BAB II

LANDASAN TEORI

2.1. Sistem Informasi

Pengertian sistem informasi tidak bisa dilepaskan dari pengertian sistem dan informasi. Definisi dari sistem adalah sekelompok dua atau lebih komponen-komponen yang saling berkaitan (*interrelated*) atau subsistem-subsistem yang bersatu untuk mencapai tujuan yang sama (*common purpose*). Sedangkan definisi dari informasi adalah data yang diambil kembali, diolah, atau sebaliknya digunakan sebagai dasar untuk peramalan atau pengambilan keputusan. Sumber dari informasi adalah data. Data adalah fakta dan angka yang tidak sedang digunakan pada proses keputusan, dan biasanya berbentuk catatan historis yang dicatatkan dan diarsipkan tanpa maksud untuk segera diambil kembali untuk pengambilan keputusan. Secara lugas sistem informasi didefinisikan sebagai kumpulan orang, prosedur, *hardware*, *software* yang saling berinteraksi untuk memberikan suatu pelayanan informasi bagi user. Sistem informasi memiliki tiga fungsi dasar :

1. Menerima data (*input*)
2. Mengubah data menjadi informasi (proses)
3. Untuk memproduksi dan mengkomunikasikan informasi ke dalam *timely fashion* bagi *user* untuk membuat keputusan (*output*). Sebagai contoh, banyak bank dan institusi keuangan yang menggunakan sistem informasi untuk membantu menentukan apakah nasabah diperbolehkan untuk melakukan pinjaman.

2.1.1 Komponen Sistem Informasi

Sistem informasi mempunyai enam buah komponen, yaitu *input*, model, *output*, teknologi, basis data, dan kontrol. Keenam komponen ini harus ada bersama-sama dan membentuk satu kesatuan. Jika satu atau lebih komponen tersebut tidak ada, maka sistem informasi tidak akan dapat melakukan fungsinya, yaitu pengolahan data dan tidak dapat mencapai tujuannya, yaitu menghasilkan informasi yang relevan, tepat waktu, dan akurat. Komponen-komponen dari sistem ini dapat digambarkan sebagai berikut ini :

1. *Input*

   Input merupakan data yang masuk ke dalam sistem informasi. Sistem sistem informasi tidak akan dapat menghasilkan *output* jika tidak mempunyai komponen *input*.

2. *Output*

   Produk dari sistem informasi adalah *output* berupa informasi yang berguna bagi para pemakainya. *Output* dari sistem informasi dibuat dengan menggunakan data yang ada di basis data dan diproses menggunakan model tertentu.

3. Basis data

   Basis data adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.

4. Model

   Model yang digunakan di sistem informasi dapat berupa model logika yang menunjukkan suatu proses

perbandingan logika atau model matematik yang menunjukkan perhitungan matematika.

5. Teknologi

   Teknologi merupakan komponen yang penting di sistem informasi. Teknologi dapat dikelompokkan ke dalam dua macam kategori, yaitu teknologi sistem komputer (perangkat keras dan perangkat lunak) dan teknologi sistem telekomunikasi.

6. Kontrol

   Kontrol ini digunakan untuk menjamin bahwa informasi yang dihasilkan oleh sistem informasi sifatnya akurat.

2.1.2 Sistem Informasi Pelayanan Pelanggan

Suatu sistem informasi pelayanan pelanggan dibuat untuk mengelola semua data transaksi yang dilakukan oleh pelanggan dalam suatu perusahaan. Sistem informasi pelayanan pelanggan bertujuan untuk mempermudah dan mempercepat pengaksesan data transaksi pelanggan tanpa melupakan unsur ketepatan dan keamanan data sehingga semua proses transaksi pelanggaan dapat berjalan dengan efisien dan baik yang berakibat pada meningkatnya hubungan baik antara pelanggan dan pihak perusahaan. Oleh karena itu suatu sistem informasi pelayanan pelanggan yang baik harus dapat menyeimbangkan sumber daya yang terdapat di dalam perusahaan dengan kemudahan yang diharapkan oleh pelanggan.

