IMPLEMENTASI *REFERENTIAL INTEGRITY CONSTRAINT* PADA MICROSOFT ACCESS DALAM UPAYA MEMELIHARA KONSISTENSI DATA

Oviliani Yenty Yuliana
Fakultas Ekonomi, Jurusan Akuntansi – Univeritas Kristen Petra
e-mail: ovi@peter.petra.ac.id

ABSTRAK: Data adalah asset perusahaan yang harus dijaga dan dipelihara. Data perusahaan dicatat pada tabel/*relation*. Struktur *relation* yang kurang baik dapat mengakibatkan *modification anomaly*, sehingga *relation* tersebut harus di-*normalisasi*. Modifikasi data pada *relation* yang sudah di-*normalisasi* akan menghadapi masalah *referential integrity constraint*. Masalah tersebut menyebabkan data tidak konsisten dan menghasilkan informasi yang salah. *Referential integrity constraint* tergantung pada minimum *relationship cardinality*. Penulis mengimplementasikan dan menguji coba *referential integrity constraint* pada Microsoft Access.

Kata kunci: *modification anomaly, normalization, referential integrity constraint, relationship cardinality constraint.*

ABSTRACT: *Data is business asset that should be safeguarded and maintained. Business data is recorded in a table/relation. Because inconeet relation structure could produce any modification anomaly, this relation should be normalized. Data modification in normalized relation will face referential integrity constraint problem. This problem could make inconsistent data and wrong information. Referential integrity constraint depends on minimum relationship cardinality. The trial and implementation of referential integrity constraint is taken on Microsoft Access.*

Keywords: modification anomaly, normalization, referential integrity constraint, relationship cardinality constrain.

1. PENDAHULUAN

Data merupakan salah satu asset perusahaan yang harus dijaga dan dipelihara, karena data merupakan sumber informasi bagi perusahaan untuk merencanakan, mengawasi, dan mengendalikan aktivitas perusahaan dalam rangka mencapai tujuan perusahaan. Data yang salah, tidak konsisten, dan tidak akurat dapat menyajikan informasi yang salah. Salah satu faktor penyebab kesalahan data adalah struktur *relation* yang tidak memenuhi aturan normalisasi.

Kroenke berpendapat: "*Unfortunately, not all relations are equally desirable. A table that meets the minimum definition of a relation may not have an effective or appropriate structure. For some relations, changing the data can have undesirable consequences, called* ***modification anomalies****. Anomalies can be eliminated by redefining the relation into two or more relations. In most circumstances, the redefined, or* ***normalized****, relations are preferred*." (2000:117-118)

Tabel 1. Contoh *Relation* Faktur Penjualan

Nomor Faktur	Tanggal Faktur	Kode Pelanggan	Nama Pelanggan	Alamat Pelanggan
F0001	4/15/97	P004	Nicholas	9020 N. W. 75 Street
F0002	4/18/97	P003	Neil	4215 South 81 Street
F0003	4/18/97	P006	Jeffrey	9522 S. W. 142 Street
F0004	4/20/97	P004	Nicholas	9020 N. W. 75 Street

Relation Faktur Penjualan pada Tabel 1 merupakan contoh *relation* yang tidak memenuhi aturan normalisasi, karena strukturnya tidak efektif. Penambahan atau penghapusan data pada *relation* tersebut mengakibatkan sesuatu yang tidak dikehendaki. Penghapusan baris faktur F0002 pada Tabel 1 tidak hanya menghilangkan data faktur F0002, tetapi secara tidak sengaja

juga akan menghilangkan data pelanggan Neil. Masalah tersebut disebut dengan *deletion anomaly*. Penambahan Pelanggan baru P011 tidak dapat dilakukan, sebelum ada faktur untuk pelanggan P011. Masalah tersebut disebut dengan *insertion anomaly*.

Masalah *deletion* dan *insertion anomaly* dapat dihindari dengan membagi *relation* FAKTUR PELANGGAN ke dalam dua *relation*, masing-masing untuk *relation* FAKTUR dan PELANGGAN. *Attribut* Nomor Faktur, Tanggal Faktur, dan Kode Pelanggan ditempatkan pada *relation* FAKTUR. Sedangkan Kode Pelanggan, Nama Pelanggan, dan Alamat Pelanggan ditempatkan pada *relation* PELANGGAN. Tabel 2 menunjukkan hasil normalisasi dari *relation* Tabel 1. Meskipun sudah terbagi ke dalam 2 *relation*, dengan bantuan *query* masih dapat disajikan informasi seperti pada Tabel 1.

Tabel 2a. *Relation* FAKTUR

Nomor Faktur	Tanggal Faktur	Kode Pelanggan
F0001	4/15/97	P004
F0002	4/18/97	P003
F0003	4/18/97	P006
F0004	4/20/97	P004

Tabel 2b. R*elation* PELANGGAN

Kode Pelanggan	Nama Pelanggan	Alamat Pelanggan
P004	Nicholas	9020 N. W. 75 Street
P003	Neil	4215 South 81 Street
P006	Jeffrey	9522 S. W. 142 Street

Penghapusan baris faktur F0002 pada Tabel 2A, tidak akan menghilangkan data pelanggan Neil. Begitu juga penambahan Pelanggan baru P011 dapat langsung ditambahkan pada Tabel 2B, tanpa harus menunggu pelanggan tersebut memiliki faktur. Sehingga *deletion* dan *insertion anomaly* dapat dihindari.

