengujian dilakukan dengan melakukan pengecekan *query* dan *constraint*. Penelitian ini diharapkan mampu menjaga aspek kebenaran data di dalam basis data dengan membatasi masuknya data input yang tidak sesuai dengan format yang ditetapkan.

2. METODE PENELITIAN

2.1. Tahap Review

Penelitian ini dilakukan dengan tahapan-tahapan yang terdiri dari identifikasi masalah, tinjauan literatur, pengumpulan data dan analisis operasi *query* di dalam tabel sebelum dan sesudah diberi CHECK *constraint* dan pemberian rekomendasi untuk *database*. Alur penelitian dapat dilihat pada Gambar 1.



Gambar 1. Alur Penelitian

2.2. Tinjauan Literatur

Penelitian oleh Raharjo membahas tentang penggunaan *constraint* CHECK untuk menyaring data yang masuk agar sesuai dengan data yang dibutuhkan [6]. Sementara itu, Utami melakukan penelitian tentang penggunaan *constraint* CHECK untuk membatasi masuknya data tanggal lahir, jenis kelamin, dan kode pelajar agar data tersebut menjadi valid [7]. Kesimpulan dari kedua penelitian ini adalah *constraint* CHECK dapat memudahkan proses validasi data yang dibuktikan dengan proses input 1000 data yang lebih cepat 1,2 hingga 5,3 milidetik dari pada tabel yang tidak memiliki *constraint* CHECK.

Selanjutnya, *constraint* CHECK juga digunakan di pengujian dalam perancangan *database* pada sistem asesmen dan pemetaan hasil *asesment* berbasis *tag* sebagai pembantu penyusunan strategi pembelajaran. [9] *Constraint* CHECK digunakan untuk menguji kualitas *database* fisik yang telah dirancang sebelumnya. Pengujian *constraint* dilakukan dengan

memasukkan query insert data pada setiap tabel di dalam database. Kelemahan dari penelitian ini adalah belum menjelaskan fokus aspek kualitas yang digunakan dalam pengujian Constraint CHECK dan hanya menggunakan query insert untuk mencari kesalahan dalam perancangan basis data.

2.3. Pengumpulan data

Proses pengumpulan data dilakukan dengan mengambil beberapa tabel dari skema database dari aplikasi *mobile* LaundryPOS sebagai sampel dari penelitian ini.

2.4. Analisis data

Pada proses analisis data query CHECK *constraint* ditambahkan ke dalam sebuah kolom di dalam tabel. Setelah penambahan *constraint* maka dilakukan pemetakan relasi tabel untuk mendapatkan rekomendasi basis data.

3. HASIL DAN PEMBAHASAN

Menurut Gordon B. Davis salah satu bagian penting dalam Sistem Informasi adalah masukan (*input*), yaitu berupa data yang akan disimpan sebagai basis data [10]. Kriteria perancangan basis data yang diperlukan untuk membuat database yang baik, antara lain 1) Setiap struktur tabel harus dibuat lebih efisien dan sistematis; 2) Penyimpanan data harus dibuat efisien; 3) Ukuran tabel harus dibuat sekecil mungkin untuk mempercepat operasi di basis data; 4) Mengoptimalkan redundansi untuk meningkatkan integritas data dan meminimalisir upaya penyebaran perubahan data dari satu tabel terkait ke tabel lainnya (catatan: dalam database relasional, redundansi data tidak dapat dihindari); 5) Ambiguitas data di semua tabel harus dihindari [11]. *Constraint* adalah aturan dalam basis data yang tidak boleh dilanggar karena berkaitan dengan kebenaran dan konsistensi data yang menghasilkan integritas basis data [12].