Dalam PDAM (Perusahaan Daerah Air Minum), pelayanan pelanggan juga menjadi suatu poin penting yang diutamakan. Dibutuhkan suatu sistem informasi pelayanan pelanggan yang dapat mengelola data transaksi

pelanggan dari proses pemasangan sambungan baru, balik nama pelanggan, dan pembayaran rekening setiap bulannya.

2.2. *Web Based Information System*

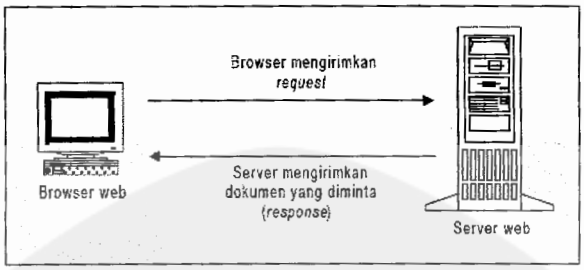
Saat ini komputer dan piranti pendukungnya telah masuk dalam setiap aspek kehidupan dan pekerjaan. Komputer yang ada sekarang memiliki kemampuan yang lebih dari sekedar perhitungan matematika biasa.

Interconnected Network yang biasanya sering disebut dengan *Internet* adalah sebuah sistem komunikasi global yang menghubungkan komputer-komputer dan bermacam jaringan komputer di seluruh dunia. Komputer dan jaringan dengan berbagai *platform* yang mempunyai perbedaan dan ciri khas masing-masing bertukar informasi dengan sebuah protokol *standard* yang dikenal dengan nama *TCP/IP*.

Web adalah fasilitas *hypertext* untuk menampilkan data berupa teks, gambar, bunyi, animasi, dan data multimedia lainnya, yang mana data tersebut saling berhubungan satu sama lainnya.

2.2.1. Web Server

Web server adalah suatu perangkat lunak yang mengatur halaman *web* dan membuat halaman-halaman *web* tersebut dapat diakses di klien, yaitu melalui jaringan lokal atau melalui jaringan *Internet*. Ada banyak *web server* yang tersedia diantaranya *Apache*, *IIS* (*Internet Information Service*), dan *IPlanet's Enterprise server*.



Gambar 2.1 Konsep dasar *browser* dan *server web*

2.2.2. Web Browser

Web browser digunakan untuk menjelajah situs *web* lewat layanan *HTTP*. Untuk mengakses layanan *WWW* (*World Wide Web*) dari sebuah komputer digunakan program *web client* yang disebut *web browser* atau *browser* saja. Jenis-jenis *browser* yang biasa digunakan adalah *Internet Explorer*, *Netscape*, *NCSA Mosaic*, *Arena*, dan masih banyak lainnya.

2.2.3. Web Statis

Web statis merupakan suatu halaman yang berisi skrip HTML editor dan disimpan sebagai *file .htm* atau *.HTML*. Disebut statis karena halaman tersebut dari waktu ke waktu isinya tidak berubah. Karena halaman *web* statis ini tidak memerlukan pemrosesan di *server*, pembuatannya dapat dilakukan menggunakan editor *HTML* dan hasilnya dapat dilihat pada *web browser*.

2.2.4. Web Dinamis

Pembuatan halaman *web* dinamis dapat dilakukan dengan dua cara yaitu secara *client side* atau secara *server side*. Penggunaan *client side* dan *server side* tidak saling bertentangan melainkan saling melengkapi. Seorang *web developer* harus dapat menentukan bagian

mana yang diletakkan secara *client side* dan mana yang diletakkan secara *server side*.

2.2.4.1. Web Dinamis *Client Side*

Di dalam model *client side* ini, terdapat modul-modul atau *plug-in* yang ditambahkan ke suatu *browser* untuk menciptakan halaman *web* dinamis. Pada umumnya skrip *HTML* dikirim ke *browser* bersama dengan *file* yang berisi sekumpulan instruksi, dan *file* tersebut mengacu pada isi halaman *HTML* tersebut. Akan tetapi, juga merupakan hal yang umum jika sekumpulan instruksi tersebut berada menjadi satu dengan *file HTML*. Kemudian, *browser* menggunakan sekumpulan instruksi tersebut untuk menghasilkan skrip *HTML* ketika terdapat *request* pada halaman tersebut.

