PENDEKATAN MODEL REA DALAM PERANCANGAN DATABASE SISTEM INFORMASI AKUNTANSI SIKLUS PENDAPATAN

Oviliani Yenty Yuliana
Dosen Fakultas Ekonomi, Jurusan Akuntansi – Universitas Kristen Petra

ABSTRAK

Database yang memenuhi aturan normalisasi diperlukan untuk menunjang Sistem Informasi Akuntansi (SIA) terkomputerisasi. Alat yang biasa digunakan untuk merancang database adalah Entity Relationship Model (Model E-R). Namun aturan penggambaran diagram tidak begitu jelas, sehingga mempersulit perancang data untuk membentuk database yang memenuhi aturan normalisasi. Model REA merupakan pengembangan dari Model E-R. Model REA menerapkan prinsip give-to-get, sehingga mempermudah pembentukan model data.

Dalam tulisan ini dibahas Logical dan Physical View data, schema, Model REA, menyusun diagram REA, tahap-tahap perancangan database dan peran serta akuntan, serta cara mengimplementasikan Model REA ke database relasional, khususnya pada siklus pendapatan.

Kata kunci: Database, Model E-R, Model REA, SIA, Siklus Pendapatan

ABSTRACT

Normalization concept in database is necessary in support in the computerized Accounting Information Systems (AIS). Entity Relationship Model (E-R Model) is usually used to design database as a common tool. However the rule of drawing E-R Model diagram is not so clear, therefore it make difficulty for data designer to construct normalization database. REA Model is a further development of E-R Model. REA Model using give-to-get principle that makes more easily to construct data model.

This paper discuss the logical and physical view data, schema, REA Model, how to construct REA diagram, database design stages and how accountant participate in database design, as well as how to implement REA model into relational database specifically for revenue cycle

Keywords: Database, E-R Model, REA Model, AIS, Revenue Cycle

1. PENDAHULUAN

Kemajuan teknologi komputer dan informasi berdampak pada cara pencatatan akuntansi tradisional, dimana penyajian informasi keuangan dari SIA manual yang

berdasarkan *historical cost*, dengan adanya teknologi komputer, maka informasi keuangan dapat disajikan berdasarkan *current replacemenrt cost* dan *market value*. SIA yang terkomputerisasi memungkinkan pemakai laporan keuangan dapat melihat laporan keuangan setiap saat secara cepat, akurat, dan benar. Dengan bantuan komputer, data yang dicatat bukan hanya data keuangan saja, melainkan data lain seperti: data pelanggan dan penjualan. Data *non*-keuangan dapat dianalisis untuk menghasilkan informasi *non*-keuangan yang dapat digunakan untuk mengambil keputusan strategik dalam mencapai tujuan perusahaan (Santosa 1999:2).

SIA terkomputerisasi dapat menyajikan informasi keuangan dan *non*-keuangan dengan mudah karena didukung oleh database. Dengan adanya database, maka data dapat terintegrasi, duplikasi dapat dikurangi, format data tidak tergantung pada aplikasi program, memudahkan pemakai data, menyajikan informasi dengan bantuan bahasa *query* (Kroenke 2000:13-14). Dalam rangka mengurangi duplikasi/pengulangan data, ada indikasi untuk meninggalkan *model double-entry bookeeping* (Romney 2000:161). Hal tersebut merupakan tantangan bagi akuntan untuk memahami database lebih jauh.

Whitten (2000:133-173) berpendapat bahwa *Joint Project Planning* (JPP) dan *Joint Requirements Planning* (JRP) merupakan strategi yang paling efektif dan tercepat dalam merancang sistem. JPP merupakan strategi dimana semua *stakeholders project* (*system owners*, *users*, *analysts*, *designers* dan *builders*) berpartisipasi dalam ruang kerja *project management*. Dari JPP dapat ditentukan: lingkup projek, rencana kerja, sumber, dan anggaran. Sedangkan JRP merupakan teknik yang menggunakan ruang kerja untuk mempertemukan *system owners*, *users*, *analysts*, *designers* dan *builders* untuk bersama-sama menganalisis sistem.

Whitten (1994:43) mengelompokkan *building block-people* menjadi beberapa *system users*, salah satunya adalah *technical and professional staff*. Orang yang berada pada kelompok *technical and professional staff* memiliki keahlian khusus, misalnya akuntan. Dengan demikian akuntan berperan dalam perencanaan di atas. Akuntan tidak hanya berperan dalam perencanaan, tetapi berperan pada keseluruhan perancangan database. Akuntan berperan dalam perancangan *conceptual*, *external*, dan *internal-level schema*. Oleh sebab itu akuntan harus memiliki pengetahuan sistem database yang baik, sehingga dapat berpartisipasi dalam merancang SIA terkomputerisasi. Partisipasi akuntan yang utama adalah menjamin bahwa pengawasan memadai diterapkan dalam sistem database, guna menjaga data. Akuntan juga harus memberi keyakinan bahwa informasi yang dihasilkan dapat dipercaya.

Berdasarkan pengamatan terhadap mahasiswa yang menempuh mata kuliah Database Manajemen Sistem, yang penulis asuh sejak tahun 1996 di Jurusan Akuntansi Universitas Kristen Petra, mahasiswa mengalami kesulitan dalam menggambarkan Diagram *Entity-Relationship* (ERD). *Entity* apa yang harus disertakan, *attribute* apa yang harus dicantumkan pada masing-masing *Entity*. Umumnya mahasiswa menggambarkan ERD seperti pada Gambar 1(A), sebenarnya penggambaran tersebut tidak terlalu salah, walaupun cukup sulit untuk menghasilkan database yang memenuhi normalisasi (Yuliana 2001:37). Penulis harus mengilustrasi seperti pada Gambar 1(B), sehingga mahasiswa dapat menggambarkan ERD seperti pada Gambar 1(C). Karena mahasiswa akuntansi sudah memahami siklus-siklus akuntansi, maka ada cara pendekatan lain, yaitu Diagram REA. Pada tulisan ini diagram REA diterapkan hanya pada siklus pendapatan.

2. PEMBAHASAN

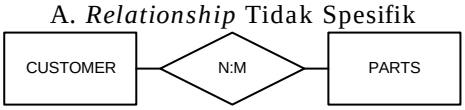
2.1 Database

2.1.1 *Logical* dan *Physical View* Data

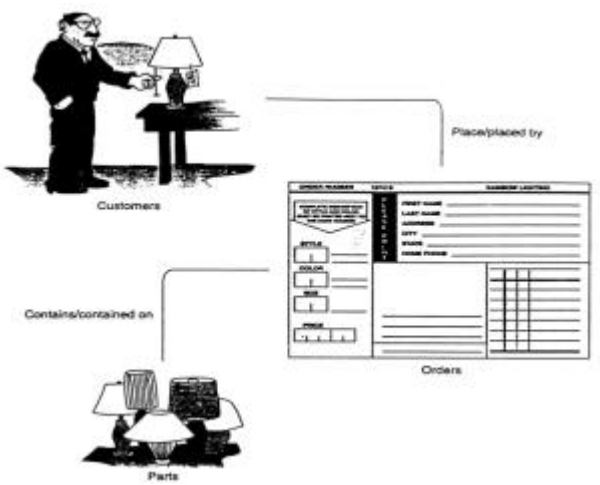
Logical View menunjukkan bagaimana pemakai dan pemrogram mengatur dan memahami data secara konsep. Seorang manajer penjualan memahami bahwa semua informasi tentang seorang pelanggan disimpan dalam satu baris pada tabel pelanggan. *Physical View* menunjukkan bagaimana dan dimana data secara fisik disusun dan disimpan pada *disk*, *tape*, CD-ROM, atau media lain. Gambar 2 menunjukkan *record layout* dari File Piutang Dagang.

