BAB II

LANDASAN TEORI

2.1 Sistem Terdistribusi

Sistem terdistribusi merupakan suatu bentuk arsitektur sistem dimana komputer-komputer yang berdiri secara otonom dapat saling berkomunikasi dan berbagi *resource* tanpa mempedulikan dimana komputer itu berada dan platform yang digunakan.

"A distributed system is a collection of independent computers that appear to the users of the system as a single computer." [TAN02]

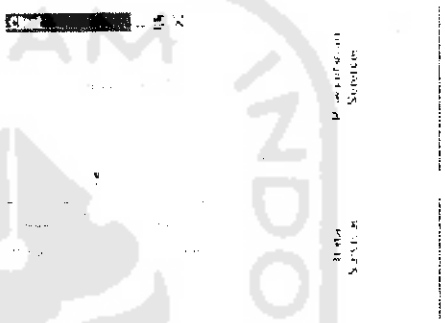
Salah satu arsitektur yang digunakan oleh sistem terdisribusi adalah client/server.

2.2 Model Client/Server

2.2.1 Model Two Tier

Pada arsitektur ini sebuah aplikasi dibagi menjadi 2 bagian, yaitu *front-end client* yang menyajikan informasi kepada *user*, dan *back-end server* yang mengelola database dan lazimnya disebut database server, seperti yang ditunjukkan pada gambar 2.1. Dalam konfigurasi yang tipikal, pembagian ini juga meliputi pembagian hardware dan software. Client biasanya terletak pada workstation yang digunakan oleh user, dan dibuat dengan menggunakan program seperti PowerBuilder, SQLWindows, Visual Basic, atau Delphi. Sedangkan

server adalah suatu komputer server yang diletakkan di bagian lain pada jaringan yang menjalankan perangkat lunak database software, seperti Sybase, atau Oracle. Pada sistem *two-tier*, aturan bisnis diterapkan pada program front end atau dapat juga di database server. Dengan menggunakan cara tersebut pada model two-tier, sistem akan lambat bila proses yang harus dilakukan adalah besar.



Gambar 2.1 Arsitektur Two Tier

2.2.2 Model Three-tier

Sedikit berbeda pada arsitektur two-tier, model three-tier dibagi menjadi tiga lapisan, yaitu :

a. User service (presentation tier)

Lapisan ini menyajikan data kepada user dan merupakan unit aplikasi yang digunakan langsung oleh user. Dua tipe utama dari user interface pada lapisan ini adalah *traditional application* dan Web-based application. Saat ini umumnya *Web-based application* berisi sebagian besar fitur manipulasi data yang digunakan oleh *traditional application*.

b. Business service (middle tier)

Lapisan ini menyediakan beberapa *service* yang dapat digunakan oleh *client application* untuk mengakses data. Unit ini merupakan penghubung antara *user service* dengan *data service*. Lapisan ini disebut juga sebagai *business logic tier*, karena bagian ini menjalankan program yang berisi aturan-aturan aplikasi, seperti menerapkan aturan bisnis dan melakukan pemrosesan yang kompleks.

c. Data service (data tier)

Lapisan ini bertugas untuk mengatur penyimpanan data. Biasanya berbentuk RDMS. Lapisan ini terdiri atas komponen-komponen *data access* untuk membantu dalam hal pembagian sumber daya dan memungkinkan client untuk mengkonfigurasi tanpa menginstall DBMS libraries dan ODBC drivers pada setiap client.

Arsitektur three-tier dapat dilihat pada gambar 2.2.

Gambar 2.2 Arsitektur Three Tier

Banyak keuntungan dari model *three-tier* ini diantaranya yang terpenting adalah peningkatan unjuk kerja atau performance. Sehingga dapat dikatakan model ini mampu menyediakan satu solusi Graphical User Interface (GUI) yang user friendly pada suatu komputer desktop client, yang mengemas proses suatu transaksi berukuran amat sangat besar yang biasanya hanya diperoleh dan ditemukan pada komputer mainframe. Hal ini memecahkan kebutuhan untuk menyediakan suatu sistem pemrosesan besar dengan GUI yang memiliki 40 sampai dengan 50 user.

