OBJECT ORIENTED DATABASE

Advanced Database Applications

Computer-Aided Design (CAD)
Database CAD menyimpan data yang berhubungan dengan rancangan mekanik dan elektrik, sebagai contoh : gedung, pesawat, dan chips IC.

Computer-Aided Manufacturing (CAM)
Database CAM menyimpan data yang jenisnya sama dengan sistem CAD, ditambah data yang berhubungan dengan produksi yang mempunyai ciri-ciri tersendiri (seperti mobil pada saat perakitan) dan produksi yang kontinyu (seperti sintesa kimia)

Computer-Aided Software Engineering (CASE)
Database CASE menyimpan data yang berhubungan dengan langkah-langkah dari siklus pengembangan software yaitu : *planning, requirements collection analysis, design, implementation, test, maintenance and documentation*.

Office Automation (OA)
Database OA menyimpan data yang berhubungan dengan pengontrolan informasi komputer dalam bidang bisnis, termasuk e-mail, dokumen-dokumen, invoice, dsb. Agar menyediakan dukungan yang lebih baik untuk area ini, dibutuhkan penanganan yang lebih luas terhadap jenis data daripada nama, alamat, tanggal dan uang. Sekarang ini sistem yang modern dapat menangani text yang berjenis bebas, foto, diagram, audio dan video. Sebagai contoh : dokumen multimedia yang mengangani teks, foto, spreadsheets dan suara.

Computer-Aided Publishing (CAP)
Database CAP menyimpan dokumen yang kompleks. Sama seperti otomatisasi kantor, applikasi CAP telah diperluas untuk menangani dokumen-dokumen multimedia yang berisikan teks, audio, gambar, video data, dan animasi.

Keterbatasan Relational DBMS

Representation of "Real World" entities
Proses normalisasi pada umumnya akan membuat relasi yang tidak berhubungan dengan entitas pada "dunia nyata". Fragmentasi dari entitas "dunia nyata" ke dalam beberapa relasi, dengan sebuah representasi fisik akan menggambarkan struktur yang tidak efisien karena akan melakukan banyak join selama proses query.

Semantic overloading
Model relasional hanya mempunyai satu konstruksi untuk menggambarkan data dan keterhubungannya. Sebagai contoh : untuk menggambarkan keterhubungan M : N dari entitas A dan B, kita membuat 3 relasi, satu untuk menggambarkan masing-masing entitas A dan B, dan satu menggambarkan keterhubungannya (*relationship*). Tidak ada

mekanisme untuk membedakan antara entitas dan *relationship*, atau untuk membedakan antara jenis *relationship* yang berbeda yang terdapat di antara entitas. Sebagai contoh : *relationship* 1 : M dapat berarti memiliki, mempunyai, mengatur, dsb. Jika perbedaan seperti itu dapat dibuat, maka mungkin dapat membangun semantik pada operasi-operasi tersebut. Oleh karena itu model relasional merupakan *semantically oevrloaded*.

Integrity constraints and enterprise constraints
Integrity berhubungan dengan validitas dan kekonsistenan dari data yang disimpan. Integrity biasanya diekspresikan dalam istilah-istilah yang berhubungan dengan constraint, dimana aturan-aturan kekonsistenan pada database tidak boleh dilanggar. Sangat disayangkan bahwa banyak sistem komersial tidak mendukung batasan-batasan itu, dan sistem tersebut lebih mengutamakan membuat batasan-batasannya di dalam aplikasi. Tentu saja hal ini sangat berbahaya dan dapat menimbulkan duplikasi data, dan lebih buruk lagi data berada dalam keadaan tidak konsisten. Lebih jauh lagi, tidak ada dukungan untuk aturan-aturan enterprise dalam model relasional, yang artinya aturan-aturan tersebut harus dibangun ke dalam DBMS atau aplikasi.

Homogeneous data
Model relasional mengasumsikan homogenitas horisontal dan vertikal. Homogenitas horisontal berarti bahwa setiap tuple dari sebuah relasi harus disusun dari atribut-atribut yang sejenis. Homogenitas vertikal berarti bahwa nilai di dalam kolom dari sebuah relasi harus berasal dari domain yang sama (sejenis). Dan irisan (intersection) antara baris dan kolom harus bernilai atomik. Struktur yang tetap seperti ini sangat terbatas untuk beberapa objek “dunia nyata” yang mempunyai struktur kompleks. Sebagai contoh : pesawat terbang.

