KONSEP SISTEM MANAJEMEN BASIS DATA BERORIENTASI OBJEK (OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM/ OODBMS)

Oleh : Yusup
Fakultas Ilmu Komputer, Universitas AKI Semarang

Abstract

This technology integrates the capability of database (DBMS) with the ones of object oriented program (OPP). Object Oriented Database Management System (ODBMS) makes the object of database is visible as the object of program at various OOP program. ODMBS is able to widen the capability of program terminologies with the capability of database as transparancy persistent data, competition control, data recovery, associate query and many other database capabilities.

Keywords : OODB, OODBMS.

Pendahuluan

Objek merupakan kesatuan entitas (baik), baik yang berwujud nyata ataupun hanya satu sistem yang memodelkan dunia nyata. Setiap object diidentifikasi oleh object identifier(OID), dan juga memiliki state dan behavior. State terdiri dari nilai object properties. Properti dari sebuah object dapat berupa atribut atau relasi antar object. Sedangkan behavior dispesifikasikan oleh operasi atau method yang dapat dieksekusi oleh sebuah object melalui propertinya.

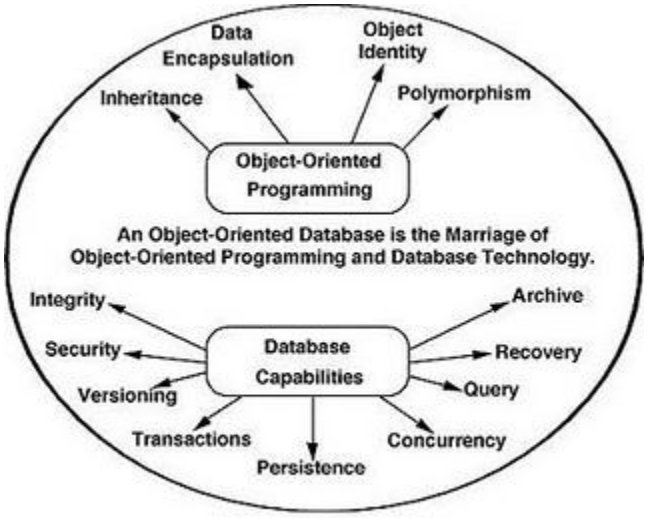

Figure 1. Makeup of an Object-Oriented Database

Karakteristik Objek

Sebuah object, mempunyai:

1. identifier : unique id
2. name : unique name dalam DB (optional)
3. lifetime : menetapkan apakah object persistent atau transient
4. structure : pembangunan object menggunakan type constructors

Struktur Objek

State (current value) dari object bias dibangun dari object lain (other values) dengan menggunakan type constructors tertentu.

Constructors :

Basic types : atom, tuple dan set

Collection type : list, bag dan array

Object Identity

Hal yang paling penting dari properti sebuah object yaitu memiliki *identity*. *Object Identifier (OID)*biasanya tidak terlihat secara langsung dan tidak dapat diakses oleh database *user*. *OID*direpresentasikan dengan sebuah nama yang unik atau lebih. Berbeda dengan *primary key* pada relational database, OID bersifat *independent* dari nilai yang dihasilkan *object state*. OID yang digunakan pada object-oriented database digunakan untuk mengidentifikasi object dan mensupport hubungan antar object melalui nilai *object properties*.

Object State

Pada *object-oriented database* , nilai dari sebuah object terdiri dari nilai dari beberapa atribut dan relasi antar object. Nilai dari suatu atribut dapat bersifat kompleks,jadi sebuah atribut dapat dibentuk dari object lain atau menggunakan tipe data yang telah tersedia. Relasi merupakan hubungan antara dua object atau lebih. Object model hanya mendukung *binary relationship,* relasi antar dua object. Sedangkan untuk mengimplementasikan hubungan satu ke banyak atau banyak ke banyak dapat menggunakan *set atau list.*

Object Behavior

Object dalam object-oriented database dimanipulasi melalui *methods. Method* adalah prosedur atau fungsi yang dimiliki oleh sebuah object dimana sebuah *method* akan mengolah/mengubah data yang terdapat didalam object sesuai dengan operasi yang telah ditentukan.