Singkatnya, halaman tersebut dihasilkan secara dinamis pada saat terdapat *request*. *Client side* sangat berguna untuk mengadakan interaksi dengan *user* dalam frekuensi yang cukup tinggi dan data yang diperlukan relatif sedikit dan telah tersedia sebelumnya.

2.2.4.2. Web Dinamis *Server Side*

Pada model *server side*, skrip *HTML* yang di dalamnya juga terdapat sekumpulan instruksi, dikirimkan ke *web server*. Seperti halnya *client side*, sekumpulan instruksi tersebut digunakan untuk menghasilkan skrip *HTML* ketika ada *request* terhadap halaman tersebut. Perbedaannya adalah kumpulan instruksi pada *client side* diproses di *client* atau di komputer *user* yang me-*request*-nya, sedangkan pada model *server side* kumpulan instruksi tersebut diproses di komputer *server*.

2.3. ORDBMS *(Object Relational Database Management System)*

Pada RDBMS (*Relational Database Management System*), data direpresentasikan ke dalam bentuk tabel dengan kolom dan baris. Baris pada tabel direpresentasikan sebagai nilai dan kolom sebagai atribut. Suatu tabel dapat berhubungan dengan tabel lainnya dengan menggunakan salah satu ataupun banyak kolom sebagai penghubungnya.

Ada beberapa masalah yang dihadapi para pengembang yang menggunakan RDBMS. Pengembang dapat menggunakan tipe data yang telah tersedia di dalam RDBMS, namun para pengembang tidak dapat menambahkan objek nyata seperti murid atau alamat sebagai suatu tipe data. Pada bahasa Pemrograman Berorientasi Objek/*Object Oriented Programming* pengembang dapat memodelkan objek nyata dengan cara merepresentasikannya ke dalam bentuk *class*.

RDBMS cocok untuk aplikasi yang bersifat memproses suatu transaksi, seperti aplikasi *accounting* dimana tipe data yang digunakan sederhana dan jumlahnya sedikit. Namun, RDBMS menawarkan fitur yang terbatas untuk pengembangan aplikasi sejenis CAD (*Computer Aided Design*) Industry dimana tipe data yang digunakan banyak dan sifatnya lebih kompleks. Aplikasi semacam itu membutuhkan struktur bersarang yang tidak disediakan dalam fitur RDBMS.

Salah satu keuntungan dari bahasa pemrograman berorientasi objek seperti C++ ataupun JAVA adalah kemudahan untuk menciptakan komponen yang sifatnya *reusable*. Empat fitur kunci OOP yakni penamaan abstrak (*namely abstraction*), pembungkusan (*encapsulation*),

pewarisan (*inheritance*), dan polimorfisme (*polymorphism*) memudahkan user menciptakan aplikasi yang sifatnya *scalable* dan efisien. Fitur-fitur tersebut memungkinkan kita untuk menggabungkan data dan operasi/*method* yang digunakan untuk memanipulasi data tersebut menjadi satu unit.

Untuk men-*support* penyimpanan dan pengambilan data di basis data dari aplikasi berbasis objek, *programmer* harus mengintegrasikan aplikasi dengan DBMS. *Programmer* dapat mengintegrasikan RDBMS dengan aplikasi yang kita buat sebagai komponen tambahan atau *network service*. Namun nerupakan hal yang sulit untuk mengintegrasikan aplikasi berorientasi objek dengan RDBMS karena secara fundamental *object oriented* dan *relational model* berbeda.

Oleh karena itu dikembangkan suatu model basis data yang menggabungkan fitur RDBMS dan fitur dalam bahasa pemrograman berorientasi objek yang disebut *Object Relational Database Management System*. Dalam hal ini beberapa vendor basis data telah mengintegrasikan konsep ORDBMS, dan salah satunya adalah Oracle yang memunculkan fitur ORDBMS dalam produknya Oracle 9i.