Namun demikian *relation* yang sudah memenuhi aturan normalisasi akan menghadapi masalah baru, yaitu *referential integrity constraint* (selanjutnya disingkat *constraint*), Jennings mendefinisikan *referential integrity* sebagai: *"Rules governing the relationships between primary keys and foreign keys of tables within a relational database that determine data consistency. Referential integrity requires that the values of every foreign key in every table be matched by the value of a primary key in another table. Microsoft access includes features for maintaining referential integrity, such as cascading updates and cascading deletion."* (1997:1195) Maksudnya *entity integrity* menuntut semua *primary key* dalam suatu tabel harus unik. *Referential integrity* menetapkan semua *foreign key* harus mempunyai hubungan dengan *primary key* pada tabel yang dihubungkan. Penambahan satu baris faktur baru pada tabel 2A akan bermasalah, jika kode pelanggan belum ada pada tabel 2B. Penghapusan satu baris pelanggan (misal pelanggan P004) yang memiliki faktur akan mengakibatkan masalah. Jika masalah-masalah tersebut tidak dihiraukan, penambahan faktur atau penghapusan pelanggan masih dapat dilakukan, dengan konsekuensi faktur tersebut tidak memiliki informasi nama dan alamat pelanggan, dapat dilihat pada Tabel 3.

Tabel 3. Hasil *Query* dari *Relation* FAKTUR dan PELANGGAN

Nomor Faktur	Tanggal Faktur	Kode Pelanggan	Nama Pelanggan	Alamat Pelanggan
F0001	4/15/97	P004		
F0002	4/18/97	P003	Neil	4215 South 81 Street
F0003	4/18/97	P006	Jeffrey	9522 S. W. 142 Street
F0004	4/20/97	P004		

Modifikasi data dapat ditolak dengan cara mendokumentasi *constraint* pada *schema design*. Implementasi *constraint* dapat didefinisikan pada DBMS, jika produk DBMS menyediakan fasilitas pemeriksaan *constraint*. Apabila produk DBMS tidak menyediakan fasilitas pemeriksaan *constraint*, maka pemeriksaan *constraint* didefinisikan pada program aplikasi.

2. TINJAUAN TEORITIS

2.1 Model *Entity Relationship* (Model E-R)

Model E-R yang diperkenalkan oleh Peter Chen pada tahun 1976 digunakan dan dikembangkan oleh Kroenke. Model E-R

didokumentasi dengan E-R diagram (ERD) (Kroenke,2000:59). Elemen-elemen ERD dapat dilihat pada Tabel 4. Kroenke berpendapat bahwa model E-R digunakan untuk mendokumentasi kebutuhan *user* dan kebijakan perusahaan pada saat merancang data base secara logis (2000:47). Pemodelan data merupakan aktivitas yang paling penting dalam pengembangan aplikasi data base. Model data yang salah akan mengakibatkan data terduplikasi, *anomaly* tidak dapat dihindari, dan data base sulit untuk digunakan atau dikembangkan. Romney berpendapat bahwa pemodelan data dilakukan pada tahap *Requirement Analysis* dan *Design* dalam proses perancangan database (2000:183).

Baik *entity* maupun *relationship* sama-sama dapat memiliki *attribute. Attribute* yang digunakan untuk mengidentifikasi *entity* disebut dengan *identifier*. *Identifier* bisa unik (contoh: NIP) dan tidak unik (contoh: Nama Pegawai). *Identifier* yang unik disebut dengan *primary key*. *Attribute primary key* diawali dengan "#" atau ditulis tebal, contoh: #NIP atau Nama Pegawai. Jika dalam *relation* tidak ada *attribute* yang dapat digunakan sebagai *identifier* yang unik, maka gabungan 2 *attribute* atau lebih dapat digunakan untuk membentuk *identifier* yang unik. Gabungan *attribute* disebut dengan *composite identifier*. Tetapi *composite identifier* dapat mengakibatkan perangkapan data yang berlebihan, untuk menghindari perangkapan dengan menambah 1 *attribute* bertipe data *autonumber*. *Attribute* yang dihasilkan disebut dengan *surrogate key* (Kroenke,2000:241-242).

Selain *key* yang disebutkan di atas, ada yang disebut dengan *foreign key*. *Foreign key* adalah *attribute* yang berasal dari *entity* lain, dimana pada *entity* lain *attribute* tersebut berperan sebagai *primary key*. *Attribute foreign key* ditulis dengan huruf miring atau diberi garis bawah putus-putus, contoh: *NIP* atau Nama Pegawai.

Kroenke mendefinisikan tipe *entity* khusus yang disebut dengan *weak entity*, digambarkan dengan kotak yang ujungnya bundar atau kotak ganda, yang digambarkan dengan cara tersebut bukan hanya *entity* saja melainkan juga *relationship*-nya. *Weak entity* adalah *entity* yang tidak boleh ada dalam data base, kecuali *entity* lain ada dalam data base. *Identifier weak entity* berasal dari *identifier entity* yang dihubungkan (sebagai *primary key* bukan sebagai *foreign key*).

Tabel 4. Elemen-elemen ERD

Nama Elemen	Simbol	Keterangan
Entity	NAMA *ENTITY*	• Kumpulan *person, place, object, event,* atau *concept* yang perlu dicatat dan disimpan datanya (Whitten et al 2000:260). • Nama *entity* ditulis menggunakan huruf besar semua. Nama *entity* dicantumkan dalam simbol.
Attribute/ Property	Nama Attribute	• Deskriptif sifat atau karakteristik *entity* (Whitten et al 2000:261). • Nama *attribute* ditulis menggunakan campuran huruf besar dan kecil. Huruf diawal kata menggunakan huruf besar. Nama *attribute* dicantumkan dalam simbol.
Relation-ship	N:M	• Hubungan antara satu *entity* dengan *entity* lain. *Relationship* mewakili peristiwa yang terjadi pada *entity* yang dihubungkan (Whitten et al 2000:264). • Dalam simbol tersebut dicantumkan maksimum *cardinality*.

