PEMETAAN HUBUNGAN GENERALISASI/SPESIALISASI PADA MODEL ER KE MODEL RELASIONAL

Ashari Imamuddin
Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Bina Nusantara
Jl. KH Syahdan 9 Kemanggisan, Jakarta Barat 11480, Jakarta, Indonesia
e-mail: ashari1844@lecturer.binus.ac.id;

ABSTRAKSI

Hubungan kompleks generalisasi/spesialisasi dalam model ER dalam perancangan database dapat ditemukan pada saat perancangan model konseptual. Turunan model konseptual adalah model logikal. Model data logikal inilah yang akan diimplementasikan ke dalam RDBMS (relational database management systems). *Oleh karena ini makalah ini akan membahas bagaimana hubungan generalisasi/spesialisasi dipetakan dari model ER ke model relasional. Di sini juga akan dibahas opsi-opsi pemetaan dan rekomendasi penggunaannya. Disamping itu juga dikemukakan modifikasi statement dalam bahasa SQL agar dapat mengenali multi referential untuk menjaga integritas referensial bagian superklas ke bagian subklas.*

Kata Kunci: *Perancangan database, generalisasi/spesialisasi, entity relationship, model relasional, referensial key, superkelas, subkelas.*

1. PENDAHULUAN

Perancangan *database* dibagi menjadi 3 fase atau tahapan yaitu tahap perancangan konseptual, tahap perancangan logikal, dan tahap perancangan fisikal. Perancangan *database* konseptual adalah proses dibangunnya suatu model informasi yang digunakan dalam perusahaan berdasarkan pandangan penguna sistem informasi (*use views*), dan tidak bergantung pada pertimbangan fisik. Perancangan *database* logikal adalah proses dibangunnya suatu model dari informasi yang digunakan oleh perusahaan yang didasari oleh suatu data model yang spesifik, tetapi tidak bergantung pada DBMS (*Data Base Management System*) dan pertimbangan fisikal lain. Perancangan *database* fisik adalah proses menghasilkan deskripsi dari suatu implementasi dari *database* pada penyimpanan (secondary storage), dan juga menjelaskan tentang *base relations*, pengorganisasian file (file organizatons), dan indeks (indexes) yang digunakan untuk mencapai efisiensi dalam mengakses data dan asosiasi batasan integrasi dan pengukuran keamanan lainnya. (Connolly, 2002)

Pada desain konseptual, digunakan model semantik dengan model ER (Entity relationship) dimana model digunakan untuk menggambarkan entitas-entitas yang ada dalam organisasi dan hubungan antar entitas. Sedangkan pada perancangan logikal berorientasi untuk menerjemahkan model logikal yang dalam bentuk ER (Entity relationship) tersebut menjadi model relasional. Atau dengan kata lain menerapkan model konseptual ke dalam model relasional.

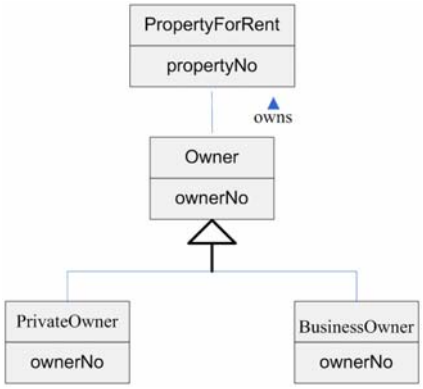
Dalam tahap konseptual, terdapat berbagai hubungan yaitu, *one to one, one to many, many to many, weak entities, strong entities, multivalued atribute*, superkelas/subkelas, agregasi, dan komposisi. Di antara hubungan di atas terdapat hubungan yang paling sulit untuk diterapkan dalam model relasional yaitu hubungan superkelas dan subkelas atau yang biasa dikenal dengan generalisasi/spesialisasi. Karena pada hubungan ini akan sangat sulit untuk menjaga integritas data yang dimiliki subkelas yang satu dengan subkelas yang lain, dimana subkelas-subkelas tersebut dimiliki oleh satu superkelas. Dengan kata lain data dari subkelas-subkelas tersebut mereferensi satu superkelas, hubungan seperti ini dalam tulisan ini disebut dengan *multi referential*. Hubungan *superclass*/ *subclass* atau biasa juga disebut generasilsasi/ spesialisasi. Model ER yang mengandung generasilsasi/spesialisasi disebut dengan EER (*Enhanced Entity Relationship*).