2.3.1. Fitur ORDBMS

ORDBMS merupakan generasi berikutnya dari DBMS yang mengkombinasikan fitur *Object Oriented Programming* ke dalam RDBMS. Keuntungan dari menggunakan ORDBMS adalah konversi data antara format RDBMS dan format *object oriented* disediakan bagi para penggunanya. Oleh karena itu, *programmer* dapat mengakses basis data

melalui bahasa pemrograman berorientasi objek. Berikut ini fitur-fitur dalam ORDBMS :

1. *Extensibility*

   Fitur ini memungkinkan *programmer* untuk mendefinisikan tipe data baru. Dengan kata lain user dapat mengelola data yang sifatnya kompleks seperti *image*, *audio* dan *video*.

2. *Encapsulation*

   Fitur ini memungkinkan *programmer* menciptakan suatu atribut dan *method* di dalam suatu *object type*, dimana jika ingin mengakses atribut atau fungsi itu *programmer* harus membuat instance dari object type tersebut. *Object type* membungkus strukur data bersama dengan *method* untuk memanipulasi data.

3. *Inheritance*

   Dengan fitur ini suatu tabel atau tipe data dapat mewarisi properti yang dimiliki oleh tabel atau tipe data lain. Properti yang ikut diwariskan dapat berupa : struktur, index, fungsi atau prosedur, *constraint*, dan *trigger*.

4. *Polymorphism*

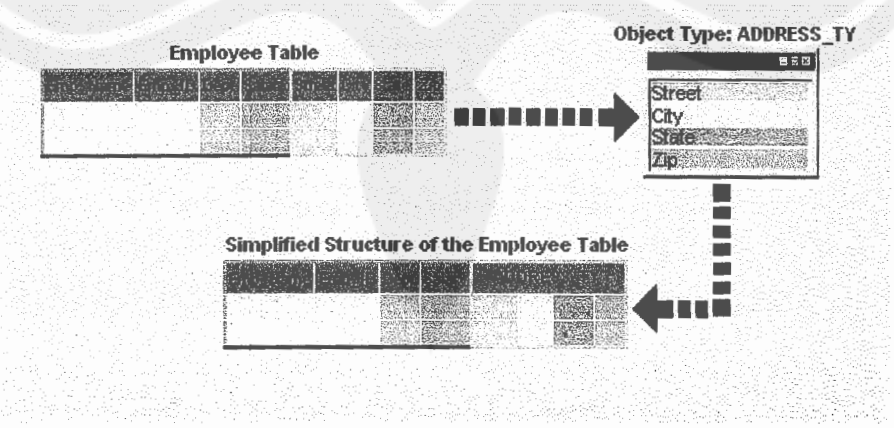
   Dengan fitur ini *programmer* dapat membuat banyak fungsi yang memiliki nama sama namun dengan paramater yang berbeda.

2.4. Oracle 9i ORDBMS

Oracle 9i telah mendukung fitur-fitur untuk mendukung konsep *Object Relational Database Management System*. Dalam menciptakan suatu basis data pada Oracle 9i *programmer* dapat menggunakan tipe data yang telah ada dan dapat menyimpan data dalam bentuk *relational table* seperti DBMS lainnya, namun *programmer* juga dapat menggunakan fitur *object oriented* dalam ORDBMS yaitu dengan menciptakan *object type, object table, object view,* serta fitur-fitur lainnya.

2.4.1. *Object Types*

Object type merupakan tipe data bentukan *user* yang terdiri dari struktur data dan fungsi/*method* yang didefinisikan jika diperlukan perubahan atau modifikasi atribut di dalam *object type* tersebut. *Object type* di dalam ORDBMS hampir sama dengan *class* di dalam pemrograman berorientasi objek. Sama dengan *class*, *object type* juga mengurangi kompleksitas dengan memungkinkan *programmer* merepresentasikan data yang kompleks sebagai suatu entitas tunggal.