Cardinality relationship menunjukkan jumlah baris suatu *entity* dalam *relationship* yang dapat dihubungkan dengan satu baris dari *entity* lain. *Cardinality* sering dinyatakan sebagai pasangan bilangan (X:Y). X menyatakan *minimum cardinality* dan Y menyatakan *maximum cardinality relationship*.

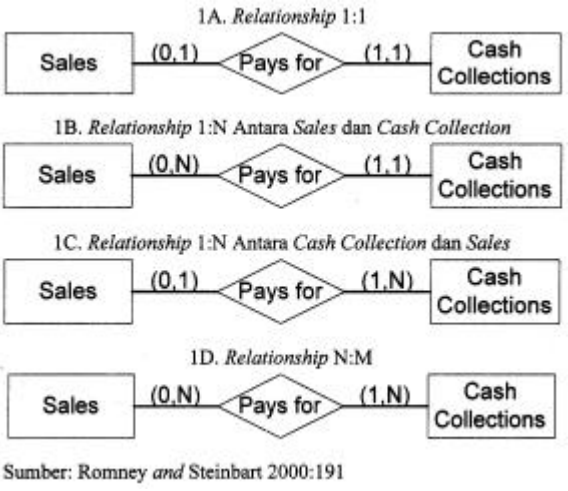
Minimum cardinality *relationship* menunjukkan jumlah baris yang paling sedikit dalam *relationship*. *Minimum cardinality* bisa 0 atau 1. *Minimum cardinality* 0 maksudnya setiap baris *entity* pada *relationship* tidak perlu dihubungkan ke

beberapa baris *entity* pada *relationship* lain. *Minimum cardinality* 1 menunjukkan bahwa setiap baris dari *entity* tersebut harus dihubungkan dengan paling sedikit satu baris dari *entity* lain.

Maximum cardinality *relationship* menunjukkan jumlah baris terbanyak dalam *relationship*. *Maximum cardinality* bisa 1 atau N, simbol tersebut menunjukkan setiap baris dalam tabel dapat dihubungkan dengan beberapa baris pada tabel lain. *Maximum cardinality* 1 menunjukkan bahwa satu baris dari *entity* dapat dihubungkan ke paling banyak satu baris dari *entity* lain. *Maximum cardinality* N menunjukkan bahwa satu baris dari *entity* dapat dihubungkan dengan lebih dari satu baris dari *entity* lain.

Tipe *relationship* tergantung pada *maximum cardinality* yang menghubungkan setiap *entity*, ada 3 tipe *relationship*:

1. *Relationship one-to-one* (1:1) pada saat *maximum cardinality* setiap *entity* adalah 1.
2. *Relationship one-to-many* (1:N) pada saat *maximum cardinality* dari satu *entity* adalah 1 dan *maximum cardinality* dari *entity* lain adalah N.
3. *Relationship many-to-many* (N:M) pada saat *maximum cardinality* kedua *entity* adalah N.



Sumber: Romney *and* Steinbart 2000:191

Gambar 1. Kemungkinan *Cardinality Relationship Sales-Cash Collection*

Gambar 1 memperlihatkan berbagai cara pemodelan *relationship* antara *sales* dan *cash collection event*. Gambar 1A menggambarkan *relationship* 1:1. Maksudnya setiap *sales event* (baris dalam tabel *sales*) dihubungkan ke paling banyak satu *cash collection event*. Hal tersebut menunjukkan suatu kebijaksanaan bahwa pelanggan tidak diijinkan untuk membayar secara angsur. Gambar 1A juga menunjukkan bahwa setiap *cash collection event* dihubungkan ke paling banyak satu *sales event*. Hal tersebut menunjukkan bahwa pelanggan harus membayar untuk setiap transaksi, tetapi tidak boleh secara sekaligus untuk beberapa transaksi.

Minimum cardinality relationship antara *sales* dan *cash collection event* dapat memodelkan penjualan tunai atau kredit. *Minimum cardinality* 1 di sebelah *sales event* menunjukkan bahwa semua penjualan adalah tunai. Pada Gambar 1A *minimum cardinality* di sebelah *sales event* adalah 0. Hal tersebut menunjukkan penjualan tunai, tetapi diberi kebijaksanaan untuk penjualan kredit (penjualan yang dilakukan mungkin tidak dihubungkan ke *cash collection event*).

Gambar 1B dan 1C menunjukkan cara untuk menyatakan *relationship* 1:N. Gambar 1B menunjukkan bahwa setiap *sales event* dihubungkan ke banyak *cash collections event*, tetapi setiap *cash collection event* dihubungkan ke paling banyak satu *sales event*. Hal tersebut menunjukkan kemungkinan pelanggan membayar secara angsur (tetapi bisa juga membayar secara tunai), tetapi setiap *sales event* harus dibayar masing-masing, tidak boleh secara sekaligus. Sedangkan Gambar 1C, menunjukkan bahwa setiap *sales event* dapat dihubungkan dengan paling banyak satu *cash collection event*, tetapi setiap *cash collection event* mungkin dihubungkan ke banyak *sales event* berbeda. Hal tersebut menunjukkan kebijaksanaan pelanggan untuk membayar secara sekaligus pada akhir bulan untuk semua pembelian yang dilakukan selama bulan tersebut, tetapi tidak diijinkan untuk pembayaran secara angsur.

