Normalisasi Tabel Pada Basisdata Relasional

Dwi Puspitasari [1], Cahya Rahmad [2], Mungki Astiningrum [3]

Politeknik Negeri Malang, Malang, Indonesia
[1] dwi_sti@yahoo.com

Abstrak

Normalisasi tabel merupakan sebuah teknik dalam *logical desain* sebuah basis data relasional yang mengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi). Pada ilmu basis data, normalisasi digunakan untuk menghindari terjadinya berbagai anomali data dan tidak konsistensinya data. Ini merupakan fungsi secara umum. Dalam beberapa kasus normalisasi ini sangat penting untuk menunjang kinerja basidata dan memastikan bahwa data dalam basisdata tersebut aman dan tidak terjadi kesalahan jika mendapat perintah SQL terutama DML yaitu *update*, *insert*, dan *delete*. Pada makalah memaparkan formula yang dapat digunakan untuk melakukan normalisasi pada tabel yang sudah diimplementasikan pada suatu RDBMS. Normalisasi yang dilakukan sampai pada bentuk normal 3 (3NF). Kami juga menyajikan hasil penerapan formula pada pemrograman basisdata yang digunakan untuk melakukan normalisasi padat tabel yang sudah diimplementasikan pada Relational Databasae Manajemen System (RDBMS) Microsoft SQL Server. Hasilnya dapat disimpulkan bahwa formula ini dapat digunakan untuk melakukan normalisasi tabel yang sudha diimplementasikan hingga pada bentuk 3NF

Kata kunci : *Normalisasi, Basisdata, Basisdata Relasional, Bentuk Normal,* Relational Databasae Manajemen System*, RDBMS*

1. Pendahuluan

Menurut Elmasri & Navathe (2013) basisdata adalah Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Basisdata juga dapat diartikan sebagai kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redundansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan, Date C.J. (2014). Basisdata mutlak dibutuhkan pada suatu sistem informasi, dimana data akan disimpan dan diolah untuk menjadi informasi-informasi yang penting bagi suatu perusahaan atau organisasi. Jenis basisdata yang masih sangat banyak digunakan adalah basisdata relasional. Dimana data disimpan didalam baris pada sebuah tabel yang memiliki relasi dengan tabel lain. Pada basisdata relasional terdapat banyak pengaturan yang dikenal dengan *primary key*, *foreign key*, *record*, *field*/ kolom/ *attibute*, *domain*/ *data type* yang menjaga agar data dapat diproses dengan cepat dan mudah, penghematan ruang menyimpanan, serta data yang dihasilkan akurat, Fathansyah (2012). Dengan perancangan yang bagus dan matang, maka manfaat basisdata tersebut dapat diperoleh. Akan tetapi untuk beberapa kasus tertentu dimana perancangan kurang matang, menghasilkan tabel tabel yang kurang baik, maka manfaat basisdata dapat diperoleh. Pada kasus seperti ini, dibutuhkan proses normalisasi basisdata.

Normalisasi tabel adalah proses pembentukan struktur basisdata relasional sehingga sebagian besar ambiguitas bisa dihilangkan. Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data relasional yang mengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi), Fathansyah (2012). Pada ilmu basis data, normalisasi digunakan untuk menghindari terjadinya berbagai anomali data dan tidak konsistensinya data. Ini merupakan funsi secara umum. Dalam beberapa kasus normalisasi ini sangat penting untuk menunjang kinerja basidata dan memastikan bahwa data dalam basisdata tersebut aman dan tidak terjadi kesalahan jika mendapat perintah SQL terutama DML yaitu update, insert, dan delete.