Makalah ini difokuskan pada perancangan *database* dalam model ER (Entity Relationship), EER (Enhanced Entity Relationship) dan model relasional yang memiliki satu *primary key*. Juga akan disajikan berbagai masalah dan teknik yang digunakan dalam pemetaan (*mapping*) model konseptual untuk EER ke model relasional, dan juga keterbatasan RDBMS dalam implementasi model ini. Hal ini akan dilakukan dengan terlebih dahulu dilakukan analisis terhadap proses *mapping* generalisasi/spesialisasi dari model ER (Entity Relationship) ke EER (Enhanced Entiy Relationship) yang kemudian dilanjutkan ke model relasional. Selain itu juga akan dibuat dua pilihan pemecahan masalah dalam implementasi generalisasi/spesialisasi ke dalam RDBMS (Relational Database Management Systems), yaitu membuat modifikasi *statement* dalam bahasa SQL agar dapat mengenali *multi referential* untuk implementasi model ini.

2. ANALISIS HUBUNGAN GENERALISASI/SPESIALISASI

Di bawah ini akan dibahas kasus yang berkaitan dengan generalisasi/spesialisasi yaitu hubungan antara **PropertyForRent**, dan **Owner** yang memiliki dua subkelas/*child* yaitu **PrivateOwner** dan **BussinessOwner**. (lihat gambar 1).

Pada kenyataannya, subkelas dari tabel **PropertyForRent** mungkin saja lebih dari dua. Untuk melakukan pemetaan ke dalam bentuk model relasional maka perlu digunakan beberapa opsi yang telah menjadi panduan pada kasus superkelas/subkelas. Berikut akan dibahas masing-masing opsi yang terjadi pada kasus **Owner**.



Gambar 1. Contoh kasus (Connolly, p443)

***Option* 1 (Mandatory Nondisjoint)**

```
AllOwner(ownerNo, address, telNo, fName,
      lName, bName, bType, contactName,
      pOwnerFlag, bOwnerFlag)
Primary Key ownerNo
```

Opsi 1 (Mandatory Nondisjoint) membutuhkan satu relasi (dengan satu atau lebih diskriminator untuk membedakan tipe macam-macam tuple atau baris). Opsi ini digunakan jika seluruh anggota entitas superkelas harus menjadi anggota dari subkelas-nya, dimana jumlah dari subkelasnya tidak dibatasi.

Pada opsi 1 jika digunakan pada kasus **Owner** maka seperti yang digambarkan di atas pada Gambar 1, maka seluruh atribut dari superkelas dan subkelas dimerger ke dalam satu tabel yaitu **AllOwner.** Oleh karena itu jika data yang dimasukkan dalam tabel tersebut adalah milik tabel **PrivateOwner** maka field yang mereferensikan tabel **BusinessOwner** akan terisi *null*, begitu pula sebaliknya. Akibatnya akan banyak sel yang dibuat sia-sia karena terisi *null*, dan itu artinya akan banyak *space memory* yang terbuang sia-sia. Dari hubungan di atas dapat dilakukan perhitungan untuk melihat banyaknya *space* kosong yang akan terjadi.

Sebagai notasi jumlah *record* = n, jumlah kolom = x, jumlah kolom umum = u, jumlah kolom C_1 = y, dan jumlah kolom C_2 = x-(u+y). Jika dimasukan sejumlah *record* $C_1 = nC_1$ dan dimasukan sejumlah *record* $C_2 = nC_2$, maka akan didapatkan suatu rumus untuk menghitung jumlah *space* yang terbuang (W):

$$\mathbf{W = nC_1 * (x - (u + y)) + nC_2 * (y)}$$

Dengan menggunakan rumus di atas dapat diketahui jika terdapat 100 *record* (n) dengan:

Jumlah kolom (x) = 10,
Kolom umumnya (u) = 3,
Kolom C_1 (y) =3 maka kolom C_2 (x-(u+y)) = 4.