Gambar 1D menunjukkan *relationship* M:N antara *sales* dan *cash collections events.* Setiap *sales event* mungkin dihubungkan ke satu atau lebih *cash collections events*, dan setiap *cash collection event* mungkin dihubungkan ke satu atau lebih *sales event*. Hal tersebut menunjukkan keadaan dimana perusahaan melakukan beberapa penjualan tunai dengan pem-

bayaran angsur, dan juga mengijinkan pelanggan membayar lebih dari satu transaksi penjualan secara sekaligus.

2.2 Normalisasi

Tidak semua *relation* sudah dalam keadaan layak. *Relation* yang dibentuk tanpa memperhatikan ketentuan normalisasi akan menghasilkan *relation* yang tidak efektif atau struktur *relation* yang tidak layak. Jika data pada *relation* tersebut diubah, akan mengakibatkan sesuatu yang tidak diharapkan, disebut *modification anomalies*. *Anomali* dapat dibatasi dengan membentuk kembali *relation* ke dalam 2 atau lebih *relation* dengan memperhatikan aturan normalisasi, sehingga *relation* yang terbentuk diharapkan lebih baik.

Aturan normalisasi menurut Kroenke, adalah sebagai berikut:

A. Bentuk normal pertama disebut dengan ***Repetitive*.**
   Ketentuan agar *relation* memenuhi bentuk normal pertama adalah:
   1. Kolom dalam tabel harus mempunyai nilai tunggal, tidak boleh merupakan grup atau array.
   2. Setiap isi data dalam 1 kolom harus memiliki tipe data yang sama.
   3. Setiap kolom harus mempunyai nama yang unik, tetapi urutan kolom dalam tabel tidak penting.
   4. Tidak boleh ada 2 baris dalam tabel yang identik, urutan baris dalam tabel tidak penting.

   Meskipun *relation* sudah memenuhi bentuk normal pertama, kemungkinan masih dapat terjadi *anomali update,* untuk itu perlu diperiksa ketentuan bentuk normal kedua.

B. Bentuk normal kedua disebut dengan ***Partial Dependency***
   Ketentuan bentuk normal keduan adalah *attribute non key* harus tergantung sepenuhnya pada *key attribute. Relation* yang memiliki *composite key* yang harus diperiksa ketentuan *partial dependency*-nya.

C. Bentuk normal ketiga disebut dengan ***Transitive Dependency***
   Relation dalam bentuk normal kedua, kemungkinan masih ditemukan *anomaly*. Ketentuan bentuk normal ketiga adalah tidak boleh ada *non key attribute* yang tergantung pada *non key attribute* lainnya.

D. ***Boyce-Codd normal form***
   Ketentuan *Boyce-Codd normal form* diberlakukan jika *relation* memiliki *key* lebih dari satu (*candidate key*). Pada bentuk normal ini memilih salah satu dari *candidate key* menjadi *primary key*.

E. Bentuk normal keempat disebut dengan ***Multivalued Dependencies***
   Relation harus memenuhi bentuk normal keempat, jika *relation* memiliki paling sedikit 3 *attribute,* dua di antaranya adalah *multivalued* (tergantung pada ketiga atau lebih *attribute*).

2.3 *Referential Integrity*

Entity integrity menuntut semua *primary key* dalam *relation* harus unik. Microsoft Access menyediakan 2 metode untuk menjaga *entity integrity*:

A. *Key field* dengan tipe data *autonumber* dapat menciptakan nilai unik.
B. *Index* pada *field primary key* dengan *property No Duplicates.* (Jennings, 1997:833)

Sedangkan *referential integrity* menetapkan bahwa semua *foreign key* harus mempunyai hubungan dengan *primary key* pada *relation* yang dihubungkan. Ketentuan ini untuk mencegah transaksi:

A. Penambahan 1 *record* pada tabel sisi N dari *relationship* 1:N tanpa keberadaan *record* yang bersesuaian pada tabel sisi 1.
B. Penghapusan 1 *record* pada tabel sisi 1 dari *relationship* 1:N tanpa menghapus terlebih dahulu semua *record* yang bersesuaian pada tabel sisi N.
C. Penghapusan atau penambahan 1 *record* pada tabel sisi 1 dari *relationship* 1:1 dengan tanpa menghapus atau menambah 1 *record* bersesuaian pada tabel yang dihubungkan.
D. Pengubahan nilai *field primary key* dimana masih ada *record* yang tergantung padanya.

E. Pengubahan nilai *field foreign key* dimana tidak ditemukan pada *field primary key* table yang digantunginya. (Jennings,1997:834)

2.3.1 *Relationship Cardinality Constraint*

Masalah *constraint* yang dilanggar dapat mengakibatkan *fragment*/*orphans*. Baris yang tidak sepatutnya ada tanpa satu *parent*/*child* disebut *fragment*, sedangkan baris *child* yang ada tanpa 1 *mandatory parent* disebut dengan *orphans*. DBMS atau aplikasi program harus mencegah terjadinya *fragment*/*orphans*. Cara pencegahan *fragment*/*orphans* tergantung pada *cardinality relationship* minimum. Terdapat empat *constraint* pada *relationship cardinality*: *mandatory to mandatory* (M-M), *mandatory to optional* (M-O), *optional to mandatory* (O-M), dan *optional to optional* (O-O).