Limited operations
Model relasional merupakan himpunan (set) yang sudah tetap, seperti operasi himpunan (set) dan operasi *tuple-oriented*. Operasi-operasi ini disediakan oleh SQL.

Recursive Query
Recursive query merupakan proses *query* dari relationship yang berelasi dengan dirinya sendiri (baik secara langsung atau tidak langsung). Sebagai contoh : terdapat relasi sederhana dari relasi Staff yang menyimpan nomor staf dan manajer-nya.

Staff_No	Manager_Staff_No
S5	S4
S4	S3
S3	S2
S2	S1
S1	Null

Bagaimana mencari para manajer yang secara langsung atau tidak langsung mengatur (manage) anggota staf no S5 ?

Terdapat 2 tingkat hirarki :

```
SELECT manager_staff_no FROM staff WHERE staff_no = 'S5'
UNION
SELECT manager_staff_no FROM staff WHERE staff_no =
            (SELECT manager_staff_no FROM staff WHERE staff_no = 'S5');
```

Di dalam contoh ini, pendekatan ini berhasil karena kita mengetahui berapa tingkatan dalam hirarki yang harus diproses. Tetapi jika untuk mendapatkan hasil yang lain seperti "untuk setiap anggota staff, tampilkan seluruh manajer baik yang secara langsung ataupun tidak langsung mengatur (manage) staf tersebut", maka pendekatan seperti di atas akan lebih sulit untuk diimplementasikan dengan menggunakan SQL. Untuk mengatasi masalah ini, SQL dapat ditempelkan (embedded) pada bahasa pemrograman tingkat tinggi, yang menyediakan konstruksi untuk fasilitas iterasi. Sebagai tambahan, banyak sistem relasional yang menyediakan *report writer* dengan konstruksi yang mirip.

Impedance mismatch
Perhitungan dengan menggunakan SQL kurang lengkap. Untuk mengatasi hal ini, SQL standar menyediakan embedded SQL untuk membangun aplikasi database yang lebih kompleks. Tetapi pendekatan ini menghasilkan sebuah "*Impedance mismatch*" karena kita mencocokan paradigma pemrograman yang berbeda. SQL merupakan sebuah bahasa yang menangani baris (record), sedangkan sebuah bahasa tingkat tinggi seperti C merupakan bahasa prosedural yang dapat menangani hanya 1 baris pada suatu saat. Kedua : SQL dan 3 GLs menggunakan model-model yang berbeda untuk menggambarkan data. Sebagai contoh : SQL bisa berisikan jenis data *date* dan *interval* yang tidak tersedia di dalam bahasa pemrograman tradisional. Oleh karena itu merupakan hal yang sangat penting bagi program aplikasi untuk mengkonversi dua tipe data tadi, sehingga menjadi tidak efisien. Lebih jauh lagi, karena kita menggunakan 2 tipe sistem yang berbeda, maka sangat tidak mungkin secara otomatis menguji aplikasi secara keseluruhan.

Object Oriented Concepts

Abstaraction and Encapsulation

Abstraction :
proses identifikasi aspek-aspek yang perlu (*essential*) dari entitas dan mengabaikan *property* yang tidak penting.

Encapsulation (information hiding) :
memisahkan aspek-aspek eksternal sebuah objek dari rincian internalnya (*internal details*), yang tidak terlihat dari dunia luar. Dengan cara ini, *internal detail* sebuah objek dapat dirubah tanpa mempengaruhi aplikasi yang menggunakan objek tersebut, begitu juga dengan *external detail*. Dengan kata lain, *encapsulation* menyediakan *data independence*.

Objects and Attributes

Object :
Sebuah entitas yang dapat diidentifikasikan secara unik, berisikan atribut-atribut yang menerangkan keadaan atau kondisi (*state*) objek dunia nyata (*real world object*) dan aksi-aksi yang berhubungan dengan sebuah objek dunia nyata. Definisi objek serupa dengan definisi entitas. Perbedaannya : objek menunjukkan keadaan (*state*) dan tingkah laku (*behaviour*), sedangkan entitas menunjukkan *models state*.