Jika jumlah *record* C_1 (nC_1) yang dimasukan sebanyak 60 dari 100 *record* dan jumlah *record* C_2 (nC_2) sebanyak 40 dari 100 *record*, maka jumlah *space memory* yang akan berisi *null* dan sia–sia sebanyak 360 sel, dengan perhitungan di bawah ini:

$W = nC_1 * (x - (u + y)) + nC_2 * (y)$
$= 60 * (4) + 40 * (3)$
$= 240 + 120$
$= 360$

Opsi 1 digunakan ketika seluruh diskriminator yang dikumpulkan diuji, maka salah satu dari atribut-atribut akan bersifat unik terhadap suatu subkelas yang *non-null*, untuk mendeterminasikan bahwa baris tersebut adalah anggota dari subkelasnya. Dan perlu dipastikan bahwa atribut yang diuji adalah atribut yang dibutuhkan atau *non-null*.

Maka kesimpulannya jika opsi 1 digunakan tidak pada kasus yang tepat, seperti pada contoh kasus **Owner** yang menggunakan opsi 1, setelah dianalisis memang tidak akan terjadi kendala pada integritas referensial, dan integritas data, akan tetapi kendala akan terjadi pada jumlah *waste space*sia, selain itu juga dengan banyaknya *null* akan memperlambat waktu *query*. Oleh karena itu opsi 1 cocok digunakan untuk kasus dimana anggota superkelasnya seluruhnya adalah anggota subkelasnya.

***Option* 2 (*Option*al Nondisjoint)**

```
Owner (ownerNo, address, telNo)
Primary Key ownerNo
OwnerDetails (ownerNo, fName, lName, bName,
    bType, contactName,
    pOwnerFlag,bOwnerFlag)
Primary Key ownerNo
Foreign Key ownerNo references Owner
(ownerNo)
```

Opsi 2 (*Option*al Nondisjoint) membutuhkan dua relasi, dimana satu relasi dibutuhkan untuk superkelas dan satu relasi untuk seluruh subkelas (dengan satu atau lebih diskriminator yang membedakan setiap jenis recordnya). Opsi ini digunakan ketika anggota dari superkelas tidak seluruhnya harus menjadi anggota subkelas, dimana jumlah dari subkelas tidak dibatasi.

Seperti halnya penggunaan opsi 1 pada kasus **Owner**, jika diselesaikan menggunakan opsi 2 maka akan terjadi pembuangan *space memory* pula, karena dalam Tabel OwnerDetail akan ada sel-sel yang berisi *null*, akibat dari merger atribut yang mewakili subkelas yang ada.

Gambar 2. Contoh *mapping Option* 2

Untuk menghitung jumlah space yang kosong dapat digunakan permisalan untuk Tabel OwnerDetail yaitu:

Jumlah *record* = n,
Jumlah kolom = x,
Jumlah kolom umum = u (digunakan hanya untuk *primary key* yang referensial dengan tabel P),
Jumlah kolom C_1 = y,
Jumlah kolom C_2 = x- (u+y),

Jika dimasukan sejumlah *record* C_1 = nC_1 dan dimasukan sejumlah *record* C_2 = nC_2, maka akan didapatkan suatu rumus untuk menghitung jumlah *space* yang terbuang (W):

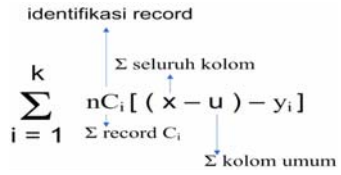
$$\mathbf{W = nC_1 * (x - (u + y)) + nC_2 * (y)}$$

Jadi jika ada 100 *record* (n), dengan
Jumlah *record* C_1 (nC_1) = 60, dan
Jumlah *record* C_2 (nC_2) = 40, dimasukkan dalam tabel P_Detail yang memliki
jumlah kolom (x) = 8,
jumlah kolom umum yang merupakan *foreign key* ke tabel P (u) = 1,
jumlah kolom C_1 (y) = 3, dan
jumlah kolom C_2 (x - (u+y)) = 4,

maka dapat dihitung jumlah sel yang akan terbuang atau berisi *null* adalah sebanyak 360 sel, dengan rumus:

$$\begin{aligned} W &= nC_1 * (x - (u + y)) + nC_2 * (y) \\ &= 60 * 4 + 40 * 3 \\ &= 240 + 120 \\ &= 360. \end{aligned}$$