Gambar 1.
Ilustrasi Penggambaran ERD

A. *Relationship* Tidak Spesifik



B. Model Data Artistik



C. *Relationship* Spesifik

(Sumber: Whitten et al. 1994:307, dimodifikasi)

Pemisahan *Logical* dan *Physical View* data mengvasilitasi pengembangan aplikasi. Sebagai contoh kalau pemrogram diminta untuk membuat laporan kredit yang menampilkan *customer number*, *credit limit*, dan *current balance.* Dari sudut pandang

Logical View data, pemrogram hanya berkonsentrasi pada pembuatan aplikasi secara logis (apa yang akan dikerjakan oleh program). Pemrogram tidak perlu memperhatikan bagaimana dan dimana macam-macam data item disimpan dan diakses. Sedangkan jika ditinjau dari sudut pandang *Physical View* data, pemrogram harus memahami lokasi dan panjang *field* (posisi *record* 1 sampai dengan 10 untuk *customer number*) serta format *field* (*alphanumeric* atau *numeric*). Pemrograman menjadi lebih kompleks jika data yang dibutuhkan berasal dari beberapa file.

Gambar 3 menunjukkan bagaimana *software* database *management system* (DBMS) mengatur hubungan antara data yang disimpan secara fisik (*physical view*) dan cara pandang logis setiap *user* (*logical view*). Jadi DBMS mengatur database sedemikian rupa, sehingga *user* dapat mengakses, meminta, atau memperbaharui data tanpa perlu tahu bagaimana dan dimana data secara fisik disimpan.

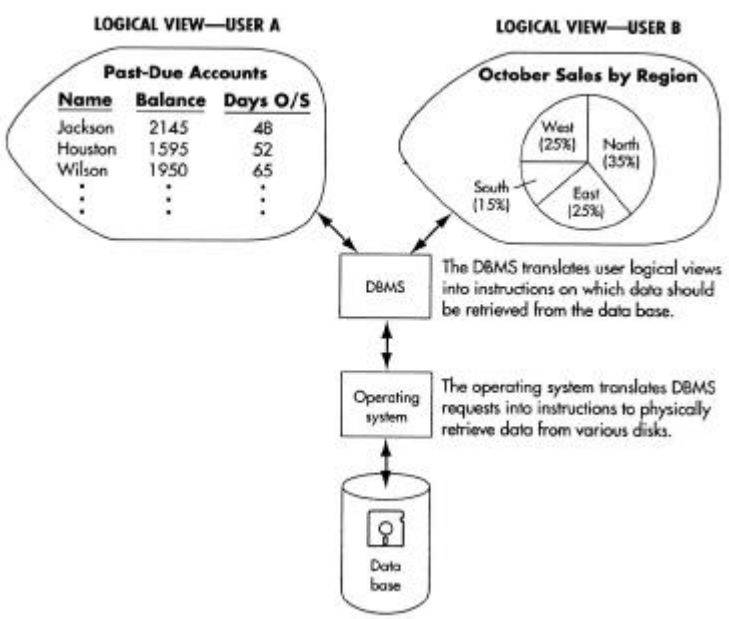
Gambar 2.
Susunan *Record File* Piutang Dagang

Customer *number*	**Customer** *name*	**Address**	**Credit** *limit*	**Balance**
A	A	A	N	N
1 10	11 30	31 60	61 68	69 76

A = *alphanumeric field* N = *numeric field*

(Sumber: Romney and Steinbart 2000:144)

Gambar 3.
Fungsi DBMS untuk Menunjang Bermacam-macam Gambaran Data Logis



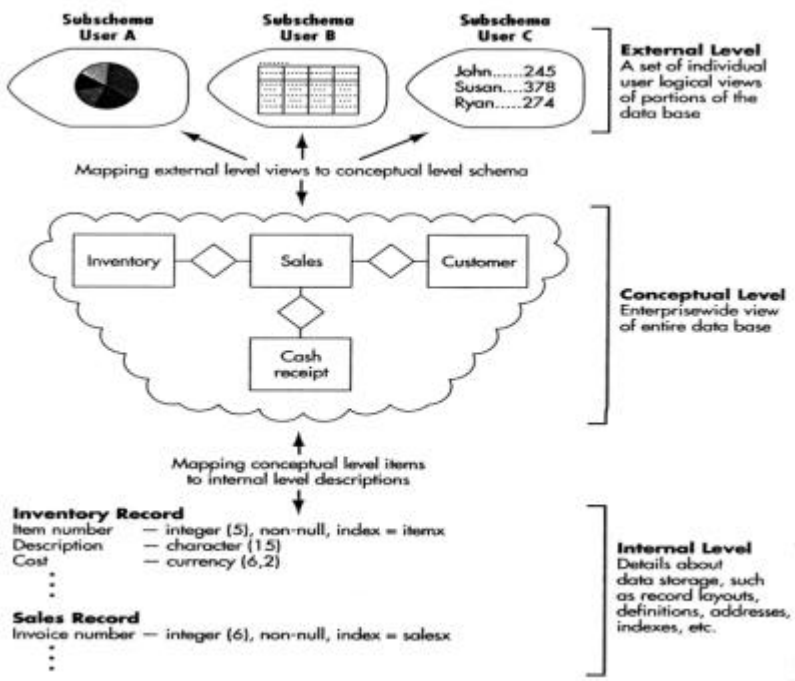
(Sumber: Romney and Steinbart 2000:145)

Pemisahan *Logical* dan *Physical View* data juga memberi keuntungan bagi *user*, dimana *user* dapat mengubah konsepnya tentang *relationship* diantara data item (memandang pekerjaan secara logis) tanpa mengubah datanya yang disimpan secara fisik. Pemisahan tersebut menunjukkan *program-data independence.*

2.1.2 Schema

Schema adalah gambaran struktur logis suatu database. Terdapat 3 tingkatan *schema*: *conceptual, external*, dan *internal*. Gambar 4 menunjukkan hubungan antar ketiga tingkatan tersebut. *Conceptual-level schema* adalah suatu cara pandang perusahaan yang menyeluruh terhadap database. *Conceptual-level schema* terdiri dari daftar semua elemen data dan hubungan diantaranya. *External-level schema* adalah kumpulan pandangan dari pemakai perorangan terhadap bagian database, dimana masing-masing bagian disebut dengan *subschema. Internal-level schema* adalah gambaran database tingkat bawah. *Internal-level schema* menggambarkan bagaimana sebenarnya data disimpan dan diakses, meliputi: informasi *pointer*, *index*, panjang *record*, dan sebagainya. Untuk menunjukkan pemetaan hubungan antar *schema* setiap tingkatan dihubungkan dengan panah dua arah. DBMS menggunakan pemetaan ini untuk menterjemahkan permintaan data dari pemakai melalui aplikasi program ke dalam *pointer*, *index*, dan pengaksesan data secara fisik.