Current state dari sebuah objek digambarkan dengan satu atau lebih *attribute* (*instance variables*). Sebagai contoh : kantor cabang di *163 Main Street* dapat memiliki atribut yang terlihat pada tabel berikut :

Tabel 1. *Object attributes for branch instance*

BNO	B3
STREET	163 Main St
AREA	Partick
CITY	Glasgow
POST_CODE	G11 9QX
TEL_NO	0141-339-2178
FAX_NO	0141-339-4439
SALES_STAFF	Ann Beech; David Ford
MANAGER	Susan Brand

Atribut dapat dikelompokkan menjadi atribut simpel dan atribut kompleks. Atribut simpel dapat berupa tipe primitif seperti integer, string, real, dsb. yang mengambil nilai literal. Sebagai contoh : Bno pada tabel 1 merupakan atribut simpel dengan nilai litaral ‘B3’. Sedangkan atribut kompleks dapat berisikan kumpulan (koleksi) dan / atau referensi. Sebagai contoh : atribut Sales_staff merupakan kumpulan dari objek staff.

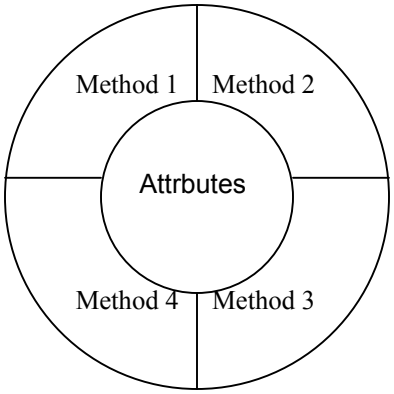
Atribut referensi (reference attribute) menggambarkan keterhubungan antar objek. Atribut referensi berisikan sebuah nilai atau kumpulan nilai yang merupakan objeknya sendiri. Sebagai contoh : Sales_staff lebih tepatnya merupakan kumpulan referensi yang menunjuk kepada objek staf. Atribut referensi secara konseptual dama seperti foreign key di dalam model data relasional atau seperti sebuah pointer dalam bahasa pemrograman.

Atribut secara umum biasanya menggunakan notasi ‘dot’. Sebagai contoh : atribut street dari objek branch : *branch_object.street*

Object Identity
Pada saat objek dibuat, *object identifier* (OID) langsung ditentukan. OID tersebut unik dan berbeda. OID membedakan objek yang satu dengan objek lainnya di dalam sistem. Sekali objek dibuat, OID tersebut tidak dapat digunakan kembali untuk objek-objek lainnya, walaupun objek tersebut telah dihapus.

Methods and Message

Methods
Dalam teknologi objek, *function* biasanya disebut *methods*. Contoh : gambar 1, memberikan suatu gambaran konseptual sebuah objek dengan atribut didalamnya, dan *methods* di bagian luar. *Methods* mendefinisikan tingkah laku dari sebuah objek. *Methods* dapat digunakan untuk merubah kondisi objek dengan memodifikasi nilai atribut-atributnya, atau meng-*query* nilai atribut yang diseleksi. Sebagai contoh kita dapat menggunakan *methods* untuk menambah *property* baru untuk disewa pada sebuah cabang, merubah gaji pegawai atau mencetak detail pegawai.



Gambar 1. Object showing attributes & methods

Sebuah *method* berisikan nama dan *body* yang membentuk tingkah laku yang berhubungan dengan nama *method*. Pada bahasa berorientasi objek, *body* berisikan kode blok yang melaksanakan aksi. Sebagai contoh berikut ini menggambarkan *method* untuk merubah gaji pegawai. Nama *method*-nya adalah *update_salary*, mempunyai parameter inputnya *increment* yang menambahkan ***instance variable*** *salary* untuk menghasilkan gaji baru

```
Method void update_salary(float increment)
{
        salary = salary + increment
}
```