Dari contoh di atas dapat disimpulkan bahwa jika opsi 2 tidak digunakan pada kasus yang tepat maka akan terjadi pembuangan *space memory* yang sia-sia. Meskipun pada opsi 2 dibuat satu tabel tambahan yang memuat *merger* seluruh atribut subkelas dan di tambah satu atribut yang digunakan atribut referensi (foreign key). Berdasarkan penggunaan opsi 1 dan 2 pada kasus Owner maka dihasilkan rumus untuk menghitung jumlah *cell* yang terbuang sia-sia yaitu:



dimana k adalah jumlah subkelas yang ada, dan y adalah perumpamaan kolom subkelas.

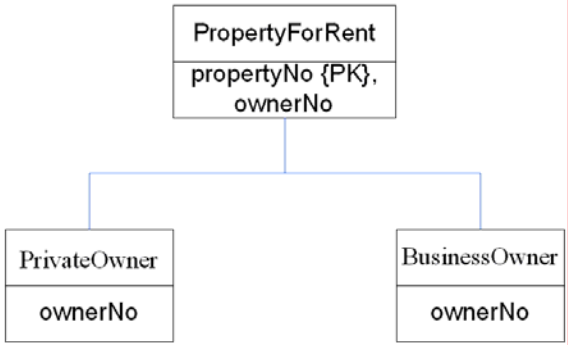
***Option* 3 (Mandatory Disjoint)**

```
PrivateOwner(ownerNo, fName, lName, address,
      telNo)
Primary Key ownerNo
BusinessOwner(ownerNo, bName, bType,
      contactName, address, telNo)
Primary Key ownerNo
```

Opsi 3 (Mandatory Disjoint) membutuhkan banyak relasi dimana setiap relasi adalah kombinasi superkelas atau subkelas. Oleh karena itu, pada kasus **Owner** hubungan antara superkelas **Owner** dengan **PrivateOwner** hanya dibentuk satu entitas **PrivateOwner** saja, begitu pula yang terjadi dengan **BusinessOwner**. Opsi ini digunakan ketika seluruh anggota dari entitas superkelas harus menjadi anggota dari hanya satu subkelas. Dan opsi ini berlaku jika superkelas memiliki lebih dari satu subkelas.

Pada opsi ini, hanya akan menyelesaikan masalah hubungan antar entitas subkelas itu sendiri, karena pada opsi ini mengkombinasikan entitas superkelas dan subkelas (menambahkan *primary key* superkelas pada setiap tabel subkelas), dan sekan-akan menghilangkan entitas superkelas sehingga entitas subkelas-subkelasnya seakan-akan berdiri sendiri, oleh sebab itu tidak ada integritas referensial yang dimiliki oleh kedua entitas tersebut. Namun kedua entitas tersebut memiliki *primary key* yang sama. Hal ini menimbulkan masalah kerancuan pada entitas yang berhubungan dengan kedua entitas tersebut.

Sebagai penjelasan lebih lanjutnya, pada kasus **Owner** yang memiliki dua subkelas yaitu **PrivateOwner** dan **BusinessOwner**, dengan menggunakan opsi ini maka seluruh anggota dari entitas superkelas yaitu **Owner** harus menjadi anggota dari salah satu subkelasnya dan setelah itu *primary key* entitas tersebut itu ditambahkan pada tabel subkelasnya sebagai *primary key* lalu entitas **Owner** dihilangkan. Dengan demikian entitas **PrivateOwner** mewakili hubungannya dengan superkelas dan **BusinessOwner** juga mewakili hubungannya dengan superkelas seolah-olah berdiri sendiri dengan *primary key* yang sama yaitu OwnerNo. Dengan demikian, dapat dilihat terdapat masalah pada hubungan kedua entitas tersebut dengan entitas yang berada di level diatasnya yaitu **PropertyForRent** yang pada awalnya berhubungan dengan **Owner**. Penyelesaian pada opsi ini tidak akan menyebabkan adanya *space memory* yang akan terbuang karena tidak ada atribut yang berisi *null*.