Gambar 2.2 *Object Type* di ORDBMS

Sebagai contoh, terdapat tabel *employee* dengan atribut seperti gambar 2.2. Atribut *street*, *city*, *state*, dan *zip* mewakili alamat dari seorang *employee*. Untuk menyederhanakan struktur tabel *employee* maka dapat *object type* Address_Ty yang mengandung atribut *street*, *city*, *state*, dan *zip*. Hal ini memungkinkan *programmer* untuk merepresentasikan alamat *employee* sebagai satu entitas.

Object type terdiri dari dua bagian yaitu atribut dan metode. Atribut adalah komponen struktural yang dapat bertipe data yang telah ada atau tipe data bentukan. Metode adalah fungsi atau prosedur dalam *object type* yang memungkinkan aplikasi untuk menjalankan operasi terhadap atribut di dalam *object type* tersebut.

Keuntungan dari penggunaan *object type*, yaitu :

1. *Encapsulation*

   Dalam ORDBMS fungsi untuk memanipulasi data tersimpan secara terpisah di dalam basis data. Dengan menggunakan *object type programmer* dapat membungkus atribut dengan metode untuk memanipulasi datanya sebagai satu kesatuan.

2. *Reusability*

   Object type yang telah dibuat dapat digunakan berulang kali tanpa perlu didefinisikan ulang.

3. *Network Performance*

   Object type memungkinkan *programmer* mengambil data yang saling berhubungan sebagai satu unit, caranya adalah dengan menggunakan *object references*. Hal ini menyebabkan koneksi ke basis data lebih cepat.

Tipe *method* dalam object type :

1. *Constructor Methods*

   Peranannya hampir sama dengan konstruktor kelas pada pemrograman berorientasi objek yakni pendefinisian atribut pada saat *instance* suatu objek.

2. *Member Methods*

   Merupakan *method* yang didefinisikan oleh *user* untuk mengambil atau memanipulasi data atribut. Jenisnya ada tiga yaitu *Simple methods*, *Map methods*, dan *Order methods* (*comparing functions*).

3. *Static Methods*

   Method yang didefinisikan untuk *object type*, tidak dapat dipanggil oleh instance dari *object type* tersebut.

2.4.2. *Nesting Of Object Types*

Object type dapat bersifat bersarang, yaitu di dalam *object type* terdapat *object type* yang lain. Sebagai contoh *programmer* akan menyimpan data employee di dalam *object type* Emp_Ty, dimana employee memiliki alamat yang disimpan dalam *object type* Address_ty. Maka *object type* Address_Ty digunakan di dalam *object type* Emp_Ty.

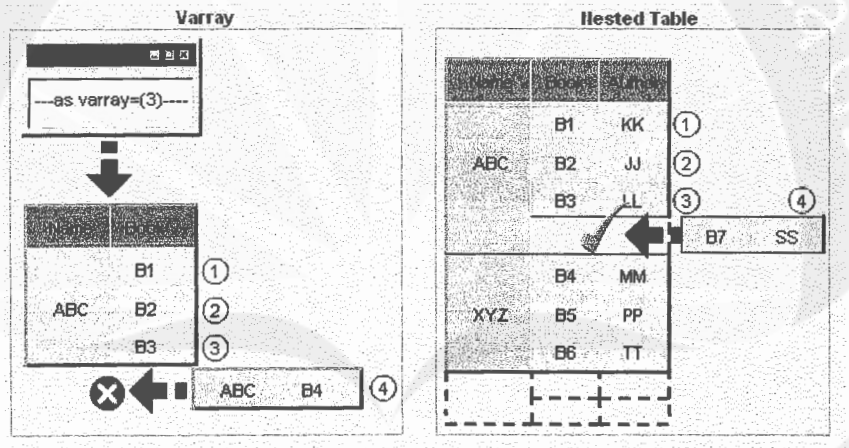
2.4.3. *Object Table*

Object table merupakan tabel yang dibentuk dari satu atau beberapa *object type* yang menyimpan data dalam bentuk *object*. Seorang *programmer* dapat membentuk *object table* dimana setiap baris diperlakukan sebagai satu *object* yang disebut *row object*. Dan apabila salah

satu kolom di dalam *object table* menggunakan *object type* sebagai tipe datanya maka kolom tersebut disebut *column object*.