Adapun proses normalisasi adalah 1) Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke beberapa tingkat, kemudian 2) Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel tersebut perlu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal. Ada enam tahapan dalam normalisasi. Setiap tahapan memiliki banyak aturan yang harus dipenuhi, Elmasri & Navathe (2013). Proses normalisasi biasanya dilakukan secara manual pada saat proses perancangan, hasil proses normalisasi merupakan desain basisdata yang memiliki tabel-tabel yang normal. Desain basisdata inilah yang selanjutnya diimplementasikan pada suatu perangkat lunak pengelola basisdata yang dikenal dengan *Database Management System* (DBMS). Proses normalisasi juga bisa dilakukan untuk memeriksa apakah suatu tabel itu normal atau tidak.

Banyaknya tahapan dan aturan pada masing-masing tahapan pada proses normalisasi menyulitkan perancangan dan pengimplementasian basisdata pada suatu DBMS. Sehingga terkadang basis data yang sudah diimplemetasikan berisi tabel-tabel yang tidak normal dan perlu dinormalisasi.

Bahmani A. (2008) melakukan penelitian untuk melakukan normalisasi hingga bentuk 3NF menggunakan pendekatan graf untuk menggambarkan ketergantungan fungsional. Demba M. (2012) mengusulkan suatu algoritma untuk melakukan proses normalisasi basisdata dengan memperhatikan primary key dan kandidat key hingga bentuk normal 3 (3NF). Keduanya membutuhkan preproses untuk membentuk himpunan ketergantungan fungsional yang cukup sulit untuk dilakukan.

Penelitian ini mencoba menentukan formula yang dapat digunakan untuk melakukan normalisasi pada tabel yang sudah diimplementasikan pada suatu *Relational Database Management* System (RDBMS). Normalisasi yang dilakukan sampai pada bentuk normal 3 (3NF). Proses normalisasi menggunakan pendekatan konten yang tersimpan pada tabel yang akan dinormalisasi.

Bab 2 makalah ini menguraikan tentang proses normalisasi mulai dari tahapan, syarat-syarat yang harus dilakukan pada masing-masing tahapan untuk menghasilkan tabel yang normal. Bab 3 berisi formula yang dapat digunakan untuk melakukan normalisasi tabel yang sudah diimplementasikan pada suatu DBMS. Bab 4 berisi contoh penerapan formula normalisasi pada pemograman. Bab 5 berisi kesimpulan

2. Proses Dan Aturan Normalisasi Basisdata

Normalisasi adalah proses pembentukan struktur basis data sehingga sebagian besar ambiguity bisa dihilangkan. Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data relasional yang mengelompokkan atribut dari suatu tabel sehingga membentuk struktur tabel yang normal. Adapun kriteria tabel dikatakan normal adalah ketika tidak ada kerangkapan data (redudansi data).

Tujuan dari normalisasi adalah untuk :

- Untuk menghilangkan kerangkapan data sehingga meminimumkan pemakaian *storage* yang dipakai oleh *base relations* (file)
- Untuk mengurangi kompleksitas
- Untuk mempermudah pemodifikasian data

Contoh tabel yang tidak normal adalah sebagaimana berikut

Hari	Jam Ke	Kode MK	Nama MK	Kode Dosen	Nama Dosen	Kls	Ruang
Senin	1-3	EL230	Fisika 1	105	Prof. Bajuri	A, B	AMPI
Senin	4-5	EL230	Fisika 1	105	Prof. Bajuri	C, D	AMPI
Selasa	1-3	EL440	Pemrograman	102	Susilowati	A,B	R1
Selasa	4-6	EL440	Pemrograman	105	Prof. Bajuri	C,D	R2
Selasa	1-3	EL540	Pancasila	109	Timbul, PhD.	E,F	R1

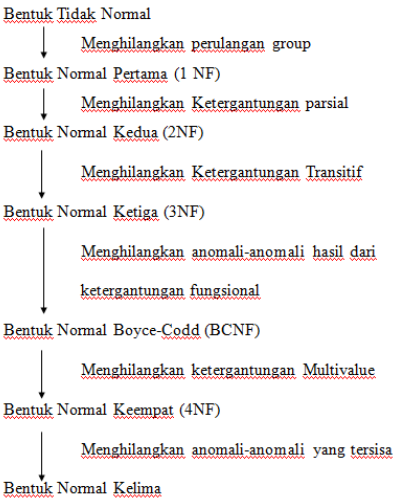
Gambar. 1 Contoh tabel yang tidak normal