Gambar 3. Contoh *mapping option* 3

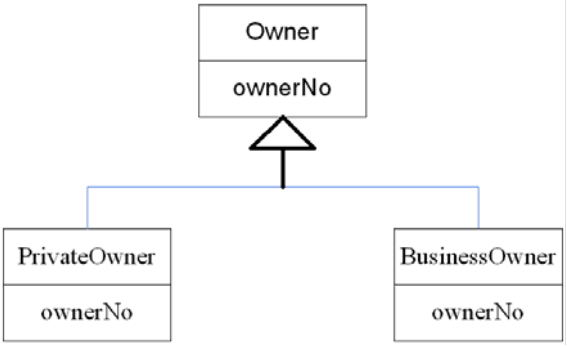
Dapat dilihat dari pemetaan di atas, terdapat masalah referensial antara ketiga tabel, karena tabel **PropertyForRent** menggunakan *foreign key* yang sama untuk mereferensial ke tabel **PrivateOwner** dan tabel **BusinessOwner,** namun datanya hanya mereferensial ke salah satu di antara kedua table tersebut. Hal ini menyebabkan adanya keilegalan dalam standar *statement* RDBMS (Relational Data Base Management Systems). Secara singkatnya, jika data dalam tabel **PropertyForRent** bereferensial dengan tabel **PrivateOwner** maka data dengan *primary key* yang sama tersebut tidak boleh berada dalam tabel **BusinessOwner**, begitu pula sebaliknya Karena pada opsi 3 ini melakukan pelanggaran pada standar *statement* RDBMS (Relational Data Base Management Systems) maka opsi ini sulit diterapkan dalam RDBMS (Relational Data Base Management Systems).

***Option* 4 (*Option*al Disjoint)**

```
Owner (ownerNo, address, telNo)
Primary Key ownerNo
PrivateOwner (ownerNo, fName, lName)
Primary Key ownerNo
Foreign Key ownerNo references Owner
      (ownerNo)
BusinessOwner (ownerNo, bName, bType,
      contactName)
Primary Key ownerNo
Foreign Key ownerNo references Owner
      (ownerNo)
```

Opsi 4 (*Option*al Disjoint) membutuhkan banyak relasi, dimana satu relasi dibutuhkan superkelas dan satu lagi untuk setiap subkelasnya. Pada opsi ini anggota dari entitas superkelas tidak harus seluruhnya menjadi anggota dari satu subkelas. Selain itu opsi ini berlaku jika superkelas memiliki lebih dari satu subkelas.

Pada opsi 4, dimisalkan hubungan antara entitas superkelas **Owner** dan entitas subkelasnya **PrivateOwner**, **BusinessOwner** yang dijadikan 3 tabel atau entitas terpisah, seperti yang terlihat pada Gambar 3.4. Sesuai dengan batasan partisipasinya, anggota dari **Owner** tidak perlu seluruhnya menjadi anggota **PrivateOwner** atau **BusinessOwner**, tetapi anggota **Owner** yang sudah menjadi anggota **PrivateOwner** tidak boleh menjadi anggota dari **BusinessOwner**. Seperti halnya penggunaan opsi 3, jika opsi 4 diterapkan untuk menyelesaikan kasus ini maka tidak akan terjadi pembuangan *space memory* yang sia-sia, akan tetapi menurut dari sifat datanya, kasus ini tidak cocok batasan partisipasinya jika diselesaikan dengan opsi 4, sebab akan menimbulkan kendala ketika melakukan manipulasi data.



Gambar 1. Contoh *mapping option* 4

Sebagai penjelasan lebih lanjut, pada kasus **Owner**, *primary key* yang terdapat pada tabel **PrivateOwner** dan **BusinessOwner** bereferensial dengan tabel **Owner**, oleh karena itu data yang ada di dalam tabel **PrivateOwner** dan **BusinessOwner** tergantung pada data dari tabel **Owner**.