Hal lainnya adalah skalabilitas. Arsitektur ini dapat dengan cepat dan mudah menaikkan jumlah transaksi pengguna tanpa perlu perubahan besar pada investasi hardware dan software. Misalkan pada suatu cliet server yang two-tiered yang meletakkan prosedur penyimpanan order pada database server. Ketika volume transaksi membesar, beban database server menjadi tinggi, sehingga berimbas pada menurunnya kinerja. Solusi penambahan database server sulit untuk dilakukan karena berkaitan dengan biaya, instalasi, dan sebagainya. Pada sistem three-tier, masalah ini dengan mudah dapat dipecahkan, yaitu dengan cara menambahkan middle-tier server. Setiap server menjalankan program business server yang sama. Tidak menjadi masalah client mana yang dilayani, karena setiap client dapat melakukan pemrosesan transaksi dengan koneksi server yang manapun, ketika server lain sibuk.

Suatu aplikasi *three-tier* juga memudahkan pihak perusahaan untuk pindah ke berbagai *database server*. Jika pada aplikasi *two-tier*, ketika butuh mengubah *database server* dari SQL Server ke Oracle dan sebagainya, maka seluruh *stored*

prosedur harus ditulis ulang, karena sangat bergantung pada dialek SQL dari tiap produk tersebut. Pada arsitektur *three-tier*, perubahan *stored prosedure* ini diminimalkan karena proses logik dilakukan di *middle-tier*.

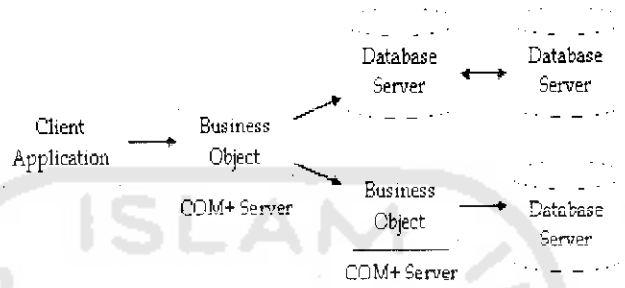
Misalkan pada model client server *two-tier* yang lain proses aturan bisnis diletakkan di program client. Hal ini memiliki potensi untuk menimbulkan masalah baru. Permasalahan perawatan sistem adalah masalah yang terbesar. Bagaimana untuk melakukan *upgrade* pada beratus-ratus *client*, dengan memrogram aturan bisnis yang baru. Arsitektur three-tier memecahkan masalah ini, karena pengubahan aturan bisnis hanya perlu dilakukan di middle server saja. Sehingga mudah untuk dilakukan perawatan perangkat lunak.

2.3 COM+

COM+ (Component Object Model *plus*) merupakan standar aplikasi dari Microsoft yang digunakan untuk menyaingi sistem CORBA yang berbasis objek. COM+ ini merupakan teknologi yang dahulu dikenal sebagai COM atau MTS (Microsoft Transaction Server).

COM+ berisi layanan yang memungkinkan aplikasi menjadi lebih fleksibel, lebih cepat, mudah dalam hal perawatan, dan memiliki kemampuan menangani banyak user. Selain itu COM+ dapat memberikan keleluasaan bagi programmer untuk membuat aplikasi dengan menggunakan arsitektur berbasis objek.

Salah satu tujuan dari COM+ adalah untuk menyederhanakan pengembangan aplikasi dari sistem terdistribusi. Skema COM+ dutunjukkan pada gambar 2.3.