Pada gambar diatas terdapat duplikasi data Nama MK untuk setiap data Kode MK serta data Nama Dosen untuk setiap data Kode Dosen. Hal ini menyebabkan pemborosan penyimpanan. Akibat yang lainnya adalah dapat menyebabkan inkonsisten data. Misalkan untuk data dengan Kode Dosen 105, data pertama untuk Nama Dosen dapat berisi Prof. Bajuri, sedangkan data kedua untuk Nama Dosen dapat berisi Profesor Bajuri. Secara kontekstual kedua data tersebut dianggap berbeda, padahal secara fakta sama

A. Proses Normalisasi

Gambaran proses normalisasi adalah 1) Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke beberapa tingkat, kemudian 2) Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel tersebut perlu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal. Untuk melakukan proses tersebut dibutuhkan beberapa tahapan.

Tahapan dalam normalisasi dimulai dari tahap paling ringan (1NF) hingga paling ketat (5NF). Biasanya hanya sampai pada tingkat 3NF atau BCNF karena sudah cukup memadai untuk menghasilkan tabel-tabel yang berkualitas baik. Urutan tahapan normalisasi tampak seperti gambar 2.



Gambar.2 Tahapan pada normalisasi

Adapun aturan dalam normalisasi adalah suatu tabel dikatakan baik (efisien) atau normal jika memenuhi 3 kriteria sbb:

1. Jika ada dekomposisi (penguraian) tabel, maka dekomposisinya harus dijamin aman (*Lossless-Join Decomposition*). Artinya, setelah tabel tersebut diuraikan / didekomposisi menjadi tabel-tabel baru, tabel-tabel baru tersebut bisa menghasilkan tabel semula dengan sama persis.
2. Terpeliharanya ketergantungan fungsional pada saat perubahan data (Dependency Preservation).
3. Tidak melanggar Boyce-Codd Normal Form (BCNF)

Jika kriteria ketiga (BCNF) tidak dapat terpenuhi, maka paling tidak tabel tersebut tidak melanggar Bentuk Normal tahap ketiga (3rd Normal Form / 3NF). Pada penelitian ini formula yang dibuat sampai memenuhi bentuk normal ke 3 yaitu 3NF.

B. Proses Normalisasi

Berdasarkan tahapan normalisasi, terdapat enam bentuk normal yaitu Elmasri & Navathe (2013):

1. Bentuk Normal Tahap Pertama (1st Normal Form / 1NF)
2. Bentuk Normal Tahap Kedua (2nd Normal Form / 2NF)
3. Bentuk Normal Tahap (3rd Normal Form / 3NF)

4. Boyce-Code Normal Form (BCNF)
5. Bentuk Normal Tahap (4th Normal Form / 4NF)
6. Bentuk Normal Tahap (5th Normal Form / 5NF)

Bentuk Normal Pertama / 1NF, memiliki aturan:

- ✓ Tidak adanya **atribut multi-value**, **atribut komposit** atau kombinasinya.
- ✓ Mendefinisikan atribut kunci.
- ✓ Setiap atribut dalam tabel tersebut harus bernilai *atomic* (tidak dapat dibagi-bagi lagi)

Contoh tabel mahasiswa yang tidak memenuhi syarat 1NF :

Nrp	nama	Hobi
12020001	Heri Susanto	Sepakbola, membaca komik, berenang
12020013	Siti Zulaiha	Memasak,mrogram komputer
12020015	Dini Susanti	Menjahit,membuat roti

atau

Nrp	nama	hobi1	hobi2	Hobi3
12020001	Heri Susanto	Sepak Bola	Membaca komik	berenang
12020013	Siti Zulaiha	Memasak	mrogram komputer	
12020015	Dini Susanti	Menjahit	membuat kue	