2.3.2 Batasan-batasan untuk *Update* Baris *Parent*

Aturan-aturan untuk mencegah *fragment* saat *insert*, *modify*, dan *delete* baris *parent* dengan *constraint* M-M, M-O, O-M, dan O-O ditunjukkan pada Tabel 5. Pada *constraint* M-M, satu baris *parent* baru dapat disisipkan, hanya jika paling sedikit satu baris *child* dibuat pada waktu yang bersamaan dengan penyisipan *parent* atau mengubah satu *child* dari *parent* lain (tindakan yang terakhir dapat menyebabkan pelanggaran *constraint*). Perubahan *key parent* diijinkan pada *relationship* M-M, hanya jika nilai *foreign key* yang berhubungan dengan baris-baris *child* juga ikut diubah. (Dimungkinkan untuk menentukan kembali semua *child* ke *parent* lain, kemudian membuat paling sedikit satu *child* baru untuk *parent* tersebut, tetapi kemungkinan ini jarang dilakukan). Catatan jika *surrogate key* digunakan, tindakan tersebut tidak pernah terjadi. Satu *parent* dari *relationship* M-M dapat dihapus sepanjang semua *child* dihapus atau ditentukan kembali.

Dalam *constraint* M-O, satu baris *parent* baru dapat ditambahkan tanpa syarat, karena *parent* tidak perlu memiliki *child*. Perubahan pada nilai *key parent* diijinkan, hanya jika nilai yang berhubungan dengan nilai pada baris-baris *child* juga diubah. Pada *relationship* dengan *constraint* M-O, baris *parent* dapat dihapus, hanya jika semua *child* dihapus atau semua *child* diubah ke *parent* lain.

Tabel 5. Aturan-Aturan Untuk *Update* Baris *Parent*

		Proposed Action on Parent		
		Insert	***Modify (key)***	***Delete***
Type of Relationship	**M-M**	***Create at least one child***	***Change matching keys of all children***	***Delete all children OR Reassign all children***
	M-O	***OK***	***Change matching keys of all children***	***Delete all children OR Reassign all children***
	O-M	***Insert new child OR Appropriate child exists***	***Change key of at least one child OR Appropriate child exists***	***OK***
	O-O	***OK***	***OK***	***OK***

Sumber: Kroenke, 2000:261

Pada *constraint* O-M, satu *parent* dapat disisipkan, hanya jika paling sedikit satu *child* ditambahkan pada saat yang sama atau jika satu *child* sudah ada. Dengan cara yang sama, *key parent* dari *relationship* O-M mungkin diubah, hanya jika satu *child* dibuat atau jika satu baris *child* yang sesuai sudah ada. Tidak ada penolakan penghapusan baris *parent* pada *relationship* O-M.

Tipe *relationship constraint* terakhir adalah O-O. Pada *constraint* O-O tidak ada prasyarat, sehingga tidak ada penolakan saat satu baris diubah, dihapus, atau ditambahkan.

2.3.3 Batasan-batasan untuk *Update* Baris *Child*

Aturan-aturan untuk mencegah *orphans* saat memperbaharui baris *child* ditunjukkan pada Tabel 6. Satu perbedaan khusus, baris-baris *child* dapat dimodifikasi atau dihapus sepanjang terdapat *sibling*. Masudnya *child* terakhir tidak pernah meninggalkan *relation*. Tanpa mempertimbangkan *sibling*, aturan-aturan untuk menghindari *fragment* saat pemrosesan baris-baris *child* adalah mirip dengan *parent*.

Tabel 6. Aturan-Aturan untuk *Update* Baris *Child*

		Proposed Action on Child		
		Insert	***Modify (key)***	***Delete***
Type of Relationship	**M-M**	***Parent exists OR Create parent***	***Parent with new value exists (or create one) AND Sibling exists***	***Sibling exists***
	M-O	***Parent exists OR Create parent***	***Parent with new value exists OR Create parent***	***OK***
	O-M	***OK***	***Sibling exists***	***Sibling exists***
	O-O	***OK***	***OK***	***OK***

Sumber: Kroenke,2000:261

3. RANCANGAN MODEL E-R DAN *RELATIONAL*

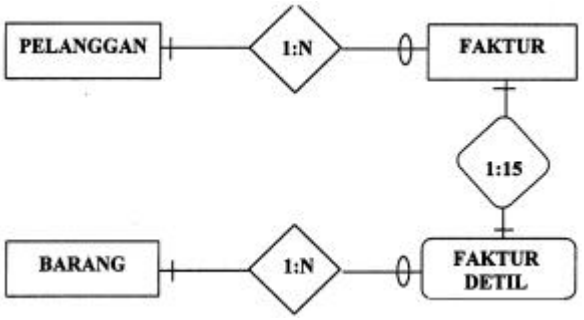
Perusahaan yang akan dimodelkan dalam tulisan ini adalah perusahaan dagang yang memiliki beberapa orang pelanggan. Setiap aktivitas penjualan dengan pelanggan dicatat pada faktur. Seorang pelanggan dapat melakukan pembelian lebih dari satu kali. Pelanggan yang sudah tidak aktif lagi tetap dicatat oleh perusahaan. Pelanggan yang masih memiliki faktur tidak bisa dihapus. Pada faktur harus selalu dicantumkan satu data pelanggan, karena satu faktur untuk seorang pelanggan.