Gambar 2.3 COM+ dalam Sistem Terdistribusi

Versi pertama teknologi yang disebut sebagai COM, diluncurkan oleh Microsoft pada tahun 1993, sejak saat itu, COM menjadi strategi andalan Microsoft mengadopsi sistem three-tier, disamping beberapa teknologi lainnya seperti OLE, DCOM, ActiveX, dan MTS.

Salah satu alasan terbaik dalam menggunakan COM+ adalah untuk merubah sistem informasi perusahaan dari model arsitektur client server 2-tier ke model three-tier. Kebutuhan yang selanjutnya timbul dalam transisi ini adalah merancang dan membuat objek - objek bisnis dalam lapisan middle-tier.

2.3.1 COM+ Service

Berikut ini adalah service-service terpenting yang disediakan oleh COM+ :

a. Administration

Tools dan service yang memungkinkan programmer dan administrator untuk mengkonfigurasi dan mengatur aplikasi *component based*. Tool terpenting adalah Window Component Services, seperti ditunjukkan pada gambar 2.4. COM+ juga menyediakan sebuah lokasi standar, katalog COM+, untuk menyimpan informasi konfigurasi.

b. Just-in-Time Activation (JITA)

Service yang memungkinkan COM+ membuat suatu objek hanya ketika objek itu dibutuhkan dan menghapusnya ketika tugasnya telah selesai.

c. Object pooling

Service yang mengizinkan penciptaan objek secara terus menerus. Object pooling dapat meningkatkan dayaguna dari aplikasi terdistribusi.

d. Transaction

Service yang mengizinkan operasi-operasi dilaksanakan oleh komponen-komponen terdistribusi dan sumber daya seperti database yang diproses sebagai operasi tersendiri.

e. Synchronization

Service yang berfungsi untuk mengontrol akses yang dilakukan secara bersamaan ke objek-objek.

f. Security

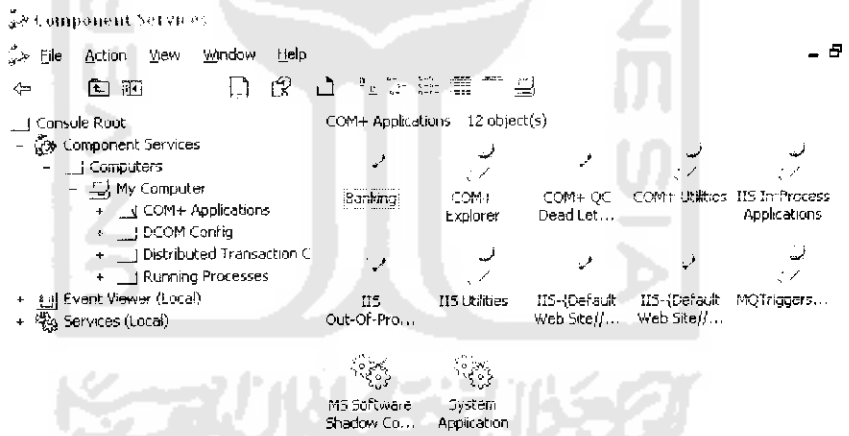
Service untuk mengautentikasi client dan mengontrol akses ke aplikasi. COM+ mendukung *role-based security*.

g. Queued component

Service yang mengizinkan komponen-komponen saling berkomunikasi melalui pesan tidak langsung (asynchronous), sebuah fitur yang memungkinkan *loosely coupled application* atau bahkan *disconnected application*.

h. Event

Service yang mengizinkan komponen-komponen untuk menginformasikan kejadian-kejadian penting, seperti perubahan data atau keadaan sistem.



Gambar 2.4 Window Component Service

2.3.2 COM+ Transaction

COM+ menyediakan konfigurasi administratif dari dukungan transaksional untuk komponen-komponen . COM+ memungkinkan sumber daya berperan dalam transaksi serta mendukung pengelolaan dan pengeksekusian transaksi. Servis manajemen transaksi COM+ didasarkan pada model manajemen transaksi MTS, dengan sedikit inovasi dan perbaikan.