Maka harus didekomposisi menjadi dua tabel,yaitu tabel mahasiswa dan hobi :

- Tabel mahasiswa

Nrp	Nama
12020001	Heri Susanto
12020013	Siti Zulaiha
12020015	Dini Susanti

- Tebel Hobi

Nrp	Hobi
12020001	Sepakbola
12020001	membaca komik
12020001	Berenang
12020013	Memasak
12020013	mrogram komputer
12020015	Menjahit
12020015	membuat roti

Bentuk Normal Kedua / 2NF, memiliki aturan :

- ✓ Sudah memenuhi dalam bentuk normal kesatu (1NF)
- ✓ Semua atribut bukan kunci hanya boleh tergantung (functional dependency) pada atribut kunci
- ✓ Jika ada **ketergantungan parsial** maka atribut tersebut harus dipisah pada tabel yang lain
- ✓ Perlu ada tabel penghubung ataupun kehadiran foreign key bagi atribut-atribut yang telah dipisah tadi

Contoh tabel berikut memenuhi 1NF tapi tidak termasuk 2NF:

Mhs_nrp	mhs_nama	mhs_alamat	mk_kode	mk_nama	mk_sks	nihuruf

- ➢ Tidak memenuhi 2NF, karena {Mhs_nrp, mk_kode} yang dianggap sebagai primary key sedangkan:
  - {Mhs_nrp, mk_kode} $\not\rightarrow$ mhs_nama
  - {Mhs_nrp, mk_kode} $\not\rightarrow$ mhs_alamat
  - {Mhs_nrp, mk_kode} $\not\rightarrow$ mk_nama
  - {Mhs_nrp, mk_kode} $\not\rightarrow$ mk_sks
  - {Mhs_nrp, mk_kode} → nihuruf
- ➢ Tabel di atas perlu didekomposisi menjadi beberapa tabel yang memenuhi syarat 2NF yaitu berdasarkan Functional dependencynya sbb:
  - {Mhs_nrp, mk_kode}→ nihuruf (fd1)
  - Mhs_nrp → {mhs_nama, mhs_alamat} (fd2)
  - Mk_kode → {mk_nama, mk_sks} (fd3)
- ➢ Ada tiga tabel yang dihasilkan, yaitu :
  - fd1 (mhs_nrp, mk_kode, nihuruf) → Tabel Nilai
  - fd2 (Mhs_nrp, mhs_nama, mhs_alamat) → Tabel Mahasiswa
  - fd3 (mk_kode, mk_nama, mk_sks) → Tabel MataKuliah

Bentuk Normal Ketiga / 3NF, memiliki aturan :

- ✓ Sudah memenuhi dalam bentuk normal kedua (2NF)
- ✓ Tidak ada ketergantungan transitif (dimana atribut bukan kunci tergantung pada atribut bukan kunci lainnya)

Contoh tabel berikut memenuhi 2NF tapi tidak termasuk 3NF:

Mhs_nrp	mhs_nama	Alm_jalan	Alm_kota	Alm_propinsi	Alm_kodepos

- ➢ karena masih terdapat atribut *non primary key* (yakni **alm_kota** dan **alm_Provinsi**) yang memiliki ketergantungan terhadap atribut *non primary key* yang lain (yakni **alm_kodepos**):
  - alm_kodepos → {alm_Provinsi, alm_kota}
- ➢ Sehingga tabel tersebut perlu didekomposisi menjadi dua tabel, yaitu :
  - Mahasiswa (Nrp, nama, alm_jalan, alm_kodepos)
  - Kodepos (alm_kodepos, alm_provinsi, alm_kota)