Gambar 4.
Tiga Tingkat *Schemas*



(Sumber: Romney and Steinbart 2000:146)

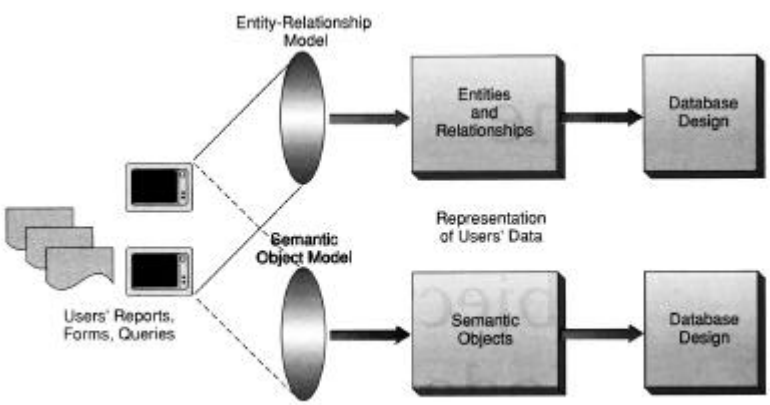
Perbedaan antara *conceptual* dan *external-level schema* dapat dilihat pada siklus pendapatan. *Conceptual schema* database siklus pendapatan berisi informasi tentang pelanggan, penjualan, penerimaan kas, staff penjualan, kas, dan sediaan. Sedangkan *external level* berisi sejumlah *subschema* yang dapat diperoleh dari *conceptual schema. Subschema* dibuat sesuai dengan kebutuhan pemakai atau aplikasi program. Setiap *subschema* yang dirancang tidak boleh mengakses pada bagian database yang tidak berhubungan dengan pekerjaan pemakai. Sebagai contoh *external-level subschema* untuk staff yang memasukkan pesanan penjualan menyajikan informasi tentang batas kredit pelanggan, saldo saat ini, jumlah persediaan, dan harga. Tetapi tidak menyertakan informasi tentang biaya persediaan atau saldo bank perusahaan saat ini. *External-level subschema* untuk staf pengiriman menyertakan informasi alamat pelanggan, tetapi tidak menyertakan informasi tentang batas kredit pelanggan atau gaji pegawai.

2.2 Model Entity Relationship (Model E-R)

Kroenke (2000:47) berpendapat bahwa model data digunakan untuk mendokumentasi kebutuhan *user* dan kebijakan perusahaan dalam rangka merancang database secara logis dengan menggunakan model E-R atau *Semantic-Object*. Model data ditunjukkan pada Gambar 5, dimana model data tersebut harus menunjang *Logical* dan *Physical View* data pemakai. Pemodelan data merupakan tugas yang paling penting dalam pembuatan aplikasi database. Pemodelan data yang salah akan berakibat perangkapan data dan database akan sulit untuk digunakan atau dikembangkan. Romney (2000:183) berpendapat bahwa pemodelan data dilakukan pada tahap *Requirement Analysis* dan *Design* dalam proses perancangan database.

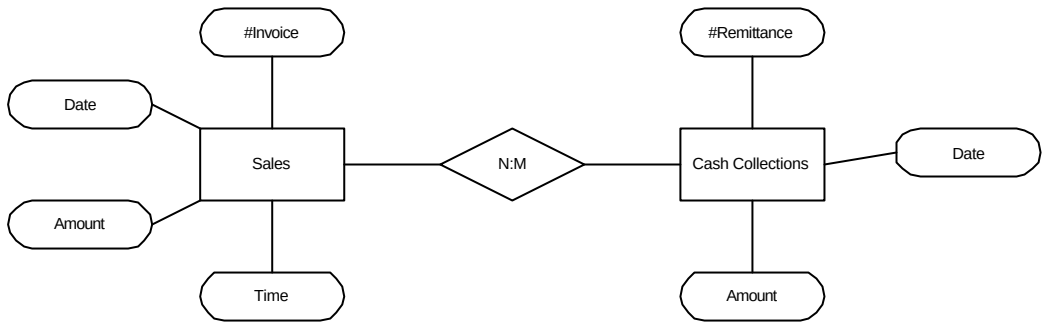
Model E-R diperkenalkan oleh Peter Chen pada tahun 1976 dan digunakan serta dikembangkan oleh Kroenke. Model E-R didokumentasikan dengan E-R diagram (ERD) (Kroenke,2000:59). Contoh ERD dapat dilihat pada Gambar 6, sedangkan elemen-elemen ERD dapat dilihat pada Tabel 1.

Gambar 5.
Penggunaan Model Data Berbeda untuk Merancang Database



(Sumber: Kroenke 2000:74)

Gambar 6.
Contoh *Entity Relationship Diagram*



(Sumber: olahan penulis)

Tabel 1.
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, dicantumkan dalam simbol.
Attribute/Property	NamaAttribute	• 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, dicantumkan dalam simbol.
Relationship	N:M	• Hubungan bisnis alamiah antara satu atau lebih *entity*. *Relationship* mungkin mewakili peristiwa yang menghubungkan *entity* atau hanya pertalian logis antara *entity* (Whitten et al 2000:264). • Dalam simbol tersebut dicantumkan maksimum *cardinality*.

(Sumber: olahan penulis)

Baik *entity* maupun *relationship* sama-sama dapat memiliki *attribute. Attribute* yang digunakan untuk mengidentifikasi *entity* disebut dengan *Identifier*. *Identifier* bisa unik (contoh: NomorIndukPegawai) dan tidak unik (contoh: NamaPegawai). *Identifier*

yang unik disebut dengan *primary key*. Cara penulisan *attribute primary key* diawali "#" pada nama *attribute* atau dipertebal, contoh: #NomorIndukPegawai atau Nomor Induk 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*. *Composite identifier* dapat mengakibatkan perangkapan data. Untuk menghindari perangkapan yang berlebihan, dengan menambah satu *attribute* dengan tipe data *autonumber*. *Attribute* tersebut disebut dengan *surrogate key* (Kroenke, 2000:241-242). Selain itu ada *foreign key* yang merupakan *attribute* dari suatu *entity* yang menjadi *primary key* dari *entity* lain. Cara penulisan *attribute foreign key* ditulis miring atau diberi garis bawah yang putus-putus, contoh: *NomorIndukPegawai* atau Nomor Induk Pegawai.

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

Minimum cardinality relationship menunjukkan jumlah baris yang paling sedikit dalam *relationship*. *Minimum cardinality* bisa 0 atau 1. *Minimum cardinality* 0 maksudnya setiap baris dalam tabel tidak perlu dihubungkan ke beberapa baris pada tabel lain. *Minimum cardinality* 1 menunjukkan bahwa setiap baris dalam tabel tersebut harus dihubungkan dengan paling sedikit satu baris dari tabel 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 tabel dapat dihubungkan ke paling banyak satu baris dari tabel lain. *Maximum cardinality* N menunjukkan bahwa satu baris dari tabel dapat dihubungkan dengan lebih dari satu baris dari tabel lain.

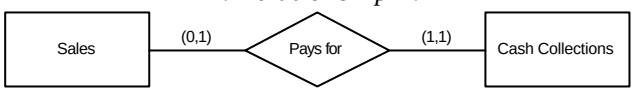
Tipe *relationship* tergantung pada *maximum cardinality* yang menghubungkan setiap *entity*, ada tiga tipe *relationship*:

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.

Gambar 7 memperlihatkan berbagai cara pemodelan *relationship* antara *sales* dan *cash collection event*. Gambar 7(A) menggambarkan *relationship* 1:1. 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 angsuran. Gambar 7(A) 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.

Gambar 7.
Kemungkinan *Cardinality Relationship Sales-Cash Collection*

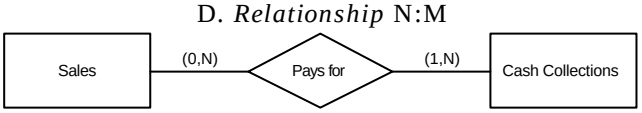
A. *Relationship* 1:1



B. *Relationship* N:1 antara *Sales* dan *Cash Collection*

C. *Relationship* 1:N antara *Cash Collection* dan *Sales*

D. *Relationship* N:M



(Sumber: Romney and Steinbart 2000:191)

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 7(A) *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 7(B) dan 7(C) menunjukkan dua cara untuk menyatakan *relationship* 1:N. Gambar 7(B) 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 angsuran (tetapi bisa juga membayar secara tunai), tetapi setiap *sales event* harus dibayar masing-masing, tidak boleh secara sekaligus. Sedangkan Gambar 7(C), 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 diijinkan untuk membayar secara sekaligus setiap bulan untuk semua pembelian yang dilakukan selama satu bulan, tetapi tidak diijinkan membayar secara angsuran.