Faktur boleh dibuat kalau ada minimal satu item yang dibeli oleh pelanggan. Kebijaksanaan tersebut dibuat untuk menghindari penyalah gunaan faktur. Jumlah baris maksimal pada satu lembar faktur adalah 15. Faktur detil hanya untuk menuliskan item dari nomor faktur yang bersangkutan. Keberadaan faktur detil tergantung pada faktur, maksudnya jika faktur dihapus, maka seluruh faktur detil juga harus dihapus.

Setiap faktur detil hanya boleh mencantumkan satu item barang. Satu item barang dapat dicantumkan pada beberapa faktur detil, ada kemungkinan item barang tidak tercantum pada faktur detil. Data barang tidak boleh dihapus, kalau masih ada faktur detil yang mencantumkan barang tersebut. Model E-R dapat dilihat pada Gambar 2.

4. IMPLEMENTASI DAN UJI COBA *CARDINALITY CONSTRAINT*

Penulis mengimplementasikan rancangan model *relation* Gambar 2B dengan salah produk DBMS, Microsoft Access. Pendefinisian *cardinality constraint* dalam Microsoft Access ditentukan pada saat mendefinisikan *relationship*, dapat dilihat pada Gambar 3A, dengan mengaktifkan menu *bar Tools* dan memilih *Relationships*. Sebaiknya pendefinisian *cardinality constraint* dilakukan terlebih dahulu, sebelum tabel diisi data. Jika tabel sudah berisi data, Microsoft Access akan memeriksa terlebih dahulu apakah *orphans* ditemukan. Kalau *orphans* ditemukan, maka Microsoft Access akan menolak pendefinisian *cardinality constraint* dengan menampilkan informasi seperti pada Gambar 3B. Sebagai contoh *select query* pada Gambar 3C dapat digunakan untuk mencari baris pada tabel FAKTUR yang mengalami *orphans*. Solusinya bisa menambah informasi pelanggan pada tabel PELANGGAN atau menghapus baris *orphans* pada tabel FAKTUR, baru kemudian didefinisikan kembali *cardinality constraint*.



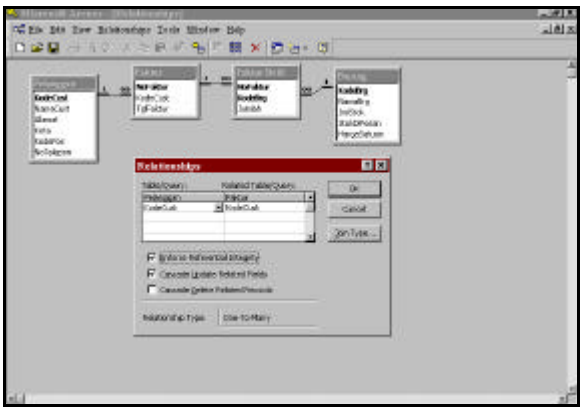
Gambar 2a. Model *Entity Relationship*

PELANGGAN(KodeCust, NamaCust, Alamat, Kota, KodePos, NoTelepon)
BARANG(KodeBrg, NamaBrg, JmlStok, StokDiPesan, HargaSatuan)
FAKTUR(NoFaktur, *KodeCust*, TglFaktur)
FAKTUR DETIL(*NoFaktur*, *KodeBrg*, Jumlah)

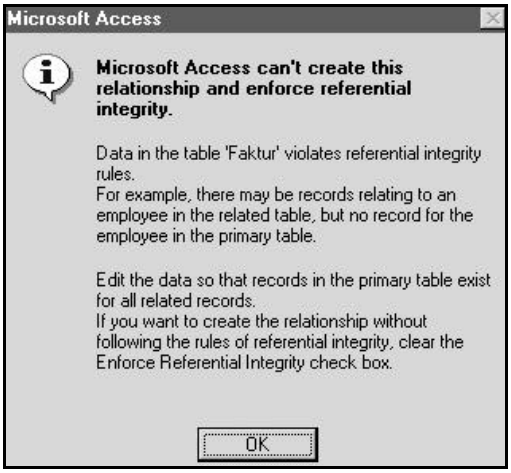
Gambar 2b. Model *Relation*

Microsoft Access tidak akan mempertahankan *referential integrity relation*, jika kotak *Enforce Referential Integrity* tidak diberi tanda. Penambahan, penghapusan, dan pengubahan baris *parent* dan *child* diijinkan tanpa pemeriksaan. Hal tersebut digunakan untuk mengimplementasikan *constraint* O-O dan O-M dari Tabel 5 dan 6.

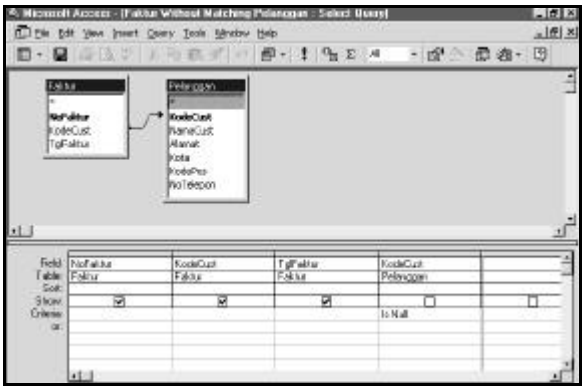
Microsoft Access hanya menampilkan *window confirmation* Gambar 4 saat menghapus baris.