2.3.3 Arsitektur Transaksi COM+

COM+ adalah sebuah Transaction Processing Monitor yang menawarkan komponen dengan kemudahan konfigurasi administratif untuk kebutuhan transaksional. Arsitektur transaksi COM+ menjelaskan beberapa konsep dasar yang diperlukan untuk memahami keuntungan yang didapat dari dukungan transaksi COM+, seperti resource manager, transaction root, two phase commit protocol, dan Distributed Transaction Coordinator (DTC).

2.3.4 Manajemen Sumber Daya dengan COM+

Salah satu trik untuk mengembangkan aplikasi yang sangat *scalable* adalah manajemen sumber daya yang efektif. COM+ mendukung banyak mekanisme baru yang memungkinkan para pengembang untuk meningkatkan alokasi sumber daya dan memori, seperti object pooling dan konstruksi objek yang terparameterisasi. Dukungan sumber daya adalah bagian lain yang penting dalam perilaku MTS/COM+ application server.

2.4 Online Transaction Processing (OLTP)

OLTP (Online Transaction Processing) adalah sebuah bentuk pemrosesan transaksi melalui jaringan komputer [WIK05]. Contoh dari aplikasi OLTP yaitu electronic banking, sistem penjadwalan pegawai, e-commerce, dan lain-lain.

OLTP system ditunjukkan dengan banyaknya user yang mengakses data online secara bersamaan. Data yang diakses dengan OLTP system terdiri atas satu atau lebih database. Saat ini banyaknya user yang mengakses OLTP system meningkat menjadi ribuan bahkan milyaran.

Saat ini sebagian besar OLTP system menggunakan arsitektur two-tier client/server. Arsitektur two-tier client/server terdiri atas dua bagian yaitu: front-end application dan back-end data storage.

Arsitektur lain yang digunakan OLTP system adalah three-tier client/server. Salah satu kelebihan arsitektur ini adalah memungkinkan application server untuk mengatur semua koneksi client ke database server, karena apabila tiap client membuat koneksi masing-masing akan menghabiskan banyak resource pada database server Konsep ini disebut dengan *connection pooling*, yang artinya *request* dari *client* diletakkan dalam suatu antrian untuk menunggu koneksi yang tersedia. Ketika koneksi tersebut ada, maka dapat digunakan untuk *request* dalam antrian. Software connection pooling tersebut dikenal sebagai Transaction Processing (TP) Monitor.

Dua faktor utama yang harus dipertimbangkan dalam membangun sebuah OLTP system menjadi *two-tier* atau *three-tier* adalah kecepatan dan efisiensi sumber daya.

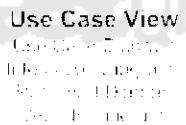
2.5 Integrasi antara ASP dan COM+

Integrasi antara ASP dn COM+ memberikan kemudahan untuk membuat dan menjalankan objek–objek bisnis yang ada pada halaman web. Hampir semua *business logic* yang dibuat dapat didistribusikan secara *embedded* di halaman-halaman ASP.

2.6 UML

UML merupakan bahasa standar untuk menulis rekonstruksi Perangkat Lunak. UML hanyalah bahasa yang merupakan salah satu bagian dari metode pembangunan perangkat lunak. UML adalah bahasa yang *process-independent*, yaitu bahasa yang saling bebas terhadap proses yang ada di dalamnya. Oleh sebab itu UML dapat digunakan dalam semua metode pembangunan Perangkat Lunak.