2.4.4. *Collection Types*

Di dalam Oracle 9i ORDBMS terdapat tipe *collection* yang memungkinkan penyimpanan banyak record data sebagai bagian dari satu baris tunggal di dalam suatu tabel atau *object type*. Hal ini mengurangi kebutuhan akan banyak tabel dan pendefinisan *constraint key*.



Gambar 2.3 *Collection Types* di ORDBMS

Tipe *collection* dalam Oracle 9i ORDBMS ada dua yaitu :

1. *Varrays*

   Tipe *collection* yang digunakan apabila telah diketahui jumlah maksimum data *collection* yang dapat dimasukkan ke dalam satu baris data. Jumlah maksimum baris yang dapat dimasukkan harus didefinisikan pada saat pembuatan *varray* tersebut.

2. *Nested Tables*

   Tabel bersarang, dimana di dalam tabel terdapat tabel yang direpresentasikan sebagai sebuah kolom.

Dengan menggunakan *nested table*, maka jumlah maksimum baris data yang dapat dimasukkan sebagai *collection* tidak terbatas.

2.4.5. *Object View*

Ada saat dimana seorang pengembang harus mengimplementasikan konsep object *oriented programming* dalam suatu aplikasi yang dikembangkan menggunakan *relational table*. Dalam kasus seperti ini dapat digunakan *object view* untuk mendefinisikan suatu objek yang menggunakan data pada *relational table* tanpa harus merubah aplikasi secara keseluruhan.

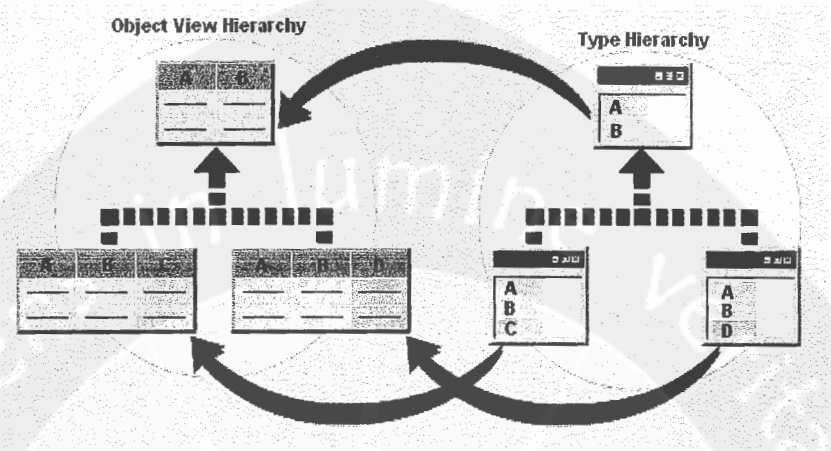
Object view dapat dikatakan sebuah *virtual object table* dimana tiap barisnya direpresentasikan sebagai sebuah objek. Data dari *object view* diambil dari *relational table* ataupun *object table*. Dengan menggunakan *object view*, *programmer* dapat mengambil data tertentu dari satu atau lebih *object table* yang saling berhubungan tanpa memperlihatkan data yang penting. Sebagai contoh dapat dibuat suatu *object view* yang memungkinkan user untuk melihat data spesifik tentang detil *employee* dan menyembunyikan data yang sifatnya rahasia seperti umur dan gaji *employee*.

Menampilkan spesifik data dengan menggunakan *object view* juga meningkatkan *network performance* karena setiap baris diambil sebagai satu unit.

2.4.6. *Object Hierarchy*

Di dalam Oracle 9i ORDBMS terdapat fitur yang *object hierarchy* yang mewakili salah satu sifat *object oriented programming* yaitu *inheritance*. Dua fitur

tersebut yaitu *object type hierarchy* dan *object view hierarchy*.