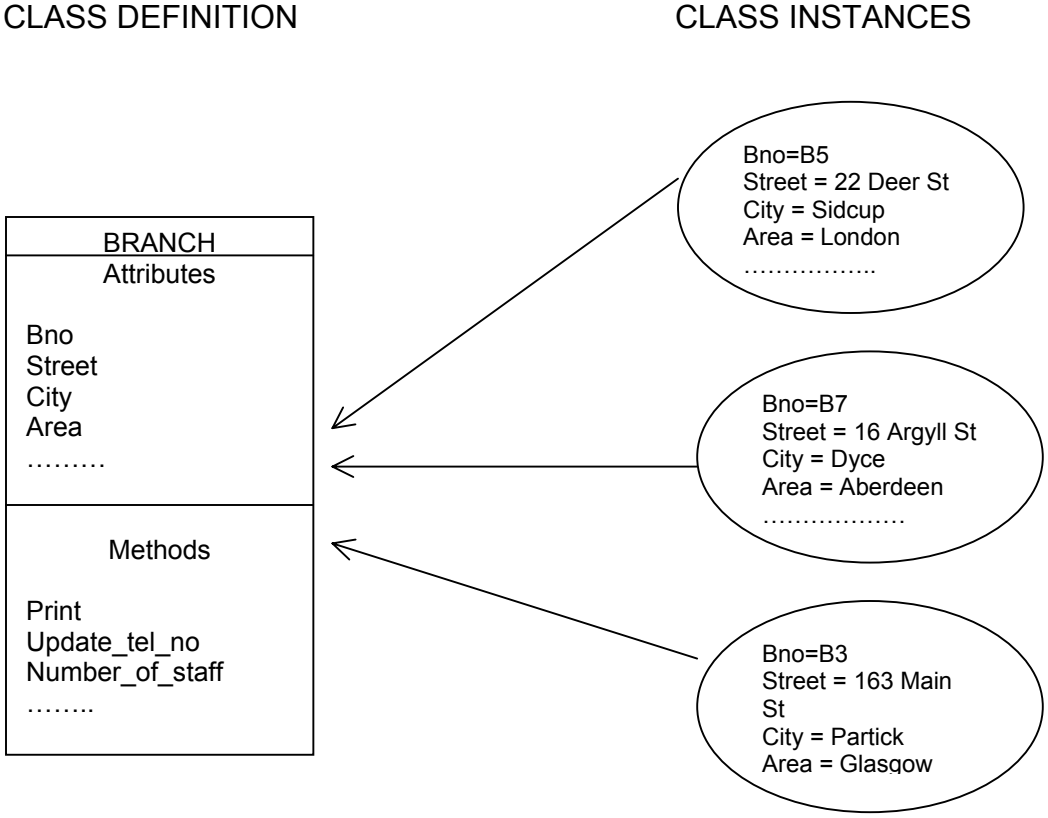
Message
Message mempunyai arti komunikasi antara objek. Sebuah *message* merupakan permintaan sederhana dari suatu objek (pengirim) ke objek lain (penerima) dan menanyakan objek tsb untuk mengeksekusi salah satu *method*-nya. Pengirim dan penerima bisa pada objek yang sama. Notasi ‘dot’ biasanya digunakan untuk mengakses sebuah *method*. Contoh : untuk mengeksekusi *method update_salary* dari objek staff dan masukkan *method* dengan pertambahan nilai 1000.

```
staff_object.update_salary(1000)
```

Pada bahasa pemrograman tradisional, sebuah *message* ditulis seperti *function call* :

update_salary(staff_object, 1000)

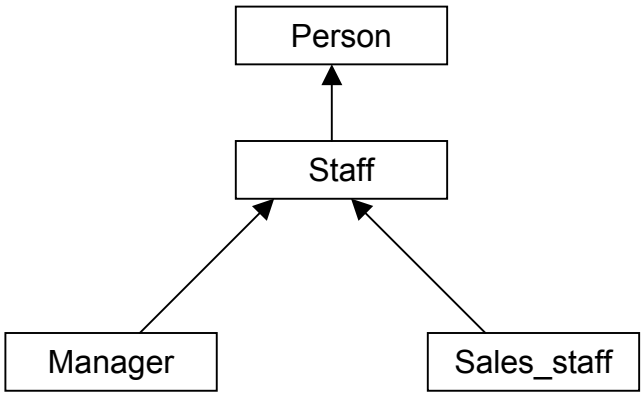
Class
Class merupakan pendefinisian himpunan objek yang sejenis. Objek yang mempunyai atribut yang sama dan meresponse *message* yang sama dapat dikelompokkan bersama membentuk sebuah *class*. Atribut dan *method* yang berhubungan cukup sekali saja didefinisikan untuk *class*, daripada didefinisikan terpisah untuk setiap objek. Contoh : seluruh objek cabang dideskripsikan oleh sebuah class cabang (branch). Objek-objek pada sebuah class disebut *instance dari class*. Setiap *instance* mempunyai nilainya sendiri untuk setiap atribut, tetapi nama atribut dan method-nya sama seperti *instance* lainnya dari sebuah *class*. Contoh : gambar 2.