Perbedaaan antara opsi 3 dan 4 adalah pada opsi 3 entitas subkelas berhubungan dengan entitas lain yang bukan merupakan superkelasnya, sementara pada opsi 4 entitas subkelas berhubungan dengan entitas superkelasnya langsung. Selain itu pada opsi 3, data dari tabel yang berada di level atas **PropertyforRent** sangat bergantung pada data subkelasnya (**PrivateOwner** dan **BusinessOwner**), sedangkan pada opsi 4, data subkelas-subkelasnya (**PrivateOwner** dan **BusinessOwner**) sangat bergantung dari data superkelasnya (**Owner**).

3. REKOMENDASI PENGUNAAN OPSI-OPSI

Option 1 - Mandatory Nondisjoint
Digunakan ketika seluruh anggota superkelas seluruhnya harus menjadi anggota dari subkelasnya, dimana jumlah subkelasnya boleh lebih dari satu.

Option 2 - Optional Nondisjoint
Digunakan ketika seluruh anggota superkelas tidak seluruhnya menjadi anggota dari subkelasnya, dimana jumlah subkelasnya boleh lebih dari satu.

Option 3 - Mandatory disjoint
Berlaku jika superkelas memiliki lebih dari satu subkelas. Digunakan ketika seluruh anggota superkelas seluruhnya harus menjadi anggota dari salah satu subkelasnya,

Option 4 - Optional disjoint
Berlaku jika superkelas memiliki lebih dari satu subkelas. Digunakan ketika tidak seluruh anggota superkelas seluruhnya menjadi anggota dari salah satu subkelasnya.

Dari rekomendasi di atas dapat dilihat pemilihan penggunaan opsi tergantung pada partisipasi data atau anggota superkelas terhadap subkelasnya serta batasan kepemilikan data antar subkelas-subkelas (Batasan Disjoint). Untuk memperjelas penggunaan dan kesimpulan pada rekomendasi berikut diberikan beberapa contoh kasus yang membutuhkan pertimbangan pemilihan opsi untuk menyelesaikan masalah generalisasi/spesialisasi.

4. DASAR PEMILIHAN OPSI

Berdasarkan tabel rekomendasi, maka jika terdapat kasus generalisasi/spesialisasi, hal yang harus dilakukan sebelum menentukan opsi yang akan digunakan, akan dijelaskan berikut ini beserta contoh-contoh kasusnya.

Kasus 1

Sebuah perusaahaan memiliki dua jenis staff yang dapat dibedakan menjadi staff *full-time* yang artinya bekerja sepanjang hari dan staff *part-time* atau staff yang bekerja setengah hari. Dari kasus di atas dapat dilihat bahwa:

- terdapat entitas **Staff**, **stafPartTime** dan **stafFullTime**
- entitas **Staff** adalah superkelas yang memilki subkelas **stafPartTime** dan **stafFullTime**
- **batasan partisipasi** data superkelas dalam subkelas adalah ***mandatory*** yang artinya seluruh staff harus seluruhnya menjadi anggota dari salah satu subkelanya
- dengan melihat jumlah subkelasnya maka jenis **batasan *disjoint*-nya** adalah ***disjoint***.
- staff yang terdaftar sebagai staff full time tidak dapat menjadi staff part time, dan begitu pula sebaliknya

Melihat dari deskripsi di atas maka solusi yang baik adalah **opsi 3** yang cocok untuk *mandatory disjoint*.

Kasus 2

Hubungan antara staff dan manager. Manager adalah staff, dan ada staff yang bekerja di bawah manager. Dari hubungan di atas dapat ditarik suatu kesimpulan:

- **Staff** dan **Manager** adalah sebuah entitas dari suatu hubungan relasional
- **Staff** adalah superkelas yang memiliki subkelas yaitu **Manager**.
- dari hubungan ini dapat dilihat bahwa seorang staff mungkin adalah seorang manager, namun tidak semua staff adalah seorang manager.
- **Batasan partisipasi** dari kasus di atas adalah ***optional***, karena tidak seluruh staff adalah manager tetapi manager adalah seorang staff
- Melihat dari jumlah subkelas-nya yang hanya satu maka dapat disimpulkan **batasan *disjoint*-nya** adalah **non-*disjoint***.