Kelas

Kelas merupakan pemodelan dari sebuah object yang berisi informasi (aturan) tentang sifat karakteristik (data) dan behavior (method) yang dimiliki oleh object tersebut.

Object types

Object types dapat digunakan untuk membuat *object tables.* Dalam *object tables,* setiap *instance* dari*tuples* memiliki *obj ect identifier*. *Object types* yang digunakan

untuk membuat *object tables* dapat digunakan sebagai tipe atribut untuk menghubungan antara tabel. Oracle mensupport beberapa tipe data baru yaitu :

1. Varrays atau Nested Table

Dua tipe data baru yang memperbolehkan koleksi data menjadi tipe data dari sebuah atribut.

2. REFs (object references)

Digunakan untuk menyimpan *logical pointer* pada object.

Pendekatan berorientasi objek adalah cara memandang persoalan menggunakan model-model yang diorganisasikan seputar konsep objek yang mengkombinasikan struktur data dan perilaku suatu entitas.

Coud-Yourdan mendefinisikan pendekatan berorientasi objek dengan persamaan :

Berorientasi Objek = Objek + Klasifikasi + Pewarisan + Komunikasi.

Konsep-konsep yang penting dalam Orientasi Objek antara lain :

1. **Kelas**

   Kelas merupakan suatu **tipe** baru yang didefinisikan oleh programmer. Dengan kata lain, Kelas adalah cetakan objek, suatu kelas bisa memiliki banyak objek. Setiap kelas memiliki atribut dan method. Kelas merupakan deskripsi satu objek atau lebih dengan sekumpulan atribut dan layanan yang seragam, termasuk deskripsi penciptaan objek baru di kelas itu. Kelas merupakan blok pembangun di pendekatan berorientasi objek. Perlu menspesifikasikan semantik dari kelas. Kelas adalah tipe data abstraks dilengkapi dengan implementasi parsial atau total. Perbedaan kelas dan objek adalah objek merupakan entitas konkrit yang ada secara ruang dan waktu sedangkan kelasa hanya merupakan representasi abstraksi. Objek adalah bukan kelas, meski kelas boleh jadi menjadi objek.

2. **Abstraksi**

   Abstraksi merupakan cara manusia yang paling dasar dalam menangani kompleksitas. Abstraksi adalah kemampuan manusia untuk mengenali keserupaan di antara objek-objek, situasi-situasi, atau proses-proses di dunia nyata serta keputusan untuk berkonsentrasi pada keserupaan-keserupaan dan mengabaikan yang disebut sebagai perbedaan-perbedaan kecil yang ada. Abstraksi adalah prinsip mengabaikan aspek-aspek dari subjek yang tidak relevan dengan

maksud tertentu untuk berkonsentrasi lebih penuh padanya.

Abstraksi adalah pemodelan dunia nyata (real), fokus pada kualitas-kualitas esensi, mengabaikan rincian-rincian, penekanan pada "apa" bukan "bagaimana".

3. **Modularitas**

   Modularitas merupakan sesuatu yang esensi untuk memecah program menjadi modul-modul kecil dimana masing-masing modul saling berinteraksi dengan lainnya lewat antar muka sempit yang terdefinisi bagus. Modularitas berhubungan dengan pengabstraksian. Masing-masing modul seharusnya berkorespondensi dengan satu abstraksi dan hanya satu-satunya.

4. **Enkapsulasi (penyembunyian informasi-informasi hiding)**

   Enkapsulasi adalah pemisahan aspek-aspek eksternal objek yang dapat diakses dari rincian-rincian implementasi internal. Enkapsulasi meredam ketergantungan yang begitu besar dengan objek kelas lainnya. Pengkapsulan sering diperoleh lewat penyembunyian informasi, yaitu proses penyembunyian semua rahasia objek yang tidak berkontribusi pada esensi. Umumnya struktur dari objek adalah tersembunyi, juga implementasi dari metode, yang tampak hanyalah layanan-layanan yang dapat diminta dari objek.