3. Proses Normalisasi Pada Tabel Yang Sudah Diimplementasikan Pada Basidata

Idealnya proses normalisasi dilakukan pada saat perancangan basisdata untuk menghasilkan tabel-tabel yang normal yang selanjutnya diimplementasikan pada suatu DBMS. Akan tetapi pada beberapa kasus khusus dimana perancangan kurang matang yang menghasilkan tabel-tabel yang tidak normal pada hasil implementasi. Pada kasus tersebut perlu dilakukan normalisasi pada tabel-tabel yang sudah diimplementasikan. Untuk proses normalisasi dibutuhkan pengamatan pada data yang tersimpan dalam suatu tabel, oleh karna itu tabel yang akan dinormalisasi tidak boleh kosong. Proses normalisasi ini hanya bisa dilakukan untuk menormalisasi satu tabel, jika ada beberapa tabel yang akan dinormalisasi maka proses dilakukan pada masing-masing tabel. Untuk melakukan proses normalisasi pada tabel yang sudah diimplementasikan ini dibuatlah suatu formula yang nantinya dapat diterapkan pada pemrograman. Dikarenakan yang diproses adalah tabel yang ada pada basisdata, maka formula yang dibuat ini akan diterapkan pada pemrograman basisdata, yaitu menggunakan bahasa SQl.

Berdasarkan hasil pengamatan dan analisis yang penulis lakukan, tabel yang dikatakan memenuhi normal hingga bentuk 3NF jika memenuhi syarat-syarat berikut:

1. Tidak ada kolom yang merupakan *multivalue* dan *composite attribute*. *Composite attribute* biasanya data terdiri dari lebih dari satu domain data, misal atribut hari dan tanggal. Data yang merupakan *composite attribute* biasanya dipisahkan dengan tanda koma. Sedangkan *multivalue attribute* biasanya data terdiri dari 1 hingga lebih dari satu nilai pada domain yang sama yang biasanya dipisahkan dengan tanda koma.

2. Tidak memiliki kolom yang memiliki ketergantungan parsial dan transitif pada *primary key*. Jadi setiap kolom yang bukan primary key harus memiliki ketergantungan total pada *primary key*. Untuk tahap ini perlu diidentifikasi kolom yang merupakan *primary key*. Kolom yang menjadi *primary key* biasanya memiliki data dengan pola sama, seperti panjang karakter sama karena biasanya menggunakan tipe data dengan *fix lenght*, mengandung angka, dan tidak mengandung tanda koma. Selain itu ada kolom lain yang memiliki ketergantungan pada *primary key*. Hal ini ditandai dengan jumlah data yang unik pada *primary key* sama dengan jumlah data yang unik pada kolom lain yang memiliki ketergantungan padanya.

Untuk menuhi kedua syarat tersebut kami mengusulkan suatu formula yang bisa digunakan untuk melakukan normalisasi tabel yang sudah diimplementasikan. Dimulai dengan pembentukan tabel-tabel master untuk memenuhi syarat kedua. Tabel master merupakan tabel yang terdiri di kolom-kolom yang memiliki ketergantungan total pada satu primary key. Kemudian dilakukan pengecekan ketergantungan parsial dimana hal ini dimungkinkan jika ada *primary key* yang tergantung dengan *primary key* yang lain. Yang terakhir adalah melakukan pengecekan kolom yang merupakan *multivalued* dan *composite attribute*. Formula yang kami usulkan adalah sebagaimana berikut :