Melihat dari deskripsi di atas maka opsi yang cocok untuk kasus di atas adalah **opsi 2** yang cocok untuk *optional* non-*disjoint*.

Dari contoh pertimbangan pemililihan opsi untuk menyelesaikan kasus-kasus di atas maka dapat disimpulkan langkah-langkah yang harus dilakukan sebelum melakukan generalisasi dan spesialisasi adalah:

a. Identifikasi entitas awal
b. Identifikasi hubungan superkelas dan subkelas
c. Identifikasi entitas superkelas, dan entitas subkelas
d. Identifikasi jenis hubungan datanya
e. Tentukan batasan partisipasi keanggotaan superkelas terhadap subkelas-nya
f. Tentukan batasan *disjoint* dari superkelas dan subkelas-nya dari jumlah subkelas-nya
g. Identifikasi opsi dengan pertimbangan tabel rekomendasi.
h. Lakukan pemetaan sesuai dengan opsi yang dipilih.

Oleh karena melihat kendala integritas referensial dan integritas data yang akan dihadapi dalam penggunaan opsi 3 dan 4, dimana keduanya merupakan solusi terbaik untuk memecahkan masalah generalisasi/spesialisasi yang bersifat *mandatory disjoint* dan *optional disjoint*, maka kedua opsi tersebut dikembangkan dan dianalisis lebih lanjut

5. PENANGANAN OPSI 3 DAN OPSI 4 OLEH RDBMS

Pada opsi 1 tidak perlu pembatasan referensial integritas karena superkelas dan subkelasnya dimerger menjadi satu tabel, begitu pula pada opsi 2 tidak perlu di lakukan pembatasan referential integritas sebab tabel subkelasnya dimerger menjadi satu tabel yang mereferensi ke tabel superkelas. Sementara untuk opsi 3 dan 4 perlu dilakukan pembatasan referensial, karena pada opsi 3 dan 4 superkelas dan subkelas masing-masing berdiri sendiri.

Untuk itu maka pada tabel di bawah ini dilakukan analisis terhadap opsi 3 dan 4 pada RDBMS (Relational Data Base Management System).

Tabel 1. Analisis Opsi 3 dan 4

Skema Relasional	*Penjelasan*	*Masalah*
OPTION 3 **PropertyForRent** (propertNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo **Primary *Key*** propertyNo **Foreign key** staffNo **references** Staff (staffNo) **Foreign key** branchNo **references** Branch (staffNo) **Foreign key** ownerNo **references** BussinessOwner (OwnerfNo) **Foreign key** OwnerNo **references** PrivateOwner (OwnerNo) **PrivateOwner** (ownerNo, fName, lName, address, telNo) **Primary *Key*** ownerNo **BusinessOwner** (ownerNo, bName, bType, contactName, address, telNo) **Primary *Key*** ownerNo	Data dari **PropertyForRent** bergantung manipulasi data dari **PrivateOwner** dan **BussinessOwner** (keduanya). Data di **PrivateOwner** dan **BusinessOwner** HARUS sama (memiliki data yang *primary key*-nya yang sama) Artinya jika data dari kedua tabel belum mengalami manipulasi, maka data di tabel **PropertyForRent** juga TIDAK DAPAT mengalami manipulasi	Data dari tabel **PropertyForRent** hanya bergantung pada salah satu dari kedua tabel tersebut. Data di **PrivateOwner** dan **BusinessOwner** TIDAK BOLEH sama (memiliki data yang *primary key*-nya yang sama) Artinya jika salah satu dari **PrivateOwner** atau **BussinessOwner** mengalami manipulasi data maka data yang ada di **PropertyForRent** DAPAT mengalami manipulasi data
OPTION 4 **Owner** (ownerNo, address, telNo) **Primary *Key*** ownerNo **PrivateOwner**(ownerNo, fName, lName) **Primary *Key*** ownerNo **Foreign *Key*** ownerNo references Owner(ownerNo) **BusinessOwner** (ownerNo, bName, bType, contactName) **Primary *Key*** ownerNo **Foreign *Key*** ownerNo references Owner(ownerNo)	Data dari kedua tabel (**PrivateOwner** dan **BusinessOwner**) bergantung pada data di **Owner** Data di **PrivateOwner** dan **BusinessOwner** DAPAT memiliki data yang sama (memiliki data yang *primary key*-nya yang sama) Manipulasi data di **PrivateOwner** dan **BusinessOwner** tergantung pada manipulasi data yang ada di **Owner**	Data dari kedua tabel (**PrivateOwner** dan **BusinessOwner**) bergantung pada data di **Owner** Data di **PrivateOwner** dan **BusinessOwner** TIDAK DAPAT memiliki data yang sama (memiliki data yang *primary key*-nya yang sama) Manipulasi data di **PrivateOwner** dan **BusinessOwner** tergantung pada manipulasi data yang ada di **Owner**