Gambar 2. Class instances share attributes and methods

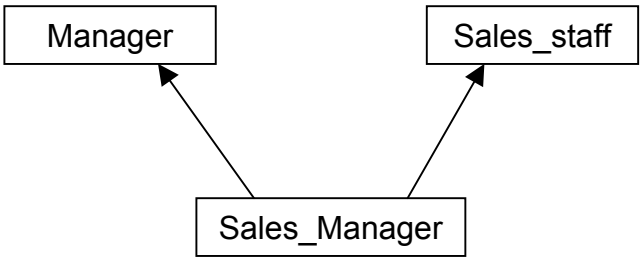
Subclass, Superclass, and Inheritance
Inheritance mengizinkan satu class objek didefinisikan sebagai kasus spesial (*special case*) dari sebuah *class* pada umumnya. *Special case* ini dikenal dengan *subclass*, dan kasus umum lainnya dikenal sebagai *superclass*. Proses pembentukan *superclass* sama seperti *generalization*, sedangkan *subclass* seperti *specialization*. Konsep dari *superclass, subclass*, dan *inheritance* sama seperti EER, kecuali dalam paradigma *object-oriented*, *inheritance* meliputi *state* dan *behaviour*. Relationship antara subclass dan superclass kadang-kadang dituliskan seperti **A KIND OF (AKO)**, sebagai contoh : a Manager is AKO staff. Relationship antara sebuah instance dan class-nya dituliskan **IS-A** ; contoh : Susan Brand IS-A Manager.

Ada beberapa bentuk inheritance :
1. Single inheritance
   Subclass merupakan turunan dari satu superclass.
   Contoh : subclass Manager dan Sales_Staff merupakan turunan property dari superclass Staff.



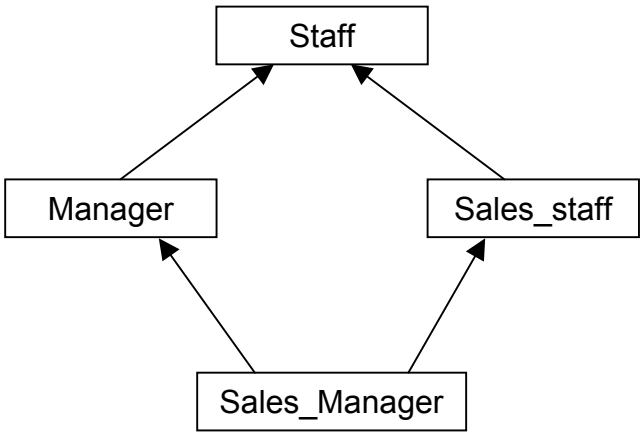
Gambar 3. Single inheritance

2. Multiple inheritance
   Subclass Sales_Manager merupakan turunan dari superclass Manager dan Sales_Staff.



Gambar 4. Multiple inheritance

3. Repeated inheritance
   Kasus spesial dari multiple inheritance, dimana sebuah superclass merupakan turunan dari sebuah superclass biasa. Melanjutkan contoh multiple inheritance, class Manager dan Sales_staff bisa saja merupakan turunan dari superclass biasa yaitu superclass Staff. Dalam kasus ini, mekanisme inheritance harus meyakinkan bahwa class Sales_manager tidak diturunkan sebanyak dua kali dari superclass Staff.



Gambar 5. Repeated inheritance

4. Selective inheritance
   Mengizinkan subclass menurunkan sejumlah *property* dari *superclass*. Keistimewaan ini secara fungsional sama seperti mekanisme *view*, dengan membatasi akses ke beberapa detail tapi tidak seluruhnya.