1. Langkah pertama adalah mengidentifikasi kolom yang memungkinkan menjadi *primary key*. Caranya pada setiap kolom :
   a. Dilakukan pengecekan data yang tersimpan pada kolom tersebut, jika ada data yang mengandung tanda koma, maka dia bukan kandidat *primary key*.
   b. Dilakukan pengukuran panjang data pada semua baris yang ada pada kolom tersebut. Jika semua data disemua baris memiliki panjang yang sama, maka kolom tersebut merupakan kandidat *primary key*.
2. Langkah kedua adalah melakukan pembentukan tabel master. Pada tahapan ini semua kolom yang bukan kandidat *primary key* dicek ketergantungannya dengan *primary key*. Caranya pada setiap kandidat primary key :
   a. Dihitung jumlah data yang unik → SELECT Count(DISTINCT(nama_kolom)) FROM nama_tabel
   b. Pada kolom lain selain kandidat *primary key* dihitung jumlah data yang unik dengan cara yang sama denga poin a.
   c. Bandingkan hasil dari poin a dan b. Jika hasilnya sama maka cek konsistensi data dengan cara :
      i. Untuk setiap nilai data unik dari kolom kandidat *primary key* cek apakah nilai dari kolom yang dicek memiliki nilai yang sama, jika iya maka kolom memiliki ketergantungan total dengan kandidat *primary key*, kemudian catat nama kolom. Jika tidak maka kolom tidak memiliki ketergantungan total dengan kandidat *primary key*.
      ii. Jika hasil sama kolom memiliki ketergantungan total dengan kandidat *primary key*.
   d. Jika tidak ada kolom yang memiliki ketergantungan total dengan kandidat *primary key*, maka kandidat batal menjadi *primary key*. Jika ada untuk setiap kandidat *primary key*, buat tabel baru yang merupakan tabel master yang memiliki kolom *primary key*, dan kolom lain yang memiliki ketergantungan fungsional. Kemudian menyalin data unik dari tabel asli ke tabel master. Terakhir menghapus kolom yang memiliki ketergantungan total pada *primary key* di tabel asli.
3. Langkah ketiga adalah melakukan pengecekan ketergantuang transitif antara *primary key* satu dengan yang lain. Caranya untuk masing-masing *primary key* cek ketergantungan transitif dengan *primary key* yang lain :
   a. Caranya untuk setiap nilai data unik dari kolom *primary key* cek apakah nilai dari kolom yang dicek (*primary key* lainnya) memiliki nilai yang sama jika iya maka kolom tersebut memiliki ketergantungan total dengan *primary key* pertama, kemudian catat nama kolom. Jika tidak maka kolom tidak memiliki ketergantungan total dengan *primary key* pertama.
   b. Atau membandingkan jumlah data dikolom kandidat primary key dengan jumlah data dikolom yang diukur yang digroupkan berdasarkan kolom tersebut → SELECT COUNT(NamaKolomKandidatPrimaryKey) FROM NamaTabel GROUP BY NamaKolomKandidatPrimaryKey ORDER BY NamaKolomKandidatPrimaryKey dibandingkan dengan SELECT COUNT(NamaKolomYangDiukur) FROM NamaTabel GROUP BY NamaKolomYangDiukur ORDER BY NamaKolomKandidatPrimaryKey. Jika hasil sama kolom memiliki ketergantungan total dengan kandidat *primary key*
   c. Untuk setiap *primary key* yang tergantung dengan *primary key* yang lain, buat kolom pada tabel master dari primary key yang diacu, masukkan nilai primary key yg mengacu sebagai foreign key. Kemudian salin nilai data sesuai dengan data asli.
   d. Hapus kolom *primary key* yang mengacu ke *primary key* lain pada tabel asli.
4. Langkah kelima mencari *composite* dan *multivalue* atribut pada semua tabel yang sudah dihasilkan

(tabel asli dan tabel master). *Composite* atribut biasanya data terdiri dari lebih dari satu domain data, misal atribut hari dan tanggal. Data yang merupakan *composite* atribut biasanya dipisahkan dengan tanda koma. Sedangkan *multivalue* atribut biasanya data terdiri dari 1 hingga lebih dari satu nilai yang biasanya dipisahkan dengan tanda koma. Maka cara untuk menentukan *composite* dan *multivalue* atribut adalah :

a. Untuk semua atribut yang bukan primary key, cari kolom yang semua datanya (yang bukan NULL) mengandung tanda koma, dan hitung tanda koma pada setiap baris data.