5. **Pewarisan**

   Pewarisan atau *Inheritance* adalah proses penciptaan kelas baru dengan mewarisi karakteristik kelas yang sudah ada, ditambah karakteristik unik kelas yang baru itu. Pewarisan mendukung penggunaan kembali (*reusability*) suatu kode. Pewarisan merupakan sarana untuk menghilangkan penulisan ulang terhadap kode yang dapat digunakan berulang kali yang menyiakan banyak waktu, menimbulkan ketidakkonsistenan, dan meningkatkan resiko menimbulkan kesalahan. Pewarisan memberi fasilitas untuk menstrukturkan kelas menjadi lebih ringkas, apa yang serupa dan apa yang berbeda di antara kelas-kelas. Pewarisan adalah abstraksi ampuh untuk berbagi keserupaan di antara kelas-kelas sambil tetap menjaga perbedaan-perbedaan di kelas-kelas itu.

6. **Polimorfisme**

   Secara harafiah, Polimorfisme berarti banyak bentuk. Dalam konsep OO, objek-objek dikatakan polimorfik bila mempunyai antarmuka yang identik, namun mempunyai perilaku yang berbeda. Contoh mudahnya adalah dua buah objek dari kelas yang berbeda dapat memiliki nama method yang sama, namun algoritma methodnya berbeda. Polimorfisme memungkinkan untuk mengenali dan mengeksploisasi keserupaan-keserupaan di antara kelas-kelas berbeda.

Object Oriented Database Management System

Istilah OODB pertama kali muncul pada tahun 1985. Proyek-proyek yang terkenal dari perkembangan OODB ini antara lain:

- Encore-Ob/Server (Brown University)
- EXODUS (University of Wisconsin-Madison)
- IRIS (HP)
- ODE (Bell Labs)
- ORION (Microelectronic and Computer Technology Corporation)

Produk komersial awal yang memakai konsep OODB ini adalah Gemstone, Gbase, dan Vbase. Pada awal sampai pertengahan tahun 1990-an, berkembang produk komersial lain dan produk-produk baru lainnya terus berkembang menggunakan OODB ini.Untuk ODBMS sendiri, konsep yang ditekankan adalah persistensi yang dimiliki oleh bahasa pemograman objek. Produk komersial awalnya diintegrasikan dengan bahasa yang bermacam-macam seperti GemStone memakai bahasa Smalltalk, Gbase yang menggunakan LISP, Vbase menggunakan COP, dan lain sebagainya. Di tahun 1990-an, C++, Java, dan C# mendominasi pasaran ODBMS.

Objek database mulai populer pada pertengahan tahun 1990 an. Bermula dari Objek Oriented Programming(OOP) yang kemudian dikembangkan menjadi Objek Oriented Design (OOD) dan pada akhirnya menjadi Objek Oriented Analysis (OOA). Didalam konsep objek oriented database kita dapat melakukan pemodelan data dari semua phenomena dan dapat dinyatakan dalam bahasa umum (natural). Objek Oriented Database pada dasarnya merupakan kosep dari pemrograman berorientasi objek secara umum ditambah dengan database sebagai media penyimpanan datanya yang berbentuk class-class, sehingga dalam hal ini masih berhubungan erat dengan E-R Model.

OODB muncul karena kekomplekan dari penyimpanan objek-objek yang akan disimpan didalam database sehingga konsep

dari Relational Database Manejemen Sistem (RDBMS) masih tetap digunakan. Mekanisme penyimpanan objek-objek didalam Relational Database Menejemen Sistem ini sering dikenal dengan istilah ORDBMS (Objek Relational Database Managemen System).

Kenapa OODB ?

Dilihat dari sisi industry, OODB memberikan pelayanan integrasi data dan data sharing. OODB adalah kombinasi terintegrasi yang baik dari operating sistem, basis data, bahasa pemograman, spreadsheets, word-processor, dan expert system intelegensia buatan. OODB memungkinkan *Referential Sharing*, yaitu aplikasi, produk,atau objek yang berbagai macam yang bisa dibagi menjadi sub-object. Referential sharing inilah yang mendukung diterapkannya identitas objek dan pewarisan.

Mengapa dibutuhkan Objek Oriented DBMS ?