Constraint dalam *database* merupakan batasan nilai yang dapat memastikan hanya data yang sesuai dengan *constraint* saja yang dapat di *input* kan dalam tabel. Fungsi utama dari penerapan *constraint* adalah untuk menjamin integritas dan konsistensi data dalam tabel [13]. *Constraint* CHECK digunakan untuk meminimalkan redudansi dan meningkatkan integritas data [14]. Penelitian pada aplikasi Laundry POS ini bertujuan untuk membuktikan pentingnya penerapan *constraint* CHECK sebagai validasi server pada proses *insert* dan *update* data. Aplikasi Laundry POS sendiri terdiri dari 34 tabel, akan tetapi di sini penulis hanya mengambil 2 tabel saja yang akan digunakan sebagai contoh untuk dilakukan analisis terkait pentingnya penerapan *constraint* CHECK yaitu tabel admin dan tabel dompet.

Tabel admin sendiri terdapat 1 kolom yaitu email yang memiliki atribut varchar dengan panjang 100 karakter dengan *constraint* tidak boleh kosong dan bersifat *unique*. Permasalahan pada kolom ini adalah format data masukan email ini dapat berupa string biasa tanpa adanya pencegahan jika format masukan yang dimasukkan bukan berupa format email karena tipe data email di basis data tidak ada. Hal ini sebenarnya dapat dicegah melalui validasi dari sisi coding interface nya, akan tetapi tidak menutup kemungkinan *developer* lupa untuk menambahkan validasi masukan berupa format email. Hal lain yang dapat terjadi adalah lemahnya validasi dari sisi sistem yang dapat ditembus oleh penyerang dari luar sistem, sehingga data email tetap bisa masuk ke basis data jika tidak adanya lapisan/validasi dari sisi basis data. Contoh serangan yang mungkin terjadi adalah SQL *Injection* di mana kode SQL dimasukkan atau ditambahkan ke parameter input aplikasi yang kemudian diteruskan ke server SQL *backend* untuk proses parsing dan eksekusi [15].

Permasalahan selanjutnya terdapat di tabel dompet, di mana di tabel ini terdapat kolom nominal dengan tipe data integer dan memiliki *constraint* tidak boleh kosong dan digunakan untuk menampung jumlah nominal *top up* saldo. Berdasarkan proses bisnis dari aplikasi ini, kolom saldo

tidak boleh diisi dengan dengan nilai negatif, namun penulis akan mencoba memasukkan nilai negatif dan proses *insert* tetap berhasil dijalankan.

Berdasarkan dua permasalahan di atas, penulis akan memberikan validasi *constraint* CHECK untuk mencegah alamat email yang tidak sesuai dan nilai angka negatif agar tidak masuk kedalam basis data. Berikut adalah cara untuk membuktikan pentingnya penerapan *constraint* CHECK pada tabel admin dan dompet.

3.1. Tabel admin

Tabel admin adalah tabel yang digunakan untuk menyimpan data *owner* (pemilik laundry) dan *operator* (pegawai dari *owner* laundry) beserta data-data yang berkaitan seperti jumlah kios, saldo, hingga paket akun yang digunakan. Kolom di dalam tabel admin bisa dilihat di Gambar 2.

Column Name	Data Type
id	int(11)
nama	varchar(45)
email	varchar(45)
telp	varchar(45)
alamat	varchar(100)
password	varchar(50)
last_update	datetime
active_until	datetime
type	varchar(45)
id_owner	int(11)
activation_code	tinytext
fcm_token	varchar(255)
trash	int(11)
paket_akun_id	varchar(20)
jml_transaksi	int(11)
jml_kios	int(11)
pesan_antar	int(1)
saldo	int(12)
reveral	varchar(10)

Gambar 2. Tabel admin

Sebelum melakukan penambahan *constraint* CHECK, penulis akan melakukan pengujian dengan memasukkan data admin baru dengan format nilai email yang tidak benar.

```
INSERT INTO `admin` VALUES (116,'check constraint email','email
asal','08384949494','yogyakarta','1f32aa4c9a1d2ea010adcf2348166a
04','2019-10-25 02:46:16','2019-10-25
14:19:54','owner',0,'817236','fuQffvs-
6gw:APA91bGO93HbGXQkyY1D3hI19qH44b6syZja NzHj-
DklkDP2qh7JorhOVyahQfd6Wl7FBhs8rUNFngdXjDJ2pz8_4vpAZ-
g58pJvZr01I4z7lbU0dDYonQ_Q010Rt6g1WdsykKh9C2',1,'free',10,1,1,70
660,'8QZmle')
```