   i. Jika semua semua baris data memiliki tanda koma maka kolom tersebut merupakan *composite* atau *multivalue* atribut. Kemudian cek jumlah tanda koma

      a) Jika jumlah tanda koma semua baris dikolom tersebut sama maka kolom tersebut merupakan *composite* atribut. Maka buat kolom baru sebanyak jumlah koma + 1. Kemudian pecah nilai per tanda koma, dan simpan masing-masing pada kolom-kolom yang baru dibuat. Kemudian hapus kolom yang lama

      b) Jika jumlah tanda koma disemua baris tidak sama maka kolom tersebut merupakan *multivalue* atribut buat tabel baru dengan jumlah kolom sebanyak 2, kedua kolom ini menjadi primary key. Kemudian pecah nilai per tanda koma, dan simpan masing-masing tabel baru pada kolom kedua. Kolom pertama isi dengan nilai primary key dari kolom yang merupakan composite atribut yang tertaut dengan nilai yang dipecah.

      Untuk langah a) dan b) pemberian nama kolom dan tipe data untuk sementara dibebaskan bisa menggunakan nama kolom asli ditambah nilai urut dengan tipe data teks.

   ii. Jika tidak ada kolom yang semua barisnya memiliki tanda koma, berarti tidak ada kolom yang *composite* atau *multivalue* atribut.

Untuk mengilustrasikan jalannya formula yang diusulkan maka akan digambarkan pada studi kasus berikut.

Noproyek	NamaProyek	Nopegawai	NamaPegawai	Golongan	BesarGaji
NP001	BRR	Peg01	Anton	A	1.000.000
NP001	BRR	Peg02	Paula	B	900.000
NP001	BRR	Peg06	Koko	C	750.000
NP002	PEMDA	Peg01	Anton	A	1.000.000
NP002	PEMDA	Peg12	Sita	B	900.000
NP002	PEMDA	Peg14	Yusni	B	900.000

Gambar 3. Contoh tabel proyek_karyawan yan akan dinormalisasi

Langkah pertama didapatkan kolom "Noproyek", "Nopegawai", "Golongan" menjadi kandidat *primary key*. Kemudian pada **langkah kedua** dilakukan pengecekan ketergantungan kolom yang bukan kandidat primary key pada kandidat primary key. Yaitu :

"NamaProyek" pada "Noproyek", "NamaPegawai" pada "Noproyek", "BesarGaji" pada "Noproyek". Dihitung jumlah data unik masing-masing kolom. Jumlah data unik(NamaProyek)=2, jumlah data unik(NamaPegawai)=5, jumlah data unik(BesarGaji)=3, jumlah data unik(Noproyek)=2. Jumlah data unik(NamaProyek)=jumlah data unik(Noproyek). Maka dilakukan pengecekan konsistensi data dikolom "NamaProyek" dan "Noproyek". Untuk setiap data "NP001" dikolom "Noproyek" data di kolom "NamaProyek" selalu bernilai "BRR". Dan untuk setiap data "NP002" di kolom "Noproyek" selalu bernilai "PEMDA" di kolom "NamaProyek". Sehingga kolom "NamaProyek" memiliki ketergantungan fungsional total pada kolom "Noproyek". Langkah kedua dilakukan pada semua kandidat *primary key* yang lain hingga didapatkan kolom "NamaPegawai" memiliki ketergantungan fungsional total pada kolom "Nopegawai" dan kolom "BesarGaji" memiliki ketergantungan fungsional total pada kolom "Golongan". Hingga didapatkan tabel tabel sebagaimana berikut :

Noproyek	NamaProyek
NP001	BRR
NP002	PEMDA

Nopegawai	NamaPegawai
Peg01	Anton
Peg02	Paula
Peg06	Koko
Peg14	Yusni

Golongan	BesarGaji
A	1.000.000
B	900.000
C	750.000

Noproyek	Nopegawai	Golongan
NP001	Peg01	A
NP001	Peg02	B
NP001	Peg06	C
NP002	Peg01	A
NP002	Peg12	B
NP002	Peg14	B