Gambar 3a. Pendefinisian *Cardinality Constraint* pada Microsoft Access



Gambar 3b. Penolakan Pendefinisian *Cardinality Constraint*



Gambar 3c. *Select Query* untuk Menampilkan Faktur yang Kehilangan Data Pelanggan

Microsoft Access akan mempertahankan *referential integrity relation*, jika kotak *Enforce Referential Integrity* diberi tanda. Hal tersebut digunakan untuk mengimplementasikan *constraint* M-O dan M-M dari Tabel 5 dan 6. Jika kotak *Enforce Referential Integrity* diberi tanda, maka kotak *Cascade Update Related Fileds* dan *Cascade Delete Related Records* dapat tidak diberi tanda, diberi tanda salah satu, atau diberi tanda semua, tergantung kebijaksanaan perusahaan yang akan diterapkan. Hasil uji coba untuk setiap kemungkinan tersebut dapat dilihat pada Tabel 7.

Tabel 7. Hasil Uji Coba *Enforce Referential Integrity* Microsoft Access

Relation	Penyisipan	Perubahan (*primary key*)	Penghapusan
Parent	Penyisipan dapat dilakukan tanpa syarat.	Perubahan bisa dilakukan, hanya jika *relation* tidak memiliki *child*.	Penghapusan bisa dilakukan, jika *relation* tidak memiliki *child*.
		# Perubahan tidak bisa dilakukan, jika *relation* memiliki *child*.	* Penghapusan tidak bisa dilakukan, jika *relation* memiliki *child*.
		## Perubahan dilakukan pada *primary key parent* dan seluruh *foreign key child* yang terkait.	** Penghapusan dilakukan pada baris *parent* dan seluruh baris *child* yang terkait
Child	Penyisipan bisa dilakukan, hanya jika *foreign key* baris yang disisipkan terdapat pada *parent*.	Perubahan dapat dilakukan, jika *foreign key* pengganti ditemukan pada *parent*.	penghapusan dapat dilakukan tanpa syarat.

Keterangan tabel:
Kotak *Cascade Update Related Fileds*: # → tidak diberi tanda dan ## → diberi tanda.
Kotak *Cascade Delete Related Records*: * → tidak diberi tanda dan ** → diberi tanda

Gambar 4. Konfirmasi Penghapusan *Record*

Untuk mempertahankan *referential integrity constraint* pada rancangan model

relational, penulis mengusulkan kotak *Enforce Referential Integrity* dan *Cascade Update Related Fileds* diberi tanda untuk keseluruhan *relationship*. Karena *relationship* antara *relation* PELANGGAN dengan FAKTUR dan *relationship* antara *relation* BARANG dengan FAKTUR DETIL *relationship constraint*-nya M-O, maka kotak *Cascade Delete Related Records* diusulkan untuk tidak diberi tanda. Sedangkan *relationship* antara *relation* FAKTUR dengan FAKTUR DETIL *relationship constraint*-nya M-M, maka kotak *Cascade Delete Related Records* diusulkan untuk diberi tanda, sehingga penghapusan satu baris faktur pada *relation parent* juga akan menghapus seluruh baris *child* yang terkait. Penentuan *Cascade Update Related Fileds* dan *Cascade Delete Related Records* untuk rancangan model *relational* Gambar 2B dapat dilihat pada Tabel 8.

Tabel 8. *Cascade Update* dan *Delete* untuk Rancangan Model *Relational*

***Relationship* antara**	***Enforce Referential Integrity***	
	Cascade Update Related Fileds	***Cascade Delete Related Records***
PELANGGAN-FAKTUR	√	
FAKTUR-FAKTUR DETIL	√	√
BARANG-FAKTUR DETIL	√	

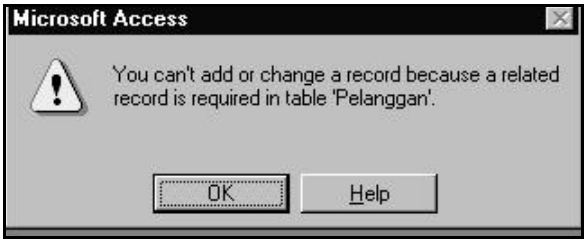
Microsoft Access mengijinkan penyisipan baris pada *relation parent* tanpa syarat, merupakan implementasi *constraint* M-O. Sedangkan penyisipan baris pada *relation parent* untuk *constraint* M-M harus dibuat program aplikasi, sebagai contoh dapat dilihat pada Gambar 5. Pada saat *form* kehilangan fokus, program aplikasi akan memeriksa apakah *parent* tersebut memiliki *child*, kalau tidak memiliki *child* program memanpilkan *window* informasi, kemudian baris *parent* dihapus. Penyisipan baris pada *relation child* dapat dilakukan hanya jika *foreign key* baris yang disisipkan terdapat pada *relation parent*. Sedangkan jika *foreign key* baris yang disisipkan tidak terdapat pada *relation parent*, maka Microsoft Access akan menampilkan *window information* yang dapat dilihat pada Gambar 6.