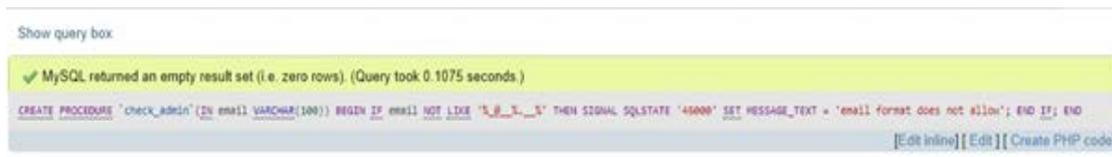
Gambar 3. hasil *insert* data admin sebelum diberikan *CHECK constraint*

Penulis melakukan proses *insert* data admin dengan nilai masukan email bertuliskan “email asal” seperti pada Gambar 3. Proses ini berhasil dijalankan dan data tersebut disimpan ke dalam basis data. Akibatnya, kebenaran data email di tabel ini menjadi tidak valid.

1.2. Penambahan Constraint CHECK

Penambahan *constraint* CHECK untuk versi MySQL 8.0.16 berbeda dengan MySQL dengan versi di bawah 8.0.16. Dalam penelitian ini, penulis menggunakan versi MySQL dibawah 8.0.16. Dalam versi ini, penulis perlu membuat suatu fungsi atau prosedur yang serupa dengan penggunaan *constraint* CHECK. Setelah itu, penulis membuat sebuah *trigger before insert* dan *before update* yang memanggil fungsi/prosedur yang telah dideklarasikan sebelumnya.

```
CREATE PROCEDURE `check_admin` (IN email VARCHAR(100)) BEGIN  
IF email NOT LIKE ('$@_$._$') THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'email format does not allow'; END IF; END  
g58pJvZr01I4z71bU0dDY0Nq_Q010Rt6g1WdsyKKh9C2',1,'free',10,1,1,  
70660,'8QZm1e')
```



Gambar 4. prosedur validasi format email

Prosedur di Gambar 4 akan mengecek nilai masukan email dari parameter email di mana jika nilai tidak mengandung format seperti “@” dan “.” seperti format email pada umumnya maka proses akan diinterupsi dan akan mengeluarkan pesan “*email format does not allow*”. Langkah selanjutnya dilakukan penambahan trigger pada proses *insert* dan *update* di tabel admin.

```
CREATE TRIGGER `admin_before_insert` BEFORE INSERT ON `admin`  
FOR EACH ROW BEGIN CALL check_admin(new.email); END  
  
CREATE TRIGGER `admin_before_update` BEFORE UPDATE ON `admin`  
FOR EACH ROW BEGIN CALL check_parts(new.email); END
```



Gambar 5. trigger *before insert* dan *update* tabel admin

Trigger yang ditampilkan di Gambar 5 akan dijalankan ketika ada proses simpan saat *insert* atau *update* data di tabel admin dengan memanggil fungsi atau prosedur untuk memvalidasi masukan nilai dari email yang telah dibuat. Setelah itu dicoba untuk memasukkan data admin dengan format email yang tidak benar.



Gambar 6. *insert* data admin setelah penambahan *constraint* CHECK

Setelah proses penambahan *constraint* CHECK dilakukan, penulis akan menguji penggunaan *constraint* CHECK dengan memasukkan kembali data admin baru dengan format

email yang tidak benar. Berdasarkan tampilan di Gambar 6, penulis dapat menyimpulkan bahwa perintah *Insert* dan *Update* nilai masukan email dengan format email yang tidak sesuai tidak dapat dilakukan lagi oleh database karena format yang dimasukkan tidak sesuai dengan yang format yang ditentukan di dalam prosedur.