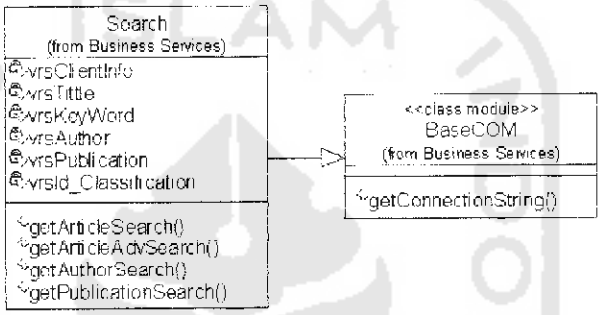
Arsitektur sistem menurut spesifikasi UML, dapat dideskripsikan dalam lima sudut pandang (*view*) yang dapat digambarkan seperti pada gambar 2.5.



Gambar 2.5 Arsitektur Sistem UML

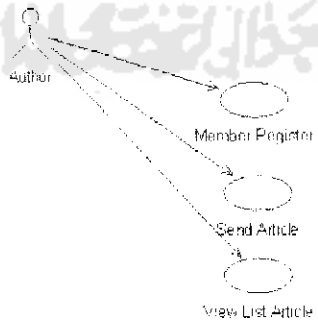
Diagram yang menyusun adalah sebagai berikut:

a. Diagram Class, menggambarkan seluruh class yang terlibat dalam sebuah perangkat lunak. Diagram ini menggambarkan seluruh atribut dan operasi suatu class beserta hubungannya dengan class lainnya, seperti yang ditunjukkan pada gambar 2.6.



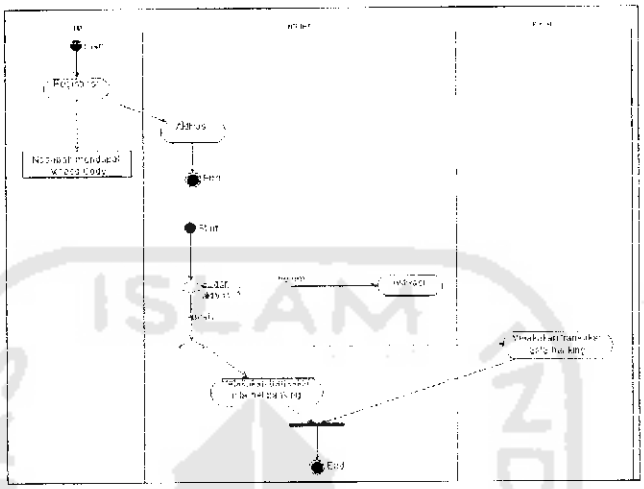
Gambar 2.6 Diagram Class

b. Diagram Use Case, menggambarkan bagaimana penggunaan suatu perangkat lunak dari sudut pandang pengguna, seperti yang ditunjukkan pada gambar 2.7. Diagram ini juga dilengkapi dengan uraian penggunaan perangkat lunak secara langkah demi langkah.



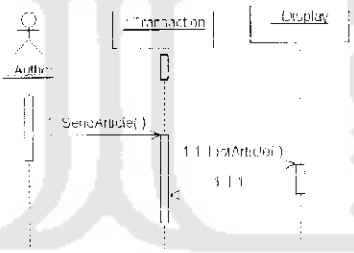
Gambar 2.7 Diagram Use Case

c. Diagram Aktivitas, adalah grafik *flowchart* yang disempurnakan. Diagram ini menggambarkan proses yang terjadi dalam sebuah organisasi atau objek.



Gambar 2.8 Diagram Aktivitas

d. Diagram Sequence, menggambarkan interaksi antar objek dalam sebuah sistem dalam sebuah urutan waktu, seperti yang ditunjukkan pada gambar 2.8.



Gambar 2.9 Diagram Sequence

e. Diagram Kolaborasi, adalah sebuah cara alternatif dalam menggambarkan interaksi antar objek.

f. Diagram Keadaan, menggambarkan keadaan sebuah objek dalam suatu waktu.

g. Diagram Komponen, menggambarkan komponen penyusun (modul) sebuah perangkat lunak.