Gambar 5a. *Form* Faktur Penjualan untuk Mengimplementasikan *Constraint* M-M

```
Private Sub Form_LostFocus()
  Dim dbs As Database
  Dim tdf As TableDef
  Dim rst As Recordset
  Dim Cari As String

  Cari = Forms!Faktur!NoFaktur
  Set dbs = CurrentDb
  Set tdf = dbs.TableDefs![Faktur Detil]
  Set rst = dbs.OpenRecordset("Faktur Detil")
  rst.Index = tdf.Indexes(2).Name
  rst.Seek "=", Cari
  If rst.NoMatch Then
    MsgBox "Untuk menerapkan aturan M-M CARDINALITY" & Chr$(13) & Chr$(10) & _
      "Faktur Detil tidak boleh kosong ! Karena " & Chr$(13) & Chr$(10) & _
      "Faktur Detil kosong, maka Faktur akan dihapus !.", _
      vbExclamation, "Peringatan"
    Set tdf = dbs.TableDefs![Faktur]
    Set rst = dbs.OpenRecordset("Faktur")
    rst.Index = tdf.Indexes(2).Name
    rst.Seek "=", Cari
    rst.Delete
  End If
  rst.Close
  Set dbs = Nothing
End Sub
```

Gambar 5b. Program Aplikasi untuk Mengimplementasikan *Constraint* M-M



Gambar 6. Informasi Penambahan atau Pengubahan Data Harus Berhubungan dengan Tabel *Parent*

Pengubahan *primary key relation parent* bisa dilakukan, hanya jika *relation parent* tidak memiliki *child*. Pengubahan *primary key relation parent* tidak bisa dilakukan, jika *relation parent* memiliki *child* dan kotak *Cascade Update Related Fileds* tidak diberi tanda. Microsoft Access akan menampilkan *window information* yang ditunjukkan pada Gambar 7. Sedangkan jika *relation parent* memiliki *child* dan kotak *Cascade Update*

Related Fileds diberi tanda, maka perubahan dilakukan pada *primary key relation parent* dan seluruh *foreign key relation child* terkait.

Gambar 7. Informasi Penghapusan dan Perubahan Tidak Dapat Dilakukan

Penghapusan baris *relation parent* dapat dilakukan, jika *relation* tidak memiliki *child*. Jika *relation parent* memiliki *child* dan kotak *Cascade Delete Related Records* tidak diberi tanda, maka penghapusan tidak dapat dilakukan. Microsoft Access akan menampilkan *window information* yang ditunjukkan pada Gambar 7. Sedangkan jika *relation parent* memiliki *child* dan kotak *Cascade Delete Related Records* diberi tanda, maka Microsoft Access akan menampilkan *window confirmation* yang ditunjukkan pada Gambar 8. Jika dipilih *Yes*, maka penghapusan dilakukan pada baris *parent* dan seluruh baris *child* terkait. Penghapusan baris pada *relation child* dilakukan tanpa syarat, Microsoft Access hanya dapat mengimplementasikan *constraint* M-O. Sedangkan *constraint* M-M harus dibuatkan program aplikasi, program aplikasi pada Gambar 5B dapat digunakan dengan menghapus baris pada tabel *parent*. Program tersebut dapat dikembangkan, agar dapat mempertahankan minimal satu baris pada tabel *child*.

Gambar 8. Konfirmasi Penghapusan Baris *Parent* dan Seluruh Baris *Child* Terkait

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil implementasi dan uji coba *cardinality constraint* pada Microsoft Access, dapat disimpulkan:

- Normalisasi menciptakan *entity integrity* dan *referential integrity.* Namun DBMS dan atau aplikasi program yang harus memelihara *integrity* selama proses modifikasi data, kelalaian dalam memelihara *integrity* data base dapat mengakibatkan kesalahan data.
- Penulis berpendapat bahwa dalam *relationship* O-M dan O-O orphan tidak dapat dihindari.
- *Referential integrity constraint* Microsoft Access dapat digunakan untuk mendefinisikan dan mengimplementasikan kebijaksanaan perusahaan dalam hal minimum *cardinality constraint*. Dengan memberi tanda pada kotak *Enforce Referential Integrity* konsistesi data dapat dipelihara.
- *Referential integrity constraint* Microsoft Access dapat digunakan untuk menerapkan ketentuan-ketentuan *relationship* M-O dan M-M dari Tabel 5 dan 6. Beberapa ketentuan *relationship* M-M yang tidak dapat ditangani oleh Microsoft Access harus diimplementasikan dengan program aplikasi.
- Perubahan *primary key* pada *relation parent* tidak bisa dilakukan, jika *relation* memiliki *child* dan kotak *Cascade Update Related Fields* tidak diberi tanda. Sedangkan jika kotak diberi tanda, perubahan dilakukan pada *primary key parent* dan seluruh *foreign key child* yang terkait.
- Penghapusan baris pada *relation parent* tidak bisa dilakukan, jika *relation* memiliki *child* dan kotak *Cascade Delete Related Records* tidak diberi tanda. Sedangkan jika kotak diberi tanda, penghapusan dilakukan pada baris *parent* dan seluruh baris *child* yang terkait

5.2 Saran

Agar *referential integrity constraint* benar-benar dapat terjaga, maka aturan-aturan pada Tabel 5 dan 6 yang tidak dapat

diimplementasikan dalam DBMS, harus diimplementasikan dalam program aplikasi.

DAFTAR PUSTAKA

1. Flaatten, Per O, et. al. *Foundation of Business Systems.* Second Edition. United States of America: Dryden Press. 1992.
2. Grauer, Robert T. *and* Maryann Barber. *Exploring Microsoft Access 97.* New Jersey: Prentice Hall. 1998.
3. Jennings, Roger *and* Matthew Harris. *Special Edition Using Microsoft Access 97.* United States of America: Que Corporation. 1997.
4. Kroenke, David M,. *Database Processing: Fundamentals, Design and Implementation.* Seventh Edition. New Jersey: Prentice Hall. 2000.
5. Romney, Marsal B. and Paul John Steinbart. *Accounting Information System*. Eight Edition. New Jersey: Prentice Hall. 2000.
6. Whitten, Jeffrey L. et. al. *Systems Analysis And Design Methods.* Fifth Edition. New York: Irwin/McGraw-Hill. 2000.