Banyak sekali beberapa keuntungan (benefit) yang diberikan apabila menggunakan konsep dari objek oriented database ini ketimbang menggunakan konsep terstruktur, keuntungan-keuntungan itu diantaranya: context sistem yang dibangun akan memberikan informasi yang jelas, mengurangi biaya maintenance, komponen sistem lebih independent artinya lebih mudah jika dilakukan perubahan dan perawatan tanpa mengganggu kinerja komponen sistem yang lain, antara pengguna dan pengembang dapat saling berkomunikasi sepanjang pembuatan sistem karena sistem yang dibangun biasanya dilengkapi dengan bebarapa diagram (usecase diagram, activity diagram, sequence diagram, class diagram, object diagram, dan lain-lain) dan visualisasi yang menjelaskan aktifitas-aktifitas yang dilakukan, dengan objek oriented lebih mencerminkan bagaimana kita dapat menguraikan atau memecahkan sistem yang komplek, mendukung semua aplikasi tidak seperti halnya RDBMS

Object Oriented Database adalah sebuah sistem database yang menggabungkan semua konsep penting dari object oriented tersebut dengan beberapa fitur tambahan antara lain :

1. Unique object identifiers
2. Persistent object handling

Teknologi ini mengintegrasikan kemampuan basis data (DBMS) dengan kemampuan pemrograman berorientesi objek (OOP). Sebuah Object Oriented Database Management System (ODBMS) membuat objek sebuah basis data terlihat seperti objek pemrograman pada beberapa bahasa pemrograman OOP. Sebuah ODBMS dapat memperluas kemampuan bahasa pemrograman dengan kemampuan basis data seperti data yang persistent secara transparan,

kontrol konkurensi, recovery data, query asosiatif dan berbagai kemampuan basis data yang lain.

OODB atau ODBMS dirancang untuk bekerja pada bahasa pemrograman berorientasi objek seperti Java, C++ dan lain-lain. Bila kita ingin menyimpan objek pada program Java atau C++ ke dalam sebuah sistem basis data, kita dapat menggunakan basis data yang berorientasi kepada objek (ODBMS).

ODBMS sangat memudahkan programmer terutama yang terbiasa dengan OOP dalam mengolah data dan variabel dalam programnya. Kebanyakan programmer basis data menghabiskan cukup banyak waktu untuk merepresentasikan variabel atau objek pada programnya ke dalam struktur basis data.

Letak perbedaan utama ODBMS dengan sistem basis data konvensional adalah pada sistem basis data konvensional data direpresentasikan ke dalam bentuk tabel-tabel dengan kolom yang mewakili kategori dari data, dan baris yang berisi data itu sendiri. Sedangkan dalam ODBMS, data direpresentasikan sebagai sebuah objek, baik dalam hal pengaksesannya maupun dalam hal pemodelannya.

Struktur Object di OODB

Paradigma OODB didasarkan pada enkapsulasi data dan kode. Seperti yang telah diketahui, enkapsulasi menyebabkan isi dari sebuah objek tidak terlihat di dunia luar objek tersebut. Secara konsep, semua interaksi antara objek dan sistem di luar dirinya dilakukan via*messages.* Interface antara objek dan sistem di luar dirinya inilah yang dinamakan kumpulan dari pesan (*set of messages*).

Secara umum, sebuah objek berasosiasi dengan

1. Sebuah himpunan variable (a set of variables) yang mengandung data untuk objek dan variable yang berkorespondensi dengan atribut di model E-R.
2. Sebuah himpunan pesan (a set of variables); kesinilah objek member respon. Tiap pesan bisa mempunyai 0, 1 atau banyak parameter.
3. Sebuah himpunan method (a set of variables) ; masing-masingnya adalah badan kode untuk mengimplementasikan pesan. Method mengembalikan nilai sebagai respon pesan.

Diilustrasikan dengan mengambil contoh entitas *employee* dalam basis data bank. Misalnya gaji tahunan karyawan dihitung dengan cara yang berbeda untuk tiap karyawan. Manajer mendapatkan bonus sesuai dengan performa bank, sedangkan teller mendapat bonus berdasarkan jumlah

jam mereka bekerja. Dalam kasus ini, semua objek *employee* berhubungan dengan message gaji-tahunan, tapi mereka dihitung dengan cara yang berbeda. Dengan mengenkapsulasi informasi bagaimana mendapatkan gaji tahunan karyawan dalam objek*employee*, semua objek *employee* bisa menampilkan interface yang sama. Karena hanya interface external yang ditampilkan oleh sebuah objek adalah *a set of message* yang berhubungan dengan objek tersebut, maka dimungkinkan untuk memodifikasi definisi method dan variable tanpa mempengaruhi sisa sistem lain.