Gambar 2.4 *Object View* dan *Object Type* di ORDBMS

2.4.6.1. *Object Type Hierarchy*

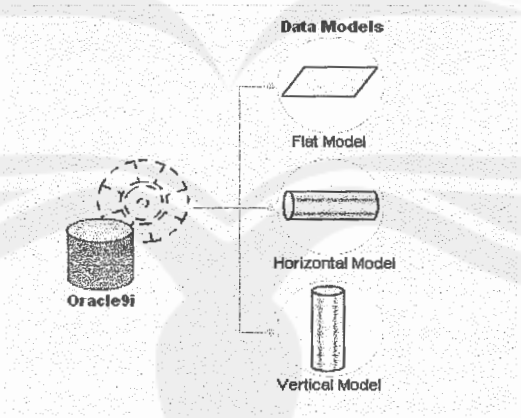
Salah satu fitur teknologi *object relational* di Oracle 9i adalah *type hierarchy*. Hirarki tipe merupakan pohon keluarga dari kumpulan *object type*. Pohon ini mengandung *parent object type* yang disebut *supertype* dan satu atau lebih *child object types* yang disebut *subtypes*.

Suatu *type hierarchy* memungkinkan diciptakannya beberapa *object types* yang sifatnya diturunkan dari satu *object type*. Sebagai contoh dibuat *object type* Employee_Ty sebagai *supertype* dan *object type* Government_Employee_Ty dan Private_Employee_Ty sebagai *subtypes* dari Employee_Ty. Dalam suatu *type hierarchy*, *supertype* dan *subtypes* saling berhubungan satu sama lain dengan sifat *inheritance*. Hal ini berarti *subtypes* secara otomatis mewarisi semua atribut dan metode yang dimiliki oleh *supertype*.

2.4.6.2. *Object View Hierarchy*

Konsep dari *Object View Hierarchy* hampir sama dengan object type hierarchy, dimana *object type hierarchy* merupakan kumpulan dari *object view*. Tiap *object view* merepresentasikan *object type* yang berbeda-beda dalam *type hierarchy*. Dengan menggunakan *object view hierarchy*, dimungkinkan eksekusi *query* dari *object view* yang berbeda dengan spesialisasi dan level yang berbeda juga tingkatannya.

Object view hierarchy dapat dibuat berdasarkan setiap tingkatan *type hierarchy*. Basis dari *object view hierarchy* tidak harus *root* dari suatu *type hierarchy*. Di dalam *object view hierarchy* juga tidak diharuskan untuk memasukkan semua *subtypes* dari suatu *type hierarchy*. Ini berarti *object view hierarchy* juga bersifat dinamis layaknya *object view* ataupun *relational view*.



Gambar 2.5 Data Models Oracle 9i

Suatu object view hierarchy dapat dibentuk berdasarkan tiga model data dalam Oracle 9i. Tiga model data ini merepresentasikan cara berbeda yang masing-

masingnya dapat digunakan untuk menciptakan *object view hierarchy*. Tiga data model tersebut yaitu:

1. *Flat Model*

   Pada model ini satu tabel digunakan untuk membuat semua *view* yang digunakan dalam *object view hierarchy*. Kelemahan dari model ini yaitu

   - Satu tabel tunggal tidak dapat mengandung lebih dari 1000 kolom.
   - Nilai null harus didefinisikan pada atribut tiap tabel tidak dari *object type*-nya.

2. *Horizontal Model*

   Berbeda dari *flat model*, pada model ini jumlah tabel yang digunakan untuk membuat view dala *object view hierarchy* adalah satu untuk masing-masing *view*. Kelemahan dari model ini yaitu tidak dapat digunakannya perintah SELECT * FROM untuk mengakses semua data dari tabel.

3. *Vertical Model*

   Hampir sama dengan *horizontal model*, pada model ini tiap *view* yang dibuat berdasarkan tabel yang berbeda. Namun tabel pada model ini hanya mengandung atribut yang sifatnya unik terhadap *subtype*-nya. Kelemahan dari model ini adalah selama proses *instance* dari suatu *object type*, satu *object identifier* harus didefinisikan untuk setiap *subtype* yang dipisahkan dari basis *object view hierarchy*.