Gambar 7(D) menunjukkan *relationship* N:M 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 pembayaran angsuran, dan juga mengijinkan pelanggan membayar lebih dari satu transaksi penjualan secara sekaligus.

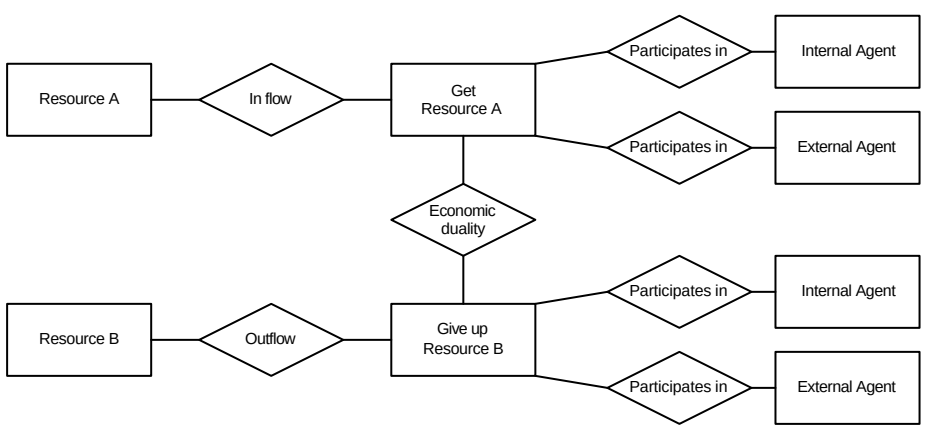
2.3 Model REA

Model REA adalah suatu alat pemodelan konseptual yang khusus dirancang untuk melengkapi struktur dalam perancangan database SIA. Dalam model REA ditentukan: *entity* apa yang harus disertakan dalam database SIA dan bagaimana susunan *relationship* antara *entity* dalam database SIA.

Tipe *entity* dalam model REA dibedakan dalam tiga kategori, yaitu: *Resources*, *Events*, dan *Agents*. *Resources* didefinisikan sebagai sesuatu yang memiliki nilai ekonomis bagi organisasi tersebut. Contoh *resources* adalah kas, inventaris, peralatan, persediaan, gudang, pabrik, dan tanah. *Events* menunjukkan aktivitas-aktivitas bisnis, dimana manajemen ingin mengumpulkan informasi untuk tujuan perencanaan atau pengawasan. Sebagai contoh, aktivitas penjualan akan mengurangi persediaan dan aktivitas penerimaan kas akan menambah jumlah kas. SIA harus dirancang untuk memperoleh dan menyimpan informasi aktivitas tersebut. Sedangkan *Agents* adalah orang dan organisasi yang berpartisipasi dalam aktivitas dan kepada siapa informasi diserahkan untuk tujuan perencanaan, pengawasan, dan pengevaluasian. Contoh *agent* adalah pengawai, pelanggan, dan pemasok.

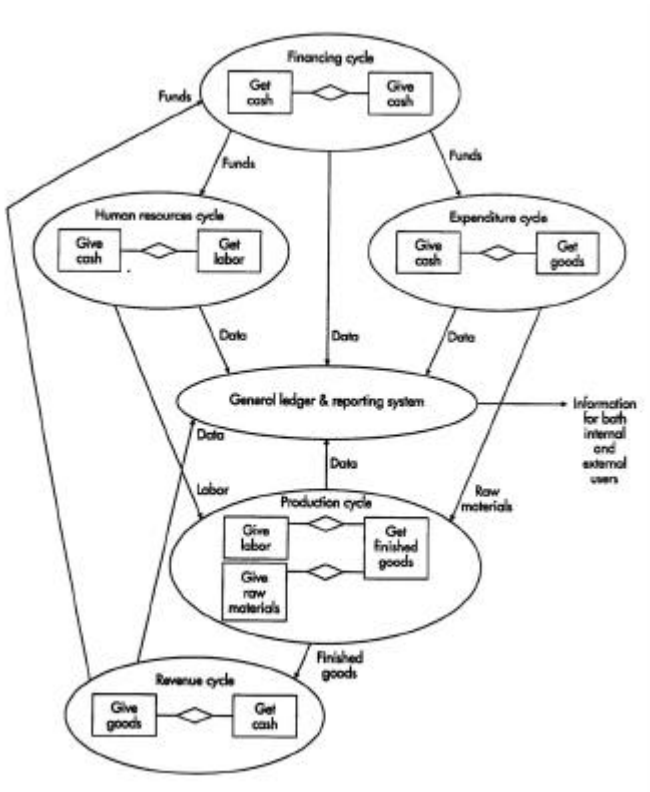
Model REA dapat dilihat pada Gambar 8. Setiap *entity event* dihubungkan dengan *entity resources* yang berpengaruh secara langsung atau tidak langsung. Setiap *entity event* juga dihubungkan dengan dua *entity agent*. *Internal agent* adalah pegawai yang bertanggung jawab pada *resources* yang terlibat dalam *event*. Sedangkan *external agent* adalah pihak luar yang berhubungan dengan transaksi. Gambar 7 menunjukkan *event* yang mengubah jumlah *resource* dihubungkan dengan *relationship give-to-get* ke *event* lain yang juga mengubah jumlah *resources*. *Relationship give-to-get* mencerminkan prinsip dasar bisnis, dimana organisasi yang menggunakan *resources* dalam aktivitas diharapkan dapat mengubah *resource* yang lain. Setiap siklus akuntansi dapat digambarkan dalam *relationship give-to-get* seperti yang ditunjukkan pada Gambar 9.

Gambar 8.
Model REA



(Sumber: Romney and Steinbart 2000:184)

Gambar 9.
Sistem dan Sub Sistem SIA



(Sumber: Romney and Steinbart 2000:185)

2.3.1 Menyusun Diagram REA

Dalam rangka menyusun diagram REA diperlukan informasi tentang: *resource*, aktivitas bisnis, *agent* dan kebijaksanaan perusahaan. Informasi tersebut dapat diperoleh dengan mewawancarai pihak manajemen. Karena aktivitas perencanaan, pengawasan, dan pengevaluasian yang ditangani manajemen untuk setiap perusahaan berbeda. Untuk menggambarkan diagram REA, kertas dibagi tiga kolom, satu kolom untuk setiap *entity*. Gunakan kolom kiri untuk *resource*, kolom tengah untuk *event*, dan kolom kanan untuk *agent*. Penggambaran *event* sebaiknya diurutkan dari atas ke bawah berdasarkan urutan aktivitas. Langkah-langkah untuk menyusun diagram REA suatu siklus transaksi adalah:

a. Tentukan pasangan aktivitas yang saling memberi dalam siklus tersebut.
   Seperti tampak dalam Gambar 8, model REA terdiri dari sepasang *event*, satu menambah *resource* dan yang lain mengurangi *resource*. Tentukan *event-event* bisnis yang perlu dimodelkan dalam siklus tersebut.
b. Tentukan *resource* yang dipengaruhi oleh *event* dan *agent* yang berpartisipasi pada *event* tersebut.
   Setelah *event* ditentukan, *resource* yang dipengaruhi oleh *event* tersebut ditentukan. *Resource* digambarkan pada kolom *resource*. Kemudian gambarkan *relationship*

antara *entity resource* dengan *entity event*. Langkah selanjutnya menentukan *agent* yang berpartisipasi dalam *event*. Akan selalu terdapat paling sedikit satu *internal agent* dan *external agent* yang terlibat dalam *event*. Gambarkan *relationship* untuk menunjukkan *agent* mana yang berpartisipasi dalam *event* tertentu. Sedapat mungkin penggambaran *agent* tidak ganda.

c. Tetapkan *cardinality* untuk setiap *relationship*.
Cardinality yang ditentukan harus mencerminkan perusahaan dan praktek bisnis yang dimodelkan.