Dalam OODB, kita mengekspresikan atribut turunan dari entitas model E-R sebagai *read-only message*. Read-only message tidak mempengaruhi nilai dari variable dalam objek. Setiap atribut pada entitas HARUS mempunyai variable dan pasangan messages dari objek yang berhubungan dalam model OO. Misalnya atribut address dari entitas employee bisa direpresentasikan dengan:

- Variable address
- Message get-address
- Message set-address (dengan parameter new-address untuk mengupdate address).

Object Classes dalam OODB

Biasanya, ada banyak objek yang sama dalam basis data. Sama di sini maksudnya mereka merespon message yang sama, menggunakan method yang sama, dan mempunyai variable untuk nama dan tipe yang sama. Akan sangat sia-sia jika kita harus mendefinisikan masing-masing objek secara terpisah. Oleh karena itu, kita menggabungkan objek-objek yang sama ini dalam sebuah class (kelas). Masing-masing objek adalah instansiasi dari kelasnya.

Kelebihan dan Kelemahan OODB

OODB memang memiliki banyak keunggulan dibandingkan dengan RDB (Relational Database). Namun dewasa ini, OODB masih belum dapat menggantikan posisi RDB. Biarpun OODB memang teknologi baru yang sangat terkenal di kalangan ahli sistem basis data, namun para pembuat aplikasi yang belum begitu mengenal teknologi OODB ini masih lebih memilih menggunakan RDB yang sudah biasa mereka gunakan.

Para pembuat aplikasi harus menentukan akan memilih OODB atau RDB dalam proyek yang akan mereka kerjakan. Sebab setiap permasalahan memerlukan penanganan yang berbeda-beda. Bisa saja dalam suatu kasus, menggunakan OODB adalah pilihan yang terbaik, namun bisa juga sebaliknya.

Kelebihan OODB

1. **Desain yang indah**

   Banyaknya waktu dan tenaga yang terbuang dalam menangani data memang menjadi masalah yang umum dalam suatu sistem basis data. Dalam OODB masalah tersebut dapat diminimisasi dengan konsep berorientasi objek yang dimilikinya sebab dengan konsep OO, proses penyimpanan dan pengambilan data akan menjadi lebih sederhana. Selain mendapatkan persistensi data, dengan OODB kita juga mendapatkan persistensi keseluruhan obyek database, bahkan termasuk implemented behaviour-nya. Kita juga dapat dengan mudah memanggil suatu method dari objek tertentu pada database di server sehingga distribusi aplikasinya lebih mudah.

2. **Penyederhanaan pembuatan aplikasi**

   Tanpa disadari, terkadang suatu proyek dapat melambung biayanya karena faktor teknis seperti penggunaan beberapa tool, bahasa program dan lingkungan dari aplikasi yang berbeda-beda. Belum lagi biaya pelatihan dan lain-lain. Dengan OODB kita dapat menyederhanakan pembuatan aplikasi dengan mengurangi penggunaan bahasa pemrograman dan teknologi yang digunakan. Programmer cukup menguasai konsep OO dan bahasa pemrograman OO dengan sedikit tambahan mengenai konektivitas aplikasi dengan database. Selain itu, programmer tinggal memfokuskan pada persistensi objek.

3. **Kinerja yang tangguh**

   Dengan RDB seorang programmer harus menghabiskan waktu dan tenaga untuk memetakan data dengan objek, menguraikan tabel-tabel ke dalam objek dan sebagainya. Terkadang hal ini mencapai sepertiga atau bahkan separuh dari program itu sendiri. Hal ini tentunya akan menyebabkan kinerja menjadi lambat karena harus memetakan objek tersebut, belum lagi bila harus melaksanakan query-query yang kompleks. Masalah tersebut tidak dijumpai dalam OODB, karena dalam OODB, program mengakses data dengan objek nya secara langsung sehingga kinerja program akan lebih tinggi. Lebih dari itu, pada beberapa produk ODBMS bahkan dimungkinkan adanya client caching. Bayangkan kecepatan yang dapat

dihasilkan bila program hanya perlu mengakses cache dari database yang sudah ada di client.