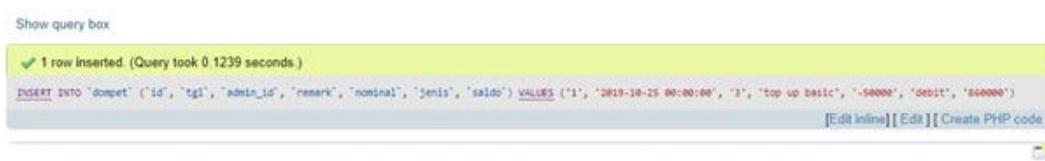
1.3. Tabel dompet

Tabel dompet adalah tabel yang digunakan untuk menyimpan data jumlah *top up* saldo untuk setiap masing-masing akun pengguna aplikasi LaundryPOS. Tabel ini berelasi dengan tabel admin yang bertipe *owner*. Permasalahan yang terdapat di tabel ini yang terkait dengan penggunaan *constraint* CHECK adalah kolom nominal yang digunakan untuk menyimpan jumlah besaran *top up* yang bisa dimasukkan nilai angka negatif. Kolom di tabel dompet dapat dilihat di Gambar 7.

Field	Type
id	int(11)
tgl	datetime
admin_id	int(11)
remark	varchar(45)
nominal	int(11)
jenis	enum('debit','credit')
saldo	float

Gambar 7. tabel dompet

```
INSERT INTO `dompet` (`id`, `tgl`, `admin_id`, `remark`,
`nominal`, `jenis`, `saldo`) VALUES ('1', '2019-10-25
00:00:00', '3', 'top up basic', '-50000', 'debit', '860000')
```

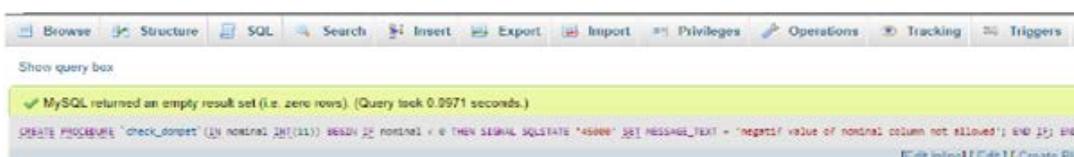


Gambar 8. *insert* data nominal bernilai negatif

Penulis mencoba untuk memasukkan nilai negatif pada kolom nominal bernilai -50000 dan proses *insert* sukses dijalankan. Berdasarkan proses bisnis, umumnya hal tersebut tidak diperbolehkan. Masuknya nominal bernilai negatif akan mengakibatkan kesalahan pada penyampaian informasi saldo yaitu setelah melakukan *top up*, total saldo dari *owner* yang melakukan *top up* justru malah berkurang -50000 seperti yang ditampilkan di Gambar 8.

1.4. Penambahan Constraint CHECK

```
CREATE PROCEDURE `check_dompet` (IN nominal INT(11)) BEGIN IF
nominal = 0 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'negatif value of nominal column not allowed'; END IF; END
```

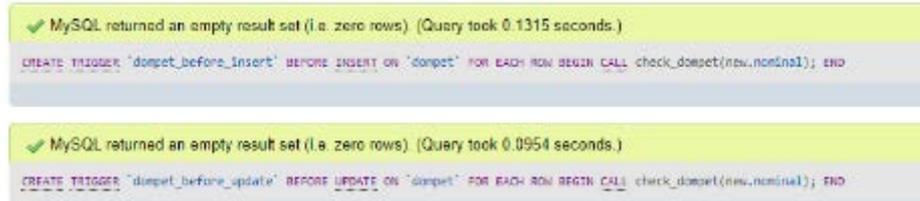


Gambar 9. prosedur validasi nilai negatif

Prosedur di Gambar 9 akan memeriksa parameter masukan variabel nominal jika nilai masukan adalah nilai negatif (nilai angka dibawah 0), maka proses *Insert* atau *Update* akan diinterupsi dengan pesan error “*negative value of nominal column not allowed*”.