2.3.2 Mengimplementasikan Diagram REA pada Database Relational

Setelah diagram REA selesai disusun, diagram REA dapat digunakan untuk merancang struktur database *relational* yang baik. Struktur database *relational* yang baik memenuhi aturan normalisasi, sehingga tidak ditemukan masalah *anomaly update, insert,* dan *delete.* Untuk mengimplementasikan diagram REA kedalam database *relational* dibutuhkan tiga langkah berikut:

a. Buat tabel untuk setiap *Entity* dan *Relationship* N:M
Database *relational* yang memenuhi aturan normalisasi memiliki satu tabel untuk setiap *entity* dan setiap *relationship* N:M. Nama setiap tabel harus sama dengan nama *entity* yang diwakilinya. Nama tabel untuk *relationship* N:M merupakan gabungan dari dua nama *entity* yang dihubungkan.

b. Menentukan *Attribute* untuk Setiap Tabel
Langkah selanjutnya adalah menentukan *attribute-atribute* yang harus dicantumkan pada setiap tabel. Setiap tabel harus memiliki *primary key* yang membuat unik baris dalam tabel. *Primary key* untuk tabel *relationship* N:M berisi minimal dua *attribute*, masing-masing mewakili *primary key* untuk setiap *entity* yang dihubungkan dalam *relationship* tersebut. Sedangkan *attribute-attribute* lain yang bukan *primary key* harus memenuhi aturan:
   - Setiap *attribute* dalam suatu tabel harus memiliki nilai tunggal.
   - Setiap *attribute* dalam suatu tabel harus menggambarkan karakteristik dari objek yang diwakili oleh *primary key*, atau *attribute* tersebut bisa juga berupa *foreign key*.

c. Mengimplementasikan *Relationship* 1:1 dan 1:N
Relationship 1:1 dan 1:N dapat diimplementasikan dengan *foreign key*. Sebagai contoh *attribute* Nomor Pelanggan adalah *primary key* tabel PELANGGAN, dimasukkan sebagai *attribute* pada tabel PENJUALAN, *attribute* ini dinyatakan sebagai *foreign key* pada tabel PENJUALAN.
Dalam database *relational*, *relationship* 1:1 dapat diimplementasikan dengan memasukkan *primary key* suatu *entity* sebagai *foreign key* pada *entity* lain. Untuk tujuan normalisasi pemilihan tabel yang menempatkan *foreign key* tidak ada ketentuan. *Minimum cardinality relationship* dapat digunakan untuk menentukan mana yang lebih efisien.
Relationship 1:1 antara *sales* dan *cash collection* yang digambarkan pada Gambar 6(A), *minimum cardinality sales event* adalah 0, menunjukkan penjualan kredit. Sedangkan *minimum cardinality cash collection event* adalah 1, menunjukkan *cash collection* hanya terjadi setelah penjualan dilakukan (contoh menunjukkan tidak adanya uang muka). Untuk masalah tersebut memasukkan nomor *invoice* sebagai

foreign key dalam *cash collection event* akan lebih effisien, karena hanya satu tabel yang diakses dan diperbaharui pada pemrosesan data *cash collection event*. Selain itu *relationship* 1:1 antara dua *event* yang berhubungan, menyertakan *primary key event* yang terjadi lebih dahulu sebagai *foreign key* pada *event* yang terjadi berikutnya. Hal tersebut dilakukan guna meningkatkan *internal control.*

Relationship 1:N diimplementasikan dalam database *relational* dengan menempatkan *foreign key*. Untuk masalah ini *primary key* dari *entity* yang berperan 1 dalam *relationship* tampak sebagai *foreign key* pada *entity* yang berperan banyak dalam *relationship*.

2.3.3 Manfaat Diagram REA

Diagram REA digunakan sebagai dokumentasi pelengkap, yang berguna untuk mendokumentasi pembentukan *advanced* SIA. Diagram REA menyediakan dua informasi database SIA, yang tidak ditunjukkan oleh bentuk dokumentasi lain. Informasi yang disajikan oleh diagram REA adalah *relationship* antara data dan praktek bisnis perusahaan. Diagram REA secara tegas menggambarkan *relationship* antara bermacam-macam data item yang disimpan dalam database akuntansi.

Cardinality diagram REA menyajikan informasi yang berguna untuk menggambarkan prinsip dan kebijaksanaan perusahaan yang dimodelkan. Menaksirkan dengan benar *cardinality* diagram REA membutuhkan pemahaman secara tepat yang menunjukkan kejadian setiap *entity*. Setiap kejadian dari *entity agent* menunjukkan orang atau organisasi tertentu. Hal yang sama setiap kejadian suatu *entity event* menunjukkan aktivitas atau transaksi bisnis spesifik.

2.4 Perancangan Database

Perancangan dan pengoperasian database meliputi enam tahap berikut: *planning, requirements analysis, design, coding, implementation, serta operation and maintenance.* Akuntan berperan pada perancangan database. Dalam tahap perencanaan, akuntan menyediakan informasi yang digunakan untuk mengevaluasi kelayakan proyek yang diusulkan dan berpartisipasi membuat keputusan. Dalam tahap *requirement analysis* dan *design*, akuntan berpartisipasi dalam menentukan informasi yang dibutuhkan oleh pemakai, membangun logical schema, merancang data *dictionary*, dan menentukan pengawasan. Akuntan dengan keahlian SIA yang baik dapat berpartisipasi pada tahap *coding*. Selama tahap implementasi, akuntan berperan mengujicoba keakuratan database dan aplikasi program yang akan menggunakan data tersebut. Akuntan menggunakan sistem database untuk memproses transaksi, kadang kala akuntan membantu mengatur sistem database.

Tahap *planning* menentukan kebutuhan dan kelayakan pengembangan sistem database baru. Sasarannya adalah menentukan apakah sistem yang diusulkan layak secara teknologi dan ekonomi. Sedangkan pada tahap requirements analysis menentukan informasi yang dibutuhkan oleh pemakai, lingkup sistem database yang diusulkan, dan menetapkan kebutuhan hardware dan *software* awal. Data tentang kebutuhan pemakai dikumpulkan dengan metode wawancara atau daftar pertanyaan. Setelah kebutuhan pemakaian dan lingkup sistem database baru ditentukan, informasi tentang jumlah pemakai dan volume transaksi yang diharapkan dapat digunakan untuk menentukan kebutuhan hardware dan *software* awal.

Setelah struktur database dibangun. Tahap *design* dibagi kedalam tiga langkah:

a. *Conceptual design*, menterjemahkan kebutuhan data pemakai yang berbeda ke dalam model database konsep. Perancangan lebih mudah jika membagi rancangan berdasarkan siklus akuntansi (*revenue*, *expenditure*, *production*, *payroll*, dan *general ledger*). Sebagai contoh skema siklus pendapatan meliputi semua data yang berhubungan dengan *sales* order *processing*, *shipping*, *billing* dan *account receivable*, serta *cash collection*.
b. *Logical design*, memilih tipe DBMS yang akan digunakan untuk menterjemahkan model konseptual ke dalam model DBMS yang dipilih.
c. *Physical design*, menterjemahkan *logical schema* kedalam model yang mendeskripsikan struktur fisik dan metode akses yang digunakan untuk mengimplementasikan sistem menggunakan paket DBMS tertentu. Pada langkah ini dihasilkan *physical schema* dan data *dictionary*.

Pada tahap *coding* diterjemahkan *physical schema* ke dalam struktur database. Selama tahap *coding* dipertimbangkan perancangan alternatif. Sangat disayangkan tidak setiap tujuan dapat maksimal, dibutuhkan uji coba. Sebagai contoh cost-effectiveness sering berbenturan dengan *flexibility* dan *accessibility*. Perancang database mencoba untuk mencapai kemungkinan terbaik dalam menyeimbangkan tujuan.

Tahap implementation mencakup semua aktivitas yang berhubungan dengan perolehan untuk sistem database baru. Meliputi uji coba sistem baru, memindahkan data dari file yang ada ke database yang baru, dan melatih pegawai tentang pengunaan sistem baru.

Tahap yang terakhir adalah *operation and maintenance* meliputi semua aktivitas yang berhubungan dengan pengoperasian dan pemeliharaan sistem baru. Memantau

kinerja sistem baru dan kepuasan pemakai untuk menentukan apakah sistem perlu dikembangkan atau tidak.

2.5 Penerapan Diagram REA pada Siklus pendapatan

2.5.1 Aktivitas Bisnis Siklus pendapatan

Satu-satunya tujuan SIA dalam siklus pendapatan adalah untuk menunjang pelaksanaan aktivitas-aktivitas bisnis dengan pemrosesan transaksi data secara efektif. Gambar 10 menunjukkan empat aktivitas bisnis siklus pendapatan berikut: *sales order entry*, *shipping*, *billing*, dan *cash collections*. Kemajuan teknologi informasi memungkinkan beberapa aktivitas tersebut dilaksanakan secara bersamaan.

Aktivitas pertama pada siklus pendapatan adalah *sales order entry*. Pada aktivitas *sales order entry customer orders* dikumpulkan dan diproses oleh *salesperson*. Agar *salesperson* dapat memutuskan *customer orders* diterima atau ditolak diperlukan informasi persediaan yang dimiliki dan status kredit. *Salesperson* dapat melihat informasi persediaan yang dimiliki dari file *inventory*. Sedangkan informasi status kredit pelanggan dapat dilihat pada file *customer*. Jika *customer orders* diterima, maka *salesperson* mencatat *customer orders* pada file *sales orders*. Keputusan yang menyangkut kebijaksanaan kredit dalam menyetujui kredit untuk pelanggan baru atau mengubah batas kredit pelanggan lama dilakukan oleh manajer kredit. Hal tersebut menunjukkan pemisahan kewajiban otorisasi dan pencatatan. Aktivitas *sales order entry* secara rinci dapat dilihat pada Gambar 11, meliputi tiga aktivitas berikut: *responding to customer inquiries*, *checking and approving customer credit*, dan *checking inventory availability*.

Pada aktivitas *responding to customer inquiries salesperson* menanggapi permintaan pelanggan yang berhubungan dengan jumlah persediaan dan harga (dapat dilihat pada file *inventory*) serta status pesanan (dari file *sales order*). Sedangkan permintaan yang berhubungan dengan *current account balances* dapat dijawab berdasarkan informasi yang diperoleh dari file *customer*.

Aktivitas *checking and approving customer credit* memutuskan apakah penjualan boleh dilakukan secara kredit. Salesperson memutuskan persetujuan kredit berdasarkan *account balance* maksimum yang diijinkan untuk pelanggan tersebut. *Account balance* ditentukan berdasarkan sejarah kredit dan kemampuan bayar masa lalu. Persetujuan kredit pelanggan dilakukan dengan memeriksa file *customer*. Bandingkan jumlah batas kredit pelanggan dengan jumlah *order* ditambah dengan jumlah *account balance current*, jika jumlah tidak melampaui batas kredit pelanggan. Sedangkan jika permintaan kredit pelanggan melebihi batas yang sudah ditentukan, maka persetujuan kredit dilakukan oleh manajer kredit. Jika permintaan ditolak, maka *salesperson* harus memberi informasi kepada *customer*.

Aktivitas *checking inventory availability* memeriksa jumlah persediaan yang dimiliki dari file *inventory*. Jika jumlah persediaan tidak mencukupi jumlah permintaan, maka dibuat *back order* untuk bagian pembelian. *Salesperson* membuat *sales orders* dan menginformasikan tanggal pengiriman barang. *Sales orders* mengakibatkan aktivitas *shipping* (berdasarkan *packing slip*), *billing*, dan bagian gudang menyiapkan barang (berdasarkan *picking ticket*).

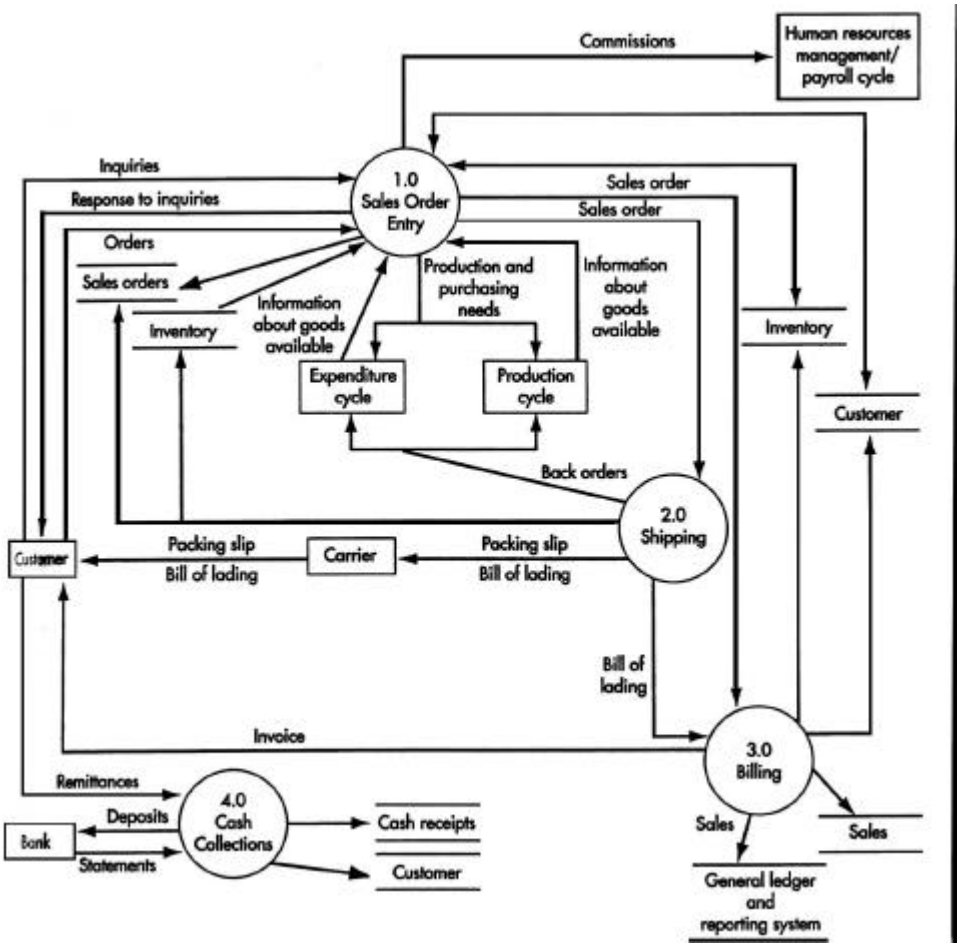
Aktivitas kedua pada siklus pendapatan adalah *shipping*. Pada aktivitas ini pesanan

pelanggan dipenuhi dan dikirim. *Warehouse clerk* yang bertanggung jawab memenuhi pesanan pelanggan berdasarkan *picking ticket*. Departemen pengiriman (*carrier*) bertanggung jawab mengirim pesanan kepada pelanggan berdasarkan *packing slip*.

Aktivitas ketiga pada siklus pendapatan adalah *billing*, meliputi aktivitas pembuatan faktur dan memelihara piutang pelanggan. Aktivitas ini dikerjakan oleh *departement billing/account receivable*.

Aktivitas terakhir yang dilakukan dalam siklus pendapatan adalah *cash collection*. Yang berpartisipasi dalam aktivitas ini adalah *cashier* dan bagian *accounts receivable*. Kasir menangani pembayaran pelanggan dan menyetor ke bank. Sedangkan bagian *accounts receivable* mengkredit piutang pelanggan atas pembayaran yang diterima.

Gambar 10.
***Data Flow Diagram* Siklus Pendapatan Tingkat 0**



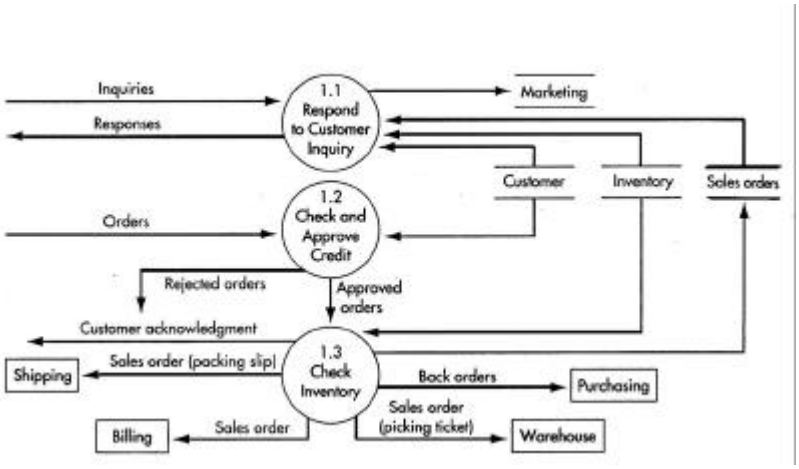
(Sumber: Romney and Steinbart 2000:417)

2.5.2 Informasi yang Dibutuhkan pada Siklus Pendapatan dan Model Data

Fungsi SIA adalah menyediakan informasi yang berguna untuk membuat keputusan. SIA harus menyediakan informasi operasional yang dibutuhkan untuk melakukan aktivitas sebagai berikut:

- Menanggapi permintaan pelanggan tentang saldo rekening dan status pesanan.
- Memutuskan apakah kredit pelanggan tertentu perlu diperbesar.
- Memutuskan persediaan mencukupi.
- Memutuskan syarat-syarat kredit yang ditawarkan.
- Menetapkan harga produk atau jasa.
- Menetapkan kebijaksanaan berkenaan dengan pengembalian penjualan dan garansi.
- Memilih metode pengiriman barang dagang.

Gambar 11.
Data Flow Diagram:* Memasukkan Pesanan Penjualan Tingkat *1



(Sumber: Romney and Steinbart 2000:418)

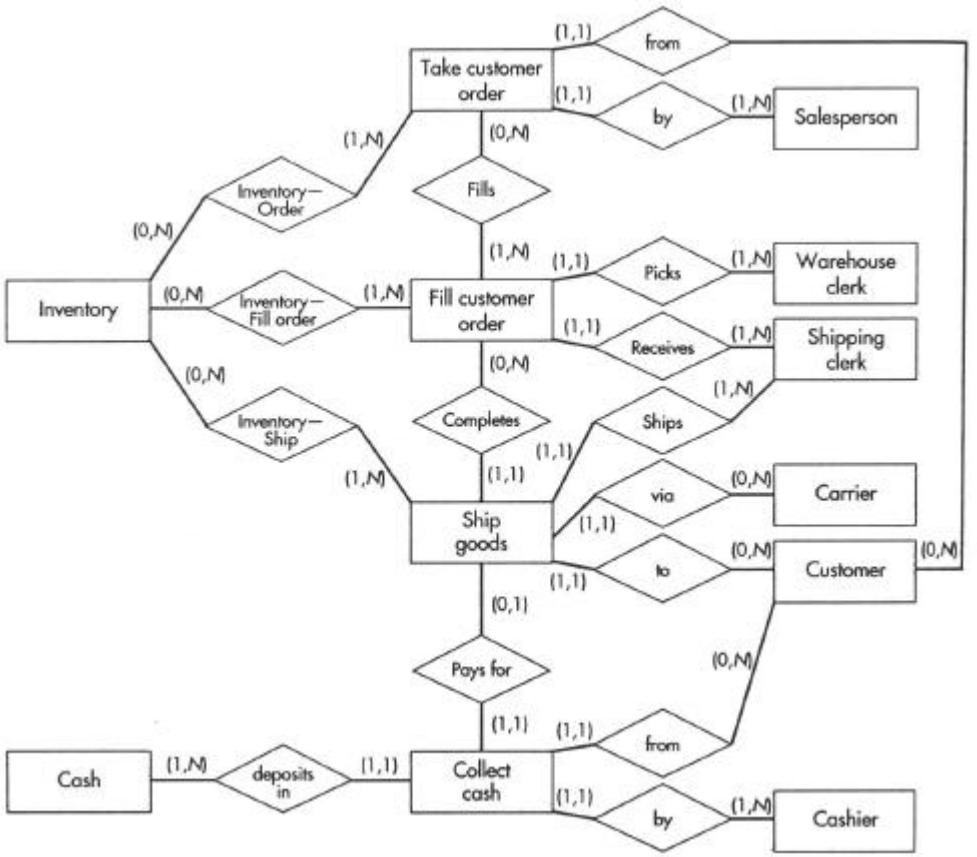
Sebagai tambahan SIA sebaiknya menyediakan informasi strategi dan evaluasi pelaksanaan:

- Waktu yang dibutuhkan untuk menangani permintaan pelanggan.
- Waktu yang dibutuhkan untuk mengisi dan menyerahkan pesanan.
- Persentase penjualan yang memerlukan *back order*.
- Kepuasan pelanggan.
- Menganalisis pengaruh pasar dan kecenderungan penjualan.
- Analisis keuntungan berdasarkan produk, pelanggan, dan wilayah penjualan.
- Volume pejualan dalam dolar dan jumlah pelanggan.
- Keberhasilan iklan dan promosi.
- Kinerja staff penjualan.
- Piutang tidak tertagih dan kebijaksanaan kredit.
- Diharapkan penerimaan kas dan pinjaman jangka pendek.

Informasi keuangan dan operasional dibutuhkan untuk mengatur dan mengevaluasi aktivitas siklus pendapatan. Dengan adanya sistem database, sudah saatnya merancang kembali SIA yang mampu menggumpulkan dan menyimpan data transaksi keuangan dan operasional pada siklus pendapatan. Dari aktivitas bisnis siklus pendapatan dari Gambar 10 dan 11 diperoleh informasi:

- Dua *resources*: *cash* dan *inventory*
- Empat bisnis *event*: *take customer order*, *fill customer order*, *ship goods*, dan *collect cash*.
- *External agent*: *customer* dan *carrier*. Sedangkan *internal agent* adalah *salesperson*, *warehouse clerk*, *shipping clerk*, dan *cashier*. Diagram REA siklus pendapatan dapat dilihat pada Gambar 12.

Gambar 12.
Diagram REA Siklus Pendapatan



(Sumber: Romney and Steinbart 2000:448)

Diagram REA pada Gambar 12 terdiri dari dua belas entity dan empat relationship N:M (relationship Inventory dan Take customer order, relationship Inventory dan Fill customer order, relationship Inventory dan Ship goods, serta relationship Take customer order dan Fill customer order). Untuk mengimplementasikan diagram REA ke database relational masing-masing entity dan relationship N:M dijadikan tabel, sehingga terbentuk enam belas tabel. Diharapkan tabel yang dihasilkan dapat memenuhi aturan normalisasi.

Langkah selanjutnya mencantumkan *attribut* pada masing-masing tabel, tentukan *attribute primary key* untuk setiap tabel. Pada Tabel 2 *attribute primary key* dicetak dengan huruf tebal, sebagai contoh tabel *Inventory*, *attribute primary key*-nya adalah *Product number*. Sedangkan untuk tabel yang mewakili *relationship* N:M, misal tabel *Inventory-Order*, *primary key* merupakan gabungan dari *primary key* tabel yang dihubungkan. Jadi *primary key* tabel *Inventory-Order* adalah *Product number* dan *sales order number* (*primary key* tabel *Order).*

Langkah terakhir adalah mengimplementasikan *relationship* 1:1 dan 1:N dengan *attribute foreign key*. Pada Tabel 2 *foreign key* ditulis menggunakan huruf miring.

Mengimplementasikan *relationship* 1:1 antara *entity Ship goods* dan *Collect cash* dengan cara menempatkan *attribute Invoice number* (sebagai *primary key entity Ship goods*) pada tabel *Collect cash* (sebagai *foreign key).* Untuk mengimplementasikan *relationship* 1:N, misalnya *relationship* antara *entity Fill customer order* (posisi 1) dengan *Ship goods* (posisi N), menempatkan *Picking ticket number* (sebagai *primary key entity Fill customer order*) pada tabel *Ship goods* (sebagai *foreign key).* Nama tabel dan *attribute* secara keseluruhan dapat dilihat pada Tabel 2.

Tabel 2.
Tabel yang Digunakan pada Siklus Pendapatan

Table Name	**Attributes** (**primary key**, *foreign keys*, other)
Inventory	**Product number**, description, unit cost, unit price, quantity on hand, weight, reorder point, ...
Cash	**Account number**, *bank ID*, balance, ...
Take customer order	**Sales order number**, date, *customer ID*, *salesperson number*, terms, desired delivery date, ...
Fill customer order	**Picking ticket number**, date, time, *warehouse clerk number*, *shipping clerk number*, ...
Ship goods	**Invoice number**, date, bill of lading number, *picking ticket number*, *shipping clerk number*, *carrier number*, *customer number*, amount due, ...
Collect cash	**Remittance number**, date, amount, *customer number*, *cashier number*, *invoice number*, *bank account number*
Salesperson	**Employee number**, name, date hired, date of birth, salary, *manager number*, ...
Warehouse clerk	**Employee number**, name, date hired, date of birth, salary, *manager number*, ...
Shipping clerk	**Employee number**, name, date hired, date of birth, salary, *manager number*, ...
Carrier	**Carrier number**, name, primary contact, ...
Customer	**Customer number**, name, bill-to address, ...
Cashier	**Employee number**, name, date hired, date of birth, salary, *manager number*, ...
Inventory-Order	**Product number**, **sales order number**, quantity
Inventory-Fill order	**Product number**, **picking ticket number**, quantity
Inventory-Ship	**Product number**, **invoice number**, quantity
Take order-Fill order	**Sale order number**, **picking ticket number**

(Sumber: Romney and Steinbart 2000:449)

2.6 Sistem Database untuk Akuntansi Masa Depan

Sistem database akan sangat berpengaruh pada akuntansi dasar. Sebagai contoh, sistem database dapat berperan untuk meninggalkan *model double-entry bookeeping* (Sentosa 1999:2-12). Dasar pemikiran model *double-entry* adalah pengulangan, karena

pencatatan jumlah transaksi dilakukan dua kali, dengan tujuan untuk memeriksa keakuratan pemrosesan data. Setiap transaksi menghasilkan catatan debet dan kredit yang sama. Kesamaan jumlah debet dan kredit diperiksa dan diperiksa kembali pada beberapa tempat pemrosesan akuntansi. Pengulangan data diantisipasi dengan konsep database. Jika jumlah yang berhubungan dengan transaksi dimasukkan ke dalam sistem database secara benar, hal tersebut dapat disimpan hanya satu kali. Pemrosesan data komputer cukup akurat untuk membantu pemeriksaan keterangan secara teliti, yang semula ditunjukkan pada model akuntansi *double-entry*.

Bagaimana dengan *account receivable*? *Account receivable* tidak disertakan, karena *account receivable* tidak dijumpai pada *resource* model REA pada Gambar 12. *Account receivable* bukan objek yang berdiri sendiri, tetapi hanya menunjukkan perbedaan waktu antara dua *event*: penjualan dan penerimaan kas. Akibatnya, jika data tentang penjualan dan penerimaan kas sudah disimpan dalam database, tidak perlu lagi menyimpan informasi tentang *account receivable* secara berlebihan.

3. KESIMPULAN

Kemajuan teknologi komputer dan informasi berpengaruh pada SIA, sehingga mengubah SIA manual ke SIA terkomputersasi yang melibatkan database. Untuk merancangkan kembali SIA, akuntan ikut berperan dalam perancangan database, karena akuntan yang mengusai pengendalian internal yang juga harus diterapkan pada SIA terkomputerisasi. Pada akhirnya akuntan yang akan menggunakan informasi yang disajikan oleh SIA terkomputerisasi.

Untuk merancang database diperlukan alat bantu, salah satunya adalah model E-R. Namun kurang jelas aturan penggambaran diagramnya, sehingga menyulitkan. Ada pendekatan lain dari model E-R yang disebut dengan model REA yang memiliki aturan yang lebih jelas dan cara pendekatannya cocok untuk seorang akuntan. Model REA merupakan *logical view* data dari pemakai yang berhubungan dengan *conceptual-level* dan *external-level schema.*

Model REA merupakan salah satu data *dictionary*, yang digunakan untuk menggambarkan aktivitas-aktivitas yang dilakukan pada suatu perusahaan. Aktivitas tersebut mempengaruhi *resource* apa saja, dengan memperhatikan prinsip ekonomi *give-to-get*. Siapa saja yang terlibat dalam aktivitas tersebut dan kepada siapa aktivitas tersebut ditujukan. Selain itu dalam diagram REA juga dapat dilihat kebijaksaan perusahaan, ditunjukkan dengan *cardinality*.

Dengan adanya database, data dapat terintegrasi, perangkapan data dapat dikurangi, format tidak tergantung pada aplikasi program, dan pemakai data dapat dengan mudah menyajikan informasi dengan bantuan bahasa *query*. Dengan prinsip mengurangi perangkapan data, database menunjukkan adanya kemungkinan untuk meninggalkan *double-entry* pada pencatatan akuntansi.

DAFTAR PUSTAKA

Kroenke, David M. (2000), *Database Processing: Fundamentals, Design & Implementation*, Seventh Edition, United State of America: Prentice Hall.

Romney, Marshall B. et al. (2000), *Accounting Information Systems*, Eighth Edition, New Jersey: Prentice Hall.

Sentosa, Setyarini dan Maya Fransiska (Mei 1999), "Pengaruh Perkembangan Basis Data Relasional Terhadap Teknik Double Entry Bookkeeping",*Jurnal Akuntansi dan Keuangan*, halaman 1-15.

Whitten, Jeffrey L. et al. (1994), *Systems Analysis and Design Methods*, Third Edition, United States of America: Irwin.

Whitten, Jeffrey L. et al. (2000), *Systems Analysis and Design Methods*, Fifth Edition, New York: Irwin/McGraw-Hill.

Yuliana, Oviliani Yenty (Mei 2001), "Implementasi Referential Integrity Constraint pada Microsoft Access dalam Upaya Memelihara Konsistensi Data", *Jurnal Informatika*, halaman 33-43.