Gambar.4 Tabel yang dihasilkan dari langkah kedua proses normalisasi

Selanjutnya pada **langkah ketiga** dilakukan pengecekan ketergantungan transitif antara kolom "Noproyek" dengan kolom "Nopegawai", kolom "Noproyek" dengan kolom "Golongan", kolom "Nopegawai" dengan kolom "Golongan", dan sebaliknya dari kolok kedua ke kolom pertama. Dari sini diperoleh kolom "Golongan" memiliki ketergantungan transitif dengan kolom "Nopegawai". Sehingga kolom "Golongan" dipindah di tabel master yang milik kolom "Nopegawai". Tabel yang dihasilkan menjadi :

Noproyek	NamaProyek
NP001	BRR
NP002	PEMDA

Nopegawai	NamaPegawai	Golongan
Peg01	Anton	A
Peg02	Paula	B
Peg06	Koko	C
Peg14	Yusni	B

Golongan	BesarGaji
A	1.000.000
B	900.000
C	750.000

Noproyek	Nopegawai
NP001	Peg01
NP001	Peg02
NP001	Peg06
NP002	Peg01
NP002	Peg12
NP002	Peg14

Gambar.5 Tabel yang dihasilkan dari langkah ketiga proses normalisasi

Selanjutnya **langkah kelima** dilakukan pengecekan *composite* dan *multivalue attribute* dengan cara melihat data yang mengandung tanda koma. Dikarenakan tidak ada data yang mengandung nilai koma, maka tabel yang dihasilkan tetap dan proses normalisasi selesai.

4. Penerapan Pada Bahasa Pemrograman

Untuk melakukan uji coba pada formula yang diusulkan, dibuatlah program sederhana dengan menggunakan pemrograman basisdata. Program berupa *store procedure* yang tersimpan pada RDBMS Microsoft SQL Server dan digunakan untuk melakukan normalisasi tabel dari basisdata yang sudah diimplementasikan pada Microsoft SQL Server. Hasil dari proses normalisasi pun kemudian disimpan pada basisdata yang sama.

Uji coba dilakukan pada tabel-tabel yang belum normal dengan berbagai macam kemungkinan. Dari hasil normalisasi dapat diketahui bahwa formula mampu melakukan normalisasi pada tabel yang sudah diimplementasikan pada RDBMS MS SQL Server hingga bentuk normal 3 (3NF).

5. Penutup

Pada makalah ini disajikan formula yang dapat digunakan untuk melakukan normalisasi tabel pada basisdata yang sudah diimplementasikan pada *Relasional Database Management System* (RDBMS). Uji coba dilakukan pada tabel yang diimplementasikan pada RDBMS Microsoft SQL Server. Hasil uji coba menunjukkan formula yang disajikan telah mampu melakukan normalisasi tabel hingga bentuk 3NF. Formula yang disajikan dapat digunakan untuk melakukan normalisasi tabel yang sudah diimplementasikan pada RDBMS yang lain.

6. Daftar Pustaka

Bahmani A., Naghibzadeh, M. and Bahmani, B., "Automatic database normalization and primary key generation", Niagara Falls Canada IEEE, 2008

Date C.J., "Pengenalan Sistem Basis Data jilid 1", PT.Indeks Group Gramedia, 2004

Elmasri & Navathe. "Fundamentals of Database Systems", Addison-Wesley, 2003

Demba M, "*An Algorithmic Approach to Database Normalization*", International Journal of Digital Information and Wireless Communications (IJDIWC) 3(2): 197-205, The Society of Digital Information and Wireless Communications, 2013 (ISSN: 2225-658X)

Fathansyah, *Basis Data*, Bandung: Informatika, 2012