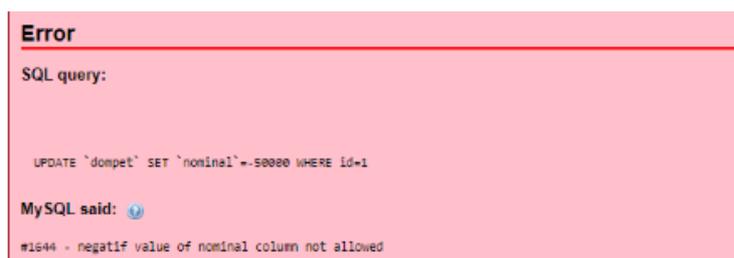
```
CREATE TRIGGER `dompet_before_insert` BEFORE INSERT ON `dompet`
FOR EACH ROW BEGIN CALL check_dompet(new.nominal); END

CREATE TRIGGER `dompet_before_update` BEFORE UPDATE ON `dompet`
FOR EACH ROW BEGIN CALL check_dompet(new.nominal); END
```



Gambar 10. Trigger *before insert* dan *update* tabel dompet

Dilakukan penambahan trigger pada proses *insert* dan *update* di tabel dompet. *Trigger* di Gambar 10 akan dijalankan ketika ada proses simpan saat *insert* atau *update* data di tabel dompet dengan memanggil fungsi atau prosedur untuk memvalidasi masukan nilai dari nominal yang telah dibuat.



Gambar 11. Trigger *before insert* dan *update* tabel dompet

Proses *update* data dompet dengan masukan kolom nominal bernilai negatif (-50000) tidak berhasil dilakukan setelah ditambahkan *CHECK constraint* seperti yang ditampilkan dalam Gambar 11. Dengan demikian, proses bisnis atau kebenaran data nominal menjadi valid.

Tabel 1. Perbandingan hasil operasi query

Sebelum Penambahan CHECK Constraint		Setelah Penambahan CHECK Constraint	
Operasi Query	Hasil	Operasi Query	Hasil
<i>Insert</i> alamat email tidak sesuai dengan format	sukses	<i>Insert</i> alamat email tidak sesuai format	gagal
<i>Update</i> alamat email tidak sesuai dengan format	sukses	<i>Update</i> alamat email tidak sesuai format	gagal
<i>Insert</i> alamat email sesuai format	sukses	<i>Insert</i> alamat email sesuai format	sukses
<i>Update</i> alamat email sesuai format	sukses	<i>Update</i> alamat email sesuai format	sukses
<i>Insert</i> nominal saldo dompet bernilai negatif	sukses	<i>Insert</i> nominal saldo dompet bernilai negatif	gagal
<i>Update</i> nominal saldo dompet bernilai negatif	sukses	<i>Update</i> nominal saldo dompet bernilai negatif	gagal
<i>Insert</i> nominal saldo dompet bernilai positif	sukses	<i>Insert</i> nominal saldo dompet bernilai positif	sukses
<i>Update</i> nominal saldo dompet bernilai positif	sukses	<i>Update</i> nominal saldo dompet bernilai positif	sukses

Berdasarkan perbandingan hasil operasi query *Insert* dan *Update* di Tabel 1, penulis menyimpulkan bahwa dengan adanya penambahan *CHECK constraint* pada kolom di sebuah tabel mampu mencegah input yang tidak sesuai kebutuhan masuk ke dalam sistem. Oleh karena itu, aspek kebenaran dari setiap kolom dalam sebuah tabel dapat terpenuhi.

4. KESIMPULAN

Berdasarkan pada penelitian yang dilakukan, penggunaan constraint *CHECK* pada basis data mampu menyaring data input yang masuk ke dalam tabel agar sesuai pada format yang telah ditentukan. Dalam constraint *CHECK* atau trigger dan prosedur yang serupa, pengguna dapat mengatur masuknya data sesuai dengan format yang dibutuhkan, dalam kasus ini digunakan untuk mengatur dan mengecek input atau *update* alamat email dan nominal saldo. Setelah diberi constraint *CHECK*, data yang tidak sesuai dengan format yang ditentukan akan ditolak dalam sistem. Oleh karena itu, penggunaan *CHECK* juga mampu menjaga basis data dalam segi aspek kebenaran data.