Kelemahan OODB

1. **Tight coupling**

   Coupling berarti keterkaitan antara aplikasi dan database. Tight coupling berarti keterkaitan yang kuat antara aplikasi dan database sehingga aplikasi dan database sulit dipisahkan. Sebenarnya tight coupling dapat menyederhanakan program dan desainnya, namun hal ini juga dapat menyebabkan hilangnya batasan antara aplikasi dan database, juga akan menimbulkan masalah baru bila akan migrasi ke OODB lainnya atau kembali ke RDB.

2. **Kurangnya dukungan platform**

   Pada dasarnya OODB diterapkan untuk dapat berintegrasi dengan semua bahasa pemrograman berorientasi objek, namun sampai sekarang kebanyakan OODBMS hanya mendukung bahasa pemrograman C++ dan Java saja.

3. **Sulit bermigrasi**

   Cara penyimpanan dan pengambilan data dalam OODB sangat berbeda dengan RDB. Begitu juga dengan cara pengaksesannya. Oleh karena itu, dibutuhkan komitmen yang kuat dalam memilih DBMS yang akan digunakan, sekali bermigrasi ke OODB, akan sulit untuk kembali ke RDB.

4. **Kebutuhan ketrampilan**

   Karena OODB masih tergolong baru dan masih relatif jarang penggunaannya, cukup sulit menemukan orang yang memiliki pemahaman OODB bila dibandingkan dengan orang yang memiliki pemahaman RDB. Selain itu, untuk memahami OODB, diperlukan pelatihan khusus sebab terdapat banyak perbedaan pendekatan dengan RDB.

5. **Query yang kompleks**

   Kemampuan logika yang mendalam sangat diperlukan dalam OODB. Masing-masing OODB dapat memiliki cara pemanggilan query yang berbeda-beda. Selain menggunakan object ID-nya saja, pengaksesan suatu data dapat menggunakan range, pola, dan berbagai kriteria lain yang mungkin kelihatan tidak berhubungan.

Aplikasi yang cocok menggunakan OODBMS:

- Aplikasi perancangan rekayasa.
- Aplikasi multimedia
- Aplikasi berbasis pengetahuan
- Aplikasi dengan kebutuhan tersebar dan kongruen
- Aplikasi yang memerlukan fitur-fitur lanjut
- Perangkat elektronik dengan perangkat lunak tempelan.

aplikasi lainnya OODB mungkin akan menjadi hambatan serius. Contohnya OODB tidak cocok digunakan untuk menyimpan data transaksi yang besar seperti data keuangan, kepegawaian dan data transaksi bank. OODB bagus digunakan menggunakan tipe data multimedia dan data yang sifatnya kompleks.

Kesimpulan:

1. OODB memodelkan data sebagai sebuah objek sehingga jika programmer yang ingin menambahkan tipe data baru dengan membuat proyek baru. Pengaksesan dan pengolahan data murni dilakukan dengan pemograman Orientasi Objek.
2. Melihat masih cukup banyaknya kekurangan OODB ini, sebelum memutuskan untuk memakai ODBMS ada baiknya mempertimbangkan kekurangan dan juga mempertimbangkan faktor kebutuhan dan ketersediaan aplikasi serta platform.
3. Pada beberapa aplikasi tertentu penggunaan OODB mungkin sangat tepat. Sebaliknya untuk beberapa

Referensi :

Hariyanto, Bambang. 2004. *Rekayasa Sisem Berorientasi Objek*. Penerbit Informatika, Bandung.

Hariyanto, Bambang. 2007. *Esensi-Esensi Bahasa Pemrograman Java*. Penerbit Informatika.

http://wargabasdat2009.wordpress.com/2009/06/11/eksplorasi-object-oriented-database/

http://en.wikipedia.org/wiki/Object_database

http://www.geocities.com/a_alaydrus/oodb/index.html

http://en.wikipedia.org/wiki/Object-relational_database

http://www.cs.pitt.edu/~chang/156/19oodb.html