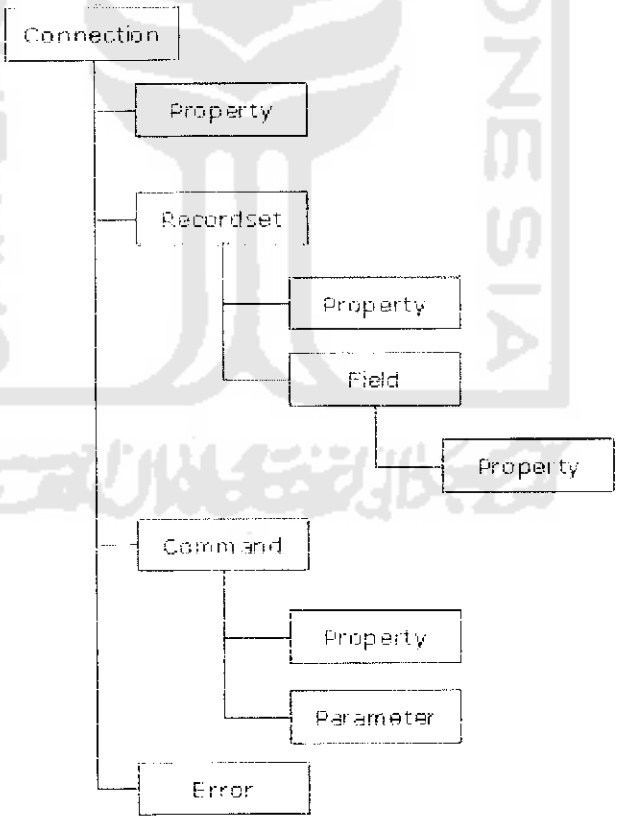
h. Diagram Deployment, menggambarkan arsitektur sistem dimana perangkat lunak tersebut diinstalasikan.

2.7 ADO

ActiveX Data Object (ADO) adalah model akses data tingkat tinggi dan metode yang direkomendasikan pada pengembangan aplikasi bervendor Microsoft. ADO menyediakan antarmuka pemrograman antara halaman web database dengan berbagai sumber data. Sama halnya untuk sumber data dasar, ADO menyediakan sekumpulan besar objek (tabel 2.1), metode dan properti. Hubungan yang ada di antara objek ADO ditampilkan pada gambar 2.10. ADO digunakan bersamaan dengan OLE DB, yakni low level interfaces (antar muka tingkat rendah) yang dapat digunakan untuk mengakses data apapun format datanya, baik database SQL, sistem e-mail, flat-file, dan lain-lain. OLE DB lebih lengkap daripada Open Database Connectivity (ODBC), yang hanya digunakan untuk database SQL.

Tabel 2.1 Objek-objek ADO Utama

Objek	Deskripsi
Connection	Menyediakan jalan yang digunakan objek lain untuk mengakses penyedia database
Recordset	Menyediakan akses ke tempat hasilnya dari operasi query atau yang lainnya
Field	Menyediakan akses ke field individual dalam record dari recordset yang ada
Property	Menyediakan akses ke berbagai karakteristik objek ADO
Command	Menangani perintah dan bisa juga parameter perintah yang akan dijalankan lewat koneksi database
Parameter	Menangani nilai yang telah dinamai yang akan digabungkan oleh penyedia database ke dalam query yang didefinisikan sebelumnya atau prosedur yang telah disimpan
Error	Berisi informasi tentang kesalahan yang dilaporkan oleh penyedia database



Gambar 2.10. Hirarki Objek ADO

Object **Recordset** adalah salah satu objek ADO yang digunakan dalam penelitian ini untuk memudahkan komunikasi data antara aplikasi *client* dan middle-tier. Setiap data yang dikirimkan lintas tier berbentuk objek recordset, sehingga memudahkan bahasa pemrograman yang mendukung ADO untuk mengaksesnya.

UNIVERSITAS ISLAM INDONESIA