5. SARAN

Untuk menindaklanjuti beberapa kekurangan pada saat analisis, ada beberapa tindakan penyempurnaan sebagai berikut:

1. Penelitian ini hanya menganalisis penggunaan *CHECK constraint* terhadap aspek kualitas kebenaran data, sehingga untuk penelitian berikutnya diharapkan mampu menganalisis penggunaan *constraint* ini terhadap aspek kualitas lainnya.
2. Penelitian berikutnya mampu mengukur waktu pengoperasian query saat tabel diberikan *CHECK constraint* agar dapat mengukur efisiensi pemakaian basis data.

DAFTAR PUSTAKA

- [1] Teorey, T., Lightstone, S., Nadeau, T., Jagadish, H. V., 2006, Database Modeling and Design, Fourth Edition, Morgan Kaufmann Publishers, San Francisco.
 - [2] Raharjo, S., Sutanta, E., Utami, E., 2007, Analisis Aspek-Aspek Kualitas Schema Database (Studi Kasus pada Database Akademik ISTA Yogyakarta), Seminar Nasional Teknologi 2007, Yogyakarta, 24 November
 - [3] Toerey, J. T., 1999, Database Modelling & Design, San Francisco, Third Edition, Morgan Kaufmann Publishers, San Francisco.
 - [4] Lubis, J. H., 2017, Analisa Performansi Query Pada Database Smell, Jurnal Manajemen dan Informatika Pelita Nusantara, Vol. 21, No.1, hal. 42-49, ST
 - [5] Benedikt, M., Leblay, J., Tsamoura, E., 2015, Querying with Access Patterns and Integrity Constraints, 41st International Conference on Very Large Data Bases (VLDB), Kohala Coast, 31 Agustus - 4 September
 - [6] Raharjo, S., 2012, Constraint Basis Data Sebagai Fondasi Yang Kuat Dalam Pengembangan Sistem Informasi. Seminar Nasional Aplikasi Sains & Teknologi (SNAST) Periode III, Yogyakarta, 3 November
 - [7] Utami, E., 2014, The Advantages of Using *CHECK* Constraints in The Academic Database Tables, Journal of Software, No.2, Vol. 9, Hal. 382-388.
-

-
- [8] Arief, R., 2005, Pemrograman Basis Data dengan Transact-SQL menggunakan SQL Server, Penerbit Andi, Yogyakarta.
- [9] Kurnianti, A., Angguningtyas, A., Isnanda, R. G., 2017, Perancangan Database Pada Sistem Aessmen Dan Pemetaan Hasil Aessmen Berbasis Tag Sebagai Pembantu Penyusunan Strategi Pembelajaran, Semesta Teknika, Vol. 20, No. 2, Hal. 106–115.
- [10] Davis, G. B., 1984, Kerangka Dasar Sistem Informasi Manajemen, Terjemahan, PT Midas Surya Grafindo, Jakarta.
- [11] Date, C. J., 1995, An Introduction to Database System, Addison-Wesley, Reading.
- [12] Connolly, T., Begg, C., 2010, Database System: A Practical Approach to Design, Implementation, and Management - Fifth Edition, Pearson Education, Essex.
- [13] Arief, R., 2010, Implementasi Constraint untuk Menjamin Konsistensi dan Integritas Data Dalam Database, Jurnal DASI, Vol. 11, No.2, Hal. 62-71.
- [14] Yunita, S., Sholeha, E. W., Hardita, V. C., Utami, E., 2018, Database Modelling for the Expertise of Special Needs Teachers, Journal of Physics Conference Series, Vol. 1140, No. 1, Hal. 1-9.
- [15] Clarke, J., 2012, SQL Injection Attacks and Defense, Second Edition, Syngress, Rockland.
-