Berdasarkan Tabel 1, opsi 3 dapat diimplementasikan di RDBMS (Relational Data Base Management System) jika nilai field ownerNo pada tabel **propertyforRent** ada di kedua tabel **PrivateOwner** dan tabel **BusinessOwner.** Hal ini melanggar ketentuan batasan opsi 3 yang mengharuskan field ownerNo dalam PropertyForRent hanya ada pada salah satu dari tabel **PrivateOwner** dan tabel **BusinessOwner.**

Sedangkan untuk opsi 4 dapat diimplementasikan di RDBMS (Relational Data Base Management System) jika nilai field ownerNo yang pada kedua tabel **PrivateOwner** dan tabel **BusinessOwner** ada di tabel **Owner,** namun dalam implementasi ini memungkinkan ownerNo yanng sama ada di **PrivateOwner** ada di **BusinessOwner**. Akan tetapi hal ini melanggar batasan yang ada di opsi 4 yang tidak memperbolehkan ownerNo yang sama berada di **PrivateOwner** dan **BusinessOwner**. Oleh karena itu hal ini harus diselesaikan. Dalam penelitian ini dirancang/dikembangkan dengan 2 cara yaitu memodifikasi konvensi *foreign key* dalam DDL (Data Definition Language) untuk *create table*.

Modifikasi DDL

Sintaks standar SQL untuk membuat suatu *foreign key* dalam *data definition language* (DDL) sebagaimana di bawah ini.

```
FOREIGN KEY (foreign key) REFERENCES
     [tabelname](primary key refered table)
     NULLS [NOT] ALLOWED
```

Untuk mengimplementasikan opsi3 maka sintaks SQL di atas harus diubah menjadi sebagaimana di bawah ini.

```
FOREIGN KEY (nama foreign key) REFERENCES
     [tabelname](primary key refered table)
     NULLS NOT ALLOWED [OR|AND]
REFERENCES [tabelname](primary key refered
     table) NULLS NOT ALLOWED
[ [OR|AND]
REFERENCES [tabelname](primary key refered
     table)]
```

6. SIMPULAN

Penanganan hubungan generalisasi/ spesialisasi sangat tergantung pada kebutuhan dan keadaan data yang dimiliki. Dengan mengetahui kondisi data dan karakteristik dari data maka kita dapat menentukan opsi mana yang paling tepat untuk kondisi tersebut.

Di samping itu adalah penting dilakukan perubahan atas sintaks SQL agar dapat mengakomodasi kebutuhan adanya data yang harus memenuhi opsi ketiga. Dengan demikian maka *database designer* dapat melakukan pekerjaan mereka sesuai dengan kebutuhan yang ada.

PUSTAKA

[1] Connolly, Thomas M. Begg, Carolyn E. (2002). *Database Systems: A Practical Approach to Design, Implementation, and Management*. edisi ke-3. Pearson Education. Inc, USA.

[2] Date, C.J. (1983). *An Introduction to Database Systems*. edisi ke-1. Addison-Wesley Publishing Company. Inc, USA.

[3] Hoffer, Jeffrey A., Prescott, Mary B., McFadden, Fred R. (2005). *Modern Database*. edisi ke-7. Pearson Education. Inc, USA.

[4] Mannino, Michael V. (2001). *Database Application Development & Design*. McGraw-Hill, USA.

[5] Silberschatz, Abraham, Korth, Henry F., Sudarshan, S. (2002). *Database System Concepts*. edisi ke-4. McGraw